UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE MATEMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM MATEMÁTICA APLICADA

Cálculo da Complexidade Exata de Algoritmos do tipo Divisão-e-Conquista através das Equações Características

por

Aline Brum Loreto

Dissertação submetida como requisito parcial
para a obtenção do grau de

Mestre em Matemática Aplicada

Prof. Dr. Rudnei Dias Cunha
Orientador

Prof. Dr.(a) Maria Paula Gonçalves Fachin
Co-orientadora

Porto Alegre, março de 2000.

CIP - CATALOGAÇÃO NA PUBLICAÇÃO

Loreto, Aline Brum

Cálculo da Complexidade Exata de Algoritmos do tipo Divisão-e- Conquista através das Equações Características / Aline Brum Loreto.— Porto Alegre:PPGMAp da UFRGS, 2000.

116 p.: il.

Dissertação (mestrado) — Universidade Federal do Rio Grande do Sul, Programa de Pós-Graduação em Matemática Aplicada, Porto Alegre, 2000. Orientador: Cunha, Rudnei Dias da ; Co-orientador: Fachin, Maria Paula Gonçalves.

Dissertação: Complexidade, Algoritmos Recursivos, Divisão-e-Conquista, Equações Características.

AGRADECIMENTOS

Agradeço a Deus, mestre dos mestres, pela proteção e por tudo que consegui consquistar na minha vida.

Ao Émerson, se não fosse ele, não estaria concluindo mais esta etapa. Obrigado pela paciência, compreensão, estímulo e principalmente, pelo companheirismo.

A amiga e professora Laira, pela confiança e incentivo no meu trabalho e em minha pessoa.

Aos professores do PPGMAp, pelos conhecimentos e ensinamentos passados no decorrer destes dois anos.

Aos colegas do mestrado, por tudo, solidariedade, companheirismo, paciência e principalmente, pela amizade.

A todos, que diretamente ou indiretamente, contribuíram para que eu chegasse onde cheguei.

SUMÁRIO

LISTA DE TABELAS

LISTA DE ABREVIATURAS

$T(n)$	tempo de execução da rotina recursiva para um problema de tamanho n.
$f(n)$	complexidade da primeira chamada recursiva do algoritmo.
m	número de subproblemas, ou número de chamadas recursivas.
$r(n)$	tamanho do subproblema.
b	complexidade da base da recursivdade.

RESUMO

A equação de complexidade de um algoritmo recursivo pode ser expressa em termos de uma equação de recorrência. A partir destas equações obtém-se uma expressão assintótica para a complexidade, provada por indução. Neste trabalho, propõe-se um esquema de solução de equações de recorrência usando equações características que são resolvidas através de um "software" de computação simbólica, resultando em uma expressão algébrica exata para a complexidade. O objetivo é obter uma forma geral de calcular a complexidade de um algoritmo desenvolvido pelo método Divisão-e-Conquista.

Palavras-chave: Equações de Recorrência, Equações Características, Complexidade, Divisão-e-Conquista, Algoritmos Recursivos.

ABSTRACT

The complexity of a recursive algorithm can be expressed in terms of a recurrence equation. From these equations it is obtained an asymptotic expression to the complexity, proved by induction. It's proposed, in this work, a method to solve recurrence equations using characteristic equations solved with a simbolic computational software. In this way an exact algebraic expression to the complexity is obtained. The purpose of this proceeding is to find a general form to compute the complexity of an algorithm developed by divide-and-conquer scheme.

Keywords: Recurrence Equations, Characteristic Equations, Complexity, Divide-and-Conquer, Recursive Algorithms.

1 INTRODUÇÃO

O presente trabalho destina-se ao cálculo da complexidade exata, no pior caso, de algoritmos desenvolvidos pelo método Divisão-e-Conquista através da resolução das equações de recorrência. Esta resolução, por sua vez, dá-se através de polinômios característicos, método utilizado na solução de equações em diferenças.

Um algoritmo é um procedimento consistindo de um conjunto de regras não ambíguas as quais especificam uma seqüência finita de operações, que produz uma solução para um problema ou uma classe de problemas. Os algoritmos recursivos são muito usados em computação, porém a sua complexidade é representada por uma equação de recorrência cuja solução é sempre a parte mais difícil para o analista de algoritmos.

Realizando uma análise em um algoritmo recursivo pode-se desenvolver uma relação de recorrência descrevendo importantes características de desempenho; dada uma relação de recorrência, pode-se calcular ou estimar os parâmetros necessários nas aplicações práticas.

A recursividade é uma técnica de programação poderosa. Sabe-se que o tempo de execução do algoritmo é penalizado em alguns compiladores, o que torna o cálculo de eficiência mais interessante para compararmos dois ou mais algoritmos. Entretanto, o cálculo da complexidade para algoritmos recursivos é muito mais complicado do que para algoritmos não recursivos. A análise da complexidade de um algoritmo é, usualmente, realizada de maneira muito particular, o que é compreensível, já que a complexidade é uma medida que tem parâmetros bem específicos do algoritmo. Em geral, fórmulas envolvendo a complexidade são provadas por indução, o que exige uma determinação de uma fórmula candidata para expressar a complexidade, a "*priori*". [TOS88]

Acredita-se que desenvolvendo técnicas do cálculo de complexidade se estará incentivando o uso da recursividade como técnica de programação. Esse trabalho teve como ponto de partida [LUE80] e o conhecimento das dificuldades do cálculo da complexidade de algoritmos recursivos.

Este trabalho propõe uma solução direta da equação de complexidade através da

redução para equações de recorrência homogênea e posterior solução. O método tem a vantagem de poder ser usado sem se ter idéia da possível forma da complexidade (não há necessidade de se encontrar uma fórmula candidata), além de permitir a solução exata, isto é, o número exato de operações e não a ordem de complexidade [LOR99]. Muitas vezes o estudo da complexidade computacional depende da solução das recorrências para estimar ou limitar o desempenho dos algoritmos recursivos.

Nosso trabalho está organizado como apresentado a seguir: o primeiro capítulo, Considerações Gerais, é subdividido em três seções: Análise da Complexidade, Relações de Recorrência e Métodos Gerais de resoluções de equações recorrência. A primeira seção apresenta os conceitos de complexidade: complexidade de tempo, medidas de complexidade, ordens de complexidade e análise da complexidade de algoritmos desenvolvidos pelo método da Divisão-e-Conquista. A seção seguinte apresenta tipos de relações de recorrências, classificação e características básicas das mesmas, enquanto a terceira seção descreve uma variedade de métodos de resolução para os diversos tipos de recorrências descritos por diferentes autores. Alguns destes métodos resultam em solução exata da recorrência e outros na solução aproximada das mesmas.

O segundo capítulo, Equações Características para Complexidade, apresenta o método e a resolução de recorrências para o cálculo da complexidade exata em função dos tamanhos dos subproblemas. A resolução genérica dos casos apresenta-se através de exemplos numa ordem crescente de generalidade.

No quarto capítulo, Análise do polinômio característico, realiza-se um estudo em relação ao formato do polinômio característico, justificando a repetição das raízes características quando aumentado o grau do polinômio da equação de recorrência.

No último capítulo, Conclusão, faz-se um relato sobre o formato do polinômio característico proveniente da homogeneização da equação de recorrência e realiza-se uma análise das complexidades exatas obtidas após a aplicação do método das equações características em cada tamanho de subproblema.

Todos os cálculos realizados neste trabalho, ou seja, as raízes das equações características, a resolução do sistema de equações (obtido da equação geral da recorrência original) e o resultado final de $T(n)$, foram obtidos através do aplicativo de programação simbólica Maple V. [ELL92]

2 CONSIDERAÇÕES GERAIS

Neste capítulo apresenta-se, de forma geral, os conceitos que serviram de base no desenvolvimento deste trabalho, e os diversos métodos de resolução de equações de recorrência.

2.1 Análise da Complexidade

A análise de um algoritmo tem como objetivo melhorar, se possível, seu desempenho e escolher dentre os algoritmos disponíveis o melhor. Existem vários critérios de avaliação de um algoritmo como: quantidade de trabalho requerido, quantidade de espaço (memória) necessário, simplicidade e exatidão da resposta.[TOS88]

2.1.1 Definição de Complexidade

O termo *Complexidade* refere-se, em geral, aos requerimentos de recursos necessários para que um algoritmo possa resolver um problema sob o ponto de vista computacional, ou seja, é a quantidade de trabalho despendido pelo algoritmo.

Segundo [TOS88], dado um problema p com solução α, seja a o conjunto de todos algoritmos que resolvem p. Para calcular a complexidade de um algoritmo $a \in \boldsymbol{a}$ é necessário escolher a operação fundamental (ou operações fundamentais) e definir a função que dá o tamanho da entrada. Seja E o conjunto de todas as seqüências de execuções de operações fundamentais. Tem-se então:

- *Execução*: a x D$\rightarrow E$; *execução*(a, d):=seqüência de execuções de operações fundamentais efetuadas na execução do algoritmo a, com entrada d.
- *Custo*: E$\rightarrow$IN; *custo*(s):= comprimento da seqüência s, definido conforme o peso estabelecido para as operações fundamentais.
- *Tamanho*: D$\rightarrow$IN; *tamanho*(d):= tamanho da entrada d.

Escolhidas as funções fundamentais e a função *tamanho*, o conjunto $\boldsymbol{a}$ é particionado (ficam na mesma classe os algoritmo com mesmas funções fundamentais e

mesma função *tamanho*). A complexidade é então definida dentro de cada classe. Considere A como uma dessas classes e defina a complexidade de um elemento de A, como uma função do tipo IN→IN. A *complexidade* é portanto um funcional do tipo A→(IN→IN), isto é *complexidade*: A→f e f:={f/f: IN→IN}. Para ***a*** ∈ A, *complexidade(a)*=f, f: IN→IN. Dado $n \in$ IN deve-se definir f(n). Para avaliar f(n) é necessário considerar todas as entradas d ∈ D com *tamanho*(d)=*n*, calcular *custo*(*execução*(a,d)) e então conforme o critério de complexidade desejado estabelecer f(n). Assim pode-se dizer que:

f(n):=*avaliação*({*custo*(*execução*(a,d))/d ∈ D e *tamanho*(d)=*n*}).

2.1.2 Medidas de Complexidade

Existem várias medidas de complexidade. Se a complexidade é tomada como a máxima para qualquer entrada de um dado "tamanho", a complexidade é chamada *Complexidade no pior caso* ou simplesmente Complexidade, como será chamado neste trabalho. Se é levada em conta a probabilidade de ocorrência de cada entrada de um mesmo "tamanho", a complexidade é chamada *complexidade esperada* ou *complexidade média*. O "tamanho" da entrada é sempre representado por um número natural.

2.1.3 Complexidade no Pior Caso

O critério de avaliação mais utilizado, entre as medidas de complexidade, é a complexidade no pior caso. Neste trabalho trataremos dessa medida.

Geralmente constrói-se o algoritmo e depois analisa-se sua complexidade. Este processo seria mais eficaz se a complexidade fosse um fator integrante do desenvolvimento do algoritmo. Uma metodologia de cálculo de complexidade poderia incentivar esta prática. Mas, apesar da complexidade no pior caso ser uma medida muito particular da classe do algoritmo que está sendo analisado, alguns aspectos do cálculo da complexidade não dependem do que faz o algoritmo mas somente de sua estrutura. Dado um algoritmo *a*, a complexidade no pior caso é definida pela função $T(a): N \rightarrow N$, a qual, dada um tamanho de entrada *n*, leva ao número máximo de operações básicas necessárias para executar *a* considerando-se todas as entradas de tamanho *n*. Normalmente, o algoritmo *a* está bem determinado, então usa-se somente a notação $T(n)$, para a complexidade no pior caso do

algoritmo a para uma entrada de tamanho n. Analisando o algoritmo, deve-se ser capaz de estimar este máximo, ou estimar seu crescimento assintótico.

2.1.4 Ordens de Complexidade

A complexidade de um algoritmo pode ser expressa em notação assintótica. Em [HOR78] e [COR90] encontram-se as ordens de complexidade mais utilizadas, suas definições e aplicações, das quais algumas são apresentadas a seguir:

Definição 2.1: *Notação-0*: $f(n)=0(g(n))$ (sendo f e g funções de IN→IR) se existirem duas constantes positivas c e n_0 de modo que $|f(n)| \leq c|g(n)|, \forall\ n \geq n_0$.

A notação-*0* descreve um limite assintótico superior sobre $f(n)$. Supõe-se que deve-se determinar o tempo $f(n)$ de processamento de um algoritmo, onde n representa o tamanho do problema ou o número de entradas ou saídas, ou ainda, a magnitude do mesmo. Uma análise a *priori* não é suficiente para determinar $f(n)$, entretanto, podemos determinar $g(n)$ de modo que $f(n)=0(g(n))$. Quando diz-se que um algoritmo tem tempo de processamento $0(g(n))$, significa que o algoritmo executando o mesmo tipo de dado para valores crescentes de n, terá o tempo resultante sempre menor que algum tempo $c|g(n)|$. Quando se determina a ordem de complexidade de $f(n)$, tenta-se, sempre, obter o menor $g(n)$ de modo que $f(n)=0(g(n))$.

TEOREMA 2.1: Se $A(n)=a_k n^k+...+a_1 n+a_0$ é um polinômio de grau k então $A(n)=0(n^k)$.
PROVA: Usa-se a definição de $A(n)$ com uma simples desigualdade; tomando valores absolutos

$$|A(n)| \leq |a_k| n^k+...+|a_1| n+|a_0|$$
$$|A(n)| \leq (|a_k|+|a_{k-1}|/n+...+|a_0|/n^k) n^k$$
$$|A(n)| \leq (|a_k|+...+|a_0|) n^k \qquad \text{para } n \geq 1.$$

O teorema 2.1 demonstra que se for possível descrever a complexidade de execução de um algoritmo por um polinômio tal como $A(n)$ de grau k, o tempo de processamento da declaração é de ordem $0(n^k)$.

Definição 2.2: *Notação-Ω*: $f(n)=\Omega(g(n))$ (sendo f e g funções de IN→IR) se existirem constantes positivas c e n_0 de modo que $|f(n)| \geq c\,|g(n)|$, $\forall\ n \geq n_0$.

A notação- Ω descreve um limite assintótico inferior sobre $f(n)$ é usada para estimar o tempo de processamento de um algoritmo. Em alguns casos, o tempo de execução de um algoritmo, $f(n)$, é tal que $f(n)=\Omega(g(n))$ e $f(n)=O(g(n))$. Para estas circunstâncias usa-se a notação-Θ.

Definição 2.3: *Notação-Θ*: $f(n)=\Theta(g(n))$ se existirem constantes positivas c_1, c_2 e n_0 de modo que $c_1\, g(n) \leq f(n) \leq c_2\, g(n)$, $\forall\ n \geq n_0$.

Se $f(n)=\Theta(g(n))$ então $g(n)$ é limitado superiormente e inferiormente por $f(n)$.

2.1.5 Análise da Complexidade do método da Divisão-e-Conquista

A Divisão-e-Conquista, conforme Toscani [TOS87], é um método de desenvolvimento de algoritmos por decomposições e recombinações que segue o paradigma do algoritmo fundamental: dado um problema, ele é decomposto em subproblemas menores, os quais são resolvidos separadamente pelo algoritmo e, as soluções parciais são combinadas para se obter a solução do problema original. Se o tamanho do subproblema é relativamente grande, com soluções não imediatas, o processo é aplicado novamente, para se obter subproblemas menores, até que os subproblemas menores possam ser resolvidos facilmente. A aplicação sucessiva de decomposições é expressa de forma natural por um algoritmo recursivo.

Um algoritmo é dito recursivo se contém uma chamada para si mesmo. Segundo Shackelford [SHA98], a recursão envolve custos de tempo e espaço, pois a implementação de rotinas recursivas é viabilizada por uma pilha, na qual são armazenados os dados utilizados cada vez que a rotina é chamada, até que seja concluído. Isto significa que todos os dados não globais ficam armazenados na pilha. A pilha é dividida em pedaços chamados blocos de localizações; cada chamada da rotina usa um destes espaços da pilha de tamanho que depende da rotina particular chamada.

O tempo requerido para uma chamada da rotina é proporcional ao tempo requerido

para avaliar o parâmetro atual e armazenar os ponteiros dos seus valores na pilha. O tempo de retorno não é superior ao da chamada.

Como a Divisão-e-Conquista utiliza rotinas recursivas na solução dos problemas, esta comporta-se de maneira semelhante aos algoritmos recursivos, pois, em ambos os casos, pode-se desenvolver uma relação de recorrência descrevendo importantes características do algoritmo.[LOR99]

Um algoritmo recursivo, assim como os de Divisão-e-Conquista, por conter em seu corpo uma chamada para si mesmo também terá, na equação de complexidade, uma chamada a si mesmo (função de complexidade), resultando numa equação de recorrência. Esta equação de recorrência é composta pela complexidade de uma chamada mais a complexidade resultante das chamadas recursivas, colocadas como uma ocorrência de uma função T para um tamanho de entrada menor que o original.

A função $T(n)$ indica o tempo de execução da rotina recursiva, o qual depende da função tamanho da entrada. Freqüentemente, só uma rotina é envolvida e $T(n)$ é escrito como $T(r(n))$; ou seja, um problema de tamanho n usa a solução de problemas menores de tamanho $r(n)$, chamado tamanho do subproblema. Utiliza-se a notação de McCarthy [BIR93] para expressar a equação de recorrência proveniente da análise de um algoritmo recursivo. Assim, tem-se a equação de recorrência geral:

$$T(n)=(n=0 \rightarrow b, f(n)+mT(r(n)))$$

onde $f(n)$ é a complexidade de uma chamada recursiva, m é o número de subproblemas em que o algoritmo foi decomposto ou o número de chamadas recursivas e $r(n)$ e o tamanho do subproblema.

Na análise dos algoritmos obtidos por Divisão-e-Conquista, Toscani [TOS88] os divide em três subcasos, que dependem do primeiro termo da equação de recorrência, $f(n)$, ter complexidade constante ($f(n)=b$), exponencial ou não polinomial ($f(n)=b^n$) e polinomial de grau maior que zero ($f(n)=bn^s$). Toscani [TOS88] calcula a complexidade destes três subcasos através de indução, variando o número e o tamanho dos subproblemas, obtendo os resultados em notação assintótica e elabora uma tabela que contém a ordem de complexidade de todos estes casos. Esta tabela, Tabela 2.1, serve de referência para compararmos com os resultados exatos obtidos por nós com o método das equações características, demonstrado no capítulo seguinte.

$f(n)$	$r(n)=n-\varepsilon$	$r(n)=n-1$	$r(n)=\frac{n}{c}$
Não polinomial	não polinomial		
Constante	$O(n)$ m=1 $O(m^p)$ m>1 $p=[\frac{n-1}{\epsilon}]$	$O(n)$ m=1 $O(m^{n-1})$ m>1	$O(\log_c n)$ m=1 $O(n^q)$ m>1 $q=[\log_c m]$
$O(n^k)$ k>0	$O(n^{k+1})$ m=1 $O(n^k m^p)$ m>1 $p=[\frac{n-1}{\epsilon}]$	$O(n^{k+1})$ m=1 $O(n^k m^{n-1})$ m>1	m>1 $O(n^k)$ $m \leq \frac{1}{2} c^k$ $O(n^k \log_c n)$ $m=c^k$ $O(n^{\log_c m})$ $m>c^k$

Tabela 2.1: Complexidade de algoritmos desenvolvidos por Divisão-e-Conquista.

Descreve-se os três subcasos a seguir:

- Constante

  Supõe-se que $f(n)$, a complexidade de uma chamada recursiva é constante, isto é, $f(n)=b$. Assim a função $T(n)$ é dada por:

$$T(n)=(n=0 \rightarrow b, b+mT(r(n))) \tag{2.1}$$

- Exponencial ou Não Polinomial

  Supõe-se que $f(n)$, a complexidade de uma chamada recursiva é exponencial, isto é, $f(n)=b^n$. Assim a função $T(n)$ é dada por:

$$T(n)=(n=0 \rightarrow b, b^n+mT(r(n))) \tag{2.2}$$

- Polinomial de grau maior que zero

  Supõe-se que $f(n)$, a complexidade de uma chamada recursiva é polinomial de grau s, isto é, $f(n)=bn^s$. Assim a função $T(n)$ é representada por:

$$T(n)=(n=0 \rightarrow b, bn^s+mT(r(n))) \tag{2.3}$$

De maneira geral, b é uma constante e é a complexidade da base da recursividade quando tem-se um elemento (*T(0) ou valor inicial*); $f(n)$ é a complexidade de uma chamada recursiva sem considerar a complexidade das chamadas recursivas do próximo nível, m é o número de subproblemas em que o problema original foi decomposto e $T(r(n))$ representa a chamada do algoritmo, agora de tamanho $r(n)$ (tamanho do subproblema). De outra forma: $f(n)$ representa a complexidade do algoritmo, exceto pelas chamadas recursivas, m o número de subproblemas e $T(r(n))$ a complexidade resultante da chamada recursiva para resolver um problema de tamanho $r(n)$. Utiliza-se a mesma variável b para representar a base da recursividade e o termo $f(n)$ quando este for constante pelo fato de tornar a resolução menos complicada. Se esta variável por diferente, a solução não ficaria tão exata. Pode-se igualar estas variáveis, b e $f(n)$ porque considera-se a análise do algoritmo no pior caso.

Neste trabalho, considera-se quatro casos para $r(n)$, ou seja, quatro tipos de tamanhos de subproblemas: $r(n)=(n-1)$, $r(n)=(n-\varepsilon)$ com $\varepsilon=2$ e $\varepsilon=3$, $r(n)=(n-1)+(n-2)$ e $r(n)=(n/c)$.

Considerando a relação entre n e $r(n)$ (tamanho do problema original e tamanho do subproblema) resolve-se para cada caso. Outro fator importante na complexidade é o número de subproblemas, nese caso a dicotomia se dá em $m=1$ e $m>1$.

2.2 Relações de Recorrência

Os algoritmos recursivos podem ser escritos como uma recorrência ou procedimento iterativo [SHA98], a partir do qual pode-se calcular o custo da resolução de um problema particular.

A construção da relação de recorrência, que descreve as características do desempenho de um algoritmo, é um passo significativo no processo de análise dos algoritmos, considerando que a própria recorrência traz uma grande quantidade de informações. Existem alguns algoritmos que não apresentam uma descrição simples; entretanto, grande parte dos algoritmos mais importantes podem ser expressos em uma formulação recursiva, e suas análises conduzem para recorrências, tanto para o caso médio como para o pior caso.

Nesta seção, verifica-se algumas características básicas de recorrências e o modo

como são classificadas. Verifica-se soluções para recorrências de primeira ordem, onde uma função avaliada em *n* é expressa em termos da função avaliada em *n-1*, recorrências lineares de ordem superior com coeficientes constantes, recorrências não-lineares, recorrências com coeficientes variáveis e recorrências de Divisão-e-Conquista.

As relações de recorrência são equivalentes a equações em diferenças; desta forma, as técnicas para resolver tais equações são relevantes ao nosso problema, já que técnicas similares são freqüentemente usadas para resolver recorrências.

Segundo Sedgewick [SED96], as recorrências são classificadas pela maneira em que os termos estão combinados, pela origem dos coeficientes envolvidos e pelo número e a origem dos termos utilizados. A Tabela 2.2 lista tipos de recorrências com os respectivos exemplos. Desta lista algumas recorrências são resolvidas neste trabalho, como as recorrências lineares de primeira e segunda ordem e a recorrência de Divisão-e-Conquista, pois, estes são os tipos de recorrências obtidos geralmente na análise da complexidade dos algoritmos de Divisão-e-Conquista.

Tipo de Recorrência	Exemplo
Primeira Ordem	
Linear	$a_n = n\, a_{n-1} - 1$
Não-linear	$a_n = 1/(1 + a_{n-1})$
Segunda Ordem	
Linear	$a_n = a_{n-1} + 2\, a_{n-2}$
Não-linear	$a_n = a_{n-1}\, a_{n-2} + \sqrt{a_{n-2}}$
Coeficientes Variáveis	$a_n = n\, a_{n-1} + (n-1)\, a_{n-2} + 1$
I-ésima Ordem	$a_n = f(a_{n-1}, a_{n-2}, \ldots, a_{n-i})$
"Full-History"	$a_n = n + a_{n-1} + a_{n-2} + \ldots + a_1$
Divisão-e-conquista	$a_n = a_{[n/2]} + a_{[n/2]} + n$

Tabela 2.2 Classificação das Recorrências

Como características básicas das recorrências, Sedgewick [SED96] considera o cálculo de valores, escala e deslocamento e linearidade:

a) Cálculo de valores: Uma recorrência destina-se a calcular uma certa quantidade de elementos solicitados pelo problema. Com base nesta quantidade, a recorrência calcula

valores iniciais em ordem crescente de índices e, a partir destes, os demais valores com índices maiores. Se o problema requer valores de índices muito grandes, recomenda-se implementar programas a fim de facilitar e agilizar o cálculo.

b) Escala e deslocamento: As recorrências dependem de seus valores iniciais; mudando a condição inicial na recorrência linear

$$a_n = f(a_{n-1}) \qquad \text{para } n>0 \text{ com } a_0 = 1 \qquad (2.4)$$

de $a_0 = 1$ para $a_0 = 2$, mudará o valor de a_n para todo n. Pode-se deslocar o valor inicial, como, por exemplo,

$$b_n = f(b_{n-1}) \qquad \text{para } n>1 \text{ com } b_0 = 1, \qquad (2.5)$$

com $b_n = a_{n-1}$. Quando modifica-se os valores iniciais diz-se que se está escalando a recorrência; quando move-se os valores iniciais diz-se que está deslocando a recorrência. Os valores iniciais são originados diretamente dos problemas, porém, às vezes, escala-se e usa-se o deslocamento para simplificar o caminho da solução.

c) Linearidade: Recorrências lineares que possuem mais de um valor inicial podem ser escaladas modificando os valores iniciais e combinando suas soluções.

Por exemplo, se $f(x,y)$ é uma função linear com $f(0,0)=0$, então a solução para a recorrência

$$a_n = f(a_{n-1}, a_{n-2}), \qquad \text{para } n>1 \qquad (2.6)$$

que possui condições iniciais $a_0=a_1=0$, é escalonada modificando a primeira condição inicial, a_0, e colocando a recorrência (2.6) em função de u, obtendo uma nova recorrência

$$u_n = f(u_{n-1}, u_{n-2}) \qquad \text{para } n>1$$

com condições iniciais $u_0 = 1$ e $u_1 = 0$; escalonando novamente, ou seja, modificando a segunda condição inicial da recorrência (2.6), a_1, e colocando a mesma recorrência em função de v, tem-se solução para uma outra nova recorrência

$$v_n = f(v_{n-1}, v_{n-2}) \qquad \text{para } n>1$$

com condições iniciais iguais a $v_0 = 1$ e $v_1 = 1$.

A condição $f(0,0)=0$ torna a recorrência homogênea: se existe uma constante em relação a f, os demais valores são calculados a partir dos valores iniciais.

A seguir, descreve-se, através de exemplos simples, os principais tipos de recorrências oriundas de algoritmos desenvolvidos por Divisão- eConquista.

2.2.1 Recorrência de primeira ordem

São recorrências simples, que podem, por exemplo, serem reduzidas a um produto. A recorrência

$$a_n = x_n\, a_{n-1}, \qquad \text{para } n>0 \text{ e } a_0 = 1 \qquad (2.7)$$

é equivalente a

$$a_n = \prod_{1 \le k \le n} x_k \ .$$

Deste modo, se $x_n=n$ então $a_n = n!$ e se $x_n=2$ então $a_n = 2^n$. Esta transformação é um exemplo de iteração: aplica-se a recorrência a si mesma com constantes e valores iniciais que estão à direita da equação e depois simplifica-se. A iteração também pode ser aplicada diretamente a um tipo de recorrência a que reduz-se diretamente a uma soma:

$$a_n = a_{n-1} + y_n \qquad \text{para } n>0 \text{ e } a_0 = 0$$

é equivalente a

$$a_n = \sum_{1 \le k \le n} y_k \ .$$

Deste modo, se $y_n=1$ então $a_n=n$ e se $y_n=n-1$ então $a_n = \frac{n(n-1)}{2}$.

TEOREMA 2.2: (Recorrência linear de primeira ordem) A recorrência

$$a_n = x_n\, a_{n-1} + y_n \qquad \text{para } n>0 \text{ com } a_0 = 0 \qquad (2.8)$$

possui como solução

$$a_n = y_n + \sum_{1 \le j \le n-1} y_j x_{j+1} x_{j+2} \ldots x_n$$

PROVA: Dividindo ambos os lados da solução de a_n , acima, por $x_n\, x_{n-1} \ldots x_1$ e iterando, tem-se:

$$a_n = y_n + \sum_{1 \le j \le n-1} y_j x_{j+1} x_{j+2} \ldots x_n \ .$$

O mesmo resultado pode ser obtido multiplicando ambos os lados por $x_{n+1}\, x_{n+2} \ldots$ e iterando.

O teorema 2.2 é uma caracterização completa da transformação das recorrências lineares de primeira ordem, com coeficientes constantes ou não constantes, para uma soma.

2.2.2 Recorrência não linear de primeira ordem

Quando uma recorrência consiste em uma função não linear relacionando a_n e a_{n-1}, surge uma variedade de situações das quais não se pode esperar um conjunto de soluções como no teorema 2.2. Neste caso, considera-se um número de casos que admitem soluções e apresentam três tipos de convergências: Simples, Quadrática e Lenta.

- *Convergência Simples*: Um dos motivos para o cálculo dos valores iniciais é que muitas recorrências, com uma aparência difícil, convergem simplesmente para uma constante. Por exemplo, considera-se a equação

$$a_n = \frac{1}{a_{n-1}+1} \qquad \text{para } n>0 \text{ e } a_0 = 1. \qquad (2.9)$$

Calculando os valores iniciais, deduz-se que a recorrência converge para uma constante:

n	a_n
1	0.50000000000
2	0.66666666666
3	0.60000000000
4	0.62500000000
5	0.61538461539
6	0.61904761905
7	0.61764705882
8	0.61818181812
9	0.61797752809

Cada iteração aumenta o número de dígitos significativos exatos avaliados, por um número constante de dígitos. Este processo é conhecido como convergência simples. Se a recorrência converge para uma constante, esta constante deve satisfazer a seguinte relação: $r = 1/(1+r)$, ou $1 - r - r^2 = 0$, a qual conduz para a solução $r = ((\sqrt{5}-1)/2) \approx 0.6180334$.

- *Convergência Quadrática e Método de Newton*: Este é um método iterativo conhecido para cálculo de raízes de funções, o qual pode ser visto como um processo de cálculo sobre uma solução aproximada para uma recorrência de primeira ordem. Por exemplo, o Método de Newton calcula a raiz quadrada de um número positivo β iterando a

equação

$$a_n = \frac{1}{2}\left(a_{n-1} + \frac{\beta}{a_{n-1}}\right) \qquad \text{para } n>0 \text{ com } a_0 = 1. \qquad (2.10)$$

Mudando as variáveis na recorrência, ou seja, fazendo $b_n = a_n - \alpha$, encontra-se a equação algébrica

$$b_n = \frac{1}{2}\frac{b_{n-1}^2 + \beta - alfa^2}{b_{n-1} + alfa} \quad .$$

Por exemplo, para calcular a raiz quadrada de 2, esta iteração retorna a seqüência:

n	a_n
1	1.50000000000
2	1.41666666667
3	1.41421568627
4	1.41421356237
5	1.41241356237

Em cada iteração, o número de dígitos significativos exatos dobra, isto é, ocorre uma convergência quadrática.

- *Convergência Lenta*: Considere a recorrência

$$a_n = a_{n-1}\,(1-a_{n-1}) \qquad \text{para } n>0 \text{ e } a_0 = \tfrac{1}{2}. \qquad (2.11)$$

Como os termos da recorrência diminuem e são positivos, pode-se ver que

$$\lim_{n \to \infty} a_n = 0.$$

Para calcularmos a velocidade da convergência, considera-se $1/a_n$ na recorrência acima,

$$\frac{1}{a_n} = \frac{1}{a_{n-1}}\left(\frac{1}{1-a_{n-1}}\right) \qquad (2.12)$$

$$\frac{1}{a_n} = \frac{1}{a_{n-1}}\left(1 + a_{n-1} + a_{n-1}^2 + \ldots\right)$$

$$\frac{1}{a_n} > \frac{1}{a_{n-1}} + 1.$$

2.2.3 Recorrência de ordem superior

Considera-se recorrência de ordem superior quando o lado direito da equação é uma combinação linear de a_{n-2}, a_{n-3},...a_{n-1} e os coeficientes envolvidos são constantes. Por exemplo, a recorrência

$$a_n = 3a_{n-1} - 2a_{n-2} \qquad \text{para } n>1 \text{ com } a_0=0 \text{ e } a_1=1 \qquad (2.13)$$

pode ser resolvida observando que $a_n-a_{n-1} = 2(a_{n-1}-a_{n-2})$. Iterando este produto obtém-se $a_n - a_{n-1} = 2^{n-1}$; iterando a soma para esta recorrência elementar, obtém-se a solução $a_n= 2^n-1$. Pode-se também resolver a recorrência (2.13) observando que $a_n-2a_{n-1} = a_{n-1}-2a_{n-2}$.

Similarmente, pode-se verificar que a solução para

$$a_n=5a_{n-1}-6a_{n-2} \qquad \text{para } n>1 \text{ com } a_0=0 \text{ e } a_1=1, \qquad (2.14)$$

é $a_n = 3^n-2^n$, resolvendo a recorrência elementar sobre a_n-3a_{n-1} ou a_n-2a_{n-1}.

TEOREMA 2.3: (Recorrência linear com coeficientes constantes) Todas as soluções para a recorrência

$$a_n= x_1 a_{n-1}+x_2 a_{n-2}+ \ldots +x_t a_{n-t} \qquad \text{para } n\geq t \qquad (2.15)$$

podem ser expressas como uma combinação linear (com coeficientes dependendo das condições iniciais a_0, a_1,..., a_{t-1}) dos termos na forma $n^j r^n$, onde r é uma raiz do polinômio característico $q(z)$,

$$q(z) \equiv z^t - x_1 z^{t-1} - x_2 z^{t-2} - \ldots\ldots -x_t \qquad (2.16)$$

e j é tal que $0\leq j <v$, se r tem multiplicidade v.

PROVA: Supõe-se que as soluções são do tipo $a_n = r^n$. Substituindo estas expressões para a_n na equação (2.15), obtém-se a equação

$$r^n = x_1 r^{n-1} + x_2 r^{n-2} + \ldots + x_t r^{n-t} \qquad \text{para } n\geq t$$

ou, de forma equivalente, $r^{n-t} q(r) =0$,

onde q é o polinômio descrito em (2.16). Isto é, r^n é uma solução para a recorrência, sendo r raiz do polinômio característico.

Supondo que r é raiz dupla de $q(z)$, prova-se que $a_n=nr^n$ é uma solução para a recorrência, assim como r^n. Substituindo nr^n na equação (2.15) tem-se

$$n r^n= x_1(n-1)r^{n-1} + x_2(n-2)r^{n-2}+\ldots + x_t(n-t)r^{n-t} \qquad \text{para } n\geq t$$

ou,

$$r^{n-1}((n-t)q(r)+rq'(r)) = 0.$$

Pois $q(r) = q'(r) = 0$ quando r é raiz dupla. Para raízes com multiplicidade maior que dois, trata-se de maneira similar. Conforme a multiplicidade das raízes do polinômio característico, obtém-se diferentes soluções para a recorrência, sendo o número de raízes o mesmo que a ordem t da recorrência. Além disso, as soluções das recorrências de ordem t são linearmente independentes. A solução geral da recorrência é expressa como uma combinação linear de termos de forma $n^i r^n$.

-Encontrando os coeficientes: Uma solução exata para qualquer recorrência linear pode ser desenvolvida pelo teorema 2.3 usando valores iniciais $a_0, a_1, \ldots, a_{t-1}$ para construir um sistema de equações onde, resolvendo este sistema, obtém-se as constantes da combinação linear da solução da recorrência. Como exemplo, considera-se a recorrência

$$a_n = 5a_{n-1}-6a_{n-2} \qquad \text{para } n \geq 2 \quad \text{com } a_0=0 \text{ e } a_1=1. \quad (2.17)$$

O polinômio característico formado de acordo com (2.16) é $r^2 - 5r + 6 = (r-3)(r-2)$, assim a solução geral de recorrência é

$$a_n = c_0\, 3^n + c_1\, 2^n,$$

pois cada solução é do tipo $a_n=r^n$, onde r é raiz da equação característica e a solução geral é combinação linear destas soluções.

Substituindo $n=0$ e $n=1$ na equação acima, obtém-se o sistema

$$a_0 = 0 = c_0 + c_1$$
$$a_1 = 1 = 3c_0 + 2c_1.$$

A solução do sistema de equações é $c_0 = 1$ e $c_1 = -1$, assim, a solução exata da equação de recorrência é:

$$a_n = 3^n - 2^n.$$

- Casos Especiais: Com base nos métodos para encontrar uma solução exata para qualquer recorrência linear, o processo torna explícito a maneira pelo qual o conjunto de soluções é determinado a partir das condições iniciais. Quando os coeficientes das equações de recorrência são iguais a zero e/ou algumas raízes tem o mesmo módulo, o resultado da solução desta equação pode apresentar-se constante ou de forma linear. Por

exemplo, seja a recorrência

$$a_n = 2a_{n-1} - a_{n-2} \qquad \text{para } n \geq 2 \quad \text{com } a_0=1 \text{ e } a_1=2. \quad (2.18)$$

O polinômio característico é $r^2 - 2r + 1 = (r-1)^2$ (com única raiz 1, de multiplicidade 2); a solução geral da equação de recorrência (2.18) é

$$a_n = c_0 1^n + c_1 n1^n.$$

Aplicando as condições iniciais na solução geral acima, obtém-se o sistema

$$a_0=1=c_0$$
$$a_1=2=c_0+c_1$$

cuja solução é $c_0=c_1=1$; assim, a solução exata da equação de recorrência (2.18) é

$$a_n = n+1.$$

Porém, se as condições iniciais fossem $a_0=a_1=1$, a solução da equação de recorrência (2.18) seria $a_n = 1$, constante.

- *Coeficientes não constantes:* Se os coeficientes não são constantes, necessita-se de técnicas mais avançadas, pois o teorema 2.3 não pode ser aplicado. Geralmente, são utilizadas Funções Geradoras [GRA88] ou Métodos de Aproximações, porém alguns problemas de ordem superior podem ser resolvidos com Soma de Fatores [LUE80]. Por exemplo, a recorrência

$$a_n = na_{n-1} + n(n-1)a_{n-2} \qquad \text{para } n>1 \text{ com } a_1=1 \text{ e } a_0=0 \quad (2.19)$$

pode ser resolvida dividindo ambos os lados da recorrência por $n!$

2.2.4 Recorrência de Divisão-e-Conquista

Existem alguns algoritmos, utilizados na solução de uma ampla variedade de problemas, que tem sido desenvolvidos aplicando o seguinte paradigma do algoritmo fundamental: "Divide o problema em dois subproblemas de mesmo tamanho, resolve-os recursivamente, então utiliza estas soluções para resolver o problema original". Por exemplo, o número de comparações realizado pelo algoritmo de ordenação "Mergesort" [SED96] é dado pela solução da recorrência

$$c_n = c_{\lceil n/2 \rceil} + c_{\lfloor n/2 \rfloor} + n \qquad \text{para } n>1 \text{ com } c_1=0, \qquad (2.20)$$

onde a notação "$\lceil x \rceil$" indica arredondamento para o número inteiro imediatamente maior que x e a notação "$\lfloor x \rfloor$" indica arredondamento para o número inteiro imediatamente menor

que x.

Esta recorrência, e outras similares a esta, surgem da análise de uma variedade de algoritmos com a mesma estrutura básica do "Mergesort". É possível determinar o crescimento assintótico das funções satisfazendo tais recorrências, porém é necessário tomar cuidado especial nos resultados exatos obtidos devido a uma simples razão: um problema de tamanho n não pode ser dividido em dois subproblemas de tamanhos iguais, se n é ímpar; nesse caso, o que se deve fazer é tomar o tamanho do problema diferente de um número ímpar. Para n suficientemente grande, este fato não é muito importante, porém para n pequeno é fundamental uma atenção maior, pois a estrutura recursiva verifica que muitos subproblemas pequenos são envolvidos.

A recorrência Divisão-e-Conquista apresenta-se como sendo uma das recorrências mais importantes na literatura sobre análise da complexidade.

A Tabela 2.3 apresenta recorrências que ocorrem comumente, e suas soluções aproximadas (ver [SED96]) . Na tabela, $a_{n/2}$ significa "$a_{\lfloor n/2 \rfloor}$ ou $a_{\lceil n/2 \rceil}$", $2a_{n/2}$ significa "$a_{\lceil n/2 \rceil} + a_{\lfloor n/2 \rfloor}$".

Recorrências	Soluções
$a_n = a_{n/2}+1$	$\log n + 0(1)$
$a_n = a_{n/2} + n$	$2n + 0(\log n)$
$a_n = a_{n/2} + n \log n$	$\theta(n \log n)$
$a_n = 2a_{n/2} + 1$	$\theta(n)$
$a_n = 2a_{n/2} + \log n$	$\theta(n)$
$a_n = 2a_{n/2} + n$	$n \log n + 0(n)$
$a_n = 2a_{n/2} + n \log n$	$\frac{1}{2} n \log n^2 + 0(n \log n)$
$a_n = 2a_{n/2} + n \log^{\delta-1} n$	$\delta^{-1} n \log^{\delta} n + 0(n \log^{\delta-1} n)$
$a_n = 2a_{n/2} + n^2$	$2n^2 + 0(n)$
$a_n = 3a_{n/2} + n$	$\theta(n^{\log 3})$
$a_n = 4a_{n/2} + n$	$\theta(n^2)$

Tabela 2.3 Recorrências de Divisão-e-Conquista e soluções.

Normalmente os resultados de aplicações, envolvendo recorrências, são avaliados no pior caso.

2.2.5 Recorrências gerais de Divisão-e-Conquista

Geralmente ocorre uma variedade de recorrências de Divisão e Conquista (conforme apresentada na seção anterior) cuja resolução depende do número e do tamanho dos subproblemas, dos termos de recorrência e do custo de recombiná-los para a solução. Normalmente, determina-se o crescimento assintótico das soluções destas recorrências.

Na busca de uma solução geral, inicia-se com a fórmula recursiva

$$a(x) = \alpha\, a(x/\beta) + f(x) \qquad \text{para } x>1 \text{ com } a(0)=0 \qquad (2.21)$$

que é definida para números reais positivos. Na realidade, esta fórmula corresponde a um algoritmo de Divisão-e-Conquista que divide o problema de tamanho x em α subproblemas de tamanho x/β recombinando-os com um custo $f(x)$. Neste caso, $a(x)$ é uma função definida para x, α e β são inteiros positivos com $\beta>1$.

Por exemplo, considere o caso em que $f(x)=x=\beta^r$ e onde assume-se que $N=\beta^n$ é um número inteiro. Neste caso, tem-se

$$a_{\beta^n}=\alpha\, a_{\beta^{n-1}}+\beta^n \qquad \text{para } n>0 \text{ com } a_1=0. \qquad (2.22)$$

Dividindo ambos os lados da equação (2.22) por α^n e iterando (aplicando teorema 2.2) obtém-se como solução

$$a_{\beta^n}=\alpha^n+\sum_{1\le j\le n}\left(\frac{\beta}{\alpha}\right)^j .$$

Fazendo uma análise da solução, considera-se três casos:

- Se $\alpha > \beta$, a soma converge para uma constante;
- Se $\alpha = \beta$, a solução é $\alpha^n+n=\beta^n+n$.
- Se $\alpha < \beta$, a soma é determinada pelos últimos termos, que são de ordem $0(\beta/\alpha)^n$.

Seja $\alpha^n = (\beta^{\log_\beta \alpha})^n = (\beta^n)^{\log_\beta \alpha}$, isso significa que a solução da recorrência é $0(N^{\log_\beta \alpha})$ quando $\alpha > \beta$, $0(N \log_\beta N)$ quando $\alpha = \beta$, e $0(N)$ quando $\alpha < \beta$ onde $(N=\beta^n)$. Apesar desta solução empregar-se somente para $N=\beta^n$, este exemplo apresenta uma estrutura para os demais casos gerais.

TEOREMA 2.4: (Funções de Divisão-e-Conquista) Se a função $a(x)$ satisfaz a recorrência

$$a(x)=\alpha a(x/\beta) + x \qquad \text{para } x>1, \quad \text{com } a(1)=0 \qquad (2.23)$$

então se $\alpha < \beta$ $\quad a(x) \sim \frac{\beta}{\beta - \alpha} x$

se $\alpha = \beta$ $\quad a(x) \sim x \log_{\beta} x$

se $\alpha > \beta$ $\quad a(x) \sim \frac{\alpha}{\alpha - \beta} \left(\frac{\beta}{\alpha} \right)^{\lfloor \log_{\beta} \alpha \rfloor} x^{\log_{\beta} \alpha}$.

PROVA: A ideia básica, a qual é aplicada a todas as recorrências de Divisão e Conquista, é iterar a recorrência até que as condições iniciais sejam conhecidas para os subproblemas. Neste caso, tem-se

$$a(x) = x + \alpha\, a(x/\beta) \tag{2.24}$$

$$a(x) = x + \alpha \frac{x}{\beta} + \alpha^2 a\left(\frac{x}{\beta^2} \right)$$

$$a(x) = x + \alpha \frac{x}{\beta} + \alpha^2 \frac{x}{\beta^2} + \alpha^3 a\left(\frac{x}{\beta^3} \right)$$

e assim por diante. Depois de $t = \lfloor \log_{\beta} x \rfloor$ iterações, o termo $a(x/\beta^t)$ pode ser substituído por 0 (zero) porque $a(1)=0$ e o processo de iteração termina. Este processo permite uma representação exata da solução:

$$a(x) = x\left(1 + \frac{\alpha}{\beta} + \ldots + \frac{\alpha^t}{\beta^t} \right) .$$

Analisando a solução acima, identifica-se três casos:

- Se $\alpha < \beta$ então a soma converge e $a(x) \sim x \sum_{j \geq 0} \left(\frac{\alpha}{\beta} \right)^j = \frac{\beta}{\beta - \alpha} x$.

- Se $\alpha = \beta$ então cada um dos termos da soma é igual a 1 e a solução é simplificada em $a(x) = x(\lfloor \log_{\beta} x \rfloor + 1) \sim x \log_{\beta} x$.

- Se $\alpha > \beta$ então o último termo predomina na soma, assim

$$a(x) = x\left(\frac{\alpha}{\beta} \right)^t \left(1 + \frac{\beta}{\alpha} + \ldots + \frac{\beta^t}{\alpha^t} \right)$$

$$a(x) \sim x \frac{\alpha}{\alpha - \beta} \left(\frac{\alpha}{\beta} \right)^t .$$

A expressão no terceiro caso pode ser isolada separando a parte inteira da fracionária e escrevendo $t = \lfloor \log_{\beta} x \rfloor = \log_{\beta} x - \{\log_{\beta} x\}$, onde $\{\log_{\beta} x\}$ é a parte fracionária de $\log_{\beta} x$, no que resulta

$$x\left(\frac{\alpha}{\beta}\right)^{t}=x\left(\frac{\alpha}{\beta}\right)^{\log_{\beta}x}\left(\frac{\alpha}{\beta}\right)^{-\{\log_{\beta}x\}}=x^{\log_{\beta}\alpha}\left(\frac{\beta}{\alpha}\right)^{\{\log_{\beta}x\}},$$

onde usa, o fato que $\alpha^{\log_{\beta}x}=x^{\log_{\beta}\alpha}$, o que completa a prova.

2.3 Métodos gerais de resolução de equações de recorrência

Como existem diversas maneiras de resolução de equações de recorrência, apresenta-se a seguir uma síntese desses métodos e dos autores que o apresentam.

Segundo Sedgewick [SED96], as recorrências não lineares ou recorrências com coeficientes variáveis podem ser resolvidas ou pode-se obter soluções aproximadas das mesmas através de uma variedade de aproximações diferentes. Em [SED96] são apresentados quatro métodos gerais de resolução de recorrências: *Mudança de variável*, o qual envolve simplificação de uma recorrência recalculando esta em termos de outras variáveis; *"Repertoire"*, trabalha em sentido contrário de uma recorrência para encontrar um espaço de solução; *"Bootstrapping"*, envolve o desenvolvimento de uma solução aproximada, continuando até que uma resposta mais exata seja obtida ou um não provável melhoramento; e *Perturbação*, o qual estuda efeitos de transformação de uma recorrência em uma similar, mais simples, para a qual a solução da equação transformada é conhecida. Os primeiros dois métodos às vezes resultam em soluções exatas; os dois últimos são tipicamente usados para obter soluções aproximadas.

Em Graham e Knuth [GRA89], encontram-se sete métodos de resolução de recorrências: *Supor uma solução e provar por indução*; *Perturbação e "Repertoire"*, conforme relatado em [SED96]; *Substituição da soma por integrais*, onde explora-se a relação entre somatório ($\sum$) e integral ($\int$), sendo que o somatório e a integração estão baseados em muitas idéias similares; *Expansão e Contração*, onde substitui a forma original por uma dupla soma aparentemente mais complicada; *Cálculo finito* e *Funções Geradoras*, as quais resolvem recorrências provenientes de algoritmos complexos, mais elaborados e de alto nível.

O Método *Prova por Indução Matemática* é citado por Manber [MAN89] e Toscani [TOS88]. Em [TOS86] e [TOS88] são analisados vários casos (os mais usuais na

computação) e elaborada uma tabela onde os resultados apresentam-se em termos de complexidade assintótica. Em [MAN89] adivinhar uma solução pode parecer como um método não científico, porém, funciona muito bem para uma ampla classe de relações de recorrência.

Cormen [COR90] apresenta três métodos para resolver recorrências, os quais apresentam solução com limites assintóticos "Θ"ou "0". No Método da *Substituição*, adivinha-se um limite à solução e usa-se a indução matemática para prová-lo, conforme citado em [MAN89] e [TOS88]; o Método da *Iteração* converte a recorrência em um somatório limitado para resolver a recorrência; e o Método *"Master"* dá limites para recorrências da forma $T(n) = aT(n/b) + f(n)$, onde $a \geq 1$, $b > 2$ e $f(n)$ é uma função fornecida.

Bentley [BEN80] descreve um método geral para resolver recorrências da forma $T(n) = kT(n/c) + f(n)$. O método é baseado em reescrever a recorrência num padrão "Template", o qual pode ser resolvido facilmente por indução matemática ou com a ajuda de uma pequena tabela apresentada em notação assintótica. Muito parecido com o Método da Iteração citado por Cormen, a diferença é que Bentley utiliza uma tabela para obter a notação assintótica da recorrência.

Lueker [LUE80] apresenta quatro métodos de resolução das recorrências através de exemplos que aparecem na análise de algoritmos. Os métodos *Soma de Fatores* e *Equações Características* (citados na seção 2.2.3, teorema 2.3) possibilitam resolver uma certa classe de equações lineares simples. O Método *Transformação de Imagem e Domínio* estende a classe de recorrências solucionáveis pelos dois métodos anteriores além do método das *Funções Geradoras* (já citado [GRA88]). Este último método é próprio para resolver recorrências mais complexas decorrentes de algoritmos mais elaborados.

Brassard [BRA95], aparentemente, partiu do trabalho de [LUE80], mas somente trabalha com exemplos simples, não generalizados, os quais não exigem utilização de ferramenta computacional. Apresenta quatro métodos: *Suposição Inteligente*, ou, como citado em [COR90], Método da Substituição; *Mudança de Variável*, já citado por [SED96]; *Transformação de domínio*, apresentado em [LUE80] e *Equações Características* considerado como a técnica principal de resolução de recorrência, já apresentado por Lueker. Na maioria dos exemplos os resultados apresentam-se na forma de notação assintótica.

Exemplifica-se alguns dos métodos citados acima:

- *Mudança de variável:* Método utilizado para originar soluções exatas de recorrências de ordem superior. Por exemplo, considera-se a recorrência não linear de segunda ordem

$$a_n=\sqrt{(a_{n-1}a_{n-2})} \qquad \text{para } n>1 \text{ com } a_0=1 \text{ e } a_1=2. \tag{2.25}$$

Aplica-se o logaritmo em ambos os lados da equação e realiza-se a mudança de variável $b_n=\log_2 a_n$; verifica-se então que b_n satisfaz

$$b_n=\frac{1}{2}\left(b_{n-1}+b_{n-2}\right) \qquad \text{para } n>1 \text{ com } b_0=0 \text{ e } b_1=1,$$

uma recorrência linear com coeficientes constantes.

-*"Repertoire"*: O método também resulta em soluções exatas, onde utilizam-se funções conhecidas para encontrar uma familia de soluções similares à procurada, as quais podem ser combinadas entre si para obter uma resposta. Este método aplica-se primeiramente a recorrências lineares, seguindo os seguintes passos:

- Relaxar a recorrência adicionando um termo funcional extra.
- Substituir funções conhecidas na recorrência para originar termos similares para a recorrência.
- Obter combinações lineares de tais termos para originar uma equação idêntica para a recorrência.

Como exemplo, considera-se a recorrência

$$a_n=(n-1)a_{n-1}-na_{n-2}+n-1 \qquad \text{para } n>1 \text{ com } a_0=a_1=1. \tag{2.26}$$

Substituindo o termo *n-1* por uma função *f(n)* no lado direito da equação (2.26), deseja-se resolver

$$a_n=(n-1)a_{n-1}-na_{n-2}+f(n) \qquad \text{para } n>1 \text{ e } a_0=a_1=1 \text{ com } f(n)=n-1. \tag{2.27}$$

Escolhe-se vários valores para a_n e observa-se o resultado em *f(n)* para obter um *"repertoire"* da recorrência, que pode ser resolvida (omitindo momentaneamente as condições iniciais). Por exemplo, considere a tabela

a_n	$f(n)=a_n-(n-1)a_{n-1}+na_{n-2}$
1	2
n	$n-1$
n^2	$n+1$

A primeira linha desta tabela, mostra que $a_n=1$ é uma solução com $f(n)=2$ (e condições iniciais $a_0=1$ e $a_1=1$); a segunda linha mostra que $a_n=n$ é uma solução com $f(n)=n-1$ (e condições iniciais $a_0=0$ e $a_1=1$); a terceira linha diz que $a_n=n^2$ é uma solução com $f(n)=n+1$ (e condições iniciais $a_0=0$ e $a_1=1$). Combinando estas soluções, também encontra-se uma segunda solução, ou seja, subtraindo a primeira linha da terceira da tabela acima, tem-se o resultado $a_n=n^2-1$, que é uma solução da equação de recorrência (2.26) com $f(n)=n-1$ (e condições iniciais $a_0=1$ e $a_1=0$). Assim, tem-se duas soluções (linearmente independentes) para $f(n)=n-1$. Combinando estas soluções obtém-se valores iniciais exatos, resultando em $a_n=n^2-n+1$.

O sucesso deste método depende da capacidade de encontrar um conjunto de soluções independentes e, também, das condições iniciais.

- *"Bootstrapping"*: Pode-se, às vezes, sugerir um valor aproximado de solução para uma recorrência. Em seguida, a própria recorrência pode ser usada para minimizar o erro sobre a solução estimada, a qual pode ser usada para obter uma solução estimada mais exata. Este método apresenta os seguintes passos:

  - Usa-se a recorrência para calcular valores numéricos.
  - Sugere-se uma forma aproximada de solução.
  - Substitui-se a solução aproximada dentro da recorrência.
  - Prova-se o limite compactado sobre a solução, baseado na solução sugerida e na substituição.

Como exemplo, aplica-se este método na Recorrência de Fibonacci:

$$a_n = a_{n-1} + a_{n-2} \qquad \text{para } n>1 \text{ com } a_0=0 \text{ e } a_1=1. \tag{2.28}$$

Primeiro, nota-se que a_n está aumentando, conseqüentemente, $a_{n-1}>a_{n-2}$ e $a_n>2a_{n-2}$. Iterando esta desigualdade obtém-se $a_n>2^{n/2}$. Uma outra combinação, $a_{n-2}<a_{n-1}$ implica que $a_n<2a_{n-1}$, ou (iterando) $a_n<2^n$. Desta maneira tem-se provado o crescimento exponencial dos

limites superior e inferior sobre a_n, então justifica-se a suposição da solução na forma $a_n \sim c_0 \alpha^n$, com $\sqrt{2} < \alpha < 2$. Da recorrência, conclui-se que α deve satisfazer $\alpha^2 - \alpha - 1 = 0$. Determinando o valor de α, pode-se aplicar o método *"bootstrapping"* e voltar para a recorrência. Assim, usando os valores iniciais, encontra-se os coeficientes apropriados.

- *Perturbação:* Método onde pode-se obter uma solução aproximada de uma recorrência resolvendo outra recorrência relacionada. O método é uma aproximação geral para resolver recorrências, no qual primeiro estuda-se as recorrências simplificadas obtidas da extração das partes dominantes, resolve-se a recorrência simplificada, e então compara-se soluções da recorrência original com as soluções da recorrência simplificada. Esta técnica é similar à da classe dos métodos familiares na análise numérica. O método apresenta os seguintes passos:

  - Modifica-se a recorrência original para encontrar uma recorrência similar.
  - Muda-se variáveis para tirar os limites conhecidos da equação de recorrência e transforma-a em uma nova recorrência.
  - Limita-se ou resolve-se o termo desconhecido "erro".

Como exemplo, considera-se a recorrência

$$a_{n+1} = 2a_n + \frac{a_{n-1}}{n^2} \qquad \text{para } n>1 \text{ com } a_0=1 \text{ e } a_1=2. \qquad (2.29)$$

Pode-se assumir que o último termo, devido ao coeficiente $1/n^2$, traz uma pequena contribuição para a recorrência, assim

$$a_{n+1} \approx 2a_n.$$

Deste modo, é intuído um crescimento da forma $a_n \approx 2^n$. Tornando a solução exata, considera-se a recorrência

$$b_{n+1} = 2b_n \qquad \text{para } n>0 \text{ com } b_0=1 \qquad (2.30)$$

(com $b_n = 2^n$) e comparando as duas recorrências, obtém-se a razão

$$\rho_n = \frac{a_n}{b_n} = \frac{a_n}{2^n} \quad .$$

Usando as recorrências ρ_n e a_n obtém-se

$$\rho_{n+1} = \rho_n + \frac{1}{4n^2}\rho_{n-1} \qquad \text{para } n>0 \text{ com } \rho_0=1.$$

Observa-se que ρ_n está crescendo. Para provar que ρ tende a uma constante, nota-se que

$$\rho_{n+1} \leq \rho_n \left(1+\frac{1}{4n^2}\right) \qquad \text{para } n \geq 1$$

assim,

$$\rho_{n+1} \leq \prod_{k=1}^{n} \left(1+\frac{1}{4k^2}\right) \quad .$$

Porém, o produto infinito correspondente ao lado direito converge monotonicamente para

$$\alpha_0 = \prod_{k=1}^{\infty} \left(1+\frac{1}{4k^2}\right) = 1.46505...$$

Assim, ρ_n está limitado por α_0 e, como está em crescimento, deve convergir para uma constante. Desta forma, tem-se provado que $a_n \sim \alpha 2^n$, para alguma constante $\alpha < 1.46505...$

- *Substituição da Soma por Integrais*: Às vezes é mais fácil utilizar o Cálculo em vez da Matemática Discreta, por ser mais familiar a integral ($\int$) do que o somatório (Σ). Este método tem o objetivo de tornar confortável o uso da integral no lugar do somatório, incentivando a explorar a relação entre Σ e $\int$.

No Cálculo, uma integral pode ser considerada como a área abaixo de uma curva, e pode-se aproximar esta área adicionando áreas alongadas de retângulos finos que tocam a curva. Pode-se, também, calcular a área de outra maneira: por exemplo, chama-se a_n a soma das áreas de retângulos de tamanhos 1x1, 1x4, ..., $1xn^2$, que é aproximadamente igual à área abaixo da curva $f(x) = x^2$ entre zero e n. A área abaixo desta curva é $\int_0^n x^2\,dx = \frac{n^3}{3}$; portanto sabe-se que a_n é aproximadamente $\frac{1}{3}n^3$.

Outra maneira de usar esse fato é examinar o erro na aproximação, $E_n = a_n - \frac{1}{3}n^3$. Sabe-se que a_n deve satisfazer a recorrência $a_n = a_{n-1} + n^2$, então verifica-se que E_n satisfaz a simples recorrência

$$E_n = a_n - \frac{1}{3}n^3 = a_{n-1} + n^2 - \frac{1}{3}n^3 = E_{n-1} + \frac{1}{3}(n-1)^3 + n^2 - \frac{1}{3}n^3$$

$$E_n = E_{n-1} + n - \frac{1}{3} \quad . \tag{2.31}$$

Outra maneira para seguir a aproximação integral é encontrar a fórmula para E_n, somando as áreas dos erros. Tem-se

$$a_n - \int_0^n x^2\,dx = \sum_{k=1}^{n} \left(k^2 - \int_{k-1}^{k} x^2\,dx \right)$$

$$a_n - \int_0^n x^2\,dx = \sum_{k=1}^{n} \left(k^2 - \frac{k^3 - (k-1)^3}{3} \right) = \sum_{k=1}^{n} \left(k - \frac{1}{3} \right) \quad .$$

De modo idêntico, poderia-se encontrar E_n e então a_n.

- *Expansão e Contração*: Uma outra maneira de descobrir a forma aproximada para a_n é substituir a soma original por uma dupla soma aparentemente mais complicada, a qual simplifica-se utilizando a propriedade dos somatórios:

$$a_n = \sum_{1 \le k \le n} k^2 = \sum_{1 \le j \le k \le n} k = \sum_{1 \le j \le n} \sum_{j \le k \le n} k \tag{2.32}$$

$$a_n = \sum_{1 \le j \le n} \left(\frac{j+n}{2} \right)(n - j + 1) = \frac{1}{2} \sum_{1 \le j \le n} \left(n(n+1) + j - j^2 \right)$$

$$a_n = \frac{1}{2} n^2 (n+1) + \frac{1}{4} n(n+1) - \frac{1}{2} a_n = \frac{1}{2} n \left(n + \frac{1}{2} \right)(n+1) - \frac{1}{2} a_n \quad .$$

O último passo é parecido com o último passo do Método da Perturbação, pois consegue-se uma equação com uma certa quantidade desconhecida de termos em ambos os lados.

No processo de transformar uma soma simples em uma dupla soma parece que se está dificultando o trabalho para a resolução da recorrência, o que na realidade não está, devido esta dupla soma ser mais fácil para trabalhar. Não se pode esperar resolver todos os problemas simplificando continuamente.

- *Suposição de solução e prova por indução, ou Método da Substituição:* A principal razão pela qual a suposição é proveitosa é que, provando que um certo limite é válido, então a solução suposta está correta. Como exemplo, considera-se a seguinte recorrência, a qual está definida somente para n potência de 2:

$$T(n) \le 2T(n) + 2n - 1, \quad T(2) = 1. \tag{2.33}$$

Esta recorrência é melhor escrita como uma desigualdade do que uma igualdade, pois a solução procurada não é de forma exata, e sim aproximada a um limite. O que é coerente com o objetivo de encontrar somente o limite superior (notação assintótica), e com o fato que o lado direito representa o pior caso.

Deseja-se encontrar uma função $f(n)$ de modo que $T(n)=0(f(n))$; porém, além disso, deseja-se que $f(n)$ não seja diferente do atual $T(n)$.

Dada uma suposição para $f(n)$, dita $f(n)=n^2$, prova-se por indução sobre n que $T(n)=0(f(n))$. Primeiro, verifica-se a base da indução, neste caso $T(2)=1 \leq f(2)=4$. Prova-se, então, que $T(n) \leq f(n)$ implica que $T(2n) \leq f(2n)$. Precisa-se provar então que

$$T(n) \leq n^2 \quad \Rightarrow T(2n) \leq (2n)^2. \tag{2.34}$$

Segue a prova como:

$T(2n) \leq 2T(n)+2n-1$,

$\leq 2n^2+2n-1$, *(por hipótese indução)*

$< (2n)^2$,

que é exatamente o que se queria provar. Deste modo, $T(n)=0(n^2)$. No último passo da prova, $2n^2+2n-1$ foi substituído por $4n^2$ para n suficientemente grande. Porém existe uma diferença essencial (de $2n^2$) entre estas duas expressões, o qual dá uma sugestão que n^2 pode ser uma estimativa para $T(n)$.

O método pode ser usado para estabelecer os limites superior e inferior sobre a recorrência.

- *Método da Iteração*: O método de iterar sobre uma recorrência não requer uma adivinhação da resposta e necessita de mais álgebra do que no método da substituição. A idéia é expandir a recorrência e expressá-la como uma soma de termos dependentes somente de n e das condições iniciais. Técnicas de avaliação e somatório podem ser usadas para provar limites na solução.

Como exemplo, considera-se a recorrência

$$T(n)=3T(\lfloor n/4 \rfloor)+n. \tag{2.35}$$

Itera-se como segue:

$$T(n)=n+3T(\lfloor n/4 \rfloor)$$
$$=n+3(\lfloor n/4 \rfloor+3T(\lfloor n/16 \rfloor))$$

$$=n+3(\lfloor n/4\rfloor+3(\lfloor n/16\rfloor)+3T(\lfloor n/64\rfloor)))$$

$$=n+3\lfloor n/4\rfloor+9\lfloor n/16\rfloor+27T(\lfloor n/64\rfloor),$$

onde $\lfloor\lfloor n/4\rfloor/4\rfloor=\lfloor n/16\rfloor$ e $\lfloor\lfloor n/16\rfloor/4\rfloor=\lfloor n/64\rfloor$. O i-ésimo termo da série é $3^i\lfloor n/4^i\rfloor$. A iteração reduz-se a *I* quando $\lfloor n/4^i\rfloor=1$ ou, equivalentemente, quando *i* excede $\log_4 n$. Continuando a iteração deste ponto e usando o limite $\lfloor n/4^i\rfloor\le n/4^i$, observa-se que o somatório contém uma série geométrica decrescente:

$$T(n)\le n+\frac{3n}{4}+\frac{9n}{16}+\frac{27n}{64}+\ldots+3^{\log_4 n}\Theta(1) \qquad (2.36)$$

$$T(n)\le n\sum_{i=0}^{\infty}\left(\frac{3}{4}\right)^i+\Theta\left(n^{\log_4 3}\right)$$

$$T(n)=4n+0(n)=0(n)\ .$$

Onde se usou o fato que $3^{\log_4 3}=n^{\log_4 3}$ e sabendo que $\log_4 3<1$ conclui-se que $\Theta(n^{\log_4 3})=0(n)$.

- *O "Template"*: O método resolve recorrências do tipo Divisão-e-Conquista, as quais são gerais na aplicabilidade e na facilidade de aplicação. Este método é sugerido como técnica de sala de aula e como ferramenta para prática de desenvolvimento de algoritmos. Como exemplo, tem-se a recorrência da forma

$$T(n)=kT(n/2)+f(n).\quad T(1)\ \text{é conhecido} \qquad (2.37)$$

Esta recorrência é definida somente para *n* potência de dois. Para resolver a recorrência reescreve-se na forma "Template"

$$T(n)=2^pT(n/2)+n^pg(n), \qquad (2.38)$$

onde $p=\log_2 k$ e $g(n)=f(n)/n^p$. Esta recorrência é solucionada usando a técnica de iteração e cancelamento:

$$T(n)=n^p[g(n)+g(n/2)+g(n/4)+\ldots+g(4)+g(2)+T(1)]\ .$$

Nota-se que esta solução é expressa em termos de *g*, pois $n^p = 2^p(n/2)^p$, o que é a generalização da propriedade de cancelamento. Abreviando a solução acima tem-se

$$T(n)=n^p[T(1)+\tilde{g}(n)]\ , \qquad (2.39)$$

onde $\tilde{g}$ é definido como

$$\tilde{g}(n)=\sum_{1\le i\le \log_2 n} g(2^i)\ .$$

A Indução Matemática, com n potência de dois, pode ser usada para mostrar que a equação (2.39) é realmente a solução única para a recorrência (2.38). A parte complicada desse processo é determinar a soma implícita na função $\tilde{g}$ da equação (2.39), o que pode ser feito com o auxílio da tabela que descreve $\tilde{g}$ em relação a g.

$g(n)$	$\tilde{g}(n)$
$0(n^q)$ $q<0$	$\Theta(1)$
$\log_2^j n$ $j\geq 0$	$(\log_2^{j+1} n)/(j+1)+\Theta(\log_2^j n)$
$\Omega(n^q)$ $q>0$	$\Theta(g(n))$

Tabela 2.4: Ordem de complexidade da função $g(n)$.

Cada uma dessas relações pode ser verificada expandindo a soma que define $\tilde{g}$. Na terceira linha da tabela 2.4 usa-se Ω limitando o seguinte: escreve-se $g(n)=\Omega(n^q)$ se existe n_0 e $a>0$ tal que $g(cn)\geq ac^q g(n)$ para todo $c>1$ e $n>n_0$.

- *Método "Master"*: O método provém do método "cookbook" para resolver recorrências da forma

$$T(n)=aT(n/b)+f(n), \tag{2.40}$$

onde $a\geq 1$ e $b>1$ são constantes e $f(n)$ é uma função assintoticamente positiva. O método "master" requer a memorização dos três casos citados por Cormen, pois a solução de muitas recorrências pode ser determinada facilmente. Se n/b não é um inteiro, a recorrência não está bem definida, desse modo, substitui-se $T(n/b)$ do termo a por outro $T(\lceil n/b \rceil)$ ou $T(\lfloor n/b \rfloor)$, o que não afeta o procedimento assintótico da recorrência.

O método "master" depende do teorema que segue:

TEOREMA 2.5: (Teorema "Master"): Sejam $a\geq 1$ e $b>1$ constantes e $f(n)$ uma função. $T(n)$ é definido para n inteiro positivo, na recorrência

$$T(n)=aT(n/b)+f(n), \tag{2.41}$$

onde interpreta-se n/b como sendo $\lfloor n/b \rfloor$ ou $\lceil n/b \rceil$. Logo $T(n)$ pode ser limitado assintoticamente como segue:

1. Se $f(n)=O(n^{\log_b a-\epsilon})$ para alguma constante $\varepsilon>0$, então $T(n)=\Theta(n^{\log_b a})$.
2. Se $f(n)=\Theta(n^{\log_b a})$ então $T(n)=\Theta(n^{\log_b a}\log n)$.
3. Se $f(n)=\Omega(n^{\log_b a+\epsilon})$ para alguma constante $\varepsilon>0$, e se $af(n/b)\leq cf(n)$ para $c>1$ e n suficientemente grande, então $T(n)=\Theta(f(n))$.

Pode-se verificar a prova deste teorema em [COR90].

Para usar o método "Master", simplesmente verifica-se qual o caso em que se aplica o Teorema "Master" e escreve-se a resposta.

Como exemplo, considera-se a recorrência

$$T(n)= 9T(n/3)+n\,. \tag{2.42}$$

Para esta recorrência, tem-se $a=9$, $b=3$, $f(n)=n$, e deste modo $n^{\log_b a}=n^{\log_3 9}=\Theta(n^2)$. Sendo $f(n)=0(n^{\log_3 9-\epsilon})$, onde $\varepsilon=1$, pode-se aplicar o caso 1 do Teorema "Master" e concluir que a solução é $T(n)=\Theta(n^2)$.

- *Soma de Fatores*: Este método resolve recorrências lineares provenientes de problemas, como por exemplo, como determinar o número de nodos, a_n, de uma árvore binária perfeita de altura n. Considere a recorrência

$$a_n = a_{n-1} + 2^n \qquad \text{para } n\geq 1 \tag{2.43}$$

com $a_0=1$. Usando a soma de fatores, reescreve-se a recorrência como

$$a_n - a_{n-1} = 2^n\,. \tag{2.44}$$

Variando n de 1 a m e somando,, tem-se no lado esquerdo da equação (2.44) $a_m - a_0$ e no lado direito $\sum_{n=1}^{m} 2^n = 2^{m+1}-2$. Assim obtém-se

$$a_m = a_0 + 2^{m+1} - 2\,,$$

como solução da equação de recorrência (2.43). Existem outras maneiras de encontrar a solução desta recorrência, utilizando o método Soma de Fatores (ver [LUE80]).

- *Transformação de Imagem e Domínio*: Uma seqüência pode ser interpretada como uma função dos inteiros nos reais. Uma transformação nos valores desta seqüência chama-se uma transformação de imagem e uma transformação sobre os índices da seqüência é uma transformação de domínio.

Como exemplo de transformação de imagem, considere a recorrência

$$a_n = 3a^2_{n-1}, \quad \text{para } n\geq 1, \text{ com } a_0=1. \tag{2.45}$$

Seja $b_n=\log_2 a_n$. Pode-se reescrever a recorrência como

$$b_n = \log_2(3a^2_{n-1}) = 2\log_2 a_{n-1} + \log_2 3$$

$$b_n = 2b_{n-1}+log_2 3, \qquad \text{com } b_0 = 0$$

a qual pode ser resolvida por Soma de Fatores ou Equações Características. Tem-se como resultado: $b_n-2b_{n-1} = log_2 3$. Com isso obtém-se $b_n=(2^n)log_2 3 - log_2 3=(2^n -1)log_2 3$. Pela propriedade de logarítmos: $b^{\log_b m}=m$, a solução da recorrência é da forma

$$a_n=2^{(2^n-1)\log_2 3}=3^{(2^n-1)} .$$

Como exemplo de transformação de domínio, considere a recorrência de complexidade do "Mergesort"[LUE80]

$$T(n)=2T(n/2)+n-1 \qquad \text{para } n\geq 2 \qquad (2.46)$$
$$T(1)=0.$$

Reescreve-se a recorrência como

$$a_n = 2a_{n/2} +n -1.$$

E, considerando que $n=2^k$ e $a_k=T(n)=T(2^k)$, pode-se escrever

$$a_k=2a_{k-1} +2^k -1, \text{ para } k\geq 1, \text{ com } a_0=0. \qquad (2.47)$$

Neste ponto, pode-se resolver esta recorrência utilizando a Soma de Fatores ou Equações Características. Assim, a solução encontrada é

$$a_k = (k-1)2^k +1.$$

Sendo $a_k=T(2^k)$, $log_2 n=k$, tem-se $T(2^k)=(k-1)2^k +1$, de onde,

$$T(n)= (log_2 n -1)n +1.$$

Este método é similar ao método Mudança de Variável citado em [SED96] e [BRA95].

Finaliza-se as apresentações dos diversos métodos de resolução de recorrências, com o método das Funções Geradoras.

- *Funções Geradoras*: Na análise de certos algoritmos, surgem recorrências muito complexas para serem resolvidas; nestes casos, utilizam-se Funções Geradoras. Estas funções ajudam a resolver recorrências de algoritmos de alto nível, como Transformadas de Laplace e Transformadas de Fourier, permitindo-nos detalhar estruturas descritas pelas recorrências em muitas aplicações. As Funções Geradoras transformam um problema de um domínio em outro onde o problema é mais facilmente resolvido.

A Função Geradora para uma seqüência é uma série infinita que possui como coeficientes os termos desta seqüência. Ou seja, se a seqüência é $\{a_n\}$, a função geradora

correspondente é

$$A(z)=\sum_{n=0}^{\infty} a_n z^n \quad . \tag{2.48}$$

Como exemplo, considera-se a recorrência

$$a_n = 2\, a_{n-1} + 1, \qquad \text{para } n \geq 1 \quad \text{com } a_0 = 1 \tag{2.49}$$

Para a recorrência (2.49), esta soma pode ser reescrita na seguinte forma:

$$A(z)=1+\sum_{n=1}^{\infty}\left(2a_{n-1}+1\right)z^n=1+z\sum_{n=1}^{\infty}2a_{n-1}z^{n-1}+\sum_{n=1}^{\infty}z^n$$

$$A(z)=1+2zA(z)+\left(\frac{1}{1-z}-1\right) \quad .$$

Agora, a equação de recorrência passa a ser resolvida por A. Este novo problema é único, onde tem-se

$$A(z)-2zA(z)=1+\left(\frac{1}{1-z}-1\right)$$

$$A(z)(1-2z)=\left(\frac{1}{1-z}\right)$$

$$A(z)=\frac{1}{(1-z)(1-2z)} \quad .$$

Deve-se retornar para o problema de domínio original, contudo, para resolver $A(z)$ usa-se, neste caso, frações parciais:

$$\frac{1}{(1-z)(1-2z)}=\frac{x}{1-z}+\frac{y}{1-2z}$$

$$1=x(1-2z)+y(1-z)$$

assim, $x=-1$ e $y=2$, de onde

$$A(z)=\frac{-1}{1-z}+\frac{2}{1-2z} \quad .$$

Para obter-se a solução da equação de recorrência (2.49), em termos de a_n, necessita-se de duas tabelas de definições de Funções Geradoras para seqüências e operações sobre as Funções Geradoras. Considera-se, para resolução do exemplo acima, apenas uma linha de cada tabela. Em [LUE80, SED96] encontra-se as tabelas completas.

Seqüência	Funções Geradoras
$1, c, c^2,...$	$1+ cz+(cz)^2+...=1/(1-cz)$

Tabela 2.5: Funções Geradoras para algumas seqüências.

Funções Geradoras	Fórmula para i-ésimo elementos da seqüência
A(z)/1-z	$\sum_{j=0}^{i} a_j$

Tabela 2.6: Operações sobre Funções Geradoras.

Usando as tabelas 2.5 e 2.6, obtém-se como solução da equação de recorrência (2.49):

$$a_n=2^{n+1}-1.$$

3 EQUAÇÕES CARACTERÍSTICAS PARA A COMPLEXIDADE

A complexidade de um algoritmo recursivo é expressa por uma equação de recorrência, apesar de apresentar-se diferentemente de um formato algébrico. Ou seja, inicialmente tem-se, por exemplo, a recorrência construída a partir da análise de um algoritmo recursivo: $T(n) = (n = 0 \rightarrow b, f(n) + mT(n-1))$. Ao passo que o formato algébrico correspondente: $x_n - mx_{n-1} = f(n)$. Neste capítulo a equação de complexidade será colocada na formulação algébrica e resolvida através de equações características.

As equações características são comumente usadas para resolver equações em diferenças, escritas na forma de equações de recorrência. Podem ser homogêneas e não-homogêneas.

Uma recorrência linear com coeficientes constantes é assim expressa

$$a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} + \ldots + a_k x_{n-k} = g(n) \qquad (3.1)$$

É chamada linear porque não ocorre operações de outras variáveis com a variável x. Se $g(n)$ é igual a zero, a equação é dita homogênea e tem a forma

$$a_0 x_n + a_1 x_{n-1} + a_2 x_{n-2} + \ldots + a_k x_{n-k} = 0 \qquad (3.2)$$

À forma homogênea corresponde uma equação característica, cuja solução resolve a recorrência. A equação característica associada à recorrência homogênea linear, assume o formato algébrico

$$a_o r^n + a_1 r^{n-1} + a_2 r^{n-2} + \ldots + a_k r^{n-k} = 0 \qquad (3.3)$$

onde cada a_i é constante.[GOL58]

O polinômio correspondente à equação (3.3) é chamado Polinômio Característico e suas raízes chamadas Raízes Características. As raízes definem a solução da recorrência homogênea.

Conforme [GOL58, LUE80, BRA95] a equação (3.2) tem solução geral do tipo $x_n = r^n$ onde r é raiz da equação característica (3.3). Se o polinômio característico possui mais de uma raiz, a solução geral tem a forma de uma combinação linear das raízes, e se as raízes possuírem multiplicidade maior que 1 aplica-se o Teorema 2.3 citado no capítulo

anterior, apresentado por [SED96].

O Teorema Fundamental da álgebra diz que qualquer polinômio de grau k possui exatamente k raízes (não necessariamente distintas). Este pode ser fatorado como um produto do tipo

$$g(x)=\prod_{i=1}^{k}(x-r_i) \tag{3.4}$$

onde r_i são as suas raízes e podem ser números complexos.

Como qualquer combinação linear das raízes é também uma solução para equação característica, pode-se concluir que

$$x_n=\sum_{i=1}^{k} c_i r_i^n \tag{3.5}$$

com constantes $c_1, c_2,...,c_k$, é a solução da recorrência.

A maioria dos problemas resolvidos recursivamente, apresentam equações de recorrência de forma não homogênea, precisando, inicialmente, serem homogeneizadas para então poder obter-se uma equação característica.

Lueker [LUE80] e Brassard [BRA95] usaram equações características para resolver equações de recorrência. Nestes trabalhos são apresentados exemplos numéricos e suas soluções, mas tais exemplos não correspondem a problemas específicos.

No presente trabalho calcula-se a complexidade exata para quatro tipos de algoritmos recursivos conforme o tamanho de subproblemas, apresentados em ordem crescente de generalidade, (n-1), (n-ε) com ε =2 e ε = 3, (n-1)+(n-2) e (n/2). Conforme citado na seção 2.1. 5, a equação de recorrência apresenta o seguinte formato geral:

$$T(n)=(n=0\rightarrow b,\ f(n)+mT(r(n))),$$

onde $f(n)$ representa a complexidade de uma chamada do algoritmo recursivo, m é o número de subproblemas em que o problema foi decomposto e $r(n)$ é o tamanho dos subproblemas.

O algoritmo decompõe o problema em m subproblemas menores independentes e do mesmo tipo, resolve esses problemas recursivamente (decompondo-o até atingirem um tamanho tão pequeno que os problemas tenham solução conhecida). Após atingir a base de recursão, os problemas são combinados passo a passo até voltar ao problema original. Esse processo define uma árvore de execução (de aridade m) que é percorrida de cima para

baixo em profundidade (decompondo) retornando de baixo para cima (combinando) da esquerda para direita). [TOS88]

A resolução das recorrências é generalizada em função do número de subproblemas e do tamanho dos subproblemas. Para cada tamanho de subproblema ocorrem três possibilidades para *f(n)* (primeiro termo da recorrência): constante, exponencial e polinomial com grau maior que zero. Para cada subproblema elabora-se uma tabela contendo a complexidade exata dos casos gerais e particulares, calculados com a utilização do software de programação simbólica Maple, conforme anexos A-1 e A-2.

3.1 Subproblema de tamanho r(n)=(n-1)

Neste caso apresentam-se recorrências do tipo linear de primeira ordem, conforme a classificação de [SED96]. No cálculo da complexidade exata obtém-se o polinômio característico completo associado à recorrência. Nos casos em que $f(n)=b$ e $f(n)=b^n$ (constante e exponencial respectivamente), o polinômio característico possui um grau a mais em relação à equação de recorrência original. No caso em que *f(n)* é um polinômio com grau maior do que zero, $f(n)=bn^s$, o polinômio característico decorrente do processo de homogeneização, terá seu grau aumentado em duas unidades, conforme o grau do polinômio da equação de recorrência, ou seja, se $s=2$ (ordem do polinômio da equação de recorrência) o polinômio característico será de quarta ordem.

Neste tipo de subproblema não existem raízes complexas devido ao polinômio característico apresentar-se sempre completo. Necessita-se somente de uma condição inicial e a equação de complexidade exata é formada a partir do número de subproblemas *m*, do formato *f(n)* da equação de recorrência do algoritmo e da condição inicial.

3.1.1 Desenvolvimento do método para *f(n)* constante

O algoritmo resolve um problema de tamanho *n* decompondo-o sucessivamente em *m* subproblemas de tamanho *n-1*. A complexidade do algoritmo, exceto as chamadas recursivas, é constante e a base da recursividade é *b* quando *n=0*. Isto é,

$$T(n)=(n=0\rightarrow b,\ b+mT(n-1)) \quad . \tag{3.6}$$

Primeiramente, coloca-se a equação de recorrência no formato algébrico, conforme

a equação (3.1), substituindo $T(n)$ por x_n e, ao mesmo tempo, define-se a base da recursividade como condição inicial:

$$x_n - mx_{n-1} = b \qquad (3.7)$$
$$x_0 = b.$$

Resolve-se a equação (3.7) para $m>1$ e $m \neq b$, tornando-a homogênea da seguinte maneira: substitui-se n por $n-1$ na recorrência (3.7) obtendo uma nova recorrência

$$x_{n-1} - mx_{n-2} = b \quad . \qquad (3.8)$$

Em seguida, subtrai-se a equação (3.8) da equação (3.7) obtendo a equação homogênea

$$x_n - (m+1)x_{n-1} + mx_{n-2} = 0 \quad . \qquad (3.9)$$

Assim, conforme (3.3) a equação característica associada à equação de recorrência (3.9) é:

$$r^n - (m+1)r^{n-1} + mr^{n-2} = 0 \quad ou \quad r^{n-2}(r^2 - (m+1)r + m) = 0 \quad .$$

Resolvendo este polinômio característico ou equação característica obtém-se as raízes, não nulas, $r_1=m$ e $r_2=1$. Assim, a solução geral da equação (3.7), conforme a solução geral da equação (3.5), $x_n=r^n$, é da forma

$$x_n = c_1 m^n + c_2 \quad . \qquad (3.10)$$

Como a partir da resolução da equação característica obteve-se duas raízes, a solução geral tem o formato de combinação linear dessas raízes. Sendo assim, é preciso calcular c_1 e c_2, o que pode ser feito substituindo $n=0$ e $n=1$ na equação (3.10). Desta forma obtém-se um sistema de equações que possui como termos independentes a condição inicial x_0 que é conhecida e x_1 que deve-se calcular substituindo $n=1$ na equação (3.7), obtendo $x_1=b+bm$, e assim

$$x_0 = c_1 + c_2 = b$$
$$x_1 = c_1 m + c_2 = b(1+m)$$

Resolvendo esse sistema, tem-se $c_1 = \dfrac{bm}{(m-1)}$ e $c_2 = \dfrac{-b}{(m-1)}$.

Substituindo as constantes c_1 e c_2 na solução geral , equação (3.10), tem-se a solução da recorrência (3.7) e a complexidade exata do algoritmo, ou seja, tem-se a quantidade exata de operações efetuadas na resolução do algoritmo, assim,

$$T(n) = x_n = \frac{b(m^{n+1}-1)}{(m-1)} \quad .$$

3.1.2 Desenvolvimento do método para *f(n)* exponencial

O algoritmo resolve o problema de tamanho n dividindo-o sucessivamente em m subproblemas de tamanho $n-1$. A complexidade do algoritmo, exceto pelas chamadas recursivas, é exponencial de base b. Isto é,

$$T(n)=(n=0\rightarrow b, b^n+mT(n-1)) \quad . \tag{3.11}$$

Como no caso anterior, coloca-se a equação de recorrência (3.11) no formato algébrico, substituindo $T(n)$ por x_n e definindo a base da recursividade como condição inicial, obtendo-se

$$\begin{aligned} &x_n-mx_{n-1}=b^n \\ &x_0=b. \end{aligned} \tag{3.12}$$

A equação (3.12) é transformada numa equação homogênea, para $m>1$ e $m\neq b$, do seguinte modo: como tem-se um exponencial de base b, multiplica-se todos os termos da recorrência (3.12) pela base da exponencial em seguida substitui-se n por $n-1$, obtendo-se

$$bx_{n-1}-bmx_{n-2}=bb^{n-1} \quad . \tag{3.13}$$

Finalizando, subtrai-se a equação (3.13) da equação (3.12), obtendo, dessa maneira, a equação homogênea:

$$x_n-(m+b)x_{n-1}+bmx_{n-2}=0 \tag{3.14}$$

Agora, escreve-se a equação característica associada à equação (3.14), conforme (3.3), obtendo-se assim,

$$r^{n-2}(r^2-(m+b)r+bm)=0 \quad .$$

Resolvendo esta equação característica obtém-se as raízes não nulas $r_1 = m$ e $r_2 = b$. Assim, a solução geral da equação (3.12), é da forma

$$x_n=c_1m^n+c_2b^n \quad . \tag{3.15}$$

Como a solução geral está na forma de combinação linear, precisa-se calcular os valores das constantes c_1 e c_2, substituindo sucessivamente $n=0$ e $n=1$ na equação (3.15). Obtém-se assim um sistema de equações que tem a condição inicial $x_0=b$ e $x_1=b+bm$ (calculado substituindo $n=1$ na equação (3.12)) como termos independentes,

$$\begin{aligned} x_0&=c_1 \quad +c_2 \ =b \\ x_1&=c_1m+c_2b=b(1+m) \end{aligned}$$

Resolvendo esse sistema, tem-se que $c_1=\dfrac{b^2-bm-b}{(b-m)}$ e $c_2=\dfrac{b}{(b-m)}$.

Como último passo, substitui-se os valores de c_1 e c_2 na solução geral da recorrência, equação (3.15), obtendo a solução da equação de recorrência (3.12) e a complexidade exata do algoritmo, assim

$$T(n)=x_n=\frac{b^{n+1}+b(bm^n-m^{n+1}-m^n)}{(b-m)} \quad .$$

3.1.3 Desenvolvimento do método para *f(n)* polinomial

O algoritmo resolve o problema de tamanho n dividindo-o sucessivamente em m subproblemas de tamanho $n-1$. A complexidade do algoritmo, exceto pela chamada recursiva, é polinomial de grau s. Isto é,

$$T(n)=(n=0 \rightarrow b, bn^s+mT(n-1)) \quad . \tag{3.16}$$

Neste caso, resolve-se a equação de recorrência com $s=1$, para demonstração do método, o que torna a complexidade linear. Porém na Tabela 3.1 encontra-se a complexidade exata para os polinômios de graus 2 e 3, onde a complexidade calculada é polinomial.

Como nos casos anteriores, coloca-se a equação (3.16) no formato algébrico e define-se a base da recursividade como condição inicial,

$$x_n-mx_{n-1}=bn \tag{3.17}$$
$$x_0=b.$$

Resolve-se a equação de recorrência (3.17) considerando $m>1$ e $m \neq b$. Para tornar homogênea esta recorrência deve-se, primeiro, tornar o lado direito da equação constante e depois o processo é como no primeiro caso. Quanto maior o grau do polinômio, maior número de reduções deverão ser realizadas. Para tornar constante o lado direito da equação, substitui-se n por $n-1$, obtendo a equação

$$x_{n-1}-mx_{n-2}=bn-b \quad . \tag{3.18}$$

Subtrai-se então, a equação (3.18) da equação (3.17) obtendo uma equação de mesmo formato que no caso 3.1.1, porém com mais termos, e a partir disto homogeneiza-se da forma já apresentada, obtendo no final a equação

$$x_n-(m+2)x_{n-1}+(2m+1)x_{n-2}-mx_{n-3}=0 \quad . \tag{3.19}$$

A equação característica associada à equação (3.19) é portanto

$$r^{n-3}(r^3-(m+2)r^2+(2m+1)r-m)=0 \quad .$$

Resolvendo-se a equação característica, obtém-se as raízes, não nulas, $r_1 = m$, $r_2 = r_3 = 1$. Assim, a solução geral da equação (3.17), possui o seguinte formato, já que $r=1$ é raiz de multiplicidade 2 (aplica-se o teorema 2.3),

$$x_n = c_1 m^n + c_2 + c_3 n \quad . \tag{3.20}$$

Agora, precisa-se calcular os valores de c_1, c_2 e c_3. Para isto substitui-se n=0, n=1 e n=2 na equação (3.20), obtendo um sistema de equações como antes. Neste caso, necessita-se de três condições iniciais, que são os termos independentes deste sistema. A primeira condição inicial x_0 já é fornecida, calcula-se x_1 e x_2 substituindo $n=1$ e $n=2$ na equação de recorrência (3.17), obtendo $x_1 = b+bm$ e $x_2 = bm^2+bm+2b$:

$$\begin{aligned} x_0 &= c_1 + c_2 = b \\ x_1 &= c_1 m + c_2 + c_3 = b(1+m) \\ x_2 &= c_1 m^2 + c_2 + 2c_3 = b(m^2+m+2) \end{aligned}$$

Resolvendo esse sistema, tem-se $c_1 = \frac{b(m^2-m+1)}{(m^2-2m+1)}$, $c_2 = \frac{-bm}{(m^2-2m+1)}$ e $c_3 = \frac{b-bm}{(m^2-2m+1)}$.

Como realizado nos casos anteriores, substitui-se os valores de c_1, c_2 e c_3 na solução geral da recorrência (3.20) para obter-se a solução exata da recorrência original (3.17) e a complexidade exata do algoritmo. Sendo assim,

$$T(n) = x_n = \frac{b(m^{n+2} - m^{n+1} + m^n - m + n - mn)}{(m^2-2m+1)} \quad .$$

3.1.4 Estudo de Casos

Apresenta-se a seguir dois exemplos de algoritmos conhecidos e como obtém-se as suas complexidades através de equações de recorrência.

3.1.4.1 Quicksort

O Quicksort [KNU73] é um exemplo clássico de algoritmo de classificação, que dada uma lista escolhe de alguma maneira um nó chamado de *pivot* e particiona a lista colocando à esquerda os elementos menores ou iguais ao *pivot* e à direita os maiores que o *pivot*.

Algoritmo Quicksort: para ordenar uma lista por concatenação de sublistas.
Entrada: (n-m+1, M_m, ..., M_n) e L=(M_m, ..., M_n) é uma lista de elementos com uma relação de ordem.
Saída: a lista L em ordem não decrescente.

$L_1 \leftarrow \phi$; $L_2 \leftarrow \phi$;
se (n-m+1=1) então
  pare-com-Saída(M_1);
  senão se (n-m+1)>1 então
    $M_i \leftarrow$"o pivot será o maior entre os dois primeiros elementos diferentes em L";
    "troque M_m e M_i de posição"
    j, L← *Particionamento* (M_m, ..., M_n);
    se j≥m então $L_1 \leftarrow$ Quicksort (j-m, M_m,..., M_{j-1}) fim-se;
    se j≤n então $L_2 \leftarrow$ Quicksort (n-j+1, M_j, ..., M_n) fim-se;
fim-se.
saída (L_1 "concatenado a " L_2).

O algoritmo Particionamento tem complexidade linear.

O algoritmo Quicksort tem pior caso quando a cada passo o pivot for sempre tal que particiona a lista em uma lista de 1 elemento (tamanho 1, já classificada) e outra de tamanho *n-1* que será classificada, chamando novamente o algoritmo e depois concatenando com outra sublista (de tamanho 1). A equação de recorrência, que pode ser facilmente determinada, é a seguinte:

$$T(n)=(n=0 \rightarrow b,\ bn+T(n-1)) \quad , \tag{3.21}$$

onde $f(n)=bn$, $m=1$,$s=1$ e $r(n)=(n-1)$, que é o tipo de recorrência apresentado em (3.16).

Primeiro, coloca-se a equação de recorrência (3.21) no formato algébrico, substituindo $T(n)$ por x_n e define-se a base da recursividade como condição inicial,

$$x_n - x_{n-1} = bn \tag{3.22}$$
$$x_0 = b.$$

Resolve-se a equação de recorrência (3.22) considerando $m=1$ e $m \neq b$. Realiza-se a homogeneização da recorrência da seguinte maneira: substitui-se n por $n-1$ obtendo a equação

$$x_{n-1} - x_{n-2} = bn - b \quad , \tag{3.23}$$

subtrai-se a equação (3.23) da equação (3.22) resultando na equação

$$x_n - 2x_{n-1} + x_{n-2} = b \tag{3.24}$$

e, como a equação não encontra-se no formato homogêneo, deve-se novamente substituir n por $n-1$, agora na equação (3.24), obtendo a equação

$$x_{n-1}-2x_{n-2}+x_{n-3}=b \quad . \tag{3.25}$$

Finalizando o processo para homogeneizar a equação de recorrência (3.22), subtrai-se a equação (3.25) da equação (3.24) obtendo a equação homogeneizada

$$x_n-3x_{n-1}+3x_{n-2}-x_{n-3}=0 \quad . \tag{3.26}$$

A equação característica associada conforme (3.3) é:

$$r^{n-3}(r^3-3r^2+3r-1)=0$$

que possui uma raiz de multiplicidade três: $r_1=r_2=r_3=1$. Assim, a solução geral da equação (3.22), conforme o teorema 2.3., possui o formato

$$x_n=c_1+c_2n+c_3n^2 \quad . \tag{3.27}$$

Para conhecer os valores de c_1, c_2 e c_3 substitui-se $n=0$, $n=1$ e $n=2$ na solução geral, equação (3.27), obtendo o seguinte sistema de equações que possui como termos independentes, x_0, x_1 e x_2. Calcula-se x_1 e x_2 substituindo $n=1$ e $n=2$ na equação de recorrência (3.22), obtendo $x_1=2b$ e $x_2=4b$.

$$\begin{aligned} x_0&=c_1 &&=b \\ x_1&=c_1+c_2+c_3 &&=2b \\ x_2&=c_1+2c_2+4c_3 &&=4b \end{aligned}$$

que tem como solução $c_1=b$ e $c_2=c_3=\frac{b}{2}$.

Por fim, substitui-se os valores de c_1, c_2 e c_3 na equação (3.27) obtendo a solução da equação de recorrência (3.22) e, conseqüentemente, a complexidade exata do algoritmo Quicksort, que é:

$$T(n)=x_n=\frac{b(n^2+n+2)}{2} \quad .$$

3.1.4.2 Torres de Hanoi

A origem das Torres de Hanoi [BRA95] deu-se da seguinte maneira: Sabe-se que depois da criação do mundo, Deus criou um conjunto de três hastes de diamantes e 64 círculos de ouro, sendo todos os círculos de tamanhos diferentes. No início foi empilhado sobre a primeira haste os círculos em ordem decrescente de tamanho, o círculo maior na

superfície e o menor no topo da haste. Deus também criou um mosteiro perto das hastes. A tarefa dos monges era transferir todos os círculos para dentro da segunda haste. A operação permitia somente mover um círculo de uma haste para outra, de modo que o círculo do topo sempre fosse menor. Quando os monges tivessem terminado suas tarefas, de acordo com a lenda, o mundo acabaria.

O problema pode ser generalizado para um número arbitrário de n círculos. Por exemplo, com $n=3$, resolve-se o problema como segue: necessita-se transferir o menor círculo n da haste i para a haste j (onde $1 \leq i \leq 3$, $1 \leq j \leq 3$, $i \neq j$ e $n \geq 1$), pode-se, primeiro, transferir o menor círculo $n-1$ da haste i para haste $k=6-i-j$, a próxima transferência é do círculo *n-ésimo* da haste i para haste j, e finalmente, transfere-se o menor círculo $n-1$ da haste k para a haste j. Segue a descrição formal do algoritmo para resolver a instância original.

```
Algoritmo Hanoi (n,i,j)
    {Move o menor círculo n da haste i para haste j}
    Se n>0 então Hanoi (n-1, i, 6-i-j)
                 escreve i "→" j;
                 Hanoi (n-1, 6-i-j, j);
    fim-se.
```

Na análise do tempo de execução deste algoritmo, tem-se que $T(n)$ é o número de vezes que é executado a chamada de Hanoi(n, ·, ·), obtendo a seguinte recorrência:

$$T(n)=(n=0 \rightarrow 0, 1+2T(n-1)) \tag{3.28}$$

onde $b=0$, $f(n)=1$, $m=2$ e $r(n)=n-1$. Trocando $T(n)$ por x_n e definindo a condição inicial pela base da recursividade, a equação de recorrência pode ser reescrita como

$$\begin{aligned} x_n - 2x_{n-1} &= 1 \\ x_0 &= 0. \end{aligned} \tag{3.29}$$

Para homogeneizar a equação de recorrência (3.29), substitui-se n por $n-1$, obtendo

$$x_{n-1} - 2x_{n-2} = 1 \quad . \tag{3.30}$$

Em seguida, subtrai-se a equação de recorrência (3.30) da equação de recorrência (3.29), obtendo a equação homogeneizada

$$x_n - 3x_{n-1} + 2x_{n-2} = 0 \quad . \tag{3.31}$$

Colocando a equação (3.31) no mesmo formato da equação (3.3), obtém-se a equação

característica

$$r^{n-2}(r^2-3r+2)=0$$

que tem como raízes características, não nulas, $r_1=1$ e $r_2=2$. Assim, a solução geral da equação de recorrência (3.29) é da forma

$$x_n=c_1+c_2 2^n \quad . \tag{3.32}$$

Neste caso, necessita-se de duas condições iniciais para determinar as constantes c_1 e c_2. A primeira condição já é conhecida, $x_0=0$, a segunda calcula-se substituindo $n=1$ na equação de recorrência original, (3.29), ou seja, $x_1=2x_0+1=1$.

Substituindo $n=0$ e $n=1$ na solução geral, equação (3.32), tem-se o seguinte sistema de equações com as condições iniciais como termos independentes,

$$\begin{aligned} x_0&=c_1+\ c_2=0 \\ x_1&=c_1+2c_2=1 \end{aligned}$$

que possui como solução $c_1=-1$ e $c_2=1$. Substituindo os valores destas constantes, c_1 e c_2, na solução geral, equação (3.32), obtém-se a solução exata da equação de recorrência original (3.29) e, em conseqüência, a complexidade exata do algoritmo Hanoi (n, ·, ·). Portanto

$$T(n)=x_n=2^n-1 \quad .$$

O número de execuções do comando "escrever" é uma boa medida do tempo obtido pelo algoritmo. Na realidade, o problema com n círculos não pode ser resolvido com número de movimentos menor que 2^n-1.

Nº de subprob.	f(n)=b	f(n)=b^n	f(n)=bn
m ≠ b	$T(n)=\frac{bm^{n+1}-b}{(m-1)}$	$T(n)=\frac{b^{n+1}+b^2m^n-bm^{n+1}-bm^n}{(b-m)}$	$T(n)=\frac{b(m^{n+2}-m^{n+1}+m^n-m+n-mn)}{(m^2-2m+1)}$
m = b	$T(n)=\frac{b^{n+2}-b}{(b-1)}$	$T(n)=b^{n+1}+nb^n$	$T(n)=\frac{b^{n+3}-b^{n+2}+b^{n+1}-b^2+bn-b^2n}{(b^2-2b+1)}$
m = 1	$T(n)=bn+b$	$T(n)=\frac{b^{n+1}+b^2-2b}{(b-1)}$	$T(n)=\frac{b(n^2+n)}{2}+b$

Nº de subprob.	f(n)=bn²
m ≠ b	$T(n)=\frac{b(m^{n+3}-2m^{n+2}+4m^{n+1}-m^n-m^2-m-2m^2n+2mn-m^2n^2+2mn^2-n^2)}{(m^3-3m^2+3m-1)}$
m = b	$T(n)=\frac{b^{n+4}-4b^{n+3}+2b^{n+2}-b^{n+1}+b^3+b^2-2b^3n+2b^2n-b^3n^2+2b^2n^2-bn^2}{(b^3-3b^2+3b-1)}$
m = 1	$T(n)=\frac{b(2n^3+3n^2+n+6)}{6}$

Nº de subprob.	f(n)=bn³
m ≠ b	$T(n)=\frac{b(m^{n+4}-3m^{n+3}+10m^{n+2}-3m^{n+1}+m^n-m^3-4m^2-m-m^3n^3+n^3+3mn(1-m^2-m^2n+2mn-n+mn^2-n^2))}{(m^4-4m^3+6m^2-3m+1)}$
m = b	$T(n)=\frac{b^{n+5}-3b^{n+4}+10b^{n+3}-3b^{n+2}+b^{n+1}-b^4+b^4n^3-4b^3-b^2+bn^3+3bn(b^2n^2+2b^2n-b^3+b^3n-bn^2-bn+b)}{(b^4-4b^3+6b^2-4b+1)}$
m = 1	$T(n)=\frac{b(n^4+2n^3+n^2+4)}{(4)}$

Tabela 3.1: Complexidade exata dos algoritmos para subproblemas de tamanho r(n)=(n-1).

3.2 Subproblema de tamanho r(n)=(n-ε)

Para este subproblema, calcula-se a complexidade exata para ε=2 e ε=3, podendo escolher outros valores inteiros e positivos para ε. Atribui-se valores para ε com o objetivo de observar o comportamento desse tipo de subproblema aplicado ao método das equações características, pois utilizando a variável ε diretamente no método não é possível concluir o cálculo devido a limitações que ocorrem com a variável ε e a falta de alguns termos no polinômio característico, ou seja, com o ε sem valor numérico não se sabe qual o termo que encontra-se ausente. Exemplificando, quando aplica-se o método das equações características, na recorrência $T(n)=(n=0\rightarrow b,\ b+mT(n-\varepsilon))$, obtém-se o seguinte polinômio: $x_n-x_{n-1}-mx_{n-\varepsilon}+mx_{n-\varepsilon-1}=0$ com condições iniciais $x_0=b$ e $x_1=b+mx_{1-\varepsilon}$ sugerindo neste caso ε=1, o que recai no subproblema de tamanho $r(n)=(n-1)$.

Este subproblema ocorre com pouca freqüência em problemas que empregam algoritmos recursivos, conseqüentemente, existem pouquíssimos exemplos a serem apresentados na seção Estudo de Casos.

De maneira geral, o algoritmo resolve o problema de tamanho n dividindo-o em m subproblemas de tamanho $n-\varepsilon$. A complexidade do algoritmo, exceto pela chamada recursiva, pode ser constante, exponencial e polinomial com grau maior que zero dependendo do termo $f(n)$ da equação de recorrência.

3.2.1 Subproblema com ε =2

A recorrência, devido ε ser igual a 2, apresenta-se do tipo linear incompleta de segunda ordem. Neste caso deve-se ter uma atenção maior devido a equação de recorrência não ser completa. No decorrer do processo, quando homogeneiza-se a equação algébrica, obtém-se um polinômio característico completo, porém com um número maior de termos O tamanho do sistema de equações será maior, assim a solução terá mais termos em função da complexidade do algoritmo. Nos casos em que o termo $f(n)$ for constante ou exponencial, o grau do polinômio característico é aumentado em dois em relação à equação de recorrência original. Já no caso em que $f(n)=bn^s$, a ordem do polinômio característico aumenta em 3(três) conforme o grau do polinômio da equação de recorrência, isto é, se $s=3$, o polinômio característico será de sexta ordem (ordem: $s+3$).

As raízes do polinômio característico são reais e inteiras, não ocorrem raízes complexas porque o polinômio característico torna-se completo após o processo de homogeneização. Para esta recorrência necessita-se de duas condições iniciais devido à recorrência ser de segunda ordem.

3.2.1.1 Desenvolvimento do método para *f(n)* constante

O algoritmo resolve o problema de tamanho *n* dividindo-o sucessivamente em *m* subproblemas de tamanho *n-2*. A complexidade do algoritmo, exceto pela chamada recursiva, é constante. Isto é,

$$T(n)=(n\leq 1 \rightarrow b, b+mT(n-2)) \quad . \tag{3.33}$$

Inicia-se o método substituindo *T(n)* por x_n para colocar a recorrência (3.33) no formato da equação (3.1) e define-se as duas condições iniciais pela base da recursividade da recorrência:

$$x_n - mx_{n-2} = b \tag{3.34}$$
$$x_0 = b$$
$$x_1 = b.$$

Para resolver a equação de recorrência (3.34) considera-se $m>1$ e $m\neq b$ e homogeneiza-se da seguinte maneira: substitui-se *n* por *n-1* obtendo a equação

$$x_{n-1} - mx_{n-3} = b \quad . \tag{3.35}$$

Em seguida subtrai-se a equação (3.35) da equação (3.34) obtendo a equação homogeneizada e completa

$$x_n - x_{n-1} - mx_{n-2} + mx_{n-3} = 0 \quad . \tag{3.36}$$

Terminado o processo de homogeneização, coloca-se a equação (3.36) no formato da equação (3.3), para obter-se a equação característica:

$$r^{n-3}(r^3 - r^2 - mr + m) = 0$$

o qual possui as raízes, não nulas, $r_1=1$, $r_2=\sqrt{m}$ e $r_3=-\sqrt{m}$. Com isso, tem-se que a solução geral da equação de recorrência (3.34) é

$$x_n = c_1 + c_2(\sqrt{m})^n + c_3(-\sqrt{m})^n \quad . \tag{3.37}$$

Novamente, calcula-se os valores de c_1, c_2 e c_3 substituindo $n=0$, $n=1$ e $n=2$ na solução geral, equação (3.37). Assim, obtém-se um sistema de equações, que possui como termos

independentes as condições iniciais, x_0 e x_1 já fornecidas e x_2 que deve ser calculado substituindo $n=2$ na equação de recorrência (3.34) obtendo $x_2=b+bm$. Assim, o sistema

$$\begin{aligned} x_0 &= c_1 + c_2 + c_3 = b \\ x_1 &= c_1 + c_2\sqrt{m} - c_3\sqrt{m} = b \\ x_2 &= c_1 + c_2 m + c_3 m = b + bm \end{aligned}$$

possui como solução $c_1=\dfrac{b\sqrt{m}}{2(\sqrt{m}-1)}$, $c_2=\dfrac{b\sqrt{m}}{2(\sqrt{m}+1)}$ e $c_3=\dfrac{-b}{(\sqrt{m}-1)(\sqrt{m}+1)}$.

Finalizando, substitui-se os valores de c_1, c_2 e c_3 na solução geral, equação (3.37), para obter-se a solução da recorrência original (3.34) e a complexidade exata do algoritmo, ou seja, a quantidade exata de operações efetuadas pelo algoritmo. Assim,

$$T(n)=x_n=\frac{b(4m^{(\frac{n+1}{2})}-1)}{(m-1)} \ .$$

3.2.1.2 Desenvolvimento do método para *f(n)* exponencial

O algoritmo resolve o problema de tamanho n dividindo-o em m subproblemas de tamanho $n-2$. A complexidade do algoritmo, exceto pela chamada recursiva, é exponencial de base b. Isto é,

$$T(n)=(n\leq 1 \rightarrow b, b^n + mT(n-2)) \ . \qquad (3.38)$$

Conforme nos casos anteriores, coloca-se a recorrência no formato algébrico, substituindo $T(n)$ por x_n e define-se duas condições iniciais a partir da base da recursividade, obtendo

$$\begin{aligned} x_n - mx_{n-2} &= b^n \qquad (3.39) \\ x_0 &= b \\ x_1 &= b. \end{aligned}$$

Para resolver a equação (3.39), considera-se $m>1$ e $m\neq b$, homogeneiza-a substituindo n por $n-1$ e, ao mesmo tempo, multiplica-se por b (base da exponencial), obtendo-se

$$bx_{n-1} - bmx_{n-3} = b(b^{n-1}) \ , \qquad (3.40)$$

em seguida, subtrai-se a equação (3.40) da equação (3.39) obtendo-se a equação homogeneizada e completa

$$x_n - bx_{n-1} - mx_{n-2} + bmx_{n-3} = 0 \ . \qquad (3.41)$$

Obtém-se a equação característica substituindo x_n por r^n:

$$r^{n-3}(r^3-br^2-mr+bm)=0 \quad .$$

Resolvendo a equação característica, obtém-se as raízes, não nulas, $r_1 = b$, $r_2=\sqrt{m}$ e $r_3=-\sqrt{m}$. Assim, a solução geral da recorrência (3.39) é da forma

$$x_n=c_1 b^n+c_2(\sqrt{m})^n+c_3(-\sqrt{m})^n \quad . \tag{3.42}$$

Substituindo $n=0$, $n=1$ e $n=2$ na solução geral (3.42) obtém-se um sistema de equações, com os termos independentes originados das condições iniciais. Tem-se duas condições iniciais conhecidas, x_0 e x_1, calcula-se x_2 substituindo $n=2$ na equação de recorrência (3.39), obtendo $x_2=b^2+bm$. Assim,

$$\begin{aligned} x_0&=c_1 \quad +c_2 \quad +c_3 \quad =b \\ x_1&=c_1 b+c_2\sqrt{m}-c_3\sqrt{m}=b \\ x_2&=c_1 b^2+c_2 m \ +c_3 m \ =b^2+bm \end{aligned}$$

Resolvendo este sistema, tem-se $c_1=\dfrac{b^2}{(b+\sqrt{m})(b-\sqrt{m})}$, $c_2=\dfrac{-b(b-\sqrt{m}-1)}{2(b-\sqrt{m})}$ e $c_3=\dfrac{b(b+\sqrt{m}-1)}{2(b+\sqrt{m})}$ e, substituindo estes valores na solução geral da recorrência, equação (3.42), tem-se a solução exata da equação de recorrência (3.39) e, conseqüentemente, a complexidade exata do algoritmo, ou seja,

$$T(n)=x_n=\frac{b^{n+2}-4bm^{\frac{n+1}{2}}}{(b^2-m)} \quad .$$

3.2.1.3 Desenvolvimento do método para *f(n)* polinomial

O algoritmo resolve o problema de tamanho n dividindo-o em m subproblemas de tamanho $n-2$. A complexidade do algoritmo, exceto pela chamada recursiva, é polinomial de ordem s. Isto é,

$$T(n)=(n\leq 1 \rightarrow b, bn^s+mT(n-2)) \quad . \tag{3.43}$$

Considera-se $s=1$ o que torna a complexidade linear. Os resultados para $s=2$ e $s=3$ encontram-se na Tabela 3.2 onde a complexidade apresenta-se na forma polinomial.

Procedendo de maneira similar aos casos anteriores, coloca-se a equação de recorrência (3.43) no formato algébrico e define-se as condições iniciais:

$$x_n - mx_{n-2} = bn \qquad (3.44)$$
$$x_0 = b$$
$$x_1 = b.$$

Homogeneiza-se a equação (3.44), considerando $m>1$ e $m \neq b$, em dois passos: primeiro substitui-se n por $n-1$ obtendo a equação

$$x_{n-1} - mx_{n-3} = bn - b \quad , \qquad (3.45)$$

subtrai-se a equação (3.45) da equação (3.44) obtendo

$$x_n - x_{n-1} - mx_{n-2} + mx_{n-3} = b \quad , \qquad (3.46)$$

como a equação (3.46) ainda não está homogeneizada, realiza-se o segundo passo: substitui-se n por $n-1$ na equação (3.46) obtendo

$$x_{n-1} - x_{n-2} - mx_{n-3} + mx_{n-4} = b \qquad (3.47)$$

e subtrai-se a equação (3.47) da equação (3.46), obtendo a equação homogeneizada

$$x_n - 2x_{n-1} - (m-1)x_{n-2} + 2mx_{n-3} - mx_{n-4} = 0 \quad . \qquad (3.48)$$

Conforme (3.3), obtém-se a equação característica

$$r^{n-4}(r^4 - 2r^3 - (m-1)r^2 + 2mr - m) = 0$$

cujas raízes, não nulas, são $r_1 = r_2 = 1$, $r_3 = \sqrt{m}$ e $r_4 = -\sqrt{m}$. Como existem raízes com multiplicidade maior que 1, a solução geral da recorrência (3.44) possui o formato

$$x_n = c_1 + c_2 n + c_3(\sqrt{m})^n + c_4(-\sqrt{m})^n \quad . \qquad (3.49)$$

A fim de calcular os valores de c_1, c_2, c_3 e c_4, substitui-se $n=0$, $n=1$, $n=2$ e $n=3$ na solução geral, (3.49), e considera-se quatro condições iniciais como termos independentes do sistema de equações, x_0, x_1, x_2 e x_3, sendo que as duas primeiras condições são conhecidas, e x_2 e x_3 devem ser calculadas substituindo $n=2$ e $n=3$ na equação de recorrência (3.44). Assim, $x_2 = 2b + bm$ e $x_3 = 3b + bm$, obtendo o sistema de equações

$$\begin{aligned}
x_0 &= c_1 + \quad + c_3 \quad + c_4 \quad = b \\
x_1 &= c_1 + c_2 \; + c_3\sqrt{m} \; - c_4\sqrt{m} \; = b \\
x_2 &= c_1 + c_2 2 + c_3 m \quad + c_4 m \quad = 2b + bm \\
x_3 &= c_1 + c_2 3 + c_3(\sqrt{m})^3 + c_4(-\sqrt{m})^3 = 3b + bm
\end{aligned}$$

que possui como solução $c_1 = \dfrac{-2(bm)}{(m^2 - 2m + 1)}$, $c_2 = \dfrac{-b}{(m-1)}$, $c_3 = \dfrac{b(m - \sqrt{m} + 1)}{2(m - 2\sqrt{m} + 1)}$ e

$$c_4=\frac{b(m+\sqrt{m}+1)}{2(m+2\sqrt{m}+1)} \quad .$$

Finalizando, substitui-se os valores de c_1, c_2, c_3 e c_4, na solução geral da recorrência, equação (3.49), obtendo a solução exata da equação de recorrência (3.44) e a complexidade exata do algoritmo,

$$T(n)=x_n=\frac{4b\left(m^{\frac{n+1}{2}}+m^{\frac{n+3}{2}}\right)+b(n-2m-mn)}{(m-1)^2} \quad .$$

3.2.1.4 Estudo de Caso

Algoritmo de Partição recursivo

O exemplo é uma versão do algoritmo Partição devido a C. A. Hoare [HOA61]. O algoritmo consiste em, dado um *pivot* particionar os elementos de um vetor, movimentando-os de tal forma que no final todos os elementos maiores ou iguais ao *pivot* estarão à direita, enquanto os menores à esquerda. O algoritmo usa dois apontadores *i* e *j* para indicar o trecho do vetor que deve ser examinado; *i* indica a posição mais à esquerda e vai sendo incrementado e *j* indica a posição mais à direita e vai sendo decrementado. Quando o apontador da direita encontra um elemento menor que o *pivot* e o apontador da esquerda um elemento maior que o *pivot,* esses dois elementos trocam de posição e o algoritmo é chamado novamente.

```
Algoritmo  Partição (A, i, j, pivot)
enquanto A(i) < pivot faça i← i+1
enquanto A(j) ≥ pivot faça j← j-1
se i ≥ j então escreve (A, j);
        senão se (j-i=1) então  troca (A(i), A(j));
                                escreve (A, j),
                        senão  troca (A(i), A(j));
                                Partição (A, i+1, j-1);
        fim-se
fim-se.
```

Este caso é especialmente interessante porque tem mais de uma operação fundamental, então a complexidade é obtida considerando as duas operações, com pesos

diferentes. Chame p_1 o peso da operação de comparação e p_2 o peso da operação de troca. O pior caso ocorre quando todos elementos são trocados, então a cada execução para um problema de tamanho n são realizadas duas comparações, uma troca e uma chamada recursiva para um problema de tamanho $n-2$. A equação de recorrência fica:

$$T(n)=(n\leq 1 \rightarrow 2p_1, b+T(n-2)) \tag{3.50}$$

onde $f(n)=b=2p_1+p_2$, $m=1$, $b=2p_1$, $r(n)=(n-2)$ e $\varepsilon=2$.

Procede-se da mesma maneira dos casos explicitados anteriormente. Coloca-se a equação de recorrência (3.50) no formato algébrico com as condições iniciais:

$$\begin{aligned} &x_n-x_{n-2}=b \\ &x_0=2p_1 \\ &x_1=2p_1. \end{aligned} \tag{3.51}$$

Realiza-se o processo de homogeneização, para tornar a equação (3.51) homogênea e completa, que resulta

$$x_n-x_{n-1}-x_{n-2}+x_{n-3}=0 \quad . \tag{3.52}$$

Em seguida, coloca-se a equação (3.52) no formato (3.3) obtendo a equação característica

$$r^{n-3}(r^3-r^2-r+1)=0$$

que possui como raízes, não nulas, $r_1=r_2=1$ e $r_3=-1$. A solução geral da equação de recorrência (3.51) possui o seguinte formato, devido às raízes apresentarem-se com multiplicidade maior do que 1:

$$x_n=c_1+c_2 n+c_3(-1)^n \quad . \tag{3.53}$$

Para obter-se a solução exata da equação de recorrência (3.51), substitui-se $n=0$, $n=1$ e $n=2$ na solução geral (3.53) e define-se três condições iniciais como termos independentes do sistema de equações. Assim, $x_0=x_1=b$ (já conhecidas) e $x_2=2b$, ou seja, substitui-se $n=2$ na equação de recorrência (3.51). Tem-se o sistema de equações:

$$\begin{aligned} &x_0=c_1 \qquad +c_3=2p_1 \\ &x_1=c_1+c_2 \quad -c_3=2p_1 \\ &x_2=c_1+2c_2+c_3=4p_1+p_2 \end{aligned}$$

que tem como solução $c_1=\dfrac{6p_1-p_2}{4}$, $c_2=\dfrac{4p_1+2p_2}{4}$ e $c_3=\dfrac{2p_1+p_2}{4}$.

Substituindo os valores de c_1, c_2 e c_3 na solução geral da recorrência (3.53) tem-se a solução exata da recorrência (3.51). Logo, a complexidade exata do algoritmo de Partição é

$$T(n)=x_n=n\left(p_1+\frac{1}{2}p_2\right)+\frac{3}{2}p_1-\frac{1}{4}p_2+(-1)^n\left(\frac{1}{2}p_1+\frac{1}{4}p_2\right) \quad , \tag{3.54}$$

ou seja, tem-se exatamente quantas vezes cada operação fundamental é executada.

Nºde subprob.	f(n)=b	f(n)=b^n	f(n)=bn
m ≠ b	$T(n)=\frac{b\left(4m^{\frac{n+1}{2}}-1\right)}{(m-1)}$	$T(n)=\frac{b^{n+2}-4bm^{\frac{n+1}{2}}}{(b^2-m)}$	$T(n)=\frac{4b\left(m^{\frac{n+1}{2}}+m^{\frac{n+3}{2}}\right)+b(n-2m-mn)}{(m^2-2m+1)}$
m = b	$T(n)=\frac{4b^{\frac{n+3}{2}}-b}{(b-1)}$	$T(n)=\frac{b^{n+1}-4b^{\frac{n+1}{2}}}{(b-1)}$	$T(n)=\frac{4\left(b^{\frac{n+3}{2}}+b^{\frac{n+5}{2}}\right)+b(n-bn-2b)}{(b^2-2b+1)}$
m = 1	$T(n)=\frac{b(2n+3+(-1)^n)}{4}$	$T(n)=\frac{b^{n+2}+2b^2(-1)^n(b-1)+2b(b^2-b-2)}{(b^2-1)}$	$T(n)=\frac{2b(n^2+2n)+b(3(-1)^n+5)}{8}$

Nº de subprob.	f(n)=bn²
m ≠ b	$T(n)=\frac{4b\left(m^{\frac{n+1}{2}}+6m^{\frac{n+3}{2}}+m^{\frac{n+5}{2}}-m+mn-m^2-m^2n\right)+bn^2(2m-m^2-1)}{(m^3-3m^2+3m-1)}$
m = b	$T(n)=\frac{4\left(b^{\frac{n+3}{2}}+6b^{\frac{n+5}{2}}+b^{\frac{n+7}{2}}\right)+4b^2(-b-bn+n-1)+bn^2(2b-b^2-1)}{(b^3-3b^2+3b-1)}$
m = 1	$T(n)=\frac{b(28n^3-66n^2+116n+9+39(-1)^n)}{48}$

Nº de subprob.	f(n)=bn³
m ≠ b	$T(n)=\frac{4b\left(m^{\frac{n+1}{2}}+23m^{\frac{n+3}{2}}+23m^{\frac{n+5}{2}}+m^{\frac{n+7}{2}}-2m-8m^2-2m^3-3m^3n+3mn\right)+b(-6m^3n^2+12m^2n^2-m^3n^3+3m^2n^3-3mn^3-6mn^2+n^3)}{(m^4-4m^3+6m^2-4m+1)}$
m = b	$T(n)=\frac{4\left(b^{\frac{n+3}{2}}+23b^{\frac{n+5}{2}}+23b^{\frac{n+7}{2}}+b^{\frac{n+9}{2}}\right)+4b(-4b-8b^2-2b^3-3b^3n+3b^2n+3b^2n^2)+3b^2n^2(bn-n-2b^2-2)+bn^3(1-b^3)}{(b^4-4b^3+6b^2-4b+1)}$
m = 1	$T(n)=\frac{b(57-136n+204n^2-8n^3+18n^4+39(-1)^n)}{96}$

Tabela 3.2: Complexidade exata dos algoritmos para subproblemas de tamanho r(n)=(n-ε), com ε=2.

3.2.2 Subproblema com ε=3

Neste caso, tem-se recorrência do tipo linear incompleta de terceira ordem. Este subproblema comporta-se de maneira semelhante ao anterior, no que diz respeito ao formato incompleto da equação de recorrência, número de raízes características, tamanho do sistema de equações e solução em função da complexidade do algoritmo, no caso, o termo *f(n)*.

Deve-se ter maior atenção quando homogeneiza-se a equação algébrica, pois o polinômio característico não se torna completo após o processo de homogeneização e, em conseqüência, pode-se obter raízes reais e complexas. Essas raízes são válidas para a solução em função da complexidade, pois aparecem sempre no formato conjugado, anulando-se ao substituir-se valores numéricos nas variáveis *b,m* e $i^2=-1$. Nos casos em que o termo *f(n)* for constante ou exponencial, o grau do polinômio característico é dois a mais em relação à equação de recorrência. Quando *f(n)* for um polinômio de grau maior que zero, a ordem do polinômio característico aumenta em 4 (quatro) conforme o grau do polinômio da equação de recorrência, ou seja, se *s=2*, o polinômio característico será de sexta ordem. Para este tipo de recorrência precisa-se de três condições iniciais.

Como trata-se de um tamanho de subproblema não muito comum em algoritmos desenvolvidos pelo método de Divisão-e-Conquista, não se encontrou exemplos a serem considerados na seção Estudo de Caso e, devido às raízes características serem complexas, não se elabora a tabela de complexidade exata para os casos gerais e particulares, conforme realizado nos casos anteriores.

3.2.2.1 Desenvolvimento do método para *f(n)* constante

O algoritmo resolve o problema de tamanho *n* dividindo-o sucessivamente em *m* subproblemas de tamanho *n-3*. A complexidade do algoritmo, exceto pela chamada recursiva, é constante, ou seja,

$$T(n)=(n\leq 2\rightarrow b, b+mT(n-3)) \quad . \tag{3.65}$$

Como nos casos anteriores, inicia-se o método substituindo *T(n)* por x_n e definindo as três condições iniciais pela base da recursividade da recorrência:

$$x_n - mx_{n-3} = b \qquad (3.66)$$
$$x_0 = b$$
$$x_1 = b$$
$$x_2 = b.$$

Resolve-se a recorrência (3.66) da seguinte maneira: primeiramente deve-se substituir n por $n-1$ na equação (3.66), obtendo

$$x_{n-1} - mx_{n-4} = b \ , \qquad (3.67)$$

em seguida, subtrai-se a equação (3.67) da equação (3.66) obtendo a equação homogeneizada e incompleta

$$x_n - x_{n-1} - mx_{n-3} + mx_{n-4} = 0 \ . \qquad (3.68)$$

No segundo passo, coloca-se a equação (3.68) no formato da equação (3.3) para obter-se a equação característica

$$r^{n-4}(r^4 - r^3 - mr + m) = 0$$

cujas raízes, não nulas, são $r_1 = 1$, $r_2 = m^{1/3}$, $r_3 = \left(\frac{-1}{2} m^{1/3} + \frac{1}{2} i\sqrt{3}\, m^{1/3}\right)$ e

$r_4 = \left(\frac{-1}{2} m^{1/3} - \frac{1}{2} i\sqrt{3}\, m^{1/3}\right)$.

Com estas raízes obtém-se a solução geral da equação de recorrência (3.66):

$$x_n = c_1 + c_2\left(m^{1/3}\right)^n + c_3\left(\frac{-1}{2} m^{1/3} + \frac{1}{2} i\sqrt{3}\, m^{1/3}\right)^n + c_4\left(\frac{-1}{2} m^{1/3} - \frac{1}{2} i\sqrt{3}\, m^{1/3}\right)^n \ . \qquad (3.69)$$

Substituindo $n=0$ e $n=1$ na equação (3.69) obtém-se um sistema de equações que possui como termos independentes as condições iniciais. Neste caso, necessita-se de quatro condições iniciais, sendo três (x_0, x_1 e x_2) já conhecidas. Calcula-se x_3 substituindo $n=3$ na equação de recorrência (3.66), obtendo $x_3 = bm + b$. Assim tem-se o seguinte sistema

$$\begin{aligned}
x_0 &= c_1 + c_2 + c_3 + c_4 = b \\
x_1 &= c_1 + c_2 m^{1/3} + c_3\left(\frac{-1}{2} m^{1/3} + \frac{1}{2} i\sqrt{3}\, m^{1/3}\right) + c_4\left(\frac{-1}{2} m^{1/3} - \frac{1}{2} i\sqrt{3}\, m^{1/3}\right) = b \\
x_2 &= c_1 + c_2 m^{2/3} + c_3\left(\frac{-1}{2} m^{1/3} + \frac{1}{2} i\sqrt{3}\, m^{1/3}\right)^2 + c_4\left(\frac{-1}{2} m^{1/3} - \frac{1}{2} i\sqrt{3}\, m^{1/3}\right)^2 = b \\
x_3 &= c_1 + c_2 m + c_3\left(\frac{-1}{2} m^{1/3} + \frac{1}{2} i\sqrt{3}\, m^{1/3}\right)^3 + c_4\left(\frac{-1}{2} m^{1/3} - \frac{1}{2} i\sqrt{3}\, m^{1/3}\right)^3 = b(m+1)
\end{aligned}$$

que possui como solução $c_1=\frac{bm^{1/3}}{\left(m^{1/3}-m^{4/3}\right)}$, $c_2=\frac{-4b\left(m^{4/3}+m^{2/3}+m\right)}{3\left(-4m^{4/3}+4m^{1/3}\right)}$,

$$c_3=\frac{4b\left(\sqrt{3}\,m^{4/3}+3im^{4/3}-2\sqrt{3}\,m^{2/3}+\sqrt{3}\,m-3im\right)}{(-\sqrt{3}-3i)\left(-4m^{4/3}+4m^{1/3}\right)} \quad \text{e}$$

$$c_4=\frac{-4bm^{2/3}\left(-1+m^{1/3}\right)\left(-4m^{1/3}+2\sqrt{3}-2i\right)}{3(-4i)\left(-4m^{4/3}+4m^{1/3}\right)} \quad .$$

Como último passo, substitui-se os valores de c_1, c_2, c_3 e c_4 na solução geral, (3.69), para obter a solução da equação de recorrência original, (3.66), e a complexidade exata do algoritmo. Assim,

$$T(n)=\left(\frac{1}{(4m^{\frac{4}{3}}-4m^{\frac{1}{3}})(4\sqrt{3}+12i)}\right)((m^{4/3})(162i-10\sqrt{3})+(m^{\frac{n+4}{3}})(48i+16\sqrt{3})+(m^{\frac{n+2}{3}})(48i+16\sqrt{3})+(m^{\frac{n+3}{3}})(48i+16\sqrt{}$$
$$+(m^{1/3})(-144i-48\sqrt{3})+\left(\frac{-1}{2}m^{1/3}+\frac{1}{2}i\sqrt{3}\,m^{1/3}\right)^n((m^{4/3})(48i+16\sqrt{3})+36im^{\frac{n+4}{3}}-32\sqrt{3}\,m^{2/3}+16\sqrt{3}\,m-48im)$$
$$+\left(\frac{-1}{2}m^{1/3}-\frac{1}{2}i\sqrt{3}\,m^{1/3}\right)^n((m^{4/3})(48i+16\sqrt{3})+(m^{2/3})(-48i+16\sqrt{3})-32\sqrt{3}\,m)$$

3.2.2.2 Desenvolvimento do método para *f(n)* exponencial

O algoritmo resolve o problema de tamanho *n* decompondo-o em *m* subproblemas de tamanho *n-3*. A complexidade do algoritmo, exceto as chamadas recursivas, é exponencial de base *b*. Isto é,

$$T(n)=(n\le 2\rightarrow b,\ b^n+mT(n-3)) \quad . \tag{3.70}$$

Iniciando a resolução da equação de recorrência (3.70), troca-se *T(n)* por x_n e define-se as três condições iniciais pela base da recursividade:

$$\begin{aligned} &x_n-mx_{n-3}=b^n \\ &x_0=b \\ &x_1=b \\ &x_2=b. \end{aligned} \tag{3.71}$$

Em seguida, substitui-se *n* por *n-1* em (3.71), multiplicando também todos os termos da equação por *b*, base da exponencial, obtendo

$$bx_{n-1}-bmx_{n-4}=b\,b^{n-1} \quad . \tag{3.72}$$

Subtrai-se a equação (3.72) da equação (3.71), resultando na equação homogeneizada

$$x_n - bx_{n-1} - mx_{n-3} + bmx_{n-4} = 0 \quad . \tag{3.73}$$

A equação característica associada à (3.73) é:

$$r^{n-4}(r^4 - br^3 - mr + bm) = 0$$

que possui como raízes, não nulas, $r_1 = b$, $r_2 = m^{1/3}$, $r_3 = \frac{-1}{2} m^{1/3} + \frac{1}{2} i\sqrt{3}\, m^{1/3}$ e

$r_4 = \frac{-1}{2} m^{1/3} - \frac{1}{2} i\sqrt{3}\, m^{1/3}$.

Para obter-se a solução geral da equação de recorrência original, (3.71), utiliza-se estas raízes características, como em (3.2):

$$x_n = c_1 b^n + c_2 (m^{1/3})^n + c_3 \left(\frac{-1}{2} m^{1/3} + \frac{1}{2} i\sqrt{3}\, m^{1/3} \right)^n + c_4 \left(\frac{-1}{2} m^{1/3} - \frac{1}{2} i\sqrt{3}\, m^{1/3} \right)^n \quad . \tag{3.74}$$

Necessita-se, agora, de quatro condições iniciais, sendo que três já são conhecidas. Precisa-se apenas calcular x_3, substituindo $n=3$ na equação de recorrência original, (3.71), obtendo $x_3 = b^3 + bm$. Com as condições iniciais, como termos independentes, e substituindo $n=0$, $n=1$, $n=2$ e $n=3$ na equação da solução geral, (3.74), obtém-se o seguinte sistema de equações:

$$\begin{aligned}
x_0 &= c_1 + c_2 + c_3 + c_4 = b \\
x_1 &= c_1 b + c_2 m^{1/3} + c_3 \left(\frac{-1}{2} m^{1/3} + \frac{1}{2} i\sqrt{3}\, m^{1/3} \right) + c_4 \left(\frac{-1}{2} m^{1/3} - \frac{1}{2} i\sqrt{3}\, m^{1/3} \right) = b \\
x_2 &= c_1 b^2 + c_2 (m^{1/3})^2 + c_3 \left(\frac{-1}{2} m^{1/3} + \frac{1}{2} i\sqrt{3}\, m^{1/3} \right)^2 + c_4 \left(\frac{-1}{2} m^{1/3} - \frac{1}{2} i\sqrt{3}\, m^{1/3} \right)^2 = b \\
x_3 &= c_1 b^3 + c_2 (m^{1/3})^3 + c_3 \left(\frac{-1}{2} m^{1/3} + \frac{1}{2} i\sqrt{3}\, m^{1/3} \right)^3 + c_4 \left(\frac{-1}{2} m^{1/3} - \frac{1}{2} i\sqrt{3}\, m^{1/3} \right)^3 = b^3 + bm
\end{aligned}$$

que possui como solução $c_1 = \frac{4b^3 m^{1/3}}{4m^{1/3}(m - b^3)}$,

$$c_2 = \left(\frac{b}{3(4m^{1/3} - 4b^3 m^{1/3})(-16m^2 + 16b^3 m)} \right) (-64m^{10/3} + m^{8/3}(-27b - 64) + m^{7/3}(128b^3 - 64b^2 + 27b))$$
$$+ m^{5/3}(-64b^4 + 128b^3) + m^{4/3}(-216b^6 + 64b^5) - 64b^6 m^{2/3} + 64m^2(b^3 - m)$$

$$c_3=\left(\frac{16b}{(3(-\sqrt{3}\,m^{1/3}-3im^{1/3})(4m^{4/3}-4b^3m^{1/3})(4m-4b^3))}\right)(m^{8/3}(-\sqrt{3}-3i)+m^{7/3}(-\sqrt{3}+3i$$
$$+m^{5/3}(2\sqrt{3}\,b^3-\sqrt{3}\,b^2-3ib^2)+m^{4/3}(\sqrt{3}\,b^3-3ib^3)+m^{2/3}(-\sqrt{3}\,b^6+\sqrt{3}\,b^5-3ib^6+3ib^5)$$
$$+2\sqrt{3}(b^4m-2b^3m-b^7+b^6+m^2)$$

e

$$c_4=\frac{8b\left(m^{5/3}(\sqrt{3}+i)+m^{4/3}(-\sqrt{3}+i)+m^{1/3}(-\sqrt{3}\,b^4+\sqrt{3}\,b^3+b^4i-b^3i)+2m(b^3i-b^2i-im)\right)}{(3(-4im)(4m-4b^3))}\ .$$

Como a solução do sistema de equações acima apresenta-se muita extensa, pode-se concluir que a solução exata da equação de recorrência (3.71), assim como a complexidade exata do algoritmo, torna-se muito extensa quando substitui-se os valores de c_1, c_2, c_3 e c_4 na solução geral, (3.74). Porém, com a equação da solução geral, (3.74), pode-se obter a ordem de complexidade, assim,

T(n) é de ordem $0(b^n)$.

3.2.2.3 Desenvolvimento do método para *f(n)* polinomial

O algoritmo resolve o problema de tamanho *n* decompondo-o em *m* subproblemas de tamanho *n-3*. A complexidade do algoritmo, exceto pela chamada recursiva, é polinomial de ordem *s*. Isto é,

$$T(n)=(n\le 2\rightarrow b,\ bn^s+mT(n-3))\ . \qquad (3.75)$$

Para demonstração do desenvolvimento do método das equações características, considera-se *s=1* no polinômio da equação de recorrência (3.75), o que torna a complexidade linear.

Como nos casos anteriores, coloca-se a equação de recorrência (3.75) no formato algébrico, trocando *T(n)* por x_n e definindo as três condições iniciais pela base da recursividade:

$$\begin{aligned} &x_n-mx_{n-3}=bn\ . \qquad (3.76)\\ &x_0=b\\ &x_1=b\\ &x_2=b. \end{aligned}$$

Agora, deve-se homogeneizar a equação (3.76) da seguinte forma: substitui-se *n* por *n-1* na equação (3.76), obtendo a equação

$$x_{n-1}-mx_{n-4}=bn-b\ . \qquad (3.77)$$

Em seguida, subtrai-se a equação (3.77) da equação (3.76), obtendo uma nova equação, porém ainda não homogeneizada,

$$x_n - x_{n-1} - mx_{n-3} + mx_{n-4} = b \quad . \tag{3.78}$$

Novamente, substitui-se n por $n-1$ na equação (3.78), obtendo

$$x_{n-1} - x_{n-2} - mx_{n-4} + mx_{n-5} = b \quad , \tag{3.79}$$

repete-se o penúltimo passo: subtrai-se a equação (3.79) da equação (3.78), obtendo a equação homogeneizada e completa:

$$x_n - 2x_{n-1} + x_{n-2} - mx_{n-3} + 2mx_{n-4} - mx_{n-5} = 0 \quad . \tag{3.80}$$

Neste momento, coloca-se a equação (3.80) no formato (3.3) obtendo, desse modo, a equação característica:

$$r^{n-5}(r^5 - 2r^4 + r^3 - mr^2 + 2mr - m) = 0 \quad ,$$

que possui como raízes, não nulas, $r_1 = r_2 = 1$, $r_3 = m^{1/3}$,

$$r_4 = \left(\frac{-1}{2} m^{1/3} + \frac{1}{2} i\sqrt{3}\, m^{1/3}\right) \quad \text{e} \quad r_5 = \left(\frac{-1}{2} m^{1/3} - \frac{1}{2} i\sqrt{3}\, m^{1/3}\right) \quad .$$

Para obter a solução geral da equação de recorrência (3.76), constrói-se a solução no formato de combinação linear destas raízes características, obtendo:

$$x_n = c_1 + c_2 n + c_3 (m^{1/3})^n + c_4 \left(\frac{-1}{2} m^{1/3} + \frac{1}{2} i\sqrt{3}\, m^{1/3}\right)^n + c_5 \left(\frac{-1}{2} m^{1/3} - \frac{1}{2} i\sqrt{3}\, m^{1/3}\right)^n \quad . \tag{3.81}$$

Precisa-se calcular os valores de c_1, c_2, c_3, c_4 e c_5 para obter a solução exata da equação de recorrência original, (3.76). Sendo assim, primeiro necessita-se de cinco condições iniciais para formar os termos independentes do sistema de equações, sendo que três já são conhecidas. Precisa-se calcular a quarta e a quinta condições iniciais, substituindo $n=3$ e $n=4$ na equação de recorrência (3.76). Obtém-se assim $x_3 = 3b + bm$ e $x_4 = 4b + bm$. Substituindo-se $n=0$, $n=1$, $n=2$, $n=3$ e $n=4$ na solução geral, (3.81), obtém-se o seguinte sistema de equações

$$x_0 = c_1 + c_3 + c_4 + c_5 = b$$

$$x_1 = c_1 + c_2 + c_3 m^{1/3} + c_4\left(\frac{-1}{2} m^{1/3} + \frac{1}{2} i\sqrt{3}\, m^{1/3}\right) + c_5\left(\frac{-1}{2} m^{1/3} - \frac{1}{2} i\sqrt{3}\, m^{1/3}\right) = b$$

$$x_2 = c_1 + 2c_2 + c_3 (m^{1/3})^2 + c_4\left(\frac{-1}{2} m^{1/3} + \frac{1}{2} i\sqrt{3}\, m^{1/3}\right)^2 + c_5\left(\frac{-1}{2} m^{1/3} - \frac{1}{2} i\sqrt{3}\, m^{1/3}\right)^2 = b$$

$$x_3 = c_1 + 3c_2 + c_3 (m^{1/3})^3 + c_4\left(\frac{-1}{2} m^{1/3} + \frac{1}{2} i\sqrt{3}\, m^{1/3}\right)^3 + c_5\left(\frac{-1}{2} m^{1/3} - \frac{1}{2} i\sqrt{3}\, m^{1/3}\right)^3 = 3b + bm$$

$$x_4 = c_1 + 4c_2 + c_3 (m^{1/3})^4 + c_4\left(\frac{-1}{2} m^{1/3} + \frac{1}{2} i\sqrt{3}\, m^{1/3}\right)^4 + c_5\left(\frac{-1}{2} m^{1/3} - \frac{1}{2} i\sqrt{3}\, m^{1/3}\right)^4 = 4b + bm$$

cuja solução é

$$c_1 = \left(\frac{-3bm^{2/3}}{(3m^{1/3} - m - 2)(-24i\sqrt{3}\, m - 12m^{1/3} + 4i\sqrt{3}\, m^{1/3} + 8i\sqrt{3}\, m^{4/3} - 24m^{4/3} + 24m^{2/3} + 8i\sqrt{3}\, m^{2/3} + 4i\sqrt{3}\, m^{5/3} + 12m^{5/3})}\right)$$
$$\left(\frac{1}{(-16m^3 + 48m^{7/3} - 96m^{4/3} + 48m^{1/3} - 72m - 32)}\right)(3456m^3 - 2496i\sqrt{3}\, m^{4/3} - 1152m^{10/3} + 1152m^{5/3} - 6336m^{4/3} - 8064m^7$$
$$-768m^{2/3} + 192m^4 + 8736m^2 + 3840m - 384m^{11/3} - 246i\sqrt{3}\, m + 256i\sqrt{3}\, m^{2/3} + 6528i\sqrt{3}\, m^{5/3} - 1920i\sqrt{3}\, m^{7/3} + 64i\sqrt{3}\, m$$
$$-384i\sqrt{3}\, m^3 - 4608i\sqrt{3}\, m^2 + 128i\sqrt{3}\, m^{11/3} + 3456i\sqrt{3}\, m^{8/3} - 760i\sqrt{3}\, m^{10/3}$$

$$c_2 = \left(\frac{4bm^{1/3}}{(3m^{1/3} - m - 2)(-24i\sqrt{3}\, m - 12m^{1/3} + 4i\sqrt{3}\, m^{1/3} + 8i\sqrt{3}\, m^{4/3} - 24m^{4/3} + 24m^{2/3} + 8i\sqrt{3}\, m^{2/3} + 4i\sqrt{3}\, m^{5/3} + 12m^{5/3})}\right)$$
$$\left(\frac{1}{(-16m^3 + 48m^{7/3} - 96m^{4/3} + 48m^{1/3} - 72m - 32)}\right)(66m^3 - 1052i\sqrt{3}\, m^{4/3} - 816m^{10/3} + 1728m^{8/3} - 432m^{5/3} + 1156m^{4/3} + 480$$
$$96m^4 + 960m^{1/3} - 288m^2 + 64i\sqrt{3} - 64i\sqrt{3}\, m^{1/3} - 1152m^{7/3} - 1584m^{2/3} - 192 + 288m^{1/3} - 48m^{13/3} + 1568i\sqrt{3}\, m - 624i\sqrt{3}\, m^{2/}$$
$$+144i\sqrt{3}\, m^{5/3} - 36i\sqrt{3}\, m^{4/3} - 768i\sqrt{3}\, m^2 - 822i\sqrt{3}\, m^3 - 32i\sqrt{3}\, m^4 - 16i\sqrt{3}\, m^{13/3} + 398i\sqrt{3}\, m^{8/3} + 199i\sqrt{3}\, m^{11/3} + 112i\sqrt{3}\, m^{10/}$$
$$+1056i\sqrt{3}\, m^{7/3}$$

$$c_3 = \left(\frac{4b}{3(3m^{1/3} - m - 2)(-24i\sqrt{3}\, m - 12m^{1/3} + 4i\sqrt{3}\, m^{1/3} + 8i\sqrt{3}\, m^{4/3} - 24m^{4/3} + 24m^{2/3} + 8i\sqrt{3}\, m^{2/3} + 4i\sqrt{3}\, m^{5/3} + 12m^{5/3})}\right)$$
$$\left(\frac{1}{-4m^{1/3}(-16m^3 + 48m^{7/3} - 96m^{4/3} + 48m^{1/3} - 48m - 32)}\right)(5568m^3 - 1856i\sqrt{3}\, m^{4/3} - 192m^6 - 3840m^{10/3} - 192i\sqrt{3}\, m^{17/3}$$
$$+9024m^{8/3} + 4416m^{4/3} - 384m + 1920m^4 + 4608m^{1/3} - 1920m^2 + 256i\sqrt{3} - 256i\sqrt{3}\, m^{1/3} - 20784m^{7/3} - 5568m^{2/3} - 2112$$
$$+72880m^{5/3} - 768 + 192m^{17/3} - 4224m^{11/3} + 1965m^{16/3} + 5952m^{13/3} - 2304m^{14/3} + 6016i\sqrt{3}\, m - 2752i\sqrt{3}\, m^{2/3} - 704i\sqrt{3}\, m$$
$$-6528i\sqrt{3}\, m^2 - 8236i\sqrt{3}\, m^{7/3} - 5184i\sqrt{3}\, m^3 - 64i\sqrt{3}\, m^6 + 4544i\sqrt{3}\, m^4 + 960i\sqrt{3}\, m^5 + 576i\sqrt{3}\, m^{16/3} - 832i\sqrt{3}\, m^{13/3}$$
$$-2560i\sqrt{3}\, m^{14/3} - 512i\sqrt{3}\, m^{11/3} - 2432i\sqrt{3}\, m^{10/3} + 6720i\sqrt{3}\, m^{8/3}$$

$$c_4 = \left(\frac{4b}{3(-24i\sqrt{3}\, m - 12m^{1/3} + 4i\sqrt{3}\, m^{1/3} + 8i\sqrt{3}\, m^{4/3} - 24m^{4/3} + 24m^{2/3} + 8i\sqrt{3}\, m^{2/3} + 4i\sqrt{3}\, m^{5/3} + 12m^{5/3})}\right)$$
$$\left(\frac{1}{im^{1/3}(-16m^3 + 48m^{7/3} - 96m^{4/3} + 48m^{1/3} - 48m - 32)}\right)(-172\sqrt{3}\, m^5 - 352\sqrt{3}\, m^2 + 48\sqrt{3}\, m^{5/3} + 384\sqrt{3}\, m^{7/3} + 32\sqrt{3}\, m^{2/}$$
$$480im^{4/3} - 816im^{5/3} - 192im^{1/3} + 672im^{2/3} + 336im^{11/3} - 96im^{10/3} - 288im^{7/3} + 112\sqrt{3}\, m^{11/3} - 288im^{8/3} - 128\sqrt{3}\, m^{1}$$
$$+128\sqrt{3}\, m + 80\sqrt{3}\, m^3 + 64\sqrt{3} - 76im - 48im^5 - 384im^4 + 144im^3 + 1056im^2 - 160\sqrt{3}\, m^{1/3} + 64\sqrt{3}\, m^4 + 96im^{14/3}$$
$$-99\sqrt{3}\, m^{13/3} + 96im^{13/3}$$

e

$$c_5=\left(\frac{-4b}{-12\,i\,m(-16m^3+4m^{7/3}-18m^{8/3}-96m^{4/3}+4m^{1/3}+120m-32)}\right)(-16\sqrt{3}\,m^{4/3}+56\sqrt{3}\,m^{8/3}+104\sqrt{3}\,m^{5/3}-40\sqrt{3}\,m$$
$$-48\sqrt{3}\,m^2+88\,i\,m^{4/3}-40\,i\,m^{5/3}-16\,i\,m^{1/3}+24\,i\,m^{2/3}+8\,i\,m^{11/3}+56\,i\,m^{10/3}+88\,i\,m^{7/3}+8\sqrt{3}\,m^{11/3}+8\,i\,m^{8/3}-8\sqrt{3}\,m^{10/}$$
$$-24\sqrt{3}\,m^{2/3}-48\sqrt{3}\,m^3-32\,i\,m-72\,i\,m^3-6\,i\,m^4-96\,i\,m^2+16\sqrt{3}\,m^{1/3}+24\sqrt{3}\,m^3+3\sqrt{3}\,m^4-3\sqrt{3}\,m$$

Como a solução do sistema de equações é muito extensa, a complexidade exata do algoritmo, ou, a solução exata da equação de recorrência (3.76) também é muito extensa, pois os valores de c_1, c_2, c_3, c_4 e c_5 são substituídos na solução geral, (3.81). Porém, com a equação da solução geral da recorrência, (3.81), pode-se obter a ordem de complexidade do algoritmo. Sendo asssim,

$T(n)$ é de ordem O(n).

3.2.2.4 Comentários

Para este tipo de subproblema, verifica-se que não é muito viável obter a complexidade exata do algoritmo, devido ao grande número de termos no resultado final, proveniente da solução do sistema de equações. A solução geral da equação de recorrência, por sua vez, não se apresenta muito simples, devido às raízes características serem complexas.

Neste caso, o melhor é obter a complexidade em notação assintótica, a qual pode ser obtida após a análise das constantes presentes na solução geral, ou seja, c_1, c_2,..., c_n. Deve-se realizar esta análise sobre as constantes com o objetivo de verificar qual é a constante dominante na solução geral.

Brassard [BRA95] obtém a ordem de complexidade do algoritmo, substituindo a equação da solução geral na equação de recorrência original e a partir desta substituição, tem-se uma nova equação em termos das constantes a serem analisadas.

Cormen [COR90] sugere uma aproximação no resultado da complexidade exata, quando este resultado não for um número real, pois o tempo de processamento de um algoritmo é definido em termos de funções, cujo o domínio é o conjunto dos números naturais $N=\{0, 1, 2,...\}$ e o tamanho da entrada é definido por um número inteiro positivo. Por essa razão, a complexidade exata de um algoritmo deve resultar em um número inteiro, ou, pertencente ao conjunto dos números naturais, $N=\{0, 1, 2, ...\}$.

3.3 Subproblema de tamanho r(n)=(n-1)+(n-2)

Este subproblema apresenta recorrência do tipo linear de segunda ordem completa, conforme a Tabela 2.2 de Classificação das Recorrências. Durante o processo de desenvolvimento do método das equações características, obtém-se o polinômio característico completo, porém com um grau a mais em relação à equação de recorrência original. Este fato ocorre quando $f(n)$ é constante e exponencial. Quando $f(n)$ é um polinômio de grau maior que zero, o grau do polinômio característico aumenta em três, em relação ao grau do polinômio da equação de recorrência original, ou seja, se $s=2$ (grau do polinômio da recorrência original, $f(n)=bn^s$) o polinômio característico será de quinta ordem. Claramente o número de raízes características aumenta conforme o grau do polinômio característico. Em conseqüência, o tamanho do sistema de equações também aumenta, pois para cada raíz tem-se uma constante c_i a determinar.

Trata-se de um tipo de subproblema não muito comum em algoritmos desenvolvidos pelo método de Divisão-e-Conquista, por isso não se encontrou exemplos a serem apresentados na seção Estudo de Casos. Elabora-se a tabela de complexidade exata para $f(n)$ constante, exponencial e linear ($f(n)=bn$), os resultados apresentam-se muito extensos devido ao formato das raízes características e do tipo do subproblema em si.

Necessita-se, neste caso, de duas condições iniciais e, por possuir m_1 subproblemas de um tipo e m_2 subproblemas de outro deve-se calcular a complexidade exata para um maior número de casos gerais, ou seja, $b\neq m_1\neq m_2$, $b\neq m_1=m_2$, $b=m_1=m_2$, $m_1=m_2=1$, $m_1=1$ e $m_2=1$. Para demonstração do método, considera-se, para os três casos de $f(n)$, $b\neq m_1\neq m_2$. A equação de complexidade exata resulta, de maneira geral, em termos do número dos dois subproblemas, m_1 e m_2 e no formato do termo $f(n)$ da equação de recorrência do algoritmo.

3.3.1 Desenvolvimento do método para $f(n)$ constante

O algoritmo resolve o problema de tamanho n decompondo-o em m_1 subproblemas de tamanho (n-1) e m_2 subproblemas de tamanho (n-2). A complexidade do algoritmo, exceto as chamadas recursivas, é constante e a base da recursividade é b quando $n=1$. Isto é,

$$T(n)=(n=1\rightarrow b,\ b+m_1T(n-1)+m_2T(n-2)) \quad . \tag{3.82}$$

Como realizado nos casos anteriores, coloca-se a equação de recorrência (3.82) no formato algébrico, substituindo $T(n)$ por x_n e, ao mesmo tempo, define-se duas condições iniciais pela base da recursividade:

$$\begin{aligned} &x_n - m_1 x_{n-1} - m_2 x_{n-2} = b \\ &x_0 = b \\ &x_1 = b. \end{aligned} \tag{3.83}$$

Substitui-se n por $n-1$ em (3.83), obtendo a equação

$$x_{n-1} - m_1 x_{n-2} - m_2 x_{n-3} = b \quad , \tag{3.84}$$

e subtrai-se a equação (3.84) da equação (3.83), obtendo-se a equação homogeneizada e completa

$$x_n - (m_1+1)x_{n-1} - (m_2 - m_1)x_{n-2} + m_2 x_{n-3} = 0 \quad . \tag{3.85}$$

Em seguida, obtém-se a equação característica:

$$r^{n-3}(r^3 - (m_1+1)r^2 - (m_2 - m_1)r + m_2) = 0$$

cujas raízes, não nulas, são: $r_1 = 1$, $r_2 = \frac{1}{2}m_1 + \frac{1}{2}\sqrt{m_1^2 + 4m_2}$ e $r_3 = \frac{1}{2}m_1 - \frac{1}{2}\sqrt{m_1^2 + 4m_2}$

. Assim, a solução geral da equação de recorrência (3.83), é da forma

$$x_n = c_1 + c_2\left(\frac{1}{2}m_1 + \frac{1}{2}\sqrt{m_1^2 + 4m_2}\right)^n + c_3\left(\frac{1}{2}m_1 - \frac{1}{2}\sqrt{m_1^2 + 4m_2}\right)^n \quad . \tag{3.86}$$

Como a solução geral encontra-se no formato de combinação linear das raízes características, devido ao número de raízes, precisa-se de três condições iniciais para formar os termos independentes do sistema de equações. Destas, x_0 e x_1 já são conhecidas, precisando-se calcular apenas x_2 substituindo $n=2$ na equação de recorrência (3.83). Obtém-se, desta forma, $x_2 = b + bm_1 + bm_2$. Substituindo $n=0$, $n=1$ e $n=2$ na equação (3.86), obtém-se o sistema de equações

$$\begin{aligned} x_0 &= c_1 + c_2 + c_3 = b \\ x_1 &= c_1 + c_2\left(\frac{1}{2}m_1 + \frac{1}{2}\sqrt{m_1^2 + 4m_2}\right) + c_3\left(\frac{1}{2}m_1 - \frac{1}{2}\sqrt{m_1^2 + 4m_2}\right) = b \\ x_2 &= c_1 + c_2\left(\frac{1}{2}m_1 + \frac{1}{2}\sqrt{m_1^2 + 4m_2}\right)^2 + c_3\left(\frac{1}{2}m_1 - \frac{1}{2}\sqrt{m_1^2 + 4m_2}\right)^2 = b + bm_1 + bm_2 \end{aligned}$$

que possui como solução $c_1 = \frac{-b}{(m_1 + m_2 - 1)}$,

$c_2 = \frac{-b(m_1^2 - 2m_1 + m_1 m_2 - 2m_2 + \sqrt{m_1^2 + 4m_2}(-m_1 - m_2))}{2(m_1^2 + 4m_2)(m_1 + m_2 - 1)}$ e

$$c_3=\frac{b(m_1^2-2m_1+m_1m_2-2m_2+\sqrt{m_1^2+4m_2}(m_1+m_2))}{2(m_1^2+4m_2)(m_1+m_2-1)} \quad .$$

Finalizando, substitui-se os valores de c_1, c_2 e c_3 na equação da solução geral (3.86) e obtém-se a solução exata da equação de recorrência original (3.83) e a complexidade exata do algoritmo, ou seja,

$$T(n)=\left(\frac{b}{2(m_1^2+4m_2)(m_1+m_2-1)}\right)(-2(m_1^2+4m_2)-\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^n(m_1^2-2m_1+m_1m_2-2m_2))$$
$$+\sqrt{m_1^2+4m_2}(-m_1-m_2)+\left(\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^n(m_1^2-2m_1+m_1m_2-2m_2+\sqrt{m_1^2+4m_2}(m_1+m_2))$$

3.3.2 Desenvolvimento do método para *f(n)* exponencial

O algoritmo resolve o problema de tamanho n decompondo-o em m_1 subproblemas de tamanho (n-1) e m_2 subproblemas de tamanho (n-2). A complexidade do algoritmo, exceto as chamadas recursivas, é exponencial de base b. A base da recursividade é b quando $n=1$. Isto é,

$$T(n)=(n=1\rightarrow b, b^n+m_1T(n-1)+m_2T(n-2)) \quad . \qquad (3.87)$$

Para a resolução da equação de recorrência (3.87) procede-se de forma semelhante aos casos anteriores. Substitui-se $T(n)$ por x_n, e define-se as duas condições iniciais pela base da recursividade:

$$x_n-m_1x_{n-1}-m_2x_{n-2}=b^n \qquad (3.88)$$
$$x_0=b$$
$$x_1=b.$$

Substitui-se n por $n-1$ em (3.88) e multiplica-se todos os termos da equação por b, base da exponencial, obtendo

$$bx_{n-1}-bm_1x_{n-2}-bm_2x_{n-3}=bb^{n-1} \quad . \qquad (3.89)$$

Em seguida, subtrai-se a equação (3.89) da equação (3.88), obtendo a equação homogeneizada e completa:

$$x_n-(m_1+b)x_{n-1}-(m_2-bm_1)x_{n-2}+bm_2x_{n-3}=0 \quad . \qquad (3.90)$$

A equação característica associada é:

$$r^{n-3}(r^2-(m_1+b)r^2-(m_2-bm_1)r+bm_2)=0 \quad ,$$

que possui como raízes, não nulas, $r_1=b$, $r_2=\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4m_2}$ e $r_3=\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4m_2}$. Com estas raízes características obtém-se a solução geral da equação de recorrência (3.88), no formato de combinação linear, isto é,

$$x_n=c_1 b^n+c_2\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^n+c_3\left(\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^n . \qquad (3.91)$$

Para obter-se a solução exata da equação de recorrência (3.88), deve-se ter três condições iniciais, sendo x_0 e x_1 já conhecidos. A terceira condição calcula-se substituindo $n=2$ na equação de recorrência (3.88) obtendo $x_2=b^2+bm_1+bm_2$. Substituindo $n=0$, $n=1$ e $n=2$ na equação da solução geral (3.91) obtém-se um sistema de equações composto pelas condições iniciais como termos independentes, assim,

$$\begin{aligned}
x_0&=c_1 \ +c_2 \qquad\qquad\qquad +c_3 \qquad\qquad\qquad =b\\
x_1&=c_1 b+c_2\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4m_2}\right)+c_3\left(\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4m_2}\right)=b\\
x_2&=c_1 b^2+c_2\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^2+c_3\left(\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^2=b^2+bm_1+bm_2
\end{aligned}$$

que possui como solução $c_1=\frac{b^2}{(b^2-bm_1-bm_2)}$,

$c_2=\frac{-b(b^2m_1-bm_1^2+bm_1-m_1m_2-2m_2+\sqrt{m_1^2+4m_2}(-b^2+bm_1+b+m_2))}{2(\sqrt{m_1^2+4m_2}(b^2-bm_1-m_2))}$ e

$c_3=\frac{b(b^2m_1-bm_1^2+bm_1-m_1m_2+2m_2+\sqrt{m_1^2+4m_2}(b^2-bm_1-b-m_2))}{2(\sqrt{m_1^2+4m_2}(b^2-bm_1-bm_2))}$.

Substituindo a solução do sistema de equações, na equação da solução geral (3.91), obtém-se a solução exata da equação de recorrência original (3.88) e a complexidade exata do algoritmo, que é

$$\begin{aligned}
T(n)=&\left(\frac{b}{2(\sqrt{m_1^2+4m_2}(b^2-bm_1-m_2))}\right)(2b^{n+1}\sqrt{m_1^2+4m_2}-\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^n(b^2m_1-bm_1^2+bm_1)\\
&-m_1m_2+2m_2+\sqrt{m_1^2+4m_2}(-b^2+bm_1+b+m_2)+\left(\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^n(b^2m_1-bm_1^2+bm_1-m_1m_2\ )\\
&+2m_2+\sqrt{m_1^2+4m_2}(b^2-bm_1-b-m_2)
\end{aligned}$$

3.3.3 Desenvolvimento do método para *f(n)* polinomial

O algoritmo resolve o problema de tamanho n decompondo-o em m_1 subproblemas de tamanho (n-1) e m_2 subproblemas de tamanho (n-2). A complexidade do algoritmo, exceto as chamadas recursivas, é polinomial de grau maior que zero. Isto é,

$$T(n)=(n=1 \rightarrow b,\ bn^s+m_1T(n-1)+m_2T(n-2)) \quad . \tag{3.92}$$

Para demonstração da resolução do método, considera-se $s=1$, obtendo uma complexidade linear.

Resolve-se a recorrência (3.92) como nos casos anteriores: substitui-se $T(n)$ por x_n e define-se duas condições iniciais pela base da recursividade:

$$\begin{aligned} &x_n-m_1x_{n-1}-m_2x_{n-2}=bn \qquad (3.93)\\ &x_0=b\\ &x_1=b. \end{aligned}$$

Substituindo n por $n-1$, obtém-se

$$x_{n-1}-m_1x_{n-2}-m_2x_{n-3}=bn-b \quad , \tag{3.94}$$

Em seguida, subtrai-se a equação (3.94) da equação (3.93), obtendo uma recorrência com o lado direito constante,

$$x_n-(m_1+1)x_{n-1}-(m_2-m_1)x_{n-2}+m_2x_{n-3}=b \quad . \tag{3.95}$$

Novamente, substitui-se n por $n-1$, obtendo

$$x_{n-1}-(m_1+1)x_{n-2}-(m_2-m_1)x_{n-3}+m_2x_{n-4}=b \quad , \tag{3.96}$$

e subtrai-se a equação (3.96) da equação (3.95), obtendo a equação homogeneizada e completa

$$x_n-(m_1+2)x_{n-1}+(2m_1-m_2+1)x_{n-2}+(2m_2-m_1)x_{n-3}-m_2x_{n-4}=0 \quad . \tag{3.97}$$

A equação característica associada é:

$$r^{n-4}(r^4-(m_1+2)r^3+(2m_1-m_2+1)r^2+(2m_2-m_1)r-m_2)=0 \quad .$$

As raízes, não nulas, desta equação característica são:

$r_1=r_2=1$, $r_3=\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4m_2}$ e $r_4=\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4m_2}$. Com estas raízes características constrói-se a solução geral da equação de recorrência (3.93):

$$x_n=c_1+c_2n+c_3\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^n+c_4\left(\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^n \quad . \tag{3.98}$$

Para determinar c_1, c_2, c_3 e c_4 precisa-se de x_0 e x_1 já conhecidos e de x_2 e x_3, que podem

ser calculadas substituindo $n=2$ e $n=3$ na equação de recorrência (3.93). Dessa forma, obtém-se $x_2=2b+bm_1+bm_2$ e $x_3=3b+2bm_1+bm_1^2+bm_2+2bm_1m_2$.. Substituindo $n=0$, $n=1$ e $n=2$ na equação (3.98), obtém-se o sistema de equações,

$$
\begin{aligned}
x_0 &= c_1 + c_3 + c_4 = b \\
x_1 &= c_1 + c_2 + c_3\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4m_2}\right) + c_4\left(\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4m_2}\right) = b \\
x_2 &= c_1 + 2c_2 + c_3\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^2 + c_4\left(\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^2 = 2b+bm_1+bm_2 \\
x_3 &= c_1 + 3c_2 + c_3\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^3 + c_4\left(\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^3 = 3b+2bm_1+bm_1^2+bm_2+2bm_1m_2
\end{aligned}
$$

que possui como solução $c_1=\dfrac{-b(2m_2+m_1)}{(m_1+m_2-1)^2}$, $c_2=\dfrac{-b}{(m_1+m_2-1)}$,

$$
c_3=\frac{-b\left(m_1^3-3m_1^2+2m_1^2m_2-4m_1m_2+m_1m_2^2+m_1-2m_2-2m_2^2+\sqrt{m_1^2+4m_2}(-m_1^2+m_1-2m_1m_2-m_2^2-1)\right)}{\left(2\sqrt{m_1^2+4m_2}(m_1+m_2-1)^2\right)}
$$

e $c_4=\dfrac{b\left(-2m_2+m_1-2m_2^2-4m_1m_2-3m_1^2+m_1m_2^2+2m_1^2m_2+m_1^3+\sqrt{m_1^2+4m_2}(m_2^2-m_1+m_1^2+2m_1^2m_2+m_1^3+2m_2m_1+1)\right)}{\left(2\sqrt{m_1^2+4m_2}(m_1+m_2-1)^2\right)}$

.

Finalizando, substitui-se os valores de c_1, c_2, c_3 e c_4 na equação da solução geral (3.98) obtendo a solução exata da equação de recorrência original (3.93) e, conseqüentemente, a complexidade exata do algoritmo, ou seja,

$$
\begin{aligned}
T(n) = &\left(\frac{b}{\left(2\sqrt{m_1^2+4m_2}(m_1+m_2-1)^2\right)}\right)\left(-\sqrt{m_1^2+4m_2}(3m_2+2m_1-1)-\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^n(m_1^3-3m_1^2)\right. \\
&+2m_1^2m_2-4m_1m_2+m_1m_2^2+m_1-2m_2-2m_2^2+\sqrt{m_1^2+4m_2}(-m_1^2+m_1-2m_1m_2-m_2^2-1) \\
&+\left(\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^n(-2m_2+m_1-2m_2^2-4m_1m_2-3m_1^2+m_1m_2^2+2m_1^2m_2+m_1^3+\sqrt{m_1^2+4m_2}(m_2^2-m_1) \\
&\left.+m_1^2+2m_1^2m_2+m_1^3+2m_2m_1+1\right)
\end{aligned}
$$

3.3.4 Comentários

O objetivo deste trabalho é obter a complexidade exata do algoritmo, porém ocorrem certos casos em que não é muito viável obter esta complexidade exata, devido ao grande número de termos no resultado final, como neste caso, subproblema de tamanho

$r(n)=(n-1)+(n-2)$.

Com o método das equações características, pode-se também, determinar a ordem exata da complexidade, não precisando calcular as constantes c_1, c_2, ..., c_n presentes na solução geral da equação de recorrência.

Para obter a ordem exata da complexidade, Brassard [BRA95] substitui a equação da solução geral na equação de recorrência original e verifica qual o termo dominante na equação, que será a ordem de complexidade do algoritmo.

Nº de subprob.	f(n)=b
$m_1 \neq m_2 \neq b$	$T(n)=\frac{b\left(-2\sqrt{m_1^2+4m_2}+\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^n(2m_2+2m_1-m_1m_2-m_1^2+(m_1+m2)\sqrt{m_1^2+4m_2})+\left(\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^n(-2m_2-2m_1+m_1m_2+m_1^2+(m_1+m_2)\sqrt{m_1^2+4m_2})\right)}{\left(2\sqrt{m_1^2+4m_2}(m_1+m_2-1)\right)}$
$m_1=m_2 \neq b$	$T(n)=\frac{b\left(-2m_1-m_1^2+8+(-m_1-4)\sqrt{m_1^2+4m_1}+\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4m_1}\right)^n((8m_1-4)\sqrt{m_1^2+4m_1})+\left(\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4m_1}\right)^n(2m_1^3+4m_1^2-16m_1+(2m_1^2+4)\sqrt{m_1^2+4m_1})\right)}{\left((\sqrt{m_1^2+4m_1}+m_1-2)(2m_1^2+7m_1-4)\right)}$
$m_1=m_2=b$	$T(n)=\frac{b\left(-b^2-2b+8+(-b-4)\sqrt{b^2+4b}+\left(\frac{1}{2}b+\frac{1}{2}\sqrt{b^2+4b}\right)^n((8b-4)\sqrt{b^2+4b})+\left(\frac{1}{2}b-\frac{1}{2}\sqrt{b^2+4b}\right)^n(2b^3+4b^2-16b+(2b^2+4)\sqrt{b^2+4b})\right)}{\left((\sqrt{b^2+4b}+b-2)(2b^2+7b-4)\right)}$
$m_1=m_2=1$	$T(n)=-b+\left(\frac{1}{2}+\frac{1}{2}\sqrt{5}\right)^n\left(b+\frac{1}{5}b\sqrt{5}\right)+\left(\frac{1}{2}-\frac{1}{2}\sqrt{5}\right)^n\left(b-\frac{1}{5}b\sqrt{5}\right)$
$m_1=1$	$T(n)=\frac{\left(-4m_1-\sqrt{4m_2+1}-1+\left(\frac{1}{2}+\frac{1}{2}\sqrt{4m_2+1}\right)^n(2m_2^2+3m_2+1+(m_2+1)\sqrt{4m_2+1})+\left(\frac{1}{2}-\frac{1}{2}\sqrt{4m_2+1}\right)^n(2m_2^2+2m_2)\right)}{(4m_2^2+m_2+m_2\sqrt{4m_2+1})}$
$m_2=1$	$T(n)=\frac{b\left(-m_1^3-4m_1+(-m_1^2-4)\sqrt{m_1^2+4}+\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4}\right)^n(m_1^3+4m_1+m_1^2+4+(m_1^2+3m_1+2)\sqrt{m_1^2+4})+\left(\frac{1}{2}-\frac{1}{2}\sqrt{m_1^2+4}\right)^n(m_1^4+3m_1-4+(m_1^3+m_1+2)\sqrt{m_1^2+4})\right)}{\left((m_1+\sqrt{m_1^2+4})(m_1^3+4m_1)\right)}$

Tabela 3.3: Complexidade exata dos algoritmos com o termo *f(n)=b*, para subproblemas de tamanho r(n)=(n-1)+(n-2).

N° de subprob.	f(n)=b^n
$m_1 \neq m_2 \neq b$	$T(n)=\dfrac{-\left(-2b^{n+2}\sqrt{m_1^2+4m_2}+\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^n(b^3m_1-b^2m_1^2+b^2m_1-bm_2m_1+2bm_2+(-b^3+b^2m_1+b^2+bm_2)\sqrt{m_1^2+4m_2})\right)}{\left(2\sqrt{m_1^2+4m_2}(b^2-bm_1-bm_2)\right)}$ $\dfrac{-\left(\left(\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^n(-b^3m_1-b^2m_1+bm_2m_1+b^2m_1^2-2bm_2+(-b^3+b^2+bm_2+b^2m_1)\sqrt{m_1^2+4m_2})\right)}{\left(2\sqrt{m_1^2+4m_2}(b^2-bm_1-bm_2)\right)}$
$m_1=m_2\neq b$	$T(n)=\dfrac{\left(\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4m_1}\right)^n(5b^3m_1+b^3m_1^2-b^4m_1-bm_1^2-4bm_1+4b^3-4b^4+(b^4+3b^3-b^3m_1-4b^2m_1-3bm_1)\sqrt{m_1^2+4m_1})\right)}{\left((m_1-2b+\sqrt{m_1^2+4m_1})(-m_1^2-bm_1^2-b^2m_1-4m_1-4bm_1+4b^2)\right)}$ $+\dfrac{\left(\left(\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4m_1}\right)^n(-3bm_1^2-bm_1^3+9b^3m_1-3b^2m_1^2+4b^2m_1-b^2m_1^3+2b^3m_1^2-b^4m_1-4b^4+4b^3+4bm_1+(-b^4+b^3-4b^2+2b^3m_1-b^2m_1-bm_1^2-b^2m_1^2-bm_1)\sqrt{m_1^2+4m_1})\right)}{\left((m_1-2b+\sqrt{m_1^2+4m_1})(-m_1^2-bm_1^2-b^2m_1-4m_1-4bm_1+4b^2)\right)}$ $+\dfrac{\left(-8b^{n+3}-2b^{n+3}m_1+4b^{n+2}m_1+b^{n+2}m_1^2+(4b^{n+2}+b^{n+2}m_1)\sqrt{m_1^2+4m_1}\right)}{\left((m_1-2b+\sqrt{m_1^2+4m_1})(-m_1^2-bm_1^2-b^2m_1-4m_1-4bm_1+4b^2)\right)}$
$m_1=m_2=b$	$T(n)=\dfrac{\left(-b^{n+3}-4b^{n+2}+(b^{n+2}+4b^{n+1})\sqrt{b^2+4b}+\left(\frac{b}{2}+\frac{1}{2}\sqrt{b^2+4b}\right)^n(b^3+3b^2-4b+(-b^2-3b)\sqrt{b^2+4b})\left(\frac{b}{2}-\frac{1}{2}\sqrt{b^2+4b}\right)^n(b^3+5b^2+4b+(-b^2-5b)\sqrt{b^2+4b})\right)}{\left((b-\sqrt{b^2+4b})(b+4)\right)}$
$m_1=m_2=1$	$T(n)=\dfrac{-2\left(-10b^{n+2}+\left(\frac{1}{2}+\frac{1}{2}\sqrt{5}\right)^n(-5b^3+b^3\sqrt{5}+10b^2+5b+b\sqrt{5})+\left(\frac{1}{2}-\frac{1}{2}\sqrt{5}\right)^n(-5b^3-b^3\sqrt{5}+10b^2+5b-b\sqrt{5})\right)}{5(2b+\sqrt{5}-1)(2b-\sqrt{5}-1)}$

Nº de subprob.	f(n)=b^n
$m_1=1$	$T(n)=\frac{\left(\left(\frac{1}{2}+\frac{1}{2}\sqrt{4m_2+1}\right)^n(-4b^2m_2^2+4b^4m_2-8b^3m_2+3b^2m_2+b^2-2b^3+b^4+4bm_2^2+bm_2+(-b^4+b^2+3b^2m_2+2bm_2^2-2b^3m_2+bm_2)\sqrt{4m_2+1})\right)}{\left((2b-\sqrt{4m_2+1}-1)(1+4m_2)(b^2-b-m_2)\right)}$ $+\frac{\left(\left(\frac{1}{2}-\frac{1}{2}\sqrt{4m_1+1}\right)^n(-3b^3+4b^4m_2-4b^2m_2^2+b^4+b^2+3b^2m_2-12b^3m_2+(b^4+b^2-b^3+5b^2m_2-2b^3m_2+2bm_2^2)\sqrt{4m_2+1})\right)}{\left((2b-\sqrt{4m_2+1}-1)(1+4m_2)(b^2-b-m_2)\right)}$ $+\frac{\left(-b^{n+2}+2b^{n+3}+8b^{n+3}m_2-4b^{n+2}m_2+(-b^{n+2}-4b^{n+2}m_2)\sqrt{4m_2+1}\right)}{\left((2b-\sqrt{4m_2+1}-1)(1+4m_2)(b^2-b-m_2)\right)}$
$m_2=1$	$T(n)=\frac{\left(\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4}\right)^n(-b^2m_1^2-b^3m_1^3+b^4m_1^2+4b^2m_1+bm_1^2+b^2m_1^3-b^3m_1^2-4b^3m_1+4b^4-4b^3+4b-4b^2+(2b-2b^3+3b^2m_1+b^2m_1^2-b^3m_1+b^3m_1^2-b^4m_1+bm_1)\sqrt{m_1^2+4})\right)}{\left((2b-m_1-\sqrt{m_1^2+4})(m_1^2+4)(b^2-bm_1-1)\right)}$ $+\frac{\left(\left(\frac{m_1-\sqrt{m_1^2+4}}{2}\right)^n(-bm_1^2-4b+bm_1^3-8b^3m_1-b^3m_1^2+3b^2m_1^2+b^2m_1^4+b^4m_1^2+4bm_1-4b^2-4b^3+4b^4-2b^3m_1^3+(b^2m_1-bm_1+b^3m_1+bm_1^2+b^2m_1^3+4b^2+2b-2b^3+b^4m_1-2b^3m_1^2)\sqrt{m_1^2+4}\right)}{\left((2b-\sqrt{4m_2+1}-1)(1+4m_2)(b^2-b-m_2)\right)}$ $+\frac{\left(8b^{n+3}-b^{n+2}m_1^3-4b^{n+2}m_1+2b^{n+3}m_1^2+(-4b^{n+2}-b^{n+2}m_1^2)\sqrt{m_1^2+4}\right)}{\left((2b-\sqrt{4m_2+1}-1)(1+4m_2)(b^2-b-m_2)\right)}$

Tabela 3.4: Complexidade exata dos algoritmos com o termo $f(n)=b^n$, para subproblemas de tamanho r(n)=(n-1)+(n-2).

Nº de subprob.	f(n)=bn
$m_1 \neq m_2 \neq b$	$T(n)=\frac{b\left(-\sqrt{m_1^2+4m_2}(3m_2+2m_1-1)-\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^n(m_1^3-3m_1^2+2m_1^2m_2-4m_1m_2+m_1m_2^2+m_1-2m_2-2m_2^2+\sqrt{m_1^2+4m_2}(-m_1^2+m_1-2m_1m_2-m_2^2-1))\right)}{\left(2\sqrt{m_1^2+4m_2}(m_1+m_2-1)^2\right)}$ $\frac{+b\left(\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4m_2}\right)^n(-2m_2+m_1-2m_2^2-4m_1m_2-3m_1^2+m_1m_2^2+2m_1^2m_2+m_1^3+\sqrt{m_1^2+4m_2}(m_2^2-m_1)+m_1^2+2m_1^2m_2+m_1^3+2m_2m_1+1)}{\left(2\sqrt{m_1^2+4m_2}(m_1+m_2-1)^2\right)}$
$m_1=m_2\neq b$	$T(n)=\frac{-b\left(-8n+6m^2m_1+24m_1+2nm_1^4+7nm_1^3+14nm_1+12m_1^3+3m_1^4+\sqrt{m_{+1^2}+4m_1}(3nm_1^2-18nm_1+2nm_1^3+8n+3m_1^3+6m_1^2-24m_1)\right)}{\left((2m_1-1)^2(m_1+4)(m_1^2+2+\sqrt{m_1^2+4m_1}(m_1-2))\right)}$ $\frac{-b\left(\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4m_1}\right)^n(\sqrt{m_1^2+4m_1}(20m_1^2-8m_1^3-14m_1+3)+7m_1+18m_1^2-8m_1^4-28m_1^3-4)\right)}{\left((2m_1-1)^2(m_1+4)(m_1^2+2+\sqrt{m_1^2+4m_1}(m_1-2))\right)}$ $\frac{-b\left(\left(\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4m_1}\right)^n(\sqrt{m_1^2+4m_1}(m_1^3+4m_1-4m_1^4+13m_1^2+5)-4-4m_1^5-7m_1^4-52m_1^2-m_1+23m_1^3)\right)}{\left((2m_1-1)^2(m_1+4)(m_1^2+2+\sqrt{m_1^2+4m_1}(m_1-2))\right)}$
$m_1=m_2=b$	$T(n)=\frac{-b\left(-8n+2nb^4+7nb^3+14nb+3b^4+12b^3+6b^2+24b+\sqrt{b^2+4b}(8n+9b^2+2nb^3+3nb^2-18nb-24b)\right)}{\left((2b-1)^2(b+4)(b^2+2+\sqrt{b^2+4b}(b-2))\right)}$ $\frac{-b\left(\frac{1}{2}b+\frac{1}{2}\sqrt{b^2+4b}\right)^n(-8b^4-28b^3+18b^2+7b-4+\sqrt{b^2+4b}(-8b^3+20b^2-14b+3))}{\left((2b-1)^2(b+4)(b^2+2+\sqrt{b^2+4b}(b-2))\right)}$ $\frac{-b\left(\frac{1}{2}b-\frac{1}{2}\sqrt{b^2+4b}\right)^n(-4b^5-7b^4+23b^3-52b^2-b-4+\sqrt{b^2+4b}(-4b^4+b^3+13b^2+4b+5))}{\left((2b-1)^2(b+4)(b^2+2+\sqrt{b^2+4b}(b-2))\right)}$
$m_1=m_2=1$	$T(n)=-b(n+3)+\left(\frac{1}{2}+\frac{1}{2}\sqrt{5}\right)^n(3b\frac{\sqrt{5}}{5}+2b)+\left(\frac{1}{2}-\frac{1}{2}\sqrt{5}\right)^n(-3b\frac{\sqrt{5}}{5}+2b)$

Nº de subprob.	f(n)=bn
$m_1=1$	$T(n)=\frac{b\left(-64m_2^3-64m_2^2-20m_2-32nm_2^3-16nm_2^2-2nm_2-2+\sqrt{4m_2+1}\,(32m_2^3+40m_2^2+16m_2+16nm_2^3+12nm_2^2+2nm_2)\right)}{\left(2m_2^2(\sqrt{4m_2+1}-2m_2-1)(4m_2+1)^{3/2}\right)}$ $+b\left(\left(\frac{1}{2}-\frac{1}{2}\sqrt{4m_2+1}\right)^n(24m_2^4+78m_2^3+66m_2^2+20m_2+2+\sqrt{4m_2+1}\,(-8m_2^4-22m_2^3-21m_2^2-8m_2-1)+(4m_2+1)^{3/2}(-m_2^2-4m_2-1))\right)$ $\frac{}{\left(2m_2^2(\sqrt{4m_2+1}-2m_2-1)(4m_2+1)^{3/2}\right)}$ $+b\left(\frac{1}{2}+\frac{1}{2}\sqrt{4m_2+1}\right)^n(8m_2^4+2m_2^3+\sqrt{4m_2+1}\,(-8m_2^4-18m_2^3-4m_2^2))$ $\frac{}{\left(2m_2^2(\sqrt{4m_2+1}-2m_2-1)(4m_2+1)^{3/2}\right)}$
$m_2=1$	$T(n)=\frac{-b\left(2m_1^7+4m_1^6+12m_1^5+24m_1^4-64m_1+2nm_1^7+12nm_1^5-64nm_1-128+\sqrt{m_1^2+4}\,(2m_1^6+4m_1^5+8m_1^4+24m_1^3+16m_1^2+32m_1+26nm_1^6+8nm_1^4+8nm_1^3+32nm_1+64)\right)}{\left(2(m_1^5+4m_1^2+\sqrt{m_1^2+4}\,(m_1^4-2m_1^2))(m_1^2+4)^{3/2}\right)}$ $-b\left(\frac{1}{2}m_1-\frac{1}{2}\sqrt{m_1^2+4}\right)^n(-2m_1^8-10m_1^6-2m_1^5+8m_1^3+64m_1^2+64m_1+128+\sqrt{m_1^2+4}\,(-m_1^7-m_1^6-6m_1^5-8m_1^4-12m_1^3-24m_1^2-16m_1-32)+(m_1^2+4)^{3/2}(-m_1^5+m_1^4+4m_1^3+2m_1^2-4m_1-8)$ $\frac{}{\left(2(m_1^5+4m_1^2+\sqrt{m_1^2+4}\,(m_1^4-2m_1^2))(m_1^2+4)^{3/2}\right)}$ $-b\left(\frac{1}{2}m_1+\frac{1}{2}\sqrt{m_1^2+4}\right)^n(-2m_1^7-6m_1^6-10m_1^5-24m_1^4-8m_1^3+\sqrt{m_1^2+4}\,(2m_1^6-6m_1^5-14m_1^4-24m_1^3-24m_1^2))$ $\frac{}{\left(2(m_1^5+4m_1^2+\sqrt{m_1^2+4}\,(m_1^4-2m_1^2))(m_1^2+4)^{3/2}\right)}$

Tabela 3.5: Complexidade exata dos algoritmos com o termo *f(n)=bn,* para subproblemas de tamanho r(n)=(n-1)+(n-2).

3.4 Subproblema de tamanho r(n)=n/c

Neste caso tem-se recorrências do tipo Divisão-e-Conquista, as quais são as mais importantes e as que ocorrem com maior freqüência em algoritmos desenvolvidos pelo método da Divisão-e-Conquista.

Para resolver a equação de recorrência deve-se, primeiro, realizar a mudança de variável [SED96] de n para 2^i. Escreve-se $T(n)$ em termos da recorrência geral e t_i em termos da nova recorrência obtida após a aplicação da mudança de variável. A nova recorrência, em termos de t_i, é do tipo linear completa de primeira ordem, segundo a Tabela 2.2 de Classificação das Recorrências, definida por [SED96]. Resolve-se esta nova recorrência da mesma maneira como foi resolvido na seção 3.1, para subproblemas de tamanho $r(n)=n-1$, precisando apenas de uma condição inicial.

Após a resolução da recorrência em termos de t_i, deve-se obter a solução em termos de $T(n)$ e, se necessário, realizar mudança de base nos termos das constantes $c_1, c_2, \ldots, c_n$.

Considera-se, em todas as soluções, n potência de dois, c divisível por dois, $m>1$ e $m \neq b$.

No final desta seção, encontra-se a Tabela 3.4 elaborada com as soluções exatas das equações de recorrência, dos casos gerais e particulares.

3.4.1 Desenvolvimento do método para $f(n)$ constante

O algoritmo resolve um problema de tamanho n decompondo-o sucessivamente em subproblemas de tamanho n/c ou $n/2$, obtendo-se m subproblemas. A complexidade do algoritmo, exceto as chamadas recursivas, é constante e a base da recursividade é b quando $n=1$. Isto é,

$$T(n)=(n=1 \rightarrow b, b+mT(\frac{n}{2})) \qquad (3.99)$$

Inicia-se a resolução da equação de recorrência (3.99) realizando a mudança de variável n por 2^i, obtendo, assim, uma nova recorrência em termos de $t_i=T(2^i)$. Esta mudança de variável é muito útil pois $n/2$ torna-se $(2^i)/2=2^{i-1}$, ou seja, a recorrência original em $T(n)$ é definida como uma função de $T(n/2)$, sendo que t_i é definido como função de t_{i-1} originando um tipo de recorrência já resolvida anteriormente na seção 3.1,

$$t_i = T(2^i) = mT(2^{i-1}) + b$$
$$t_i = mt_{i-1} + b \qquad (3.100)$$

que pode ser reescrita como

$$t_i - mt_{i-1} = b \qquad (3.101)$$
$$t_1 = b,$$

obtendo uma recorrência no mesmo formato da equação (3.1). Para homogeneizar a equação (3.101) procede-se de mesma maneira que nos casos anteriores. Substitui-se *i* por *i-1* na equação (3.101), obtendo

$$t_{i-1} - mt_{i-2} = b \quad . \qquad (3.102)$$

Em seguida, subtrai-se a equação (3.102) da equação (3.101) obtendo a equação homogeneizada

$$t_i - (m+1)t_{i-1} + mt_{i-2} = 0 \quad . \qquad (3.103)$$

Substituindo t_i por r^i obtém-se a equação característica:

$$r^{i-2}(r^2 - (m+1)r + m) = 0$$

cujas raízes, não nulas, são $r_1 = m$ e $r_2 = 1$. Tem-se, com isso, a solução geral da equação (3.101), conforme (3.2) é

$$t_i = c_1 m^i + c_2 1^i \quad . \qquad (3.104)$$

Utiliza-se o fato que $T(2^i) = t_i$ e, deste modo, $T(n) = t_{\log_2 n}$ pois $n = 2^i$ e $i = log_2 n$, para obter a solução geral da equação de recorrência (3.99),

$$T(n) = c_1 m^{\log_2 n} + c_2 1^{\log_2 n} \quad . \qquad (3.105)$$

Usando o fato que $m^{log_2 n} = n^{log_2 m}$, obtém-se como solução geral da recorrência (3.99)

$$T(n) = c_1 n^{\log_2 m} + c_2 \quad . \qquad (3.106)$$

Substituindo $n = 1$ e $n = 2$ na solução geral, equação (3.106), obtém-se um sistema de equações que possui como termos independentes a condição iniciais *T(1)* já conhecida, e *T(2)* que pode ser calculado substituindo $n = 2$ na equação de recorrência (3.99) obtendo $T(2) = b + bm$. Assim, o sistema é:

$$T(1) = c_1 \quad + c_2 = b$$
$$T(2) = c_1 m + c_2 = b(1+m)$$

cuja solução é $c_1 = \dfrac{bm}{(m-1)}$ e $c_2 = \dfrac{-b}{(m-1)}$.

Substituindo os valores de c_1 e c_2 encontrados na solução geral, (3.106), tem-se a solução

exata da equação de recorrência (3.99) e, conseqüentemente, a complexidade exata do algoritmo desenvolvido por Divisão-e-Conquista:

$$T(n)=\frac{b(m n^{\log_2 m}-1)}{(m-1)} \quad .$$

3.4.2 Desenvolvimento do método para *f(n)* exponencial

O algoritmo resolve um problema de tamanho *n* decompondo-o sucessivamente em subproblemas de tamanho *n/2*, obtendo-se *m* subproblemas. A complexidade do algoritmo, exceto as chamadas recursivas, é exponencial e a base da recursividade é *b* quando *n=1*. Isto é,

$$T(n)=(n=1 \rightarrow b,\ b^n+mT(\frac{n}{2})) \quad . \tag{3.107}$$

Como já realizado no caso anterior, realiza-se a mudança de variável substituindo *n* por 2^i obtendo uma nova recorrência em termos de $t_i=T(2^i)$, onde *n/2* torna-se 2^{i-1}, ou seja, a recorrência original em *T(n)* é definida em função de *T(n/2)*, sendo que t_i é definido em função de t_{i-1}:

$$\begin{aligned} t_i&=T(2^i)=mT(2^{i-1})+b^{2^i} \\ t_i&=mt_{i-1}+b^i \end{aligned} \tag{3.108}$$

reescrevendo, tem-se

$$\begin{aligned} t_i-mt_{i-1}&=b^i \\ t_1&=b. \end{aligned} \tag{3.109}$$

Como a equação não está na forma homogênea, multiplica-se todos os termos da equação (3.109) por *b* (base da exponencial) e substitui-se *i* por *i-1*, obtendo

$$bt_{i-1}-bmt_{i-2}=bb^{i-1} \quad . \tag{3.110}$$

Subtrai-se a equação (3.110) da equação (3.109) e obtém-se a equação homogeneizada

$$t_i-(b+m)t_{i-1}+bmt_{i-2}=0 \quad . \tag{3.111}$$

A equação característica associada é:

$$r^{i-2}(r^2-(b+m)r+bm)=0 \quad ,$$

cujas raízes não nulas são $r_1=m$ e $r_2=b$. Com estas raízes obtém-se a solução geral da equação (3.109), de acordo com (3.2):

$$t_i=c_1 m^i+c_2 b^i \quad . \tag{3.112}$$

Como $T(2^i)=t_i$ e, conseqüentemente, $T(n)=t_{\log_2 n}$ (pois $n=2^i$) tem-se a solução geral da equação de recorrência (3.107),

$$T(n)=c_1 m^{\log_2 n}+c_2 b^{\log_2 n} \quad . \tag{3.113}$$

Mudando as bases das exponenciais obtemos (já que $m^{log_2 n}=n^{log_2 m}$ e $b^{log_2 4}=n^{log_2 b}$)

$$T(n)=c_1 n^{\log_2 m}+c_2 n^{\log_2 b} \quad . \tag{3.114}$$

Substituindo $n=1$ e $n=2$ na equação (3.114), obtém-se o sistema de equações onde os termos independentes são a condição inicial$T(1)$ e $T(2)$, que é calculado substituindo $n=2$ na equação de recorrência (3.107) obtendo $T(2)=b^2+bm$. Assim, tem-se

$$\begin{aligned} T(1)&=c_1 \quad +c_2 \quad =b \\ T(2)&=c_1 m+c_2 b=b(b+m) \end{aligned}$$

que possui como solução $c_1=\dfrac{-bm}{(b-m)}$ e $c_2=\dfrac{b^2}{(b-m)}$.

Substituindo a solução do sistema de equações na solução geral, (3.114), obtém-se a solução exata da equação de recorrência original, (3.107), e a complexidade exata do algoritmo:

$$T(n)=\frac{b^2(n^{\log_2 b})-bm(n^{\log_2 m})}{(b-m)} \quad .$$

3.4.3 Desenvolvimento do método para $f(n)$ polinomial

O algoritmo resolve um problema de tamanho n dividindo-o sucessivamente em subproblemas de tamanho $n/2$, obtendo-se m subproblemas. A complexidade do algoritmo, exceto as chamadas recursivas, é polinomial de ordem s. Isto é,

$$T(n)=(n=1 \rightarrow b,\ bn^s+mT(\frac{n}{2})) \quad . \tag{3.115}$$

Para demonstração do método, considera-se $s=1$, obtendo uma complexidade linear. Os resultados calculados para $s=2$ e $s=3$ encontram-se na Tabela 3.4, onde tem-se complexidade polinomial de grau s.

Resolve-se a equação de recorrência (3.115) realizando a mudança de variável, de n para 2^i, obtendo uma nova recorrência em termos de $t_i=T(2^i)$. Com esta mudança, $n/2$ torna-se 2^{i-1} e a recorrência original em $T(n)$ fica :

$$t_i = T(2^i) = mT(2^{i-1}) + b2^i$$
$$t_i = mt_{i-1} + 2^i b \quad . \tag{3.116}$$

Reescrevendo, tem-se

$$t_i - mt_{i-1} = 2^i b \tag{3.117}$$
$$t_1 = b.$$

Como a equação de recorrência (3.117) não é homogênea, multiplica-se todos os termos da equação (3.117) por *2* e substitui-se *i* por *i-1*, obtendo

$$2t_{i-1} - 2mt_{i-2} = 22^{i-1} b \quad . \tag{3.118}$$

Em seguida, subtrai-se a equação (3.118) da equação (3.117), obtendo a equação homogeneizada

$$t_i - (m+2)t_{i-1} + 2mt_{i-2} = 0 \quad . \tag{3.119}$$

Coloca-se a equação (3.119) no formato (3.3) obtendo-se a equação característica

$$r^{i-2}(r^2 - (m+2)r + 2m) = 0$$

cujas raízes não nulas são $r_1 = m$ e $r_2 = 2$. Com estas raízes características obtém-se a solução geral da equação (3.117),

$$t_i = c_1 m^i + c_2 2^i \quad . \tag{3.120}$$

Como $T(2^i) = t_i$ e, em conseqüência, $T(n) = t_{\log_2 n}$ pois $n = 2^i$ e $i = \log_2 n$, tem-se a solução geral da equação de recorrência (3.115),

$$T(n) = c_1 m^{\log_2 n} + c_2 2^{\log_2 n} \quad . \tag{3.121}$$

Neste ponto, realiza-se a mudança de base da exponencial, obtendo a solução geral da recorrência original (3.115):

$$T(n) = c_1 n^{\log_2 m} + c_2 n \quad . \tag{3.122}$$

Para se obter os valores de c_1 e c_2, substitui-se $n=1$ e $n=2$ na equação (3.122) obtendo um sistema de equações, com termos independentes, *T(1)*e *T(2)*, sendo *T(1)* conhecida. Calcula-se *T(2)* substituindo $n=2$ na equação de recorrência (3.115), obtendo $T(2) = 2b + bm$. Assim o sistema fica:

$$T(1) = c_1 + c_2 = b$$
$$T(2) = c_1 m + 2c_2 = b(m+2)$$

que tem como solução $c_1 = \frac{bm}{(m-2)}$ e $c_2 = \frac{-2b}{(m-2)}$.

Finalizando, substitui-se a solução do sistema de equações, c_1 e c_2, na solução geral da equação de recorrência (3.115), equação (3.122), obtendo a solução exata da recorrência original (3.115), e a complexidade exata do algoritmo desenvolvido pelo método da Divisão-e-Conquista:

$$T(n)=\frac{b\left(m\left(n^{\log_2 m}\right)-2n\right)}{(m-2)} \quad .$$

3.4.4 Estudo de Casos

3.4.4.1 Algoritmo *Ray-tracing*

O método *Ray-tracing* [KAY79,WHI79], baseia-se na simulação da interação da luz com os objetos, permitindo obter os efeitos de reflexão, transparência e sombras.

A cena é modelada através de informações que representam os objetos existentes, as fontes de luz e a posição do observador. A imagem é sintetizada sobre um plano de sintetização colocado entre o observador e os objetos. Define-se um raio por dois vetores no espaço euclidiano tridimensional, representando a origem e direção.

Os raios são disparados do observador em direção aos objetos. No pior caso, cada raio é testado contra todos os objetos à procura de intersecções; se estas não ocorrerem, a cor do ponto pelo qual o raio passou, é igual à cor escolhida como cor de fundo. Ocorrendo intersecções, é selecionado o ponto de intersecção mais próximo do ponto de origem do raio. O objeto ao qual pertence o ponto de intersecção é utilizado para a determinação dos raios refletido e transmitido. Para cada um destes raios, é repetida a determinação das intersecções, até que o raio não intercepte nenhum objeto. Ao final, representa o tráfego da luz dentro da cena até alcançar o ponto no plano de sintetização.

A fim de se acelerar o cálculo das intersecções, o algoritmo proposto em [CUN88] utiliza uma subdivisão espacial isométrica do espaço ocupado pelos polígonos. A estrutura de dados resultante de tal subdivisão (uma "octree") é empregada para determinar o caminho que um raio percorre dentro da cena.

A subdivisão espacial isométrica consiste na divisão sucessiva de volumes do espaço, com as mesmas dimensões. Inicialmente, o espaço é visto como um conjunto de 8 volumes, ou octantes primários.

Os octantes são subdivididos em novos conjuntos de oito octantes sempre que o número máximo de polígonos permitido em cada octante for excedido.

Sempre que um octante é dividido, são criados seus oito octantes filhos, e os polígonos que estavam atribuídos a ele devem ser testados contra cada um dos octantes filhos, de forma a se redistribuir os polígonos entre eles. A lista de polígonos do octante pai é destruída ao final da redistribuição dos polígonos entre os octantes filhos.

Ao final do processo de subdivisão, existe uma lista de polígonos atribuídos a cada octante.

Algoritmo Ray-tracing (r, n, o, c)
para i=1 até n
    procedimento localiza_intersecções (r,o)
    "guarda a intersecção mais próxima de *o*";
fim-para
se "houve intersecção" então
    "calcula o raio transmitido *T* e refletido *S*, sobre o objeto para
    o qual houve a intersecção mais próxima";
    chama *Ray-tracing* (S, c_s);
    chama *Ray-tracing* (T, c_t);
    c=cor do objeto +c_s+c_t;
fim-se.
Procedimento *localiza_intersecções* (r, o)
    "testa um raio apenas contra o objeto que se encontra em seu caminho";
    "divide os octantes até que o número de objetos em um nó é menor ou igual
    a um máximo pré-especificado *a*";
fim.

O procedimento *localiza_intersecções* tem pior caso quando os oito nós filhos são subdivididos novamente.

O algoritmo *Ray-tracing* realiza apenas uma chamada recursiva de mesmo tamanho, ou seja, com o mesmo número de objetos por octante, não enquadrando-se em um algoritmo desenvolvido por Divisão-e-Conquista. Já o procedimento *localiza_intersecções* divide o problema em oito subproblemas de tamanho $r(n)=v(n)/8=n`/8$ e, se for preciso, subdivide novamente em oito e assim sucessivamente, o que origina uma equação de recorrência.

Da análise do algoritmo *Ray-tracing*, verifica-se que a complexidade é dada por

$$T(n)=nc_{localiza(v(n))}+nc_{guardar(v(n))}+c_{(calcula\ novo\ raio)(v(n))}.$$

Primeiro calcula-se a complexidade de $c_{(localiza)(v(n))}$, chamada $T`(v(n))$, chama-se $v(n)$

o número máximo de triângulos por octante, representado por $n`$ para simplificar a resolução da equação de recorrência, e n é o total de triângulos do problema, a qual origina a seguinte recorrência:

$$T'(v(n))=(n=1\rightarrow 0, b+8T'(\frac{v(n)}{8})) \tag{3.123}$$

onde $f(n)=b$, $m=8$, $c=8$, $r(n)=v(n)/8=n`/8$, sendo $n`$ de potência 8.

Como o problema é sempre dividido em 8 realiza-se a mudança de variável, fazendo $n`=8^i$, obtendo uma nova recorrência em termos de $t_i=T(8^i)$, sendo $t_i$ definido em função de $t_{i-1}$. Nesta caso $n`/8$ torna-se $(8^i)/8=8^{i-1}$, assim

$$t_i=T'(8^i)=8T'(8^{i-1})+b$$
$$t_i=8t_{i-1}+b \ , \tag{3.124}$$

que pode ser reescrito como

$$t_i-8t_{i-1}=b \tag{3.125}$$
$$t_1=0.$$

Com esta mudança de variável, obtém-se uma recorrência linear de primeira ordem, que pode ser resolvida de mesmo modo que na seção 3.1.1 (subproblema de tamanho $r(n)=n-1$). Sendo assim, substitui-se i por $i-1$ na equação (3.125), obtendo

$$t_{i-1}-8t_{i-2}=b \ , \tag{3.126}$$

e depois subtrai-se a equação (3.126) da equação (3.125), obtendo a equação homogeneizada

$$t_i-9t_{i-1}+8t_{i-2}=0 \ . \tag{3.127}$$

A equação característica é:

$$r^{i-2}(r^2-9r+8)=0$$

que possui raízes não nulas $r_1=8$ e $r_2=1$. Com estas raízes características obtém-se a solução geral da equação (3.125), ou seja,

$$t_i=c_1 8^i+c_2 \ . \tag{3.128}$$

Sabendo que $T`(8^i)=t_i$, tem-se $T'(n')=t_{\log_8 n}$ pois $n`=8^i$ e $i=\log_8 n$. Com isso obtém-se a solução geral da equação (3.123),

$$T'(n')=c_1 8^{\log_8 n}+c_2$$

ou,

$$T'(n')=c_1 n'+c_2 \ . \tag{3.129}$$

Neste caso necessita-se de $T(1)=0$ já conhecida e a próxima, $T(8)$, obtida substituindo $n`=8$

na equação (3.123). Assim, substituindo $n=1$ e $n=8$ na equação (3.129) tem-se o seguinte sistema de equações,

$$T(1)= c_1 + c_2=0$$
$$T(8)=8c_1 + c_2=b,$$

que possui como solução $c_1=b/7$ e $c_2=-b/7$. Substituindo c_1 e c_2, na solução geral (3.129), obtém-se a solução exata da equação de recorrência (3.123) e a complexidade exata do procedimento *localiza_intersecções*. Assim,

$$T'(n')=\frac{b(n'-1)}{7} \quad .$$

Pode-se concluir, também, que $T`(v(n))$ é de ordem $0(v(n))$, considerando $v(n)$ em função de n.

Na complexidade de $T(n)$ do algoritmo *Ray-tracing*, conclui-se que

$$T(n)=n\left(\frac{b(v(n))-b}{7}\right)+nc+c$$

$$T(n)=\frac{bn(v(n)-1)+7c(n+1)}{7} \quad ,$$

ou seja,

$T(n)$ é de ordem $0(n)$.

3.4.4.2 Multiplicação de dois inteiros

O algoritmo clássico de multiplicação INT_MULTC [WIN96] procede multiplicando todos os dígitos da primeira entrada por todos os dígitos da segunda entrada e adicionando os resultados após mudança adequada. A complexidade de INT_MULTC é proporcional ao produto do tamanho das duas entradas, e se as entradas são de mesmo tamanho n, então a complexidade de INT_MULTC é proporcional a n^2. Aqui, define-se o tamanho de um inteiro como sendo o número de dígitos inteiros na base β.

Um algoritmo de multiplicação mais rápido foi desenvolvido por A. Karatsuba e Yu. Ofman (1962). A idéia básica no algoritmo Karatsuba é reduzir as duas entradas x, y de tamanho $\leq n$ em partes de tamanho $\leq n/2$ de modo que

$$x=a.\beta^{n/2}+b \quad , \quad y=c.\beta^{n/2}+d \quad . \qquad (3.130)$$

Uma aproximação de Divisão-e-Conquista poderia reduzir o produto de dois inteiros de tamanho n para quatro produtos de inteiros de tamanho $n/2$. A complexidade

deste algoritmo poderia ainda ser $0(n^2)$. Karatsuba e Ofman, contudo, observam que uma das quatro multiplicações pode ser dispensada.

$$x.y = ac\,\beta^n + (ad+bc)\,\beta^{n/2} + bd$$

$$x.y = ac\,\beta^n + ((a+b)(c+d) - ac - bd)\,\beta^{n/2} + bd \quad . \tag{3.131}$$

Assim, três multiplicações de inteiros de tamanho $n/2$, poucos deslocamentos e adições são suficientes para calcular o produto xy.

No algoritmo de Karatsuba, INT_MULTK, omite-se os sinais dos inteiros.

```
Algoritmo INT_MULTK(entrada: x, y; saída: z)
[x, y inteiros; z=x.y]
n:=max(tamanho(x), tamanho(y));
se n=1 então
        {z:=INT_MULTC(x,y); retorna};
se n é ímpar então
         n:=n+1;
        (a,b):=(Remove(n/2,x), Inicializa(n/2,x);
        (c,d):=(Remove(n/2,y), Inicializa(n/2,y);
        u:=INT_MULTK(a+b, c+d);
        v:=INT_MULTK(a,c);
        w:=INT_MULTK (b,d);
        z:= v β^n + (u − v − w) β^(n/2) + w ;
retorna.
```

TEOREMA 3.1: A complexidade do algoritmo Karatsuba INT_MULTK é $0(n^{\log_2 3})$, onde n é o tamanho das entradas.

PROVA: Inicialmente assume-se que n é potência de 2. Sejam x e y inteiros de tamanhos não maiores que n, e sejam a, b, c, d partes de x, y como em (3.130). Durante a execução do algoritmo Karatsuba, tem-se que calcular os produtos (a+b)(c+d), ac, bd. Todas as outras operações são de adição e deslocamento, as quais tomam tempo proporcional a n. Os fatores ac e bd são de tamanhos não maiores que $n/2$, considerando que os fatores $(a+b)(c+d)$ podem ser de tamanho $n/2+1$. Escreve-se os fatores como

$$a+b = a_1\,\beta^{n/2} + b_1 \quad , \quad c+d = c_1\,\beta^{n/2} + d_1 \quad , \tag{3.132}$$

onde a_1 e c_1 são os principais dígitos de $a+b$ e $c+d$, respectivamente. Agora

$$(a+b)(c+d) = a_1 c_1\,\beta^n + (a_1 d_1 + b_1 c_1)\,\beta^{n/2} + b_1 d_1 \quad . \tag{3.133}$$

No produto b_1d_1 os fatores são de tamanho não maior que $n/2$. Todas as outras operações são multiplicações por um simples dígito ou deslocamento, e juntas elas tem complexidade proporcional a n.

Sendo assim, denotando o tempo de multiplicação de dois inteiros de tamanho n por $T(n)$, obtém-se a equação de recorrência

$$T(n)=(n=1\rightarrow k,\ kn+3T(\frac{n}{2}))\ , \tag{3.134}$$

onde $f(n)=kn$, $m=3$, $r(n)=n/2$. A constante k é o limite da complexidade da multiplicação de dígitos bem como o fator constante na função da complexidade das operações adição e deslocamento.

Resolve-se a equação de recorrência (3.134) como segue:

Realiza-se uma mudança de variável n por 2^i, originando uma nova recorrência em termos de t_i, definida por $t_i=T(2^i)$. A transformação faz com que $n/2=2^{i-1}$. Deste modo, a recorrência em $T(n)$ é definida como uma função de $T(n/2)$, assim

$$t_i=T(2^i)=3T(2^{i-1})+2^ik\ . \tag{3.135}$$

Reescrevendo em termos de t_i, tem-se

$$t_i-3t_{i-1}=2^ik \tag{3.136}$$

$$t_0=k.$$

Multiplica-se todos os termos da equação (3.136) por 2 e substitui-se i por $i-1$, para homogeneizar a equação (3.136). Assim obtém-se

$$2t_{i-1}-6t_{i-2}=2.2^{i-1}k \tag{3.137}$$

e subtrai-se a equação (3.137) da equação (3.136), obtendo a equação homogeneizada

$$t_i-5t_{i-1}+6t_{i-2}=0\ . \tag{3.138}$$

Colocando no formato (3.3) obtém-se a equação característica

$$r^{i-2}(r^2-5r+6)=0$$

cujas raízes não nulas são $r_1=3$ e $r_2=2$. Com as raízes características obtém-se a solução geral da equação (3.136), isto é,

$$t_i=c_13^i+c_22^i\ . \tag{3.319}$$

Considerando que $T(2^i)=t_i$ e, em conseqüência, $T(n)=t_{\log_2 n}$ pois $n=2^i$ e $i=log_2n$, tem-se a solução geral da equação de recorrência (3.134),

$$T(n)=c_1 3^{\log_2 n}+c_2 2^{\log_2 n}$$

$$T(n)=c_1 3^{\log_2 n}+c_2 n \quad . \tag{3.140}$$

Realiza-se uma mudança na base da exponencial no termo de c_1, obtendo

$$T(n)=c_1 n^{\log_2 3}+c_2 n \quad . \tag{3.141}$$

Neste caso, necessita-se de $T(1)=k$ e de $T(2)$ que é obtido substituindo $n=2$ na equação de recorrência (3.134). Desta forma obtém-se $T(2)=5k$ e, substituindo $n=1$ e $n=2$ na equação (3.141), obtém-se o seguinte sistema de equações:

$$T(1)=c_1 \quad +c_2=k$$
$$T(2)=3c_1+2c_2=5k$$

que possui como solução $c_1=3k$ e $c_2=-2k$. Substituindo os valores de c_1 e c_2 na solução geral (3.141), obtém-se a solução exata da recorrência (3.134) e a complexidade exata do algoritmo Karatsuba INT_MULTK, ou seja,

$$T(n)=3kn^{\log_2 3}-2kn \quad .$$

3.4.4.3 Multiplicação de matrizes

Considera-se o problema da multiplicação de duas matrizes A e B, ambas de dimensão nxn, resultando a matriz produto C. A matriz resultante $C=AB$ é, por definição uma matriz nxn cujo elemento C_{ij}, da linha i e coluna j é obtido a partir da i-ésima linha de A e da j-ésima coluna de B, através do somatório

$$c_{ij}=\sum_{k=1}^{n} a_{ik}.b_{kj} \quad . \qquad \text{para } 1\leq i, j\leq n \tag{3.142}$$

Cada elemento C_{ij} requer n multiplicações e $(n-1)$ adições e como, a matriz C possui n^2 destes elementos, são necessários n^3 multiplicações para se obter C.

Este problema pode ser resolvido por um algoritmo baseado no método da Divisão-e-Conquista. Sabe-se que o tempo de execução do algoritmo é de $0(n^{\log_2 7})$, o que é aproximadamente $0(n^{2.81})$. A idéia do algoritmo consiste em particionar as matrizes A e B em quatro submatrizes quadradas, tendo cada uma delas dimensão $(n/2)x(n/2)$ (supondo n potência de dois) conforme o esquema abaixo:

$$\begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} x \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix} = \begin{pmatrix} C_{11} & C_{12} \\ C_{21} & C_{22} \end{pmatrix}$$

onde $C_{ij}=A_{i1}.B_{1j}+A_{i2}.B_{2j}$ para i e j pertencentes a (1,2).

Se $n=2$, as submatrizes tem dimensão 1x1 e o produto AB pode ser calculado diretamente pela fórmula de C_{ij}, por multiplicação de elementos.

Para $n>2$, os cálculos dos C_{ij} necessitam de multiplicações de submatrizes $(n/2)x(n/2)$. Como $n/2$ é também uma potência de dois, a multiplicação das submatrizes pode ser feita pela aplicação recursiva do particionamento em quatro submatrizes (de dimensão $(n/4)x(n/4)$). Recursivamente, estes particionamentos sucessivos resultarão em vários casos de multiplicação de matrizes 2x2, que podem ser efetuados diretamente.

O algoritmo recursivo descrito acima requer oito multiplicações e quatro somas de matrizes $(n/2)x(n/2)$ resultando na equação de recorrência

$$T(n)=cn^2+8T(n/2), \tag{3.143}$$

onde o termo $f(n)=cn^2$ refere-se aos laços de controle da linha e da coluna da matriz.

No mesmo algoritmo, as multiplicações C_{ij} podem ser substituídas por adições resultando em novas fórmulas C_{ij}, o que resulta na equação de recorrência

$$T(n)=(n=2\rightarrow a,\ bn^2+7T(\frac{n}{2})) \ . \tag{3.144}$$

Para resolver a equação de recorrência (3.144) realiza-se uma mudança de variável, substituindo n por 2^i, obtendo uma nova recorrência em termos de $t_i=T(2^i)$. Com esta transformação $n/2=2^{i-1}$, t_i é definido em função de t_{i-1}, resultando numa recorrência linear de primeira ordem:

$$t_i=T(2^i)=7T(2^{i-1})+b(2^i)^2$$

$$t_i=7t_{i-1}+b4^i \ . \tag{3.145}$$

Esta recorrência pode ser reescrita como,

$$t_i-7t_{i-1}=b4^i \tag{3.146}$$

$$t_1=a.$$

Para homogeneizar a equação (3.146) procede-se como nos casos anteriores, quando $f(n)$ é exponencial: substitui-se i por $i-1$ e multiplica-se todos os termos da equação (3.146) por 4, obtendo

$$4t_{i-1}-28t_{i-2}=4b4^{i-1} \ . \tag{3.147}$$

Em seguida, subtrai-se a equação (3.147) da equação (3.146), obtendo a equação homogeneizada

$$t_i-11t_{i-1}+28t_{i-2}=0 \ . \tag{3.148}$$

A equação característica associada, é, portanto

$$r^{i-2}(r^2-11r+28)=0$$

cujas raízes não nulas são $r_1=7$ e $r_2=4$. Com as raízes características constrói-se a solução geral da equação (3.146):

$$t_i=c_1 7^i+c_2 4^i \quad . \tag{3.149}$$

Considerando que $T(2^i)=t_i$ e assim $T(n)=t_{\log_2 n}$ pois $n=2^i$ e $i=log_2 n$, obtém-se a solução geral da equação de recorrência (3.144),

$$T(n)=c_1 7^{\log_2 n}+c_2 n^2 \quad . \tag{3.150}$$

Realiza-se mudança de base no termo c_1, obtendo

$$T(n)=c_1 n^{\log_2 7}+c_2 n^2 \quad . \tag{3.151}$$

Neste caso necessita-se da condição inicial $T(1)$ já fornecida pelo problema, e $T(2)$ que é calculado substituindo $n=2$ na equação de recorrência (3.144), obtendo $T(2)=7a+4b$. Substituindo $n=1$ e $n=2$ na equação da solução geral, (3.151), obtém-se o seguinte sistema de equações:

$$\begin{aligned} T(1) &= c_1+c_2=a \\ T(2) &= 7c_1+4c_2=7a+4b \end{aligned}$$

que possui como solução $c_1=\dfrac{3a+4b}{3}$ e $c_2=\dfrac{-4b}{3}$.

Para obter a solução exata da equação de recorrência (3.144) e a complexidade exata do algoritmo de multiplicação de matrizes, substitui-se os valores das constantes c_1 e c_2 na equação da solução geral (3.151), obtendo

$$T(n)=\frac{(3a+4b)(n^{\log_2 7})-4bn^2}{3} \quad .$$

Nº de subprob.	f(n)=b	f(n)=b^n	f(n)=bn
m ≠ b	$T(n)=\frac{b\left(n^{\log_2 m}-1\right)}{(m-1)}$	$T(n)=\frac{b^2\left(n^{\log_2 b}\right)-bm\left(n^{\log_2 m}\right)}{(b-m)}$	$T(n)=\frac{b\left(m\left(n^{\log_2 m}\right)-2n\right)}{(m-2)}$
m = b	$T(n)=\frac{b\left(bn^{\log_2 b}-1\right)}{(b-1)}$	$T(n)=b\left(n^{\log_2 b}+\log_2 n\left(n^{\log_2 b}\right)\right.$	$T(n)=\frac{b^2\left(n^{\log_2 b}\right)-2bn}{(b-2)}$
m = 1	$T(n)=b(\log_2 n+1)$	$T(n)=\frac{b^2\left(n^{\log_2 b}\right)-b}{(b-1)}$	$T(n)=b(2n-1)$

Nº de subprob.	f(n)=bn²	f(n)=bn³
m ≠ b	$T(n)=\frac{b\left(m\left(n^{\log_2 m}\right)-4n^2\right)}{(m-4)}$	$T(n)\frac{b\left(m\left(n^{\log_2 m}\right)-8n^3\right)}{(m-8)}$
m = b	$T(n)=\frac{b\left(b\left(n^{\log_2 b}\right)-4n^2\right)}{(b-4)}$	$T(n)=\frac{b\left(b\left(n^{\log_2 b}\right)-8n^3\right)}{(b-8)}$
m = 1	$T(n)=\frac{b(4n^2-1)}{3}$	$T(n)=\frac{b(8n^3-1)}{7}$

Tabela 3.6: Complexidade exata dos algoritmos para subproblemas de tamanho r(n)=n/2.

3.5 Considerações finais

Em todos os casos resolvidos neste trabalho, considerou-se três casos para o termo *f(n)* (primeiro termo da equação de recorrência): constante, exponencial e polinomial de grau maior que zero. Porém este termo, *f(n)*, pode apresentar-se de forma diferente, como: n^{log3}, $n^2 log\ n$, $n\ log\ n$, $n\ log^2 n$, ou outros.

Quando *f(n)* ocorre de modo diferente dos três casos estudados, na resolução da equação de recorrência procede-se de maneira semelhante aos casos anteriores já calculados, ou seja, considera-se, por exemplo, a equação de recorrência

$$T(n)=(n<2 \rightarrow b, n\log_2 n+2T(\frac{n}{2})) \quad . \tag{3.152}$$

Com *n* potência de dois, *b=b*, *f(n)= n log₂ n*, *m=2* e *r(n)=n/2*. Realiza-se a mudança de variável trocando *n* por 2^i e $i=log_2 n$, obtendo uma nova recorrência em termos de $t_i=T(2^i)$, ou seja,

$$t_i=T(2^i)=2T(2^{i-1})+i2^i$$

$$t_i=2t_{i-1}+i2^i \quad . \tag{3.153}$$

Reescrevendo a equação (3.153), obtém-se

$$t_i-2t_{i-1}=i2^i \tag{3.154}$$

$$t_1=b.$$

Homogeneiza-se a equação (3.154) da seguinte maneira: substitui-se *i* por *i-1* e multiplica-se por *2* todos os termos da equação (3.154), obtendo

$$2t_{i-1}-4t_{i-2}=(i-1)2.2^{i-1} \quad . \tag{3.155}$$

Subtrai-se a equação (3.155) da equação (3.154), obtendo

$$t_i-4t_{i-1}+4t_{i-2}=2^i \quad . \tag{3.156}$$

Novamente, substitui-se *i* por *i-1* e multiplica-se por *2* (base da exponencial) todos os termos da equação (3.156), obtendo

$$2t_{i-1}-8t_{i-2}+8t_{i-3}=2.2^{i-1} \quad , \tag{3.157}$$

subtrai-se a equação (3.157) da equação (3.156), obtendo a equação homogeneizada

$$t_i-6t_{i-1}+12t_{i-2}-8t_{i-3}=0 \quad . \tag{3.158}$$

A equação característica associada (3.3), é:

$$r^{i-3}(r^3-6r^2+12r-8)=0$$

que possui como raízes não nulas $r_1=r_2=r_3=2$. Com estas raízes tem-se a solução geral da equação de recorrência (3.154),

$$t_i=c_1 2^i+c_2 i2^i+c_3 i^2 2^i \quad . \tag{3.159}$$

Considerando que $T(2^i)=t_i$ e $T(n)=t_{\log_2 n}$ pois $n=2^i$ e $i=log_2\, n$, reescreve-se a equação (3.159) como $T(n)$,

$$T(n)=c_1 n+c_2 n\log_2 n+c_3 n\log_2^2 n \quad . \tag{3.160}$$

Para obter a solução da recorrência (3.152), necessita-se de $T(1)=b$, $T(2)$ e $T(4)$, que são obtidos substituindo $n=2$ e $n=4$ (pois n é potência de dois) na equação de recorrência (3.152). Logo, tem-se $T(2)=2b+2$ e $T(4)=4b+12$. Substituindo $n=1$, $n=2$ e $n=4$ na solução geral, (3.160), obtém-se o sistema de equações

$$\begin{aligned} T(1)&=c_1 &&=b \\ T(2)&=2c_1+2c_2+2c_3 &&=2b+2 \\ T(4)&=4c_1+8c_2+16c_3 &&=4b+12 \end{aligned}$$

que possui como solução os valores de $c_1=b$, $c_2=c_3=1/2$.

Substituindo os valores das constantes c_1, c_2 e c_3 na equação da solução geral (3.160), obtém-se a solução da equação de recorrência (3.152) e a complexidade exata do algoritmo, ou seja, a quantidade exata de operações efetuadas na resolução do algoritmo, assim,

$$T(n)=bn+\frac{n\log_2 n+n\log_2^2 n}{2} \quad .$$

No caso anterior, seção 3.4, a recorrência dada por $T(n)$, aplicava-se somente quando n era potência de 2, ou, como no exemplo 3.4.4.1 (Algoritmo *Ray-tracing*), n devia ser potência de 8.

Brassard [BRA95] resolve uma das mais importantes recorrências que aparecem na formulação geral de recorrência (seção 2.2.5) que são definidas como recorrências de algoritmos obtidas pelo método de Divisão-e-Conquista. As constantes $n_0\geq 1$, $m\geq 1$, $c\geq 1$ e $s\geq 0$ são inteiras, e b é um número estritamente positivo e real. Seja $T:N\rightarrow R^+$ uma função não decrescente, de modo que

$$T(n)=mT(n/c)+bn^s \qquad n>n_0 \tag{3.161}$$

onde n/n_0 é uma potência exata de c e $n \in \{cn_0, c^2n_0, c^3n_0, \ldots\}$. Neste caso, a mudança de variável apropriada é $n=c^i n_0$. Assim,

$$t_i = T(c^i n_0) = mT(c^{i-1} n_0) + b(c^i n_0)^s$$

ou

$$t_i = mt_{i-1} + bn_0^s c^{is} \quad . \tag{3.162}$$

Reescreve-se esta equação, obtendo

$$t_i - mt_{i-1} = bn_0^s (c^s)^i \quad . \tag{3.163}$$

O lado direito da equação (3.163) é da forma $a^i p(i)$ onde $p(i)=bn_0^s$ é uma constante e $a=c^s$. Deste modo, a equação característica é:

$$r^2 - (m + c^s)r + mc^s = 0$$

cujas raízes são $r_1=m$ e $r_2=c^s$. Disto, tenta-se concluir que a solução geral é da forma

$$t_i = c_1 m^i + c_2 (c^s)^i \quad . \tag{3.164}$$

Escreve-se a equação (3.164) em termos de $T(n)$, considerando que $i=\log_c(n/n_0)$ quando n é de forma apropriada, e conseqüentemente $d^i=(n/n_0)^{\log_b d}$ para valores arbitrários de d. Portanto,

$$T(n) = \left(\frac{c_1}{\log_c m}\right) n^{\log_c m} + \left(\frac{c_2}{n_0^s}\right) n^s \tag{3.165}$$

$$T(n) = c_3 n^{\log_c m} + c_4 n^s \tag{3.166}$$

para novas constantes apropriadas c_3 e c_4. Para conhecer estas constantes, substitui-se a equação (3.166) na equação de recorrência original (3.161), obtendo

$$bn^s = T(n) - mT\left(\frac{n}{c}\right)$$

$$bn^s = c_3 n^{\log_c m} + c_4 n^s - m\left(c_3 \left(\frac{n}{c}\right)^{\log_c m} + c_4 \left(\frac{n}{c}\right)^s\right)$$

$$bn^s = \left(1 - \frac{m}{c^s}\right) c_4 n^s \quad , \tag{3.167}$$

pois $c_3 n^{\log_c m} - mc_3 \left(\frac{n}{c}\right)^{\log_c m} = 0$.

Sendo assim, $c_4 = \frac{b}{\left(1 - m/c^s\right)}$. (3.168)

Em [BRA95] encontra-se somente a análise em notação assintótica. Para expressar *T(n)* em notação assintótica, necessita-se verificar quem é o termo dominante na equação (3.166). Existem três casos a serem considerados, dependendo se m é menor, maior ou igual a c^s.

- Se $m<c^s$ então $c_4>0$ (pois $b>0$) e $s>\log_c m$. Portanto o termo $c_4 n^s$ é o termo dominante da equação (3.166). Conclui-se que *T(n)* é de ordem $\Theta(n^s/(n/n_0)$ com n potência de c). Porém n^s é uma função e *T(n)* é não decrescente por suposição. Logo *T(n)* é de ordem $\Theta(n^s)$.
- Se $m>c^s$ então $c_4<0$ e $\log_c m>s$. O fato de c_4 ser negativo implica que c_3 é positivo. Por outro lado *T(n)* da equação (3.166) implica ser negativo, contrário à especificação de que $T:N\rightarrow R^+$. Portanto o termo $c_3 m^{\log_c m}$ é o termo dominante na equação (3.166). Além disso, $n^{\log_c m}$ é uma função suave e *T(n)* é não decrescente. Logo *T(n)* é de ordem $\Theta(n^{\log_c m})$.
- Se $m=c^s$, surge uma preocupação porque a fórmula para c_4 envolve uma divisão por zero. O que acontece é que neste caso o polinômio tem uma única raiz distinta de multiplicidade 2, que é c^s . Assim, a solução geral neste caso é

$$t_i = c_5 (c^s)^i + c_6 i (c^s)^i \quad . \tag{3.169}$$

Em termos de *T(n)* fica,

$$T(n) = c_7 n^s + c_8 n^s \log_c (n/n_0) \tag{3.170}$$

para constantes apropriadas c_7 e c_8. Substituindo na recorrência original, equação (3.162), com a manipulação já realizada anteriormente, obtém-se $c_8=b$. Logo, $bn^s \log_c n/n_0$ é o termo dominante na equação (3.170), pois b foi definido ser estritamente positivo no início do exemplo. Sendo $n^s \log_c n$ suave e *T(n)* não decrescente, conclui-se que *T(n)* é de ordem $\Theta(n^s \log_c n)$.

Colocando todas as soluções juntas, tem-se

$$T(n) \in \begin{cases} \Theta(n^s) & se\ m < c^s \\ \Theta(n^s \log_c n) & se\ m = c^s \\ \Theta(n^{\log_c m}) & se\ m > c^s \end{cases} \quad .$$

4 ANÁLISE DO POLINÔMIO CARACTERÍSTICO

Neste capítulo realiza-se uma análise sobre os polinômios característicos provenientes do processo de homogeneização da equação de recorrência, isto é, verifica-se o motivo pelo qual as raízes deste polinômio mantém o mesmo valor, mesmo aumentando o grau do polinômio presente na equação de recorrência.

Para tanto, fez-se um estudo sobre uma das leis de formação da fórmula do binômio de Newton, Produto de Stevin.

As potências de um binômio obedecem a uma lei de formação pela qual pode-se obter qualquer potência sem passar pelas anteriores e determinar qualquer termo sem conhecer os precedentes.

A lei de sucessão dos termos do desenvolvimento da potência de um binômio, pela qual cada termo se forma do precedente, chama-se *Lei Binomial de Newton.* [FAR64]

Geralmente deduz-se a fórmula de Newton, partindo do Produto de Stevin, que é o produto de fatores de binômios do 1° grau, tendo os primeiros termos iguais e os segundos diferentes, por exemplo, $(x+a)(x+b)$ onde a e b são denominados de termos independentes.

PRODUTO DE STEVIN: Considere o produto

$$(x+a_1)(x+a_2)(x+a_3)(x+a_4)\ldots(x+a_n) \tag{4.1}$$

de n fatores.

Para obter-se a lei de formação deste produto, multiplica-se o primeiro fator pelo segundo, o resultado pelo terceiro e assim sucessivamente, empregando a regra usual da multiplicação de polinômios e ordenando os produtos em relação às potências decrescentes de x. Efetuando as multiplicações, obtém-se que

$$(x+a)(x+b)=x^2+(a+b)x+ab$$

$$(x+a)(x+b)(x+c)=x^3+(a+b+c)x^2+(ab+ac+bc)x+abc$$

$$(x+a)(x+b)(x+c)(x+d)=x^4+(a+b+c+d)x^3+(ab+ac+ad+bc+cd)x^2+(abc+abd+acd+bcd)x+abcd$$

e assim por diante.

Analisando estes produtos, verifica-se que:

1°) O grau de cada polinômio, em relação a x, é igual ao número de fatores. Assim, também cada produto tem tantos termos quantos são os fatores mais um.

2°) O coeficiente de x no termo de maior grau (ou primeiro termo) de cada produto é sempre igual a unidade.

3°) O coeficiente do segundo termo de cada produto é a soma algébrica dos termos independentes dos fatores.

4°) O coeficiente do terceiro termo é a soma algébrica dos produtos binários distintos dos termos independentes dos fatores.

5°) O coeficiente do quarto termo é a soma algébrica dos produtos ternários dos termos independentes dos fatores.

6°) Enfim, o coeficiente de qualquer outro termo é a soma algébrica dos produtos distintos dos termos independentes dos fatores, combinados a uma taxa igual ao número de termos precedentes. Considerando o polinômio ordenado segundo potências decrescentes de x.

7°) O último termo de cada produto é o produto de todos os termos independentes dos fatores.

Designando a soma algébrica dos produtos dos termos independentes dos fatores por S, com um índice que represente a taxa de combinação, tem-se

$$S_1=a+b+c+...$$

$$S_2=ab+ac+...$$

$$S_3=abc+abd+...$$

$$\vdots =....$$

e os produtos considerados podem ser escritos simbolicamente, do seguinte modo

$$(x+a)(x+b)=x^2+S_1x+S_2$$

$$(x+a)(x+b)(x+c)=x^3+S_1x^2+S_2x+S_3$$

$$(x+a)(x+b)(x+c)(x+d)=x^4+S_1x^3+S_2x^2+S_3x+S_4$$

Verifica-se que, em cada termo, a soma do índice de S com o expoente de x é igual ao número de fatores ou ao grau do produto.

De acordo com as observações feitas, pode-se induzir a expressão do produto de n fatores, sendo n um número inteiro e positivo qualquer,

$$(x+a_1)(x+a_2)(x+a_3)\ldots(x+a_n)=$$

$$=x^n+S_1x^{n-1}+S_2x^{n-2}+S_3x^{n-3}+\ldots+S_px^{n-p}+\ldots+S_{n-3}x^3+S_{n-2}x^2+S_{n-1}x+S_n \qquad (4.2)$$

onde S_1, S_2, S_3,..., S_n, representam as somas algébricas das combinações dos termos independentes dos fatores 1 a 1, 2 a 2, 3 a 3,..., n a n.

Para completar a indução, é preciso verificar se a lei de formação do produto de n fatores é verdadeira para $n+1$ fatores. Designando por P o produto indicado dos n fatores e por $x+m$ o novo fator, temos

$$P(x+m)=(x^n+S_1x^{n-1}+S_2x^{n-2}+S_3x^{n-3}+\ldots+S_n)(x+m)=$$

$$=x^{n+1}+(S_1+m)x^n+(S_2+S_1m)x^{n-1}+(S_3+S_2m)x^{n-2}+\ldots+S_nm. \qquad (4.3)$$

Analisando o novo produto, de $n+1$ fatores, verifica-se que a lei de formação é confirmada em relação ao grau e aos coeficientes. Com efeito, o grau do novo produto é $n+1$, igual ao número de fatores; o coeficiente do segundo termo é S_1+m, soma dos termos independentes dos fatores; o do terceiro é S_2+S_1m, soma das combinações 2 a 2 dos termos independentes dos fatores; e assim por diante.

Com base na lei de formação do Produto de Stevin, pode-se dizer que os polinômios característicos, obtidos após o processo de homogeneização da equação de recorrência, seguem esta mesma lei de formação do produto. Como exemplo considera-se $s=2$ (grau do polinômio da equação de recorrência, ou do termo $f(n)=bn^2$) e a equação de recorrência

$$T(n)=(n=0\rightarrow b,\ bn^2+mT(n-1)). \qquad (4.4)$$

Homogeneizando, conforme realizado em todos os casos anteriores, obtém-se a seguinte equação:

$$\begin{gathered}
x_n-mx_{n-1}=bn^2 \\
\hline
x_n-(m+1)x_{n-1}+mx_{n-2}=2bn-b \\
\hline
x_n-(m+2)x_{n-1}+(2m+1)x_{n-2}-mx_{n-3}=2b \\
\hline
x_n-(m+3)x_{n-1}+(3m+3)x_{n-2}-(3m+1)x_{n-3}+mx_{n-4}=0
\end{gathered}$$

Fazendo $x_n=r^n$, obtém-se a seguinte equação característica:

$$r^{n-4}(r^4-(m+3)r^3+(3m+3)r^2-(3m+1)r+m)=0 \qquad (4.5)$$

Para verificar que esta equação equivale a realizar o Produto de Stevin, necessita-se

conhecer os fatores para realizar este produto. Para isto, parte-se da equação de recorrência (4.4), substituindo n por n-1 na mesma equação de recorrência (4.4), obtendo uma nova equação, subtraindo estas duas equações, e colocando em evidência o termo que possui o menor grau, considerando $x_n=r^n$, tem-se

$$(r^n-mr^{n-1})-(r^{n-1}-mr^{n-2})=r^{n-1}(r-m)-r^{n-2}(r-m)$$
$$(r^n-mr^{n-1})-(r^{n-1}-mr^{n-2})=(r-m)(r^{n-1}-r^{n-2})=(r-m)r^{n-2}(r-1)$$
$$(r^n-mr^{n-1})-(r^{n-1}-mr^{n-2})=r^{n-2}(r-m)(r-1)\quad . \tag{4.6}$$

Na equação (4.6), novamente substitui-se n por n-1, obtendo uma nova equação, e subtrai-se estas duas equações, colocando sempre em evidência o termo que possui o menor grau, assim,

$$r^{n-2}(r-m)(r-1)-r^{n-3}(r-m)(r-1)=r^{n-3}r(r-m)(r-1)-r^{n-3}(r-m)(r-1)$$
$$r^{n-2}(r-m)(r-1)-r^{n-3}(r-m)(r-1)=r^{n-3}(r-m)(r-1)(r-1)\quad . \tag{4.7}$$

Finalizando, realiza-se o mesmo processo dos casos anteriores, porém trabalhando com a expressão (4.7), sendo assim,

$$r^{n-3}(r-m)(r-1)^2-r^{n-4}(r-m)(r-1)^2=r^{n-4}r(r-m)(r-1)^2-r^{n-4}(r-m)(r-1)^2$$
$$r^{n-3}(r-m)(r-1)^2-r^{n-4}(r-m)(r-1)^2=r^{n-4}r(r-m)(r-1)^2(r-1)\quad . \tag{4.8}$$

Com base no resultado final deste processo, equação (4.8), verifica-se que os fatores para o Produto de Stevin são $(r$-$m)$ e $(r$-$1)^3$. Realizando este produto, obtém-se o polinômio característico, associado à equação (4.5).

O número de vezes que o processo de homogeneização é repetido é determinado pelo grau do polinômio da equação de recorrência mais um, devido ao termo constante que surge no lado direito da equação de recorrência. Por esse motivo, o número de fatores para realizar o Produto de Stevin depende do grau do polinômio da equação de recorrência. Como neste exemplo, tem-se que $s=2$ (grau do polinômio da equação de recorrência), então tem-se três fatores iguais $(r$-$1)$ mais o fator $(r$-$m)$ da própria equação de recorrência.

De outro modo pode-se aplicar o produto $P(r+m)$, equação (4.3), porém com os fatores conhecidos do exemplo acima:

$$P(r-m)=(r^3+S_1r^2+S_2r+S_3)(r-m)$$
$$P(r-m)=r^4+(-m-S_1)r^3+(-S_1m+S_2)r^2+(-S_2m+S_3)r-S_3m\quad . \tag{4.9}$$

Para calcular os valores das somas algébricas, S_1, S_2 e S_3, tem-se que $a=b=c=-1$, ou seja, $(r+a)(r+b)(r+c)=(r-1)^3$, assim,

$S_1=a+b+c=(-1)+(-1)+(-1)=-3$

$S_2=ab+ac+bc=((-1)(-1))+((-1)(-1))+((-1)(-1))=3$

$S_3=abc=-1.$

Substituindo estes valores das somas algébricas na equação (4.9) obtém-se

$$P(r-m)=r^4-(m+3)r^3+(3m+3)r^2-(3m+1)r+m \quad , \tag{4.10}$$

o qual é equivalente ao polinômio característico (4.5).

Embora os processos, para obter o polinômio característico, sejam diferentes, pode-se concluir que os polinômios característicos obtidos com o processo de homogeneização, realizado neste trabalho, seguem a lei de formação da fórmula do Binômio de Newton, o Produto de Stevin e, por este motivo, à medida que aumenta o grau do polinômio da equação de recorrência, mais precisamente o termo $f(n)=bn^s$, as raízes características continuam com o mesmo valor numérico, aumentando apenas a sua multiplicidade.

Esta análise é válida para todos os casos estudados neste trabalho, ou seja, para tamanhos de subproblemas, variando, conforme o caso, os valores das raízes características.

5 CONCLUSÃO

A complexidade segundo [AHO74] é o coração da computação, o que faz sentido, pois um problema só é tratável se tem um algoritmo polinomial que o resolva [COO83]. Assim a determinação da complexidade de um algoritmo é de extrema importância. A complexidade é calculada caso a caso, e para algoritmos recursivos esse cálculo é potencialmente mais difícil envolvendo provas por indução.

Este trabalho propõe uma solução direta através da redução da equação de complexidade a equações de recorrência homogêneas e posterior solução. Esse método, equações características, possui a vantagem de poder ser usado sem se ter idéia da possível forma da complexidade (não há necessidade de se encontrar uma função candidata) e permite, nos casos de subproblemas de tamanho r(n)=(n-1), r(n)=(n-2) e r(n)=(n/2), a solução exata, isto é, o número exato de operações efetuadas pelo algoritmo, e não somente a ordem de complexidade.

Para a aplicação do método, considera-se quatro tipos de algoritmos recursivos, conforme o tamanho dos subproblemas: $r(n)=(n-1)$, $r(n)=(n-\varepsilon)$, $r(n)=(n-1)+(n-2)$ e $r(n)=n/2$. Para cada tamanho de subproblema calcula-se a complexidade exata, de forma generalizada, considerando três casos para o termo $f(n)$ (primeiro termo da equação de recorrência): constante, exponencial e polinomial de grau maior que zero.

Nos casos dos subproblemas de tamanho $r(n)=(n-1)$, $r(n)=n-2$ com $\varepsilon=2$ e $r(n)=n/2$, pode-se concluir que é viável a aplicação do método das equações características, obtendo, em todos os casos, expressões para a complexidade exata não muito extensas e de fácil aplicação. Ou seja, se existir um algoritmo que possua as características de algum tipo de tamanho de subproblema acima, basta substituir, por valores numéricos, o tamanho da entrada, o número de subproblemas, o termo $f(n)$ e a condição inicial na tabela elaborada para casos gerais e particulares para obter a complexidade exata deste algoritmo.

Quando o subproblema possui tamanhos $r(n)=(n-3)$ ($\varepsilon=3$ para o caso $n-\varepsilon$) e $r(n)=(n-1)+(n-2)$ vê-se que a aplicação do método das equações características não é muito

indicada, pois nestes casos os resultados da complexidade exata apresentam-se de forma muito extensa, o que dificulta a aplicação de diversos algoritmos recursivos no resultado geral. Ou seja, não é direta a substituição dos valores numéricos das variáveis *m, b, f(n)* e *n* na solução final (resultados apresentados nas tabelas de complexidade exata). Nestes casos, aconselha-se obter a ordem da complexidade, a partir da equação de solução geral da equação de recorrência, realizando uma análise sobre as constantes c_1, c_2,..., c_n para determinar qual das constantes é dominante na equação. Conhecendo esta constante dominante, tem-se a ordem de complexidade.

Quando o subproblema possui tamanho $r(n)=n-\varepsilon$, com $\varepsilon>2$, durante a aplicação do método das equações características, surgem algumas raízes complexas, além das reais, o que dificulta ainda mais a obtenção da complexidade exata do algoritmo. Isto acontece devido ao grande número de termos no resultado final, pois as raízes complexas apresentam-se aos pares. No entanto, para calcular o valor da complexidade, estas raízes complexas não apresentam dificuldade, pois ao substituir valores númericos nas variáveis da equação de recorrência genérica, estes números complexos se anulam. O que pode ocorrer na solução final, após a substituição destas variáveis, é obter um número não inteiro, ou seja, obter a complexidade exata do algoritmo no formato de números reais. Isso não é válido, no entanto, pois a complexidade de um algoritmo, fixado um *n*, é representada por um número inteiro, devida ao tamanho da entrada ser inteiro e a função de complexidade pertencer ao conjunto dos números naturais. Quando isso ocorre, pode-se realizar uma aproximação deste número, tornando-o inteiro (normalmente, arredonda-se para o número inteiro imediatamente superior ao número real encontrado).

Em todos os casos de tamanhos de subproblemas utilizados na resolução do cálculo da complexidade, obteve-se polinômios característicos semelhantes, ou seja, com o mesmo tipo de raízes e, quando aumentado a ordem do polinômio, suas raízes continuam as mesmas, isto é, aumenta o número de raízes mas os valores se repetem. Este fato ocorre devido os polinômios característicos, obtidos após o processo de homogeneização da equação de recorrência, seguirem a lei de formação da fórmula do Binômio de Newton, o Produto de Stevin (ver seção 4).

Para auxiliar na resolução das equações de recorrência, foram desenvolvidos procedimentos no Maple, conforme o tipo de recorrência (linear ou divisão-e-conquista).

Os procedimentos calculam, a partir da equação característica, as raízes, o sistema de equações obtido da equação geral da equação de recorrência original e a substituição da solução do sistema de equações na solução geral, obtendo a solução exata da complexidade do algoritmo ou da equação de recorrência. Nos resultados obtidos da aplicação do método das equações características nos quatro casos de tamanhos de subproblemas considerados neste trabalho, obteve-se resultados muito extensos no Maple, que ainda podiam ser simplificados, com ocorreu em alguns casos. Nos anexos A-1 e A-2 encontram-se os comandos e bibliotecas utilizadas para a resolução dos problemas citados acima.

Como trabalhos futuros, deseja-se aplicar este método, resolução das equações de recorrências utilizando equações características, no cálculo da complexidade do algoritmo para o caso médio, em Divisão-e-Conquista paralela, ou seja, algoritmos desenvolvidos pelo método da Divisão-e-Conquista em computadores com arquitetura paralela e em algoritmos desenvolvidos para sistemas concorrentes, isto é, sistemas que possuem paralelismo implícito. Pretende-se, ainda, realizar um estudo sobre as Funções Geradoras e efetuar uma comparação com os resultados obtidos pelo método das equações características.

ANEXO A-1 PROCEDIMENTO PARA CÁLCULO DA COMPLEXIDADE, PARA RECORRÊNCIAS DO TIPO LINEAR, ATRAVÉS DO MAPLE

Para auxiliar na resolução das equações de recorrência, foi desenvolvido um procedimento no Maple que, a partir do polinômio característico, calcula as raízes características, o sistema de equações obtido da equação geral da equação de recorrência original e a substituição da solução dos sistemas de equações na solução geral, obtendo a solução exata do algoritmo ou da equação de recorrência. Este procedimento aplica-se a recorrência do tipo linear de i-ésima ordem.

Procedimento A-1

```
# Cálculo das raízes da equação caracterísitica, fatorando a equação em f, expandindo o mesmo em g e resolvendo as raízes em raiz:
> f:=factor(r^3-2*r^2+1);
> g:=expand(f);
> raiz:=solve(g,r);
# Chama a biblioteca que contém os procedimentos para resolução de problemas de álgebra linear.
> with(linalg):
# Construção do sistema de equações a partir da solução geral, x_n, e das condições iniciais:
>sys:=(c1+c2+c3=b,c1+(raiz[2])*c2+(raiz[3])*c3=b,c1+(raiz[2])^2*c2+(raiz[3])^2*c3=3*b);
# Resolve o sistema de equações, sys, fatora e simplifica os valores das constantes calculadas, c1, c2,.., cn.
> sol:=solve({sys},{c1,c2,c3}):
> sol1:=factor(simplify(sol)):
> sol1[1];
> sol1[2];
> sol1[3];
# Constrói a solução geral, x_n, com as raízes características.
> xn:=c1+(raiz[2])^n*c2+(raiz[3])^n*c3;
# Substitui os valores das constantes c1, c2 e c3 na solução geral, x_n, e atribui a T(n) a solução final, simplificada.
> Tn:=simplify(subs(sol1,xn),power);
```

ANEXO A-2 PROCEDIMENTO PARA CÁLCULO DA COMPLEXIDADE, PARA RECORRÊNCIAS DO TIPO DIVISÃO-E-CONQUISTA, ATRAVÉS DO MAPLE

Este procedimento aplica-se a recorrência do tipo Divisão-e-Conquista, ou seja, algoritmos que possuem subproblemas de tamanho $r(n)=n/2$.

Procedimento A-2

```
# Cálculo das raízes da equação caracterísitica, fatorando a equação em f,
expandindo o mesmo em g e resolvendo as raízes em raiz:
> f:=factor(r^3-5*r^2+8*r-4);
> g:=expand(f);
> raiz:=solve(g,r);
# Bloco de comandos para raízes com multiplicidade maior que 1.
# Define-se a solução geral em termos de t_i e a partir desta uma nova
solução geral em termos de T(n).
> t[i]:=c1+c2*raiz[2]^i+c3*i*raiz[3]^i;
> i:=log[2](n);
> T(n):=eval(simplify(algsubs(i=log[2](n),ti)));
# Bloco de comandos para raízes distintas entre si.
# Define-se a solução geral em termos de t_i e a partir desta define-se uma
nova solução geral em termos de T(n).
> t[i]:=c1+c2*raiz[2]^i+c3*raiz[3]^i;
> T(n):=c1+c2*n^log[2](raiz[2])+c3*n^log[2](raiz[3]);
# Chama a biblioteca que contém os procedimentos para resolução de
problemas de álgebra linear.
> with(linalg):
# Construção do sistema de equações a partir das condições iniciais e da
solução geral T(n).
> sys:=(c1+c3=b,2*c1+2*c2+c3=2*b+2,4*c1+8*c2+c3=4*b+12);
# Resolução do sistema de equações, fatorando e simplificando os valores
das constantes calculadas c_1, c_2, ..., c_n.
> sol:=solve({sys},{c1,c2,c3}):
> sol1:=factor(simplify(sol)):
> sol1[1];
> sol1[2];
> sol1[3];
# Substituição das variáveis c_1, c_2 e c_3 na solução geral T(n).
> Tn:=simplify(subs(sol1,T(n)),power);
```

BIBLIOGRAFIA

[AHO74] AHO, A.V., HOPCROFT, J. E. and ULLMAN, J. D., "The design and Analysis of computer Algorithms". Addison-Wesley, 1974.

[BAL90] BALCAZAR, José Luiz; DIAZ, Josep; GABARRO, Joaquim. "Structural Complexity". Berlin: Springer-Verlang, 1990.

[BEN80] BENTLEY, Jon L; HAKEN, Dorothea; SAXE, James B. "A general method for solving divide-and-conquer recurrence". SIGACT News,12(3):36-44,1980.

[BIR93] BIRD, Steven; MARK, Ellison. "Course computational phosology lectures". Faculdade de Letras, Universidade de Lisboa, Portugal, August 16-27, 1993.

[BOV93] BOVET, D. & Crescenzi, P. "Introduction to the Theory of Complexity", Prentice-Hall, 1993.

[BRA95] BRASSARD, G., BRATLEY, P. "Fundamentals of Algorithmics". Prentice-Hall, 1995.

[BRO77] BRONSHTEIN, I.; SEMENDIAEV, K. "Manual de Matematicas para ingenieros y estudients ". 3 ed. Traduccion al español. Editorial Mir, 1977.

[COO83]COOK, S. A. "An Overview of Computational Complexity". Comunications of the ACM. 26 (6):401-7, jun. 1983.

[COR90] CORMEN, Thomas H., LEISERSON, Charles E. and RIVEST, Ronald L. "Introduction to Algorithms". 2.ed. Massachusetts Institute of Technology,1990.

[CUN88] CUNHA, Rudnei D. "Algoritmos para a síntese de imagens através do método *Ray-tracing*". Monografia de Graduação. Depto. De Informática, UFRGS/RS, 1988.

[ELL92] ELLIS,Wade Jr; JOHNSON, Eugene W.; LODI, Ed. "Maple V Fllight Manual: Tutorials for Calculus, Linear Algebra and Differential Equations".Brooks/Cole Publishing Company, Pacific Grove, California, 1992.

[FAR64] FARIAS, Sinésio de. "Curso de Álgebra". Rio de Janeiro: Globo, 1964.

[GOL58] GOLDBERG, Samuel. "Introduction to Difference Equations". Library of Congress Catalog card:58-10223, United of America, 1958.

[GRA89] GRAHAM, Ronald Lewis; KNUTH, Donald E.; PATASHNISK, Oren. "Concrete mathematics: a foundation for compute science", Addison-Wesley, 1989.

[GRE81] GREENE, Daniel H., KNUT, Donald E. "Mathematics for the Analysis of Algorithms". (Progress in computer Science; vol.1) Boston; Basel, Stuttgart: Birkhäuser, 1981.

[HOA61] HOARE, C. A. R. "Partition (Algorithm63) ". CACM, vol. 4, n°7, Jul. 1961, p.321.

[HOR78] HOROWITZ, Ellis, SAHNI, Sartaj. "Fundamentals of Computer Algorithms". Computer Science Press, 1978.

[KAY79] KAY, Douglas, S., and GREENBERG, Donald. "Transparency for Computer Synthesized Images" *Computer Graphics*, Vol. 13, pp. 158-164, 1979 (Proc. SIGGRAPH 79).

[KNU72] KNUTH, Donald E. "Fundamental Algorithms". United States of America: Addison-Wesley, 1972.

[KNU73] KNUTH, Donald E. "The art of computer programming". vol III/Sorting and Searching. United States of America: Addison-Wesley, 1973.

[KNU81] KNUTH, Donald E. "The art of computer programming". 2 ed. vol. II/Seminumerical Algorithms. United States of America: Addison-Wesley, 1981.

[LOR99] LORETO, Aline B.; TOSCANI, Laira V.; NEGRON, Manuel M. & FACHIN, Maria Paula. "Resolução de equações de Recorrências e sua aplicação na complexidade de Algoritmos".In: XXII Congresso Nacional de Matemática Aplicada e Computacional, Santos, 13-17 setembro 1999, Anais. SBMAC, Santos-SP, p. 373.

[LUE80] LUEKER, George S. "Some Techniques for Soving Recurrences". Computing Surveys, vol. 12, n° 4, December 1980.

[MAN89] MANBER, Udi. "Introduction to Algorithms: A creative approach". Addison-Wesley,1989.

[PAP94] PAPADIMITRIOU, C. "Computational Complexity", Addison-Wesley, 1994.

[SED96] SEDGEWICK, Robert; FLAJOLET, Philipe. "An introduction to analysis of algorithms". Addison-Wesley, 1996.

[SHA98] SHACKELFORD, Russel L. "Introduction to Computing and Algorithms". Addison-Wesley, 1998.

[TOS86] TOSCANI, Laira V. & VELOSO, Paulo A. S. "Divisão e Conquista: Análise da complexidade". In: Seminário Integrado de Software e Hardware, 13, Olinda, jul.19-25. Anais. Recife, SBC/UFPE,1986.p.89-104.

[TOS87] TOSCANI, Laira V. "Análise da complexidade de algoritmos em arquiteturas paralelas. Estudo de Caso: a técnica de divisão e conquista. Pesquisa Operacional, v. 7, n. 2, p. 66-86, dez, 1987.

[TOS88] TOSCANI, L. V. "Métodos de Desenvolvimento de Algoritmos: Análise comparativa e de Complexidade". Tese de doutorado. Depto. De Informática, PUC/RJ. 1988.

[WHI79] WHITTED, J. T., "An Improved Illumination Model for Shaded Display,"CACM, Vol. 23, pp. 343-349, (Proc. SIGGRAPH79).

[WIN96] WINKLER, Franz. "Polynomial algorithms in computer algebra". Springer-Verlang/Wien, 1996.