Análise e Projeto de Algoritmos

Divisão e Conquista (Parte 1)

Prof. Bruno Bruck

Centro de Informática
UFPB

Divisão e Conquista

- Paradigmas de Projeto de Algoritmos
  - Técnicas genéricas para construir algoritmos
  - Fornecem uma abstração do algoritmo usado para resolver um problema
  - Exemplos:
    - Divisão e Conquista
    - Algoritmos Gulosos
    - Programação Dinâmica
    - Etc...

Divisão e Conquista

- É uma técnica que consiste em dividir um problema maior e mais complexo em subproblemas menores
  - Posteriormente combinamos as soluções dos subproblemas para gerar a solução do problema original!

- Geralmente os subproblemas possuem somente uma fração do tamanho do problema original
  - Pode reduzir bastante o tamanho dos subproblemas

Divisão e Conquista

- Resolver recursivamente em **três** etapas:
  - **Divisão**
    - Dividir o problema em subproblemas que são instâncias menores do mesmo problema
  - **Conquista**
    - Resolve (conquista) os subproblemas recursivamente, se o subproblema for suficientemente pequeno, basta resolvê-lo de forma direta
  - **Combinação**
    - Combinar soluções fornecidas pela conquista até resolver o problema original

Divisão e Conquista

- Algoritmos criados baseados no paradigma de Divisão e Conquista geralmente possuem recorrências com um formato típico

  - $T(n) = a\,T\left(\frac{n}{b}\right) + f(n)$

  - $a =$ numero de subproblemas criados
  - $b =$ tamanho de cada subproblema em relação ao problema original
  - $f(n) =$ custo da resolução de cada subproblema

Divisão e Conquista

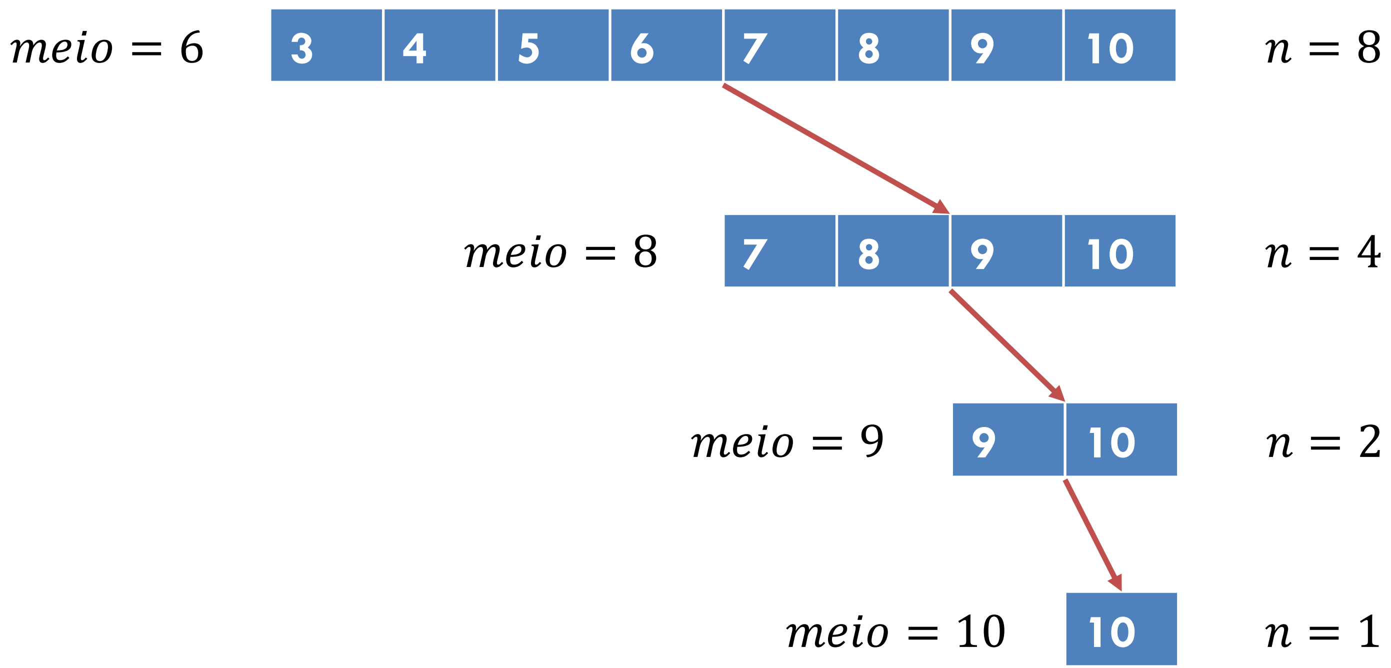
- Como exemplo, vamos considerar o algoritmo de **Busca Binária**

Busca Binária

- Problema
  - Encontrar um elemento em um vetor ordenado
- Ideia Básica
  - Supõe que o vetor de entrada já está ordenado
  - Divide o vetor ao meio, e procurar o elemento em uma das metades baseado em seu valor
    - Repete o processo até encontrar o elemento

Busca Binária

- Exemplo: queremos encontrar o elemento **10**
  - **Suponha que o elemento do meio do vetor seja dado por** $\boldsymbol{meio = \lfloor n/2 \rfloor}$



Busca Binária

- Psedocódigo

```
binarySearch(A, E, L, R)
    if R < L then
        return -1
    m ← floor( ( L + R ) / 2)
    if A[m] < E then
        binarySearch(A, E, m+1, R)
    else if A[m] > E then
        binarySearch(A, E, L, m-1)
    else then       // Caso A[m] == E
        return m
```

Busca Binária

- Exemplo funcionamento : **binarySearch**(A, 9, 0, 7)

3	4	5	6	7	8	9	10

E = 9

L = 0
R = 7
m = 3

A[3] < E
binarySearch(A, 9, m+1, 7)

				7	8	9	10

L = 4
R = 7
m = 5

A[5] < E
binarySearch(A, 9, m+1, 7)

						9	10

L = 6
R = 7
m = 6

A[6] = E
Elemento encontrado!

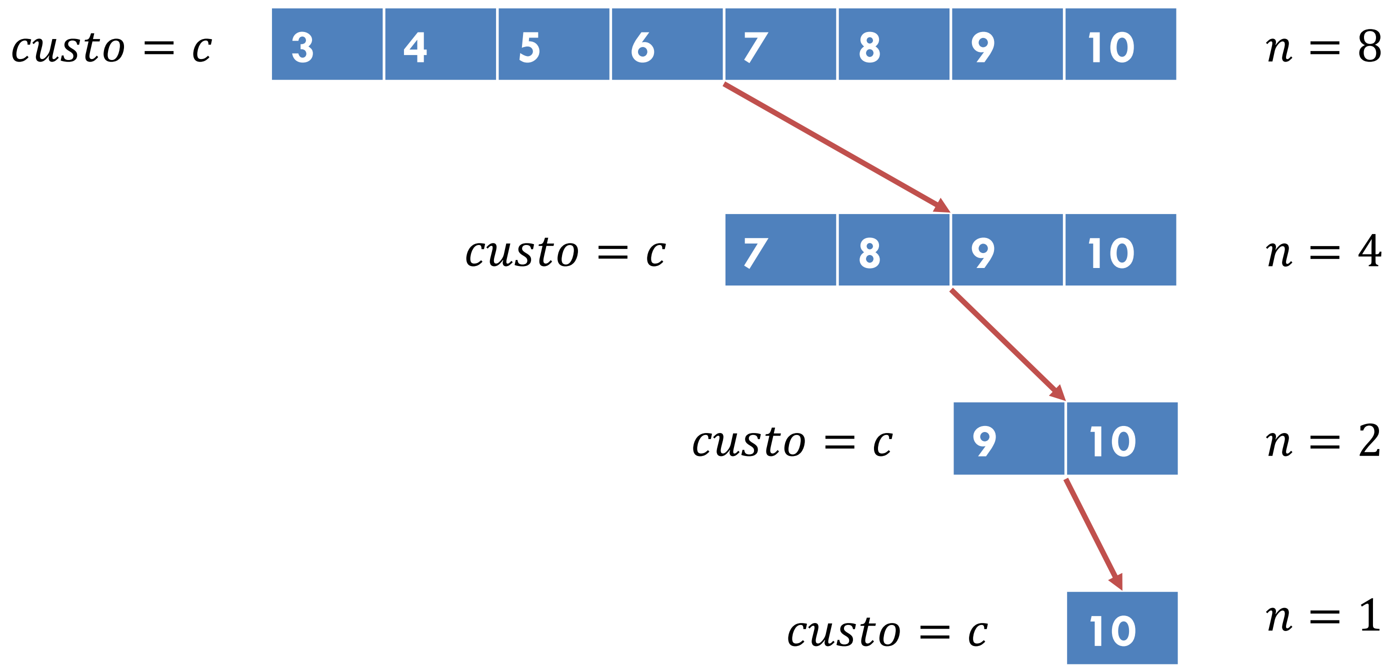
Busca Binária

- Qual a recorrência que descreve o tempo de execução deste algoritmo?

- Relembrando...
  - $T(n) = a\,T\left(\frac{n}{b}\right) + f(n)$
  - a = numero de subproblemas criados
  - b = tamanho de cada subproblema em relação ao problema original
  - $f(n)$ = custo da resolução de cada subproblema

Busca Binária

- Árvore de Recursão
  - para encontrar o elemento 10)



Busca Binária

- Qual a recorrência que descreve o tempo de execução deste algoritmo?
  - $a = 1$
  - $b = 2$
  - $f(n) = c$

$$T(n) = \begin{cases} T(1) = 1 \\ T\left(\frac{n}{2}\right) + c & , n > 1 \end{cases}$$

Merge Sort

Merge Sort

- Problema
  - Ordenar um vetor
- Ideia Básica
  - Usar a técnica de divisão e conquista
  - Dividir o vetor ao meio até que o caso base, mesclar soluções dos subproblemas menores até o original

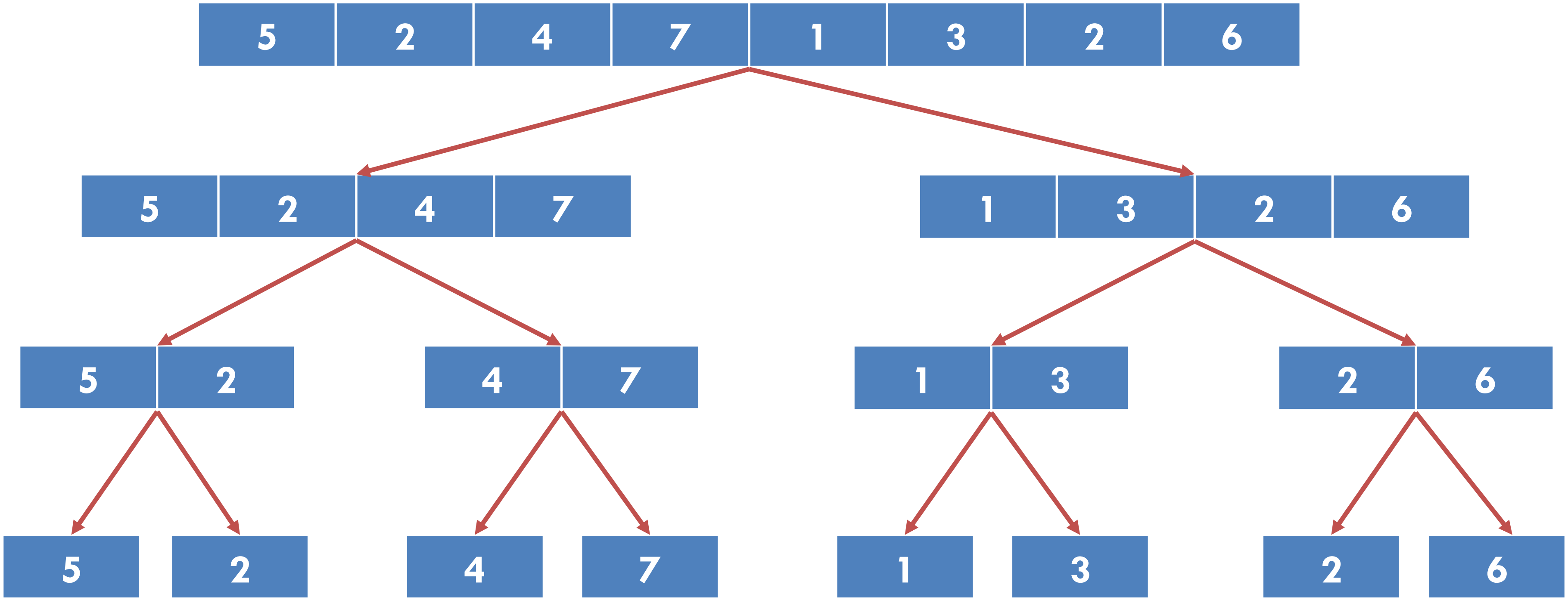
Merge Sort

- Psedocódigo (função **mergeSort**)

```
mergeSort(A, p, r)
    if p < r then
        q ← floor( ( p + r ) / 2)
        mergeSort (A, p, q)
        mergeSort (A, q+1, r)
        merge(A, p, q, r)
```

Merge Sort

- Divisão (função **mergeSort**)



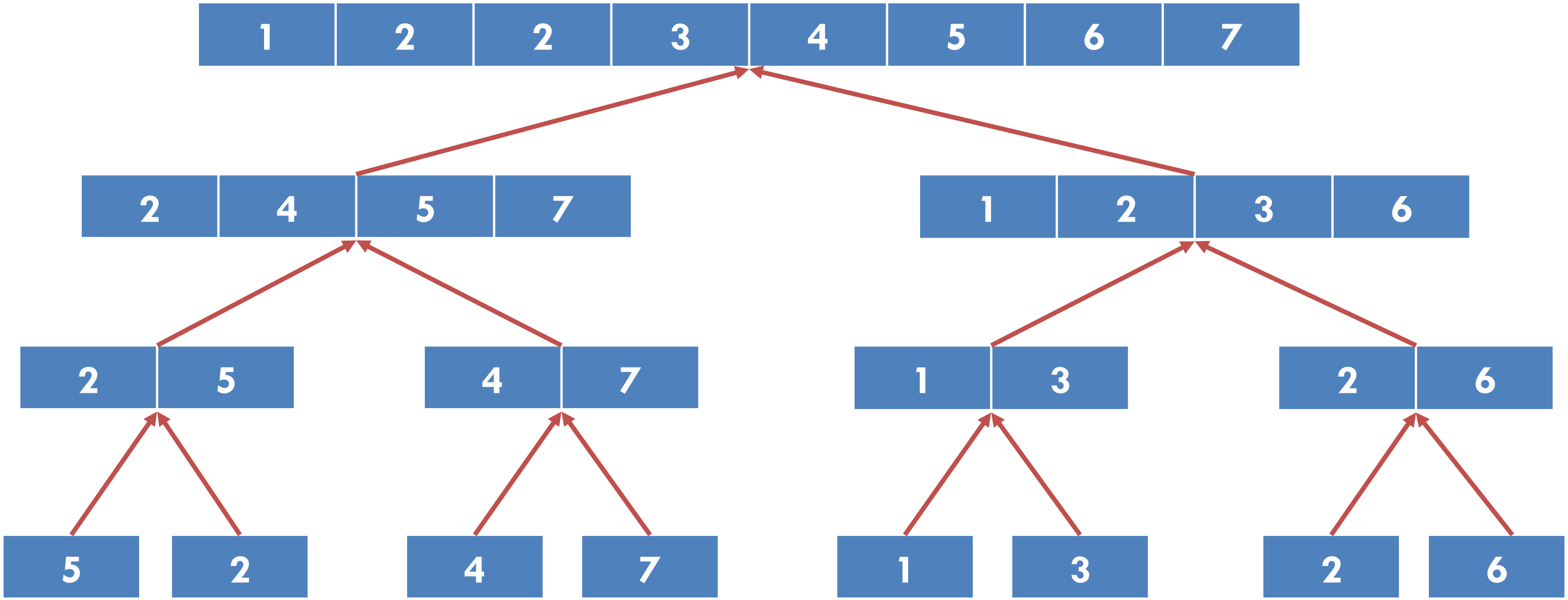
Merge Sort

- Psedocódigo (função **merge**)

```
merge(A, p, q, r)
    L ← A[p .. q]
    R ← A[q+1 .. r]
    L.inserir(INF)
    R.inserir(INF)
    i ← j ← 1
    for k ← p to r do
        if L[i] < R[j] then
            A[k] ← L[i]
            i ← i + 1
        else then
            A[k] ← R[j]
            j ← j + 1
```

Merge Sort

- Conquista (função **merge**)



Merge Sort

- Qual a recorrência que descreve o tempo de execução deste algoritmo?
  - $a = 2$
  - $b = 2$
  - $f(n) = n$

Está correto?

$$T(n) = \begin{cases} T(1) = 1 \\ 2T\left(\frac{n}{2}\right) + n & , n > 1 \end{cases}$$

Merge Sort

- Essa recorrência só é válida caso n seja potência de 2!

- A recorrência para qualquer valor de n é:

$$T(n) = \begin{cases} T(1) = 1 \\ T\left(\left\lceil \frac{n}{2} \right\rceil\right) + T(\left\lfloor \frac{n}{2} \right\rfloor) \ + c\ n \end{cases}$$