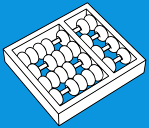
Análise de algoritmos recursivos

MC458 - Projeto e Análise de Algoritmos I

09/25

08



Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br

UNICAMP

"As recorrências andam de mãos dadas com o paradigma divisão e conquista" [1]

Thomas H. Cormen et al.

[1] Thomas H. Cormen et al., *"Algoritmo: Teoria e Prática"*, Tradução da 3ra Edição, 2009.

Projetando algoritmos

Entendendo e melhorando

Até agora:

- Ordenamos **incrementalmente** com o INSERTION-SORT.
- Vimos que sua complexidade de pior caso é $\Theta(n^2)$.

Vamos estudar uma maneira alternativa de ordenar números:

- Utilizaremos uma técnica recursiva chamada de **divisão e conquista**.
- Muitas vezes, obtemos algoritmos mais rápidos do que os incrementais.

Algoritmos recursivos

- Um **algoritmo recursivo** resolve um problema:
  - Diretamente, se a instância for pequena.
  - Chamando a si mesmo uma ou mais vezes, se a instância não for pequena.
- As chamadas recursivas devem receber **instâncias menores**.

Divisão e conquista

Um algoritmo de **divisão e conquista** tem três etapas:

1. **DIVISÃO:** Dividir o problema em subproblemas semelhantes, mas com instâncias menores.

2. **CONQUISTA:** Cada subproblema é resolvido recursivamente, ou diretamente se os subproblemas forem pequenos.

3. **COMBINAÇÃO:** As soluções dos subproblemas são combinadas para obter uma solução da instância original.

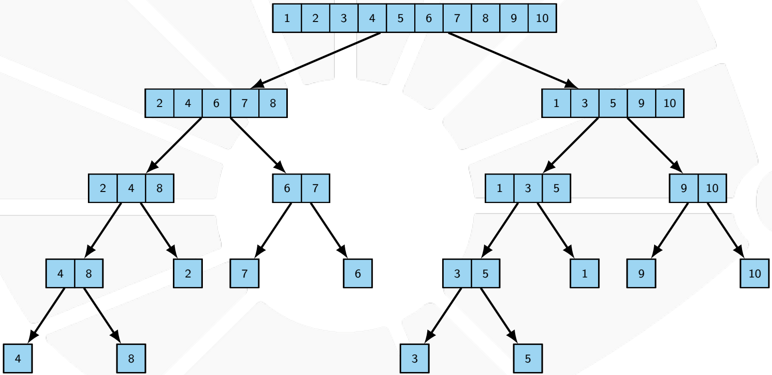
Exemplo: ordenando usando divisão e conquista

Merge-Sort é um exemplo clássico de divisão e conquista.

Ideia:

1. **DIVISÃO:** Divida um vetor de tamanho n em dois subvetores de tamanhos $\lfloor n/2 \rfloor$ e $\lceil n/2 \rceil$.
2. **CONQUISTA**: Ordene os dois subvetores recursivamente.
3. **COMBINAÇÃO**: Intercale os dois subvetores obtendo um vetor ordenado.

Simulação



Mergesort

O vetor de entrada é representado como $A[p \dots r]$, com $p \leq r$.

Algoritmo: $\textsc{Merge-Sort}(A, p, r)$

se $p < r$

$\quad q \leftarrow \lfloor (p + r)/2 \rfloor$

$\quad \textsc{Merge-Sort}(A, p, q)$

$\quad \textsc{Merge-Sort}(A, q + 1, r)$

$\quad \textsc{Intercala}(A, p, q, r)$

	p				q				r
A	6	3	5	4	9	1	7	2	8

Mergesort

O vetor de entrada é representado como $A[p \dots r]$, com $p \leq r$.

Algoritmo: $\textsc{Merge-Sort}(A, p, r)$

se $p < r$
- $q \leftarrow \lfloor (p + r)/2 \rfloor$
- $\textsc{Merge-Sort}(A, p, q)$
- $\textsc{Merge-Sort}(A, q + 1, r)$
- $\textsc{Intercala}(A, p, q, r)$

	p				q				r
A	**3**	**4**	**5**	**6**	**9**	1	7	2	8

Mergesort

O vetor de entrada é representado como $A[p \dots r]$, com $p \leq r$.

Algoritmo: $\textsc{Merge-Sort}(A, p, r)$

```
se p < r
    q ← ⌊(p + r)/2⌋
    MERGE-SORT(A, p, q)
    MERGE-SORT(A, q + 1, r)
    INTERCALA(A, p, q, r)
```

	p				q				r
A	3	4	5	6	9	1	2	7	8

Mergesort

O vetor de entrada é representado como $A[p \dots r]$, com $p \leq r$.

Algoritmo: $\textsc{Merge-Sort}(A, p, r)$

se $p < r$
- $q \leftarrow \lfloor (p + r)/2 \rfloor$
- $\textsc{Merge-Sort}(A, p, q)$
- $\textsc{Merge-Sort}(A, q + 1, r)$
- $\textsc{Intercala}(A, p, q, r)$

	p				q				r
A	1	2	3	4	5	6	7	8	9

Combinando soluções dos subproblemas

Problema (Intercalar dois subvetores)

- ***Entrada:*** *Um vetor $A[p \dots r]$ tal que os subvetores $A[p \dots q]$ e $A[q+1 \dots r]$ estão ordenados.*
- ***Saída:*** *Um rearranjo de $A[p \dots r]$ ordenado.*

Entrada:

	p				q				r
A	2	3	5	7	9	1	4	6	8

Saída:

	p				q				r
A	1	2	3	4	5	6	7	8	9

Pseudocódigo de INTERCALA

Algoritmo: INTERCALA(A, p, q, r)

para $i \leftarrow p$ **até** q
    $B[i] \leftarrow A[i]$
para $j \leftarrow q + 1$ **até** r
    $B[r + q + 1 - j] \leftarrow A[j]$
$i \leftarrow p$
$j \leftarrow r$
para $k \leftarrow p$ **até** r
    se $B[i] \leq B[j]$
        $A[k] \leftarrow B[i]$
        $i \leftarrow i + 1$
    senão
        $A[k] \leftarrow B[j]$
        $j \leftarrow j - 1$

Complexidade de INTERCALA

Entrada:

	p				q				r
A	2	3	5	7	9	1	4	6	8

Saída:

	p				q				r
A	1	2	3	4	5	6	7	8	9

Tamanho da entrada: $n = r - p + 1$.

Consumo de tempo: $\Theta(n)$.

Complexidade de MERGE-SORT

Algoritmo: MERGE-SORT(A, p, r)

se $p < r$
- $q \leftarrow \lfloor (p+r)/2 \rfloor$
- MERGE-SORT(A, p, q)
- MERGE-SORT$(A, q+1, r)$
- INTERCALA(A, p, q, r)

- Tamanho da entrada: $n = r - p + 1$.
- Seja $T(n)$ o número de instruções executadas no pior caso.

Complexidade de Merge-Sort

Algoritmo: Merge-Sort(A, p, r)

```
1 se p < r
2     q ← ⌊(p + r)/2⌋
3     Merge-Sort(A, p, q)
4     Merge-Sort(A, q + 1, r)
5     Intercala(A, p, q, r)
```

Linha	Tempo
1	$\Theta(1)$
2	$\Theta(1)$
3	$T(\lceil n/2 \rceil)$
4	$T(\lfloor n/2 \rfloor)$
5	$\Theta(n)$

$$T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) + \Theta(2).$$

Recorrência

O tempo de MERGE-SORT é dado pela fórmula

$$T(n) = \begin{cases} \Theta(1) & \text{se } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + \Theta(n) & \text{se } n = 2, 3, 4, \ldots \end{cases}$$

Obtemos uma **fórmula de recorrência**:

- É a descrição de uma função em termos de si mesma.
- O tempo de um algoritmo recursivo costuma se descrito por uma recorrência.

Mas queremos uma **fórmula fechada**!

Recorrências

Resolução de recorrências

Considere a recorrência:

$$T(n) = \begin{cases} d & \text{se } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + an + b & \text{se } n \geq 2, \end{cases}$$

- Queremos uma **FÓRMULA FECHADA** para $T(n)$.
- Não é necessária a solução exata.
- Basta encontrar uma função $f(n)$ tal que $T(n) \in \Theta(f(n))$.

Resolução de recorrências

Existem alguns métodos comuns para resolver recorrências:

- Substituição.
- Iteração.
- Árvore de recorrência.

Veremos também o chamado **Teorema Master**:

- Aplicável a uma família comum de recorrências.
- Fornece uma fórmula fechada diretamente.

Método da substituição

Método da substituição

Ideia:

1. **ADIVINHAR** uma solução.
2. Demonstrar que é válida usando indução.

Nem sempre é fácil chutar a solução:

- É necessário ter experiência.
- Mas vamos obter sugestões com os métodos estudados.

Exemplo

Exemplo

Encontre uma fórmula fechada para:

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + n & \text{se } n \geq 2 \end{cases}$$

Chutamos que $T(n) \in O(n \log_2 n)$.

- Observe que há uma constante escondida na notação O.
- Para usar indução, precisamos conhecer essa constante.
- Então chutamos que $T(n) \leq 3n \log_2 n$.

Exemplo: passo da indução

Suponha que a desigualdade vale para $k < n$.

$$
\begin{aligned}
T(n) &= \mathbf{T}(\lceil \mathbf{n}/\mathbf{2} \rceil) + \mathbf{T}(\lfloor \mathbf{n}/\mathbf{2} \rfloor) + n \\
&\leq \mathbf{3}\left\lceil \frac{\mathbf{n}}{\mathbf{2}} \right\rceil \log_2 \left\lceil \frac{\mathbf{n}}{\mathbf{2}} \right\rceil + \mathbf{3}\left\lfloor \frac{\mathbf{n}}{\mathbf{2}} \right\rfloor \log_2 \left\lfloor \frac{\mathbf{n}}{\mathbf{2}} \right\rfloor + n \qquad \text{(pela h.i.)} \\
&\leq 3\left\lceil \frac{n}{2} \right\rceil \log_2 n + 3\left\lfloor \frac{n}{2} \right\rfloor (\log_2 n - 1) + n \\
&= 3\left(\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor\right) \log_2 n - 3\left\lfloor \frac{n}{2} \right\rfloor + n \\
&= 3n \log_2 n - 3\left\lfloor \frac{n}{2} \right\rfloor + n \\
&\leq 3n \log_2 n.
\end{aligned}
$$

Mostramos o passo da indução!

Exemplo: base da indução

Ainda falta a base.

- Temos $T(1) = 1$, mas $3 \cdot 1 \cdot \log_2 1 = 0$.
- A desigualdade não vale para $n = 1$.
- Não temos uma base da indução.

Ok, queremos mostrar apenas $T(n) \in O(n \log_2 n)$.

- Basta mostrar que a desigualdade vale para $n \geq n_0$.
- Vamos escolher $n_0 = 2$.

Base da indução:

- Para $n = 2$ e $n = 3$, temos:

$$T(2) = T(1) + T(1) + 2 = 4 \leq 3 \cdot 2 \cdot \log_2 2 = 6$$
$$T(3) = T(2) + T(1) + 3 = 8 \leq 3 \cdot 3 \cdot \log_2 3 \approx 14{,}26$$

- Por que precisamos de **DOIS CASOS BÁSICOS**?

Modificando um pouco o exemplo

Mas se tivéssemos $T(1) = 8$?

- A desigualdade não vale para $n = 2$
- Temos $T(2) = 8 + 8 + 2 = 18$, mas $3 \cdot 2 \cdot \log_2 2 = 6$
- Podemos escolher uma **constante multiplicativa** maior.
- Tentando $T(n) \leq 10n \log_2 n$, obtemos

$$T(2) = 18 \leq 10 \cdot 2 \cdot \log_2 2 = 20$$
$$T(3) = 29 \leq 10 \cdot 3 \cdot \log_2 3 \approx 47{,}55$$

Conclusão:

- Se o passo da indução vale, então podemos escolher valores adequados para as constantes c e n_0.

Encontrando as constantes

Suponha que já adivinhamos que $T(n) \in O(n \log_2 n)$:

- Chutamos que $T(n) \leq cn \log_2 n$ para $c = 3$.
- Depois escolhemos $n_0 = 2$.
- Como podemos encontrar essas constantes?

Uma maneira possível:

1. Suponha $T(n) \leq cn \log_2 n$ para algum c genérico.
2. Substitua e desenvolva a expressão para $T(n)$.
3. Determine c e n_0 de forma a provar o passo da indução.

Primeira tentativa

$$
\begin{aligned}
T(n) &= \mathbf{T}(\lceil \mathbf{n}/\mathbf{2} \rceil) + \mathbf{T}(\lfloor \mathbf{n}/\mathbf{2} \rfloor) + n \\
&\leq \mathbf{c}\left\lceil \frac{\mathbf{n}}{\mathbf{2}} \right\rceil \log_{\mathbf{2}} \left\lceil \frac{\mathbf{n}}{\mathbf{2}} \right\rceil + \mathbf{c}\left\lfloor \frac{\mathbf{n}}{\mathbf{2}} \right\rfloor \log_{\mathbf{2}} \left\lfloor \frac{\mathbf{n}}{\mathbf{2}} \right\rfloor + n \\
&\leq c\left\lceil \frac{n}{2} \right\rceil \log_2 n + c\left\lfloor \frac{n}{2} \right\rfloor \log_2 n + n \\
&= c\left(\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor\right) \log_2 n + n \\
&= cn \log_2 n + n
\end{aligned}
$$

Não deu certo...

Segunda tentativa

$$
\begin{aligned}
T(n) &= \mathbf{T}(\lceil \mathbf{n}/\mathbf{2} \rceil) + \mathbf{T}(\lfloor \mathbf{n}/\mathbf{2} \rfloor) + n \\
&\leq \mathbf{c}\left\lceil \frac{\mathbf{n}}{\mathbf{2}} \right\rceil \log_2 \left\lceil \frac{\mathbf{n}}{\mathbf{2}} \right\rceil + \mathbf{c}\left\lfloor \frac{\mathbf{n}}{\mathbf{2}} \right\rfloor \log_2 \left\lfloor \frac{\mathbf{n}}{\mathbf{2}} \right\rfloor + n \\
&\leq c\left\lceil \frac{n}{2} \right\rceil \log_2 n + c\left\lfloor \frac{n}{2} \right\rfloor (\log_2 n - 1) + n \\
&= c\left(\left\lceil \frac{n}{2} \right\rceil + \left\lfloor \frac{n}{2} \right\rfloor\right) \log_2 n - c\left\lfloor \frac{n}{2} \right\rfloor + n \\
&= cn \log_2 n - c\left\lfloor \frac{n}{2} \right\rfloor + n \\
&\leq cn \log_2 n. \qquad \text{(escolhendo } n \geq n_0 \text{ adequado)}
\end{aligned}
$$

- Para a última desigualdade, queremos $-c\lfloor n/2 \rfloor + n \leq 0$.
- Basta que $c = 3$ e $n_0 \geq 2$.

Completando o exemplo

- Já mostramos que $T(n) \in O(n \log_2 n)$.
- Mas queremos mostrar que $T(n) \in \Theta(n \log_2 n)$.
  - Resta mostrar $T(n) \in \Omega(n \log_2 n)$.
  - A prova é similar.
  - Faça como exercício.

Adivinhando uma solução

Não há receita genérica para adivinhar soluções.

- A experiência é um fator importante.
- Normalmente, recorrências similares têm a mesma ordem de crescimento.

Exemplo

Resolva a recorrência

$$T(n) = \begin{cases} 1 & \textit{se } n = 1 \\ 2T(\lfloor n/2 \rfloor) + n & \textit{se } n \geq 2 \end{cases}$$

- Ela muito parecida com a anterior.
- Então chutamos $T(n) \in \Theta(n \log_2 n)$.
- Mostre isso como exercício.

Adivinhando uma solução (cont)

Exemplo

Resolva a recorrência

$$T(n) = \begin{cases} 1 & se\ n = 1 \\ 2T(\lfloor n/2 \rfloor + 17) + n & se\ n \geq 2 \end{cases}$$

- Ela parece mais difícil por causa do termo 17.
- Intuitivamente, parece que $T(\lfloor n/2 \rfloor) \approx T(\lfloor n/2 \rfloor + 17)$
- Chutamos que $T(n) \in \Theta(n \log_2 n)$.
- Mostre isso como exercício.

Sutilezas

Algumas vezes é necessário fortalecer a hipótese de indução.

Exemplo

Resolva a recorrência

$$T(n) = \begin{cases} 1 & se\ n = 1 \\ T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + 1 & se\ n \geq 2 \end{cases}$$

Chutamos que $T(n) \leq cn$ para alguma constante c.

$$\begin{aligned} T(n) &= \mathbf{T(\lceil n/2 \rceil)} + \mathbf{T(\lfloor n/2 \rfloor)} + 1 \\ &\leq \mathbf{c\lceil n/2 \rceil} + \mathbf{c\lfloor n/2 \rfloor} + 1 \\ &= cn + 1. \end{aligned}$$

- Não deu certo.
- Mas de fato $T(n) \in \Theta(n)$!

Fortalecendo a hipótese

- Vamos fazer uma hipótese de indução um pouco diferente.
- Vamos mostrar que $T(n) \leq cn - b$ para $c > 0$ e $b > 0$:

$$
\begin{aligned}
T(n) &= \mathbf{T}(\lceil \mathbf{n}/\mathbf{2} \rceil) + \mathbf{T}(\lfloor \mathbf{n}/\mathbf{2} \rfloor) + 1 \\
&\leq \mathbf{c}\lceil \mathbf{n}/\mathbf{2} \rceil - \mathbf{b} + \mathbf{c}\lfloor \mathbf{n}/\mathbf{2} \rfloor - \mathbf{b} + 1 \\
&= cn - 2b + 1 \\
&\leq cn - b.
\end{aligned}
$$

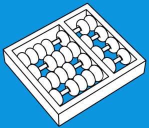
- Para a última de desigualdade, basta escolher $b \geq 1$.
- Isso mostra o passo indutivo.

Análise de algoritmos recursivos

MC458 - Projeto e Análise de Algoritmos I

08

09/25



UNICAMP

Santiago Valdés Ravelo
https://ic.unicamp.br/~santiago/
ravelo@unicamp.br