Projeto de Algorítmo

Projeto de Algorítmo

- Principais abordagens
  - **Divisão e conquista:** subproblemas análogos e disjuntos;resoluções recursivas; combinação entre as subsoluções.
  - **Método Guloso:** solução incremental; otimização de um critério local.
  - **Programação Dinâmica:** subproblemas análogos, mas sobrepostos; resoluções em ordem crescente de tamanho; armazenamento das subsoluções em tabelas.

Divisão e Conquista

- Dividir o problema original em um determinado número de subproblemas independentes.

- Conquistar os subproblemas, resolvendo-os recursivamente até obter o caso base.

- Combinar as soluções dadas aos subproblemas, a fim de formar a solução do problema original.

Divisão e Conquista

- Recorrências

  O tempo de execução dos algoritmos recursivos pode ser descrito por uma recorrência.

  Uma recorrência é uma equação ou desigualdade que descreve uma função em termos de seu valor em entradas menores.

Divisão e Conquista

Exemplo do Paradigma da divisão e conquista do Algoritmo de ordenação por intercalação (Mergesort).

Divisão: divide arranjo de n elementos em dois de n/2

- Na realidade, um com n/2 e outro com n/2 elementos

Conquista: ordena recursivamente os dois arranjos

- Quando há 1 elemento: arranjo já está ordenado

Combinação: intercala os dois arranjos ordenados, produzindo um arranjo com n elementos ordenado

Divisão e Conquista

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

Divisão e Conquista

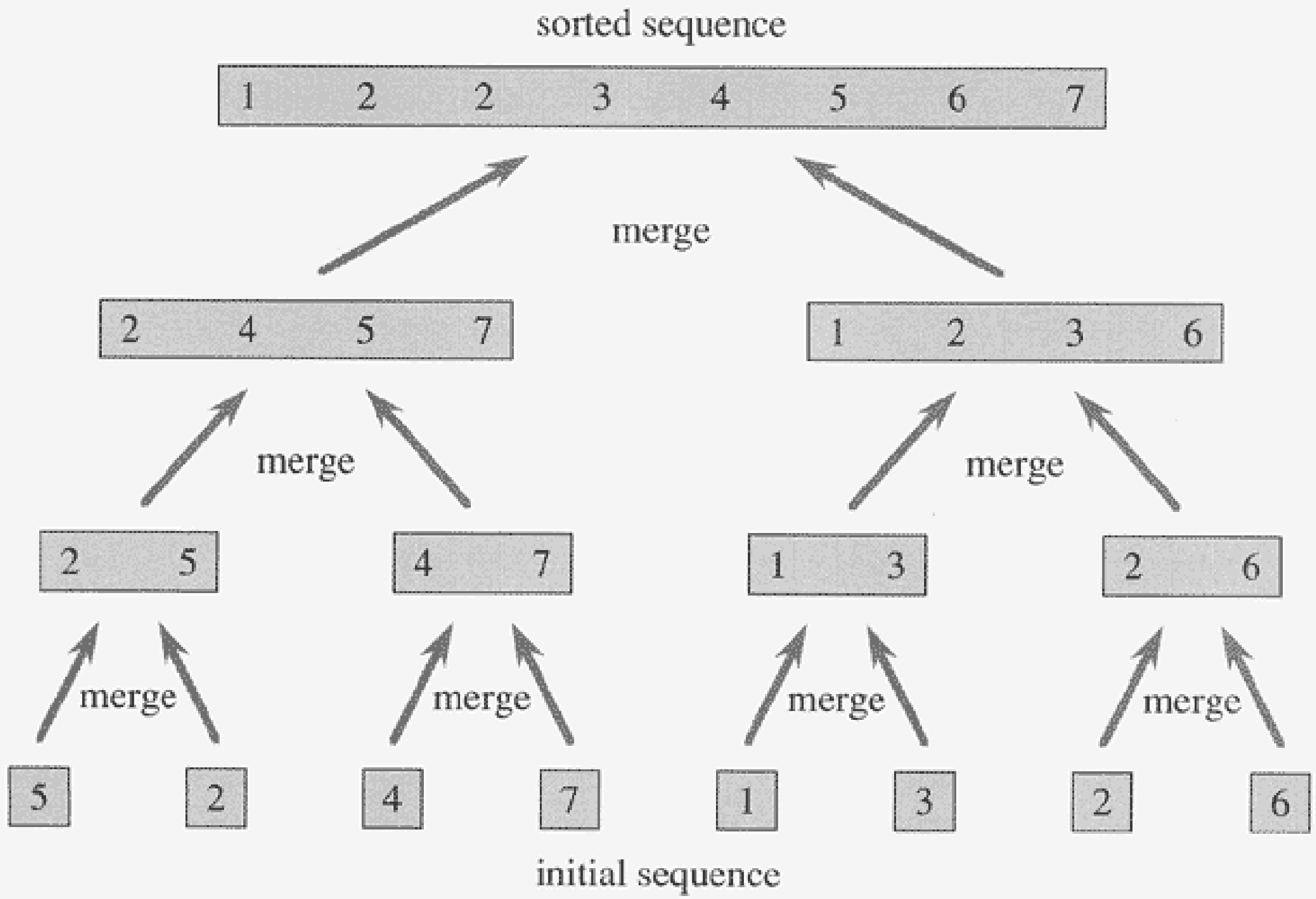


Figure 2.4 The operation of merge sort on the array $A = \langle 5, 2, 4, 7, 1, 3, 2, 6 \rangle$. The lengths of the sorted sequences being merged increase as the algorithm progresses from bottom to top.

Divisão e Conquista

Tempo de execução escrito por uma recorrência

- **Recursão natural**
  - Seja d(n) o tempo para a divisão.
  - Seja s(n) o tempo para computar a solução final.
  - Podemos somar: f(n) = d(n) + s(n) e obtemos

  - **Em geral, temos:**

$$T(n)=\begin{cases} \Theta(1) & se \quad n<c \\ aT(n/b)+D(n)+C(n) & caso \quad contrário \end{cases}$$

a: quantidade de subproblemas

n/b: tamanho dos subproblemas

D(n): tempo gasto na etapa de divisão

C(n): tempo gasto na etapa de conquista

Divisão e Conquista

- **Recorrências: caso do Mergesort**

A equação de recorrência do Mergesort é:

$$\mathbf{T(n) = 2T\,(n/2) + \Theta(n)}$$

Sendo que:

- $T(n)$ representa o tempo da chamada recursiva da função para um problema de tamanho n.
- $2T\,(n/2)$ indica que, a cada iteração, duas chamadas recursivas (2T) serão executadas para entradas de tamanho $n/2$.
- Os resultados das duas chamadas recursivas serão combinados (merged) com um algoritmo com complexidade de pior caso $\Theta(n)$.

Divisão e Conquista

- **Métodos para resolver recorrências**

  Existem vários métodos para resolver recorrências:

  - Método da substituição;
  - Método de árvore de recursão;
  - Método mestre.

Divisão e Conquista

- A árvore de de recursão é um método muito útli para estimar a solução de uma recorrência.

- A ideia da árvore de recursão é representar o desenvolvimento da recorrência por um diagrama (geralmente uma árvore) onde cada nó representa um subproblema, e somar os custos por nível.

Divisão e Conquista

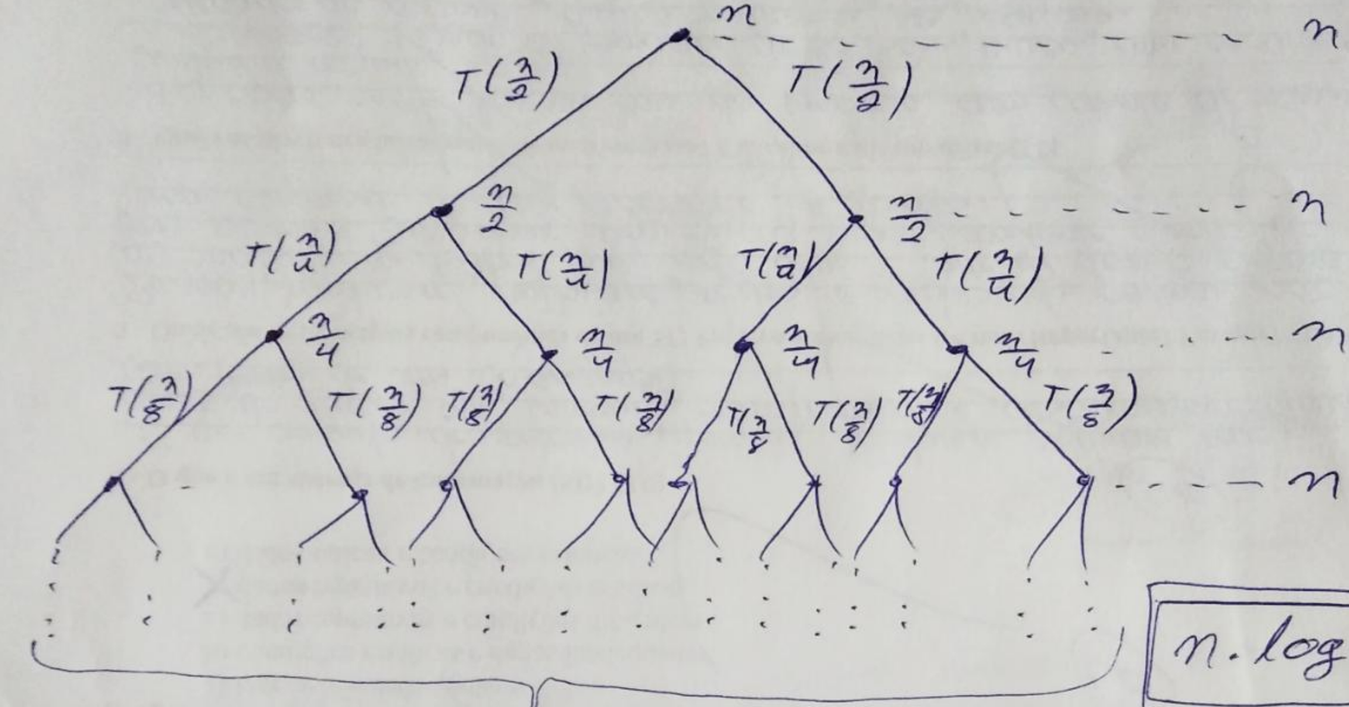
- **Método da árvore de recursão**

  Uma árvore de recursão apresenta uma forma bem intuitiva para a análise de complexidade de algoritmos recursivos

  - Numa árvore de recursão cada nó representa o custo de um único subproblema da respectiva chamada recursiva;
  - Somam-se os custos de todos os nós de um mesmo nível, para obter o custo daquele nível;
  - Somam-se os custos de todos os níveis para obter o custo da árvore.

Exemplo: considere a recorrência: *T(n)= 2T(n/2) + n*

EXEMPLO 1: $T(n) = 2T\left(\frac{n}{2}\right) + n$



$$\boxed{n \cdot \log n}$$

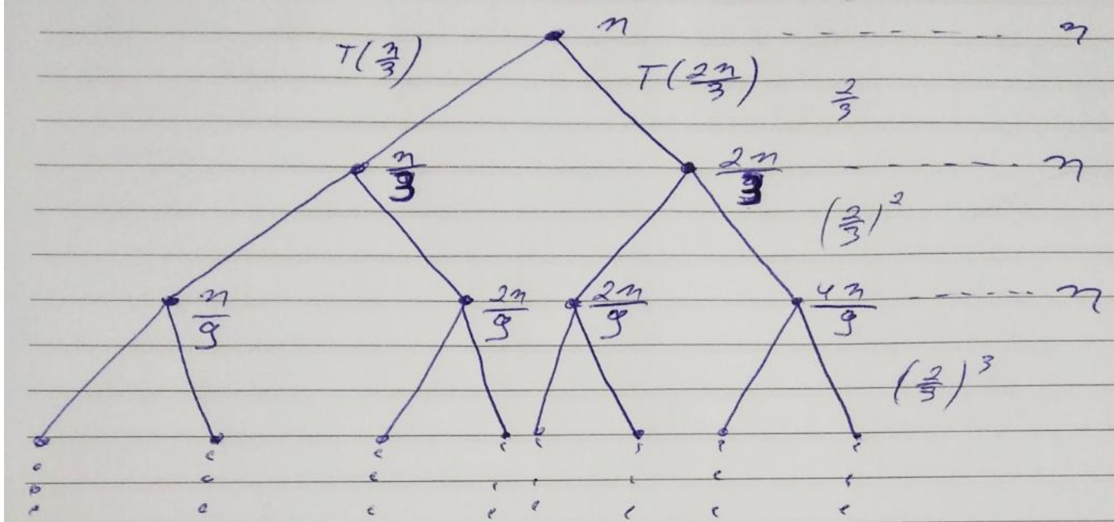
$$\frac{n}{2^i} \Rightarrow i = \log n$$

* VAI PARAR QUANDO n FICAR PEQUENO AO PONTO DE RESOLVER O PROBLEMA.

$i = \log n$, SIGNIFICA QUE TEMOS $\log n$ NÍVEIS.

Exemplo: considere a recorrência: $T(n)= 2T(n/3) + T(2n/3) + n$

$T(n) = T(\frac{n}{3}) + T(\frac{2n}{3}) + n$



* OS RAMOS SÃO DIFERENTES
* O RAMO DA DIREITA CRESCE MAIS PROFUNDAMENTE
* QUANTOS NÍVEIS TEM?

- CADA NÍVEL CONTRIBUI COM n E TEM $\log n$ NÍVEIS

O PADRÃO É $\left(\frac{2}{3}\right)^i n$

$$\left(\frac{2}{3}\right)^i n = 1 \Rightarrow n = \left(\frac{3}{2}\right)^i \Rightarrow i = \log_{3/2} n$$

* A BASE DO LOG NÃO INFLUENCIA NO VALOR FINAL DA SOLUÇÃO

$$\boxed{\Theta(n \,.\, \log n)}$$

Divisão e Conquista

Exercício: através da árvore de recursão, resolva a seguinte recorrência:

$$T(n) = T(n/2) + T(n/4) + n$$

Divisão e Conquista