CES-11

Algoritmos e Estruturas de Dados

Carlos Alberto Alonso Sanches
Juliana de Melo Bezerra

Alguns algoritmos de ordenação

- A ordenação é o problema mais clássico da computação.
- Inicialmente, veremos algumas das suas resoluções mais simples:
  - Ordenação pelo método da bolha (*BubbleSort*)
  - Ordenação por seleção (*SelectionSort*)
  - Ordenação por inserção (*InsertionSort*)
- Consideraremos sempre a ordenação de um vetor v de índices [1..n].

CES-11

- Alguns algoritmos de ordenação
  - Método da bolha (*BubbleSort*)
  - Ordenação por seleção (*SelectionSort*)
  - Ordenação por inserção (*InsertionSort*)

CES-11

- Alguns algoritmos de ordenação
  - Método da bolha (*BubbleSort*)
  - Ordenação por seleção (*SelectionSort*)
  - Ordenação por inserção (*InsertionSort*)

Método da bolha (*BubbleSort*)

- É um dos algoritmos mais simples e conhecidos.
- Princípio:
  - Os elementos vizinhos são comparados e, caso estejam fora de ordem, são trocados.
  - A propagação dessas comparações permite isolar o maior (ou o menor) elemento do vetor.
  - Repetindo-se esse processo com as demais posições do vetor, é possível ordená-lo completamente.
  - Este método recebe o nome de bolha, pois os elementos 'sobem' até a sua posição final, de modo semelhante a uma bolha em um tubo com água.

Exemplo para n=8

1	44	44	12	12	12	12	6	6
2	55	12	42	42	18	6	12	12
3	12	42	44	18	6	18	18	18
4	42	55	18	6	42	42	42	42
5	94	18	6	44	44	44	44	44
6	18	6	55	55	55	55	55	55
7	6	67	67	67	67	67	67	67
8	67	94	94	94	94	94	94	94

Algoritmo

```
BubbleSort(){
     for (i=1; i<n; i++)
         for (j=1; j<=n-i; j++)
             if (v[j] > v[j+1]) {
                  x = v[j];
                  v[j] = v[j+1];
                  v[j+1] = x;
             }
}
```

- É lento, pois só faz comparações entre posições adjacentes.
- Pode ser melhorado com testes intermediários para verificar se o vetor já está ordenado.
- Mesmo assim, gasta tempo $O(n^2)$ no pior caso.

CES-11

- Alguns algoritmos de ordenação
  - Método da bolha (*BubbleSort*)
  - Ordenação por seleção (*SelectionSort*)
  - Ordenação por inserção (*InsertionSort*)

Ordenação por seleção (*SelectionSort*)

- Procedimento:
  - Selecione o menor elemento do vetor e troque-o com o que está na posição 1.
  - Desconsiderando a primeira posição do vetor, repita essa operação com as restantes.

Exemplo com n=6

	1	2	3	4	5	6
Vetor inicial:	55	62	21	35	48	13
	13	62	21	35	48	**55**
	13	**21**	**62**	35	48	55
	13	21	**35**	**62**	48	55
	13	21	35	**48**	**62**	55
	13	21	35	48	**55**	**62**

Algoritmo

```
SelectionSort() {
    for (i=1; i<n; i++) {
         min = i;
         for (j=i+1; j<=n; j++)
              if (v[j] < v[min])
                   min = j;
         x = v[min];
         v[min] = v[i];
         v[i] = x;
    }
}
```

Sempre gasta tempo $O(n^2)$

CES-11

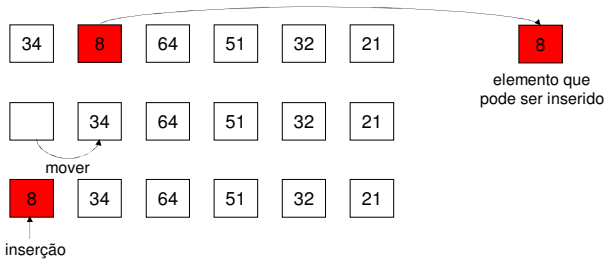
- Alguns algoritmos de ordenação
  - Método da bolha (*BubbleSort*)
  - Ordenação por seleção (*SelectionSort*)
  - Ordenação por inserção (*InsertionSort*)

Ordenação por inserção (*InsertionSort*)

- Semelhante ao método de ordenação das cartas de um baralho.
- Procedimento:
  - Verifica-se se o valor da posição 2 do vetor poderia ser colocado na posição 1.
  - Repete-se este processo para as posições subsequentes, verificando-se o local adequado da inserção.
- A inserção de um elemento na sua nova posição exige a movimentação de vários outros.

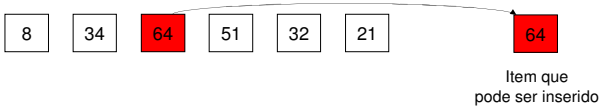
Exemplo com n=6

Passo 1



Exemplo com n=6

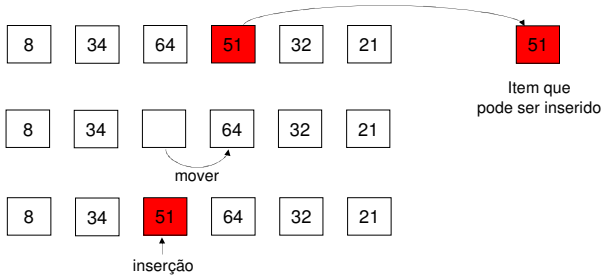
Passo 2



Não ocorre inserção, pois esse elemento já está no seu lugar

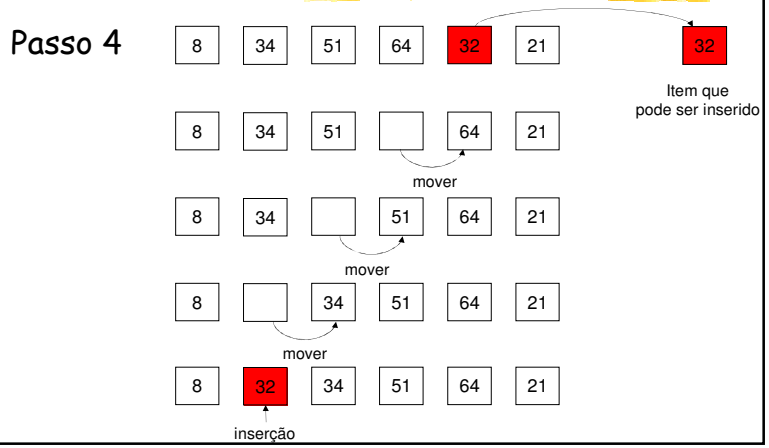
Exemplo com n=6

Passo 3



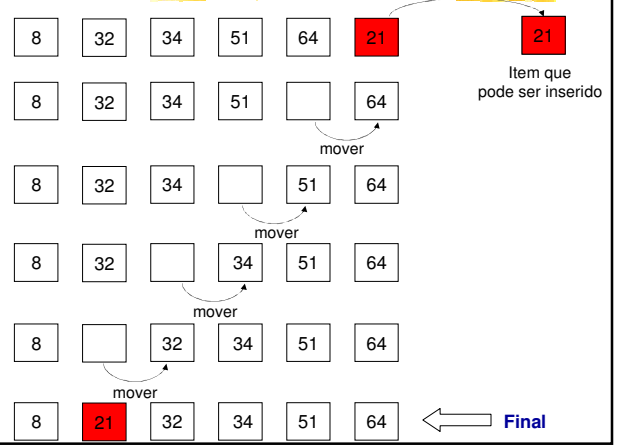
Exemplo com n=6

Passo 4



Exemplo com n=6

Passo 5



Algoritmo

```
InsertionSort() {
    for (i=2; i<=n; i++) {
        x = v[i];
        for (j=i; j>1 && x<v[j-1]; j--)
            v[j] = v[j-1];
        v[j] = x;
    }
}
```

Gasta tempo $O(n^2)$ no pior caso.