Teoria da Computação

Guilherme Henrique de Souza Nakahata

Universidade Estadual do Paraná - Unespar

15 de Maio de 2024

Recorrência

- Definição:
  - Uma equação ou inequação que descreve uma função em termos de seu valor sobre em termos da entrada com valor menor;
- Recorrências aparecem quando um algoritmo contém chamadas recursivas para ele mesmo;
- Para resolver uma recorrência:
  - Encontrar uma fórmula explícita de uma expressão;
  - Limitar a recorrência por uma expressão que envolva n;

Exemplo

Algorithm 1 Fatorial(n)

Require: Um número inteiro n

```
if n ≤ 1 then
    return 1
else
    return n × Fatorial(n − 1)
end if
```

Exemplo

Algorithm 2 Fibonacci(n)

Require: Um número inteiro n

```
if n ≤ 1 then
    return n
else
    return Fibonacci(n − 1) + Fibonacci(n − 2)
end if
```

Exemplo

Algorithm 3 MergeSort(A, p, r)

if $p < r$ **then**
  $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$
  MergeSort(A, p, q)
  MergeSort($A, q + 1, r$)
  Merge(A, p, q, r)
end if

Complexidade de Algoritmos Recursivos

- $T(n) = T(n-1) + n$
  - $\Theta(n^2)$
  - Algoritmo recursivo que a cada loop examina a entrada e elimina um item;
- $T(n) = T(n/2) + c$
  - $\Theta(\log n)$
  - Algoritmo recursivo que divide a entrada em cada passo;
- $T(n) = T(n/2) + n$
  - $\Theta(n)$
  - Algoritmo recursivo que divide a entrada, mas precisa examinar cada item na entrada;
- $T(n) = 2T(n/2) + 1$
  - $\Theta(n)$
  - Algoritmo recursivo que divide a entrada em duas metades e executa uma quantidade constante de operações.

Métodos de Resolução

Métodos que veremos:

- Método de Substituição
- Método da Árvore de Recursão
- Método Mestre

Método de substituição

- O método consiste de duas etapas:
  - Pressupor uma função como solução;
  - Usar **indução matemática** para encontrar as contantes da definição assintótica;
- Provar no passo da indução examente o que foi proposto;
- Provar que $T(n) \leq f(n)$.

Exemplo

- Resolver a recorrência $T(n) = 2T(\frac{n}{2}) + n$;

Exemplo

- Resolver a recorrência $\mathrm{T(n)} = 2\mathrm{T}(\frac{n}{2}) + \mathrm{n}$;
- Suposição que T(n) é $O(n^2)$;
- Provar por indução que $\mathrm{T(n)} \leq \mathrm{c}n^2$.

Exemplo

- Resolver a recorrência $\mathrm{T(n)} = 2\mathrm{T}(\frac{n}{2}) + \mathrm{n}$;
- Suposição que T(n) é $O(n^2)$;
- Provar por indução que $\mathrm{T(n)} \leq \mathrm{c}n^2$.

Exemplo

- Resolver a recorrência $\mathsf{T(n)} = 2\mathsf{T}(\frac{n}{2}) + \mathsf{n}$;
- Suposição que T(n) é $O(n^2)$;
- Provar por indução que $\mathsf{T(n)} \leq \mathsf{c}n^2$;
- Base: $\mathsf{n} = 1$;
  - Sabemos que $\mathsf{T}(1) = 1$ e $\mathsf{c}.1^2 = \mathsf{c}$;
  - Caso base vale se $\mathsf{c} \geq 1$;

Exemplo

- Resolver a recorrência $T(n) = 2T(\frac{n}{2}) + n$;
- Suposição que $T(n)$ é $O(n^2)$;
- Provar por indução que $T(n) \leq cn^2$;
- Base: $n = 1$;
  - Sabemos que $T(1) = 1$ e $c.1^2 = c$;
  - Caso base vale se $c \geq 1$;
- Queremos mostrar que $T(n) \leq cn^2$ se $n > 1$;
- Hipótese?

Exemplo

- Resolver a recorrência $\mathrm{T(n)} = 2\mathrm{T}(\frac{n}{2}) + \mathrm{n}$;
- Suposição que T(n) é $\mathrm{O}(n^2)$;
- Provar por indução que $\mathrm{T(n)} \leq \mathrm{c}n^2$;
- Base: $\mathrm{n} = 1$;
  - Sabemos que $\mathrm{T}(1) = 1$ e $\mathrm{c}.1^2 = \mathrm{c}$;
  - Caso base vale se $\mathrm{c} \geq 1$;
- Queremos mostrar que $\mathrm{T(n)} \leq \mathrm{c}n^2$ se $\mathrm{n} > 1$;
- Hipótese: $\mathrm{T(k)} \leq \mathrm{ck}^2\ \forall 1 \leq k < n$;
- Sabemos que $\mathrm{T(n)} = 2\mathrm{T}(\frac{n}{2}) + \mathrm{n}$;
- Como $\frac{n}{2} < n$ para $\mathrm{n} > 1$, vale por hipótese que:
  - $T\frac{n}{2} \leq c(\frac{n}{2})^2$;
  - $\frac{cn^2}{4}$
- Logo:

Exemplo

- $T\frac{n}{2} \leq c(\frac{n}{2})^2 = \frac{cn^2}{4}$;
- $\mathrm{T(n)} = 2\mathrm{T}(\frac{n}{2}) + n \leq 2(\frac{cn^2}{4}) + n$;
- $\frac{cn^2}{2} + n$;
- $\leq cn^2$

Exemplo

- $T\frac{n}{2} \leq c(\frac{n}{2})^2 = \frac{cn^2}{4}$;
- $\mathrm{T(n)} = 2\mathrm{T}(\frac{n}{2}) + n \leq 2(\frac{cn^2}{4}) + n$;
- $\frac{cn^2}{2} + n$;
- $\leq cn^2$;
- $\mathrm{c} \geq 2$;
- Provamos por indução que $\mathrm{T(n)} \leq \mathrm{c}n^2$ ($\mathrm{c} = 2$ e $n_0 = 1$).

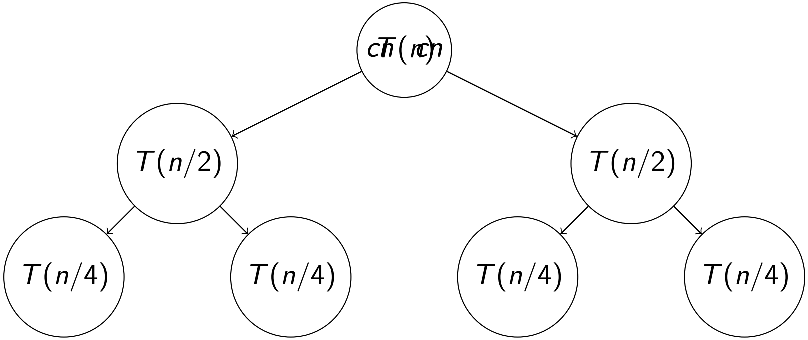
Exercícios

- Resolva a recorrência $\mathsf{T(n)} = 2\mathsf{T}(\frac{n}{2}) + \mathsf{n}$ é $\Omega(n^2)$;
- Resolva a recorrência $\mathsf{T(n)} = 2\mathsf{T}(\frac{n}{2}) + \mathsf{n}$ é $\mathsf{O(n\ log\ n)}$;

Áravore de recursão

- Cada nó representa o custo de um único subproblema;
- Durante o processo de chamadas de funções recursivas;
- Permite somar o custo de cada nível da árvore;
- Determinar o custo total pela soma de todos os níveis.

Árvore de Recursão para $T(n) = 2T(n/2) + cn$



Exemplo

- Cada nível da árvore tem custo de cn;
- Altura da árvore é lg n;
- O(n lg n);

Exercicíos

- $T(n) = 3\ T(n/4) + cn^2$
- $T(n) = T(n/3) + T(2n/3) + cn$;

O Método Mestre é um método para resolver recorrências de divisão e conquista. Ele fornece uma estrutura para analisar a complexidade de algoritmos recursivos.

- O método é expresso na forma $T(n) = aT(n/b) + f(n)$.
- a é o número de subproblemas.
- b é o fator pelo qual o tamanho do problema é reduzido em cada chamada recursiva.
- $f(n)$ é o custo da divisão e conquista.

Se $f(n)$ é assintoticamente menor que $n^{\log_b a}$ para algum $\varepsilon > 0$, então a solução é $T(n) = \Theta(n^{\log_b a})$.
Exemplo: Considere a recorrência $T(n) = 8T(n/2) + n^2$.
$a = 8$, $b = 2$, $f(n) = n^2$.
$\log_b a = \log_2 8 = 3$.
Como $f(n) = O(n^{\log_b a - \varepsilon}) = O(n^{3-\varepsilon})$ para qualquer $\varepsilon > 0$, a solução é $T(n) = \Theta(n^3)$.

Se $f(n)$ é assintoticamente igual a $n^{\log_b a}$, então a solução é $T(n) = \Theta(n^{\log_b a} \log n)$.
Exemplo: Considere a recorrência $T(n) = 2T(n/2) + n$.
$a = 2$, $b = 2$, $f(n) = n$.
$\log_b a = \log_2 2 = 1$.
Como $f(n) = \Theta(n^{\log_b a}) = \Theta(n^1)$, a solução é $T(n) = \Theta(n \log n)$.

Se $f(n)$ é assintoticamente maior que $n^{\log_b a}$ e $af(n/b) \leq kf(n)$ para algum $k < 1$ e n suficientemente grande, então a solução é $T(n) = \Theta(f(n))$.

Exemplo: Considere a recorrência $T(n) = 2T(n/2) + n^2$.

$a = 2$, $b = 2$, $f(n) = n^2$.

$\log_b a = \log_2 2 = 1$.

Como $f(n) = \Omega(n^{\log_b a+\varepsilon}) = \Omega(n^{1+\varepsilon})$ para alguma constate $\varepsilon > 0$ $(\varepsilon = 1)$

$af(n/b) = 2(n/2)^2 = \frac{1}{2}n^2 \leq kf(n)$.

Se $f(n)$ é assintoticamente maior que $n^{\log_b a}$ e $af(n/b) \leq kf(n)$ para algum $k < 1$ e n suficientemente grande, então a solução é $T(n) = \Theta(f(n))$.

Exemplo: Considere a recorrência $T(n) = 2T(n/2) + n^2$.

$a = 2,\ b = 2,\ f(n) = n^2$.

$\log_b a = \log_2 2 = 1$.

Como $f(n) = \Omega(n^{\log_b a+\varepsilon}) = \Omega(n^{1+\varepsilon})$ para alguma constate $\varepsilon > 0$ $(\varepsilon = 1)$

$af(n/b) = 2(n/2)^2 = \frac{1}{2}n^2 \leq kf(n)$ para $k = \frac{1}{2}$, a solução é $T(n) = \Theta(n^2)$.

Exercícios

- 1) $T(n) = 9T(n/3) + n$
- 2) $T(n) = T(2n/3) + 1$
- 3) $T(n) = 2T(n/2) + n\lg n$

Exercício

- Implemente um algoritmo recursivo e analise sua complexidade de tempo para:
  - Encontrar o máximo elemento em uma lista de números inteiros;
  - Calcular a soma dos elementos de um vetor de números inteiros;
  - Calcule a potência ($T(n) = T(\frac{n}{2}) + O(1)$) ;

Bibliografia Básica

- LEWIS, H. R.; PAPADIMITRIOU, C. H. **Elementos de Teoria da Computação**. 2 ed. Porto Alegre: Bookman, 2000.
- VIEIRA, N. J. **Introdução aos Fundamentos da Computação**. Editora Pioneira Thomson Learning, 2006.
- DIVERIO, T. A.; MENEZES, P. B. **Teoria da Computação: Máquinas Universais e Computabilidade**. Série Livros Didáticos Número 5, Instituto de Informática da UFRGS, Editora Sagra Luzzato, 1 ed. 1999.

Obrigado!
Dúvidas?

Guilherme Henrique de Souza Nakahata

guilhermenakahata@gmail.com

https://github.com/GuilhermeNakahata/UNESPAR-2024