Estruturas de Dados - Ordenação 02

Aula passada

O que vimos:

- O problema de ordenação
- Bubble Sort
- Bubble Sort (versão 2)
- Selection Sort

Complexidades:

Algoritmo	melhor caso	pior caso
Bubble Sort	$O(n^2)$	$O(n^2)$
Bubble Sort (v2)	$O(n)$ (vetor ordenado)	$O(n^2)$
Selection Sort	$O(n^2)$	$O(n^2)$

Insertion Sort

- Ordenação por inserção
- Dado uma estrutura (vetor, lista, ...), construir o vetor final (ordenado) um elemento de cada vez
- Analogia: ordenar as cartas de um baralho
  - para cada carta que você receber, você deve procurar a "posição correta" para ela

Insertion Sort

- Ideia para ordenar:
  - ordenar o vetor $S[p..r-1]$ elemento a elemento
  - para cada novo elemento $S[i]$ que recebermos, devemos colocá-lo na sua posição correta no vetor $S[p..i]$
    - $S[p..i-1]$ é um subvetor ordenado de S, e $S[i]$ é um elemento que possivelmente quebra essa ordenação caso seja adicionado diretamente na posição i
  - após isso, o vetor $S[p..i]$ também estará ordenado
  - fazemos isso para os índices i em $\{2, \ldots, r-1\}$
- Tem a ideia de inserir novos elementos em um conjunto já ordenado
- Corretude: essa ideia funciona?

Insertion Sort

- O vetor unitário $S[p]$ é um vetor ordenado
- Recebemos $S[p+1]$, e procuramos sua posição no subvetor ordenado $S[p..p+1]$
  - se $S[p] > S[p+1]$, então trocamos $S[p]$ por $S[p+1]$
- Recebemos $S[p+2]$, e procuramos sua posição no subvetor ordenado $S[p..p+2]$
  - se $S[p-1] > S[p+2]$, então trocamos $S[p+1]$ por $S[p+2]$
  - se $S[p] > S[p+1]$, então trocamos $S[p]$ por $S[p+1]$

$\vdots$

Insertion Sort: algoritmo

Algoritmo: InsertionSort(S, p, r)

Entrada: vetor S, índices p e r

Saída: vetor S ordenado de p a $r-1$

```
1 para i = p + 1 até r − 1 faça
2 |   x = S[i]
3 |   j = i − 1
4 |   enquanto j > p − 1 e S[j] > x faça
5 |   |   S[j + 1] = S[j]
6 |   |   j = j − 1
7 |   S[j + 1] = x
```

Insertion Sort: complexidade

- Complexidade do Insertion Sort:
  - complexidade média: $O(n^2)$
    - loop das linhas 1 a 7 é $O(n)$
    - loop aninhado das linhas 4 a 6 é $O(n)$
  - melhor caso: $O(n)$
    - vetor S ordenado por completo
    - execução não entra no loop das linhas 4 a 6

Insertion Sort

- Vantagens:
  - simples implementação (inclusive para outras estruturas que não vetor)
  - não necessita de vetor auxiliar; utilizamos apenas uma variável auxiliar (economiza memória)
  - estável
  - tem um melhor caso mais eficiente que Bubble Sort e Selection Sort ($O(n)$ quando o vetor já estiver ordenado)
- Desvantagens:
  - efetua muitas trocas
  - pior caso: $O(n^2)$