MergeSort, Divisão e Conquista

Fernando Lobo

Algoritmos e Estrutura de Dados

MergeSort

- Nesta aula vamos aprender e analisar outro algoritmo de ordenação: MergeSort.
- O algoritmo faz uso de uma técnica chamada divisão e conquista.
- Na realidade, a técnica deveria chamar-se dividir, conquistar, e combinar.
- Trata-se de uma técnica que é usada frequentemente para resolver problemas.

Divisão e Conquista

3 passos essenciais:

- Divide o problema em vários subproblemas.
- Conquista (resolve) cada um dos problemas recursivamente.
  (caso base: se o problema for suficientemente pequeno, resolve-o de qualquer maneira e feitio.)
- Combina as soluções dos subproblemas para obter a solução do problema original.

Merge Sort

MERGE-SORT$(A, left, right)$

if $left < right$ // Verifica o caso base
  $mid = \lfloor (left + right)/2 \rfloor$ // Divide
  MERGE-SORT$(A, left, mid)$ // Conquista
  MERGE-SORT$(A, mid + 1, right)$ // Conquista
  MERGE$(A, left, mid, right)$ // Combina

Chamada inicial: MERGE-SORT$(A, 1, n)$

Merge

- **Input:** Array A e índices *left*, *mid* e *right* de tal forma que,
  - $left \leq mid < right$.
  - o subarray $A[left \,..\, mid]$ está ordenado.
  - o subarray $A[mid + 1 \,..\, right]$ está ordenado.
- **Output:** array $A[left \,..\, right]$ ordenado.

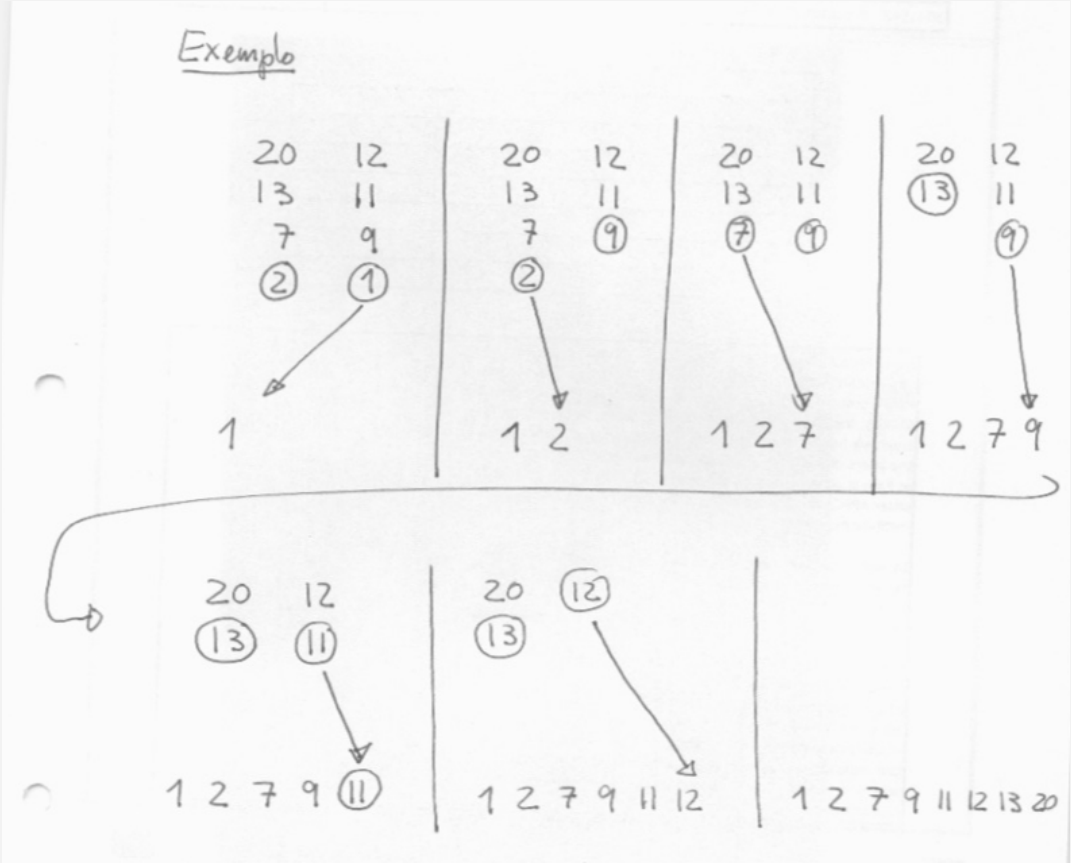
  A operação será feita com complexidade $\Theta(n)$, onde $n = right - left + 1$ é o número total de elemenos.

Operação Merge em tempo linear

Imaginar os 2 subarrays como sendo 2 pilhas. No topo da pilha está sempre o elemento mais pequeno de cada array.

- Comparar o topo de cada um dos sub-arrays e retirar o menor deles.
- Repetir o passo anterior até uma das pilhas ficar vazia.
- Cada um destes passos demora tempo constante e temos no máximo n passos. Logo, a operação será feita em $\Theta(n)$.

Exemplo



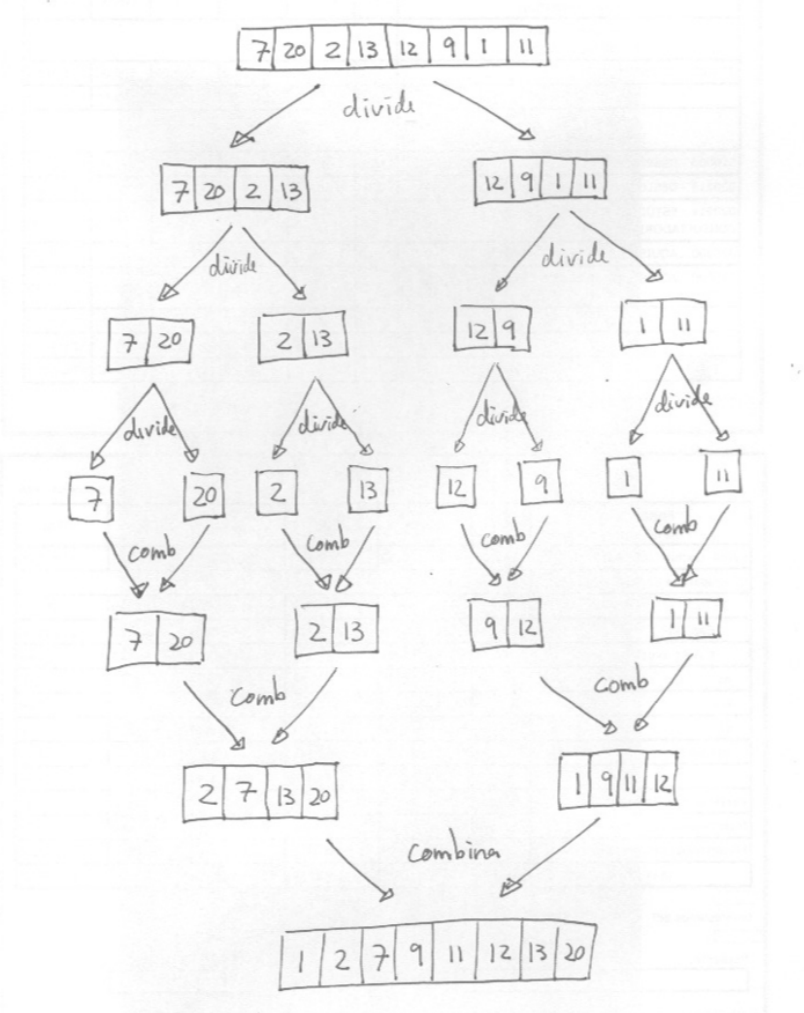
Implementação com sentinelas

- Pode-se implementar de forma eficiente usando sentinelas.
- A ideia é colocar um valor muito grande (infinito) ao fim de cada sub-array.
- Evita-se o teste de verificar se a pilha está vazia.

Pseudocódigo de Merge (com sentinelas)

MERGE($A, left, mid, right$)

$n1 = mid - left + 1$
$n2 = right - mid$
// Cria arrays $L[1\,..\,n1+1]$ e $R[1\,..\,n2+1]$
for $i = 1$ **to** $n1$
    $L[i] = A[left + i - 1]$
for $j = 1$ **to** $n2$
    $R[j] = A[mid + j]$
$L[n1+1] = R[n2+1] = \infty$ // Sentinelas
$i = j = 1$
for $k = left$ **to** $right$
    if $L[i] \leq R[j]$
        $A[k] = L[i]$
        $i = i + 1$
    else $A[k] = R[j]$
        $j = j + 1$



Análise do algoritmo MergeSort

- No caso base, $n = 1$ e o algoritmo nada faz.
- Quando $n \geq 2$, os tempos de execução para os vários passos do algoritmo são os seguintes:
  - **Dividir:** Cálculo de *mid*. $\Rightarrow \Theta(1)$
  - **Conquistar:** Temos de resolver 2 subproblemas de tamanho $n/2$. $\Rightarrow 2T(n/2)$.
  - **Combinar:** Fazer MERGE. $\Rightarrow \Theta(n)$.

Análise do algoritmo MergeSort

$$T(n) = \begin{cases} \Theta(1) & \text{, se } n = 1 \\ \Theta(1) + 2T(n/2) + \Theta(n) & \text{, se } n > 1 \end{cases}$$

$$= \begin{cases} \Theta(1) & \text{, se } n = 1 \\ 2T(n/2) + \Theta(n) & \text{, se } n > 1 \end{cases}$$

Recorrência de MERGE-SORT

Notação O e Θ em expressões

$$T(n) = n^3 + \Theta(n^2)$$

é equivalente a

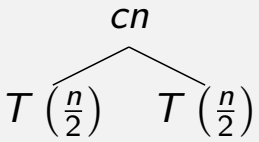
$$T(n) = n^3 + h(n) \text{ , em que } h(n) = \Theta(n^2)$$

Resolução da recorrência

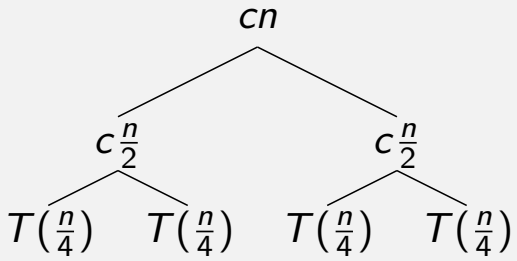
- Método intuitivo: "desenrolar" a recursividade através de uma árvore.
- $T(n) = 2T(n/2) + cn$, com $c > 0$ constante.
  (para já vamos ignorar o caso base.)
- vamos escrever a soma em forma de árvore:

$$T(n) = \begin{array}{c} cn \\ \diagup \quad \diagdown \\ T\left(\frac{n}{2}\right) \quad T\left(\frac{n}{2}\right) \end{array}$$

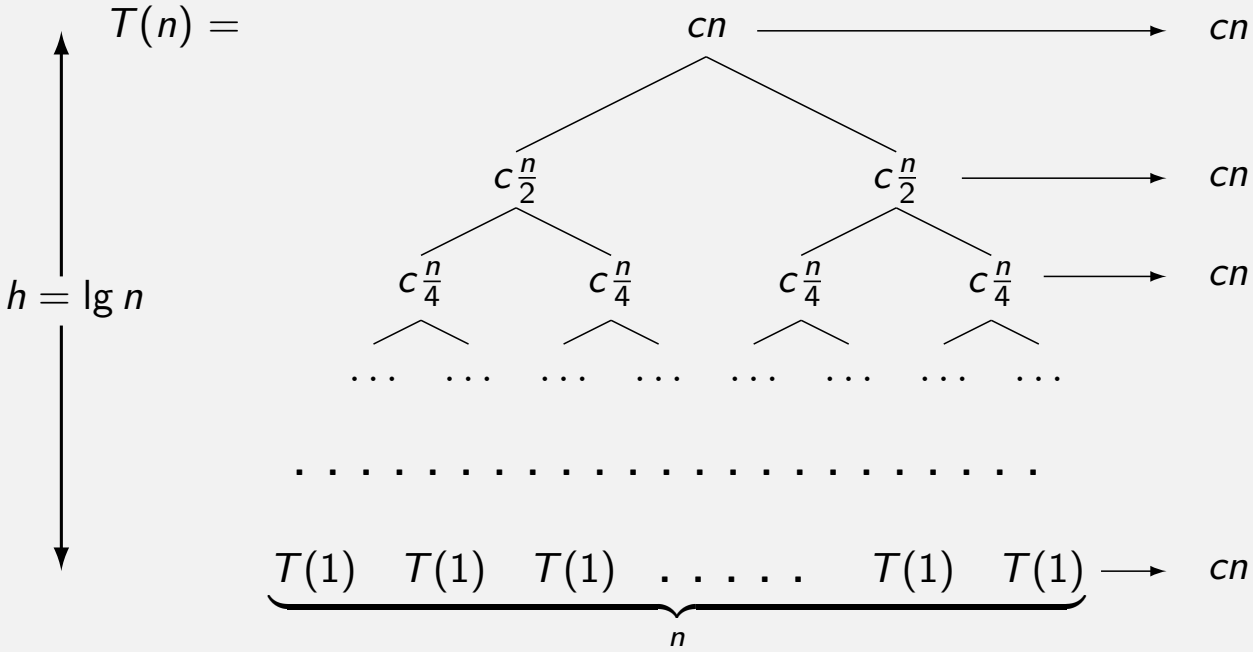
$T(n) =$



$=$



$T(n) =$



Árvore de altura h tem $1 + h$ níveis.

Total: $cn(1 + \lg n)$

$= \Theta(n \lg n)$

MergeSort vs. HeapSort vs. InsertionSort

.

- MERGE-SORT e HEAPSORT são ambos $\Theta(n \lg n)$
- São ambos assimptoticamente mais rápidos que o INSERTION-SORT, que tem complexidade $\Theta(n^2)$.
- Na prática, INSERTION-SORT é melhor para valores pequenos de n. Para n grande, MERGE-SORT ou HEAPSORT são muito melhores.