① 5ª Aula

Problema de ordenação

Algoritmo	Pior caso	Melhor caso	Caso médio	Complexidade de espaço
Insertion-Sort	$\Theta(n^2)$	$\Theta(n)$	$\Theta(n^2)$	$\Theta(n)$
Merge-Sort	$\Theta(n \lg n)$	$\Theta(n \log n)$	$\Theta(n \log n)$	$\Theta(n)$
Selection-Sort	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n^2)$	$\Theta(n)$
Heap-Sort	$O(n \lg n)$	$O(n \lg n)$	$O(n \lg n)$	$\Theta(n)$
Quick-Sort	$\Theta(n^2)$	$\Theta(n \lg n)$	$\Theta(n \lg n)$	? $\Theta(k+n)$
Counting-Sort	$\Theta(k+n)$	$\Theta(k+n)$	$\Theta(k+n)$	? $\Theta(d(n+k))$
Radix-Sort	$\Theta(d(n+k))$	$\Theta(d(n+k))$	$\Theta(d(n+k))$	? $\Theta(n)$
Bucket-Sort	$\Theta(n^2)$	$\Theta(n)$	$\Theta(n)$	

② Insertion-Sort recursivo

```
Insertion-Sort(A, n)
1  if n ≥ 2 do
2      Insertion-Sort(A, n-1)
3      v = A[n]
4      j = n-1
5      while (j > 0) and (A[j] > v) do
6          A[j+1] = A[j]
7          j = j-1
8      A[j+1] = v
```

③ Análise do Insertion-Sort recursivo

→ Quantas comparações e quantas trocas o algoritmo executa no pior caso?

→ Tanto o número de comparações quanto o de trocas é dado pela recorrência:

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

Exercício 1: Resolva a recorrência $T(n)$.

→ $\Theta(n^2)$ comparações e trocas são executadas no pior caso.

④ Algoritmo Selection-Sort

```
Selection-Sort(A)                                  | Custo (Pior caso)
1. for i = 1 to n-1 do                             | Θ(n)
2.     min = i                                     | Θ(n)
3.     for j = i+1 to n do                         | Θ(n²)
4.         if A[j] < A[min] then                   | } Θ(n²)
5.             min = j                             |
6.     A[i] ↔ A[min]                               | Θ(n)
```

Portanto, a complexidade do algoritmo é $\Theta(n^2)$

⑤ Análise do Selection-Sort iterativo

- Complexidade de pior caso (elementos em ordem decrescente):
  $\Theta(n^2)$, com $\Theta(n^2)$ comparações e $\Theta(n)$ trocas
- Complexidade de melhor caso (elementos em ordem crescente):
  $\Theta(n^2)$
- Complexidade de caso médio (vetor aleatório):
  $\Theta(n^2)$
- Complexidade de espaço
  $\Theta(n)$, pois comparações e trocas são efetuadas diretamente no vetor

Note que o Selection-Sort realiza mais comparações que o Insertion-Sort, mas menos trocas!

⑥ Corretude do Selection-Sort

Invariante:

O vetor $A[1..i-1]$ está ordenado e $A[1..i-1] < A[i..n]$

Exercício 2: Demonstre/Prove a corretude do algoritmo Selection-Sort.

⑦ Selection-Sort recursivo

```
Selection-Sort ( A, i, n)
1. if i<n do
2.     min = i
3.     for j = i+1 to n do
4.         if A[j] < A[min] then min = j
5.     t = A[min]
6.     A[min] = A[i]
7.     A[i] = t
8.     Selection-Sort (A, i+1, n)
```

⑧ Análise do Selection-Sort recursivo

→ Quantas comparações e quantas trocas o algoritmo executa no pior caso?

→ O número de comparações é dado pela recorrência:

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

Exercício 3: Resolva a recorrência $T(n)$.

→ $\Theta(n^2)$ comparações são executadas no pior caso.

⑨ Análise do Selection-Sort recursivo

→ O número de trocas é dado pela recorrência:

$$T(n) = \begin{cases} 1 & , \text{se } n = 1 \\ T(n-1) + O(1) & , \text{se } n > 1 \end{cases}$$

Exercício 4: Resolva a recorrência $T(n)$.

→ $\Theta(n)$ trocas são executadas no pior caso.

→ Insertion-Sort e Selection-Sort tem a mesma complexidade assintótica, porém em situações em que a operação de troca é muito custosa, é preferível utilizar o Selection-Sort.

⑩ Algoritmo Merge-Sort

Merge-Sort recursivo:

```
Merge-Sort (A, p, r)
1  if p < r then
2      q = ⌊(p+r)/2⌋
3      Merge-Sort (A, p, q)
4      Merge-Sort (A, q+1, r)
5      Merge (A, p, q, r)
```

Exercício 5: Escreva uma versão iterativa do algoritmo Merge-Sort, determine um invariante e prove a corretude deste algoritmo.

(11) Web sites

www.sorting-algorithms.com

nicholasandre.com.br/sorting

AlgoRythmics (Canal do youtube)