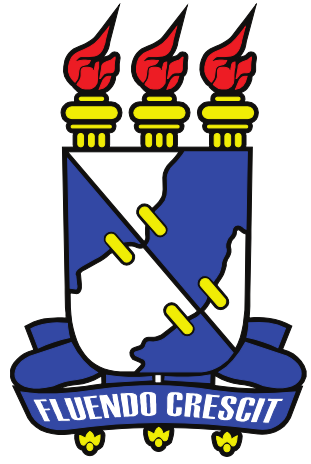


Estrutura de Dados I

Ordenação por inserção

Bruno Prado

Departamento de Computação / UFS

Introdução

- O que é ordenação?
  - É um algoritmo que coloca um conjunto de elementos em uma determinada ordem
  - Ordem numérica (crescente ou decrescente)
  - Ordem lexicográfica (alfabética)

Introdução

- Por que ordenação é importante?
  - Organização e busca eficiente das informações
    - Ata de presença
    - Lista de contatos
    - Banco de dados
    - ...

Introdução

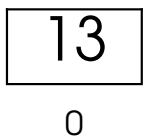
- Por que ordenação é importante?
  - Imagine trabalhar sem ordenação
    - Listagem de nomes em ordem aleatória
    - Livros com páginas fora de ordem
    - Dicionário sem ordem alfabética
    - ...

Ordenação por Inserção

- O que é necessário para ordenar?
  - Conhecer o tipo de dados usado
    - Números
    - Letras
    - ...
  - Definir um critério de ordenação mensurável
    - Crescente
    - Decrescente
    - ...

Ordenação por Inserção

- Tipo de dados
  - Pelo menos duas posições para armazenamento
  - Definição de operações de $>$, $<$ e $==$



Laranja > Abacaxi ?

Ordenação por Inserção

- Critério de ordenação
  - Define que aspecto será medido
  - Medida de preço por quilo
  - Ordem crescente

Laranja	Banana	Pitomba	Kiwi
R$ 2,50	R$ 3,00	R$ 0,50	R$ 9,00

Pitomba	Laranja	Banana	Kiwi
0	1	2	3

Ordenação por Inserção

- Ideia do algoritmo
  - É baseado em comparação, consistindo em inserir de forma ordenada os elementos nas posições
  - São feitas comparações e trocas de posições até que todos os elementos do vetor sejam verificados
  - Um método similar é utilizado na ordenação de cartas de baralho

Ordenação por Inserção

- Implementação em C

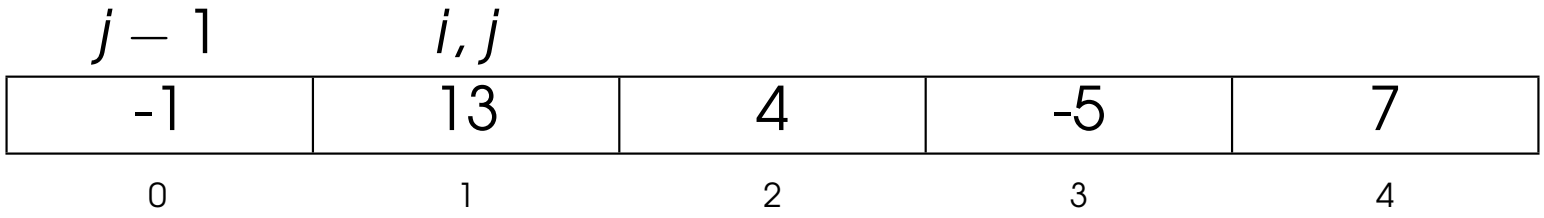
```
void ordenacao_por_insercao(int V(), int n) {
        int i, j;
        for(i = 1; i < n; i = i + 1)
                for(j = i; j > 0 && V(j - 1) > V(j); j–)
                        trocar(&V(j), &V(j - 1));
}
```

-1	13	4	-5	7
0	1	2	3	4

Ordenação por Inserção

- Implementação em C

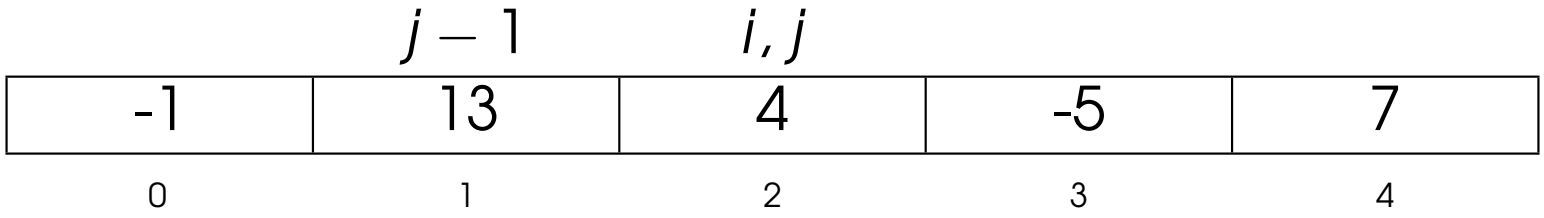
```
void ordenacao_por_insercao(int V(), int n) {
        int i, j;
        for(i = 1; i < n; i = i + 1)
                for(j = i; j > 0 && V(j - 1) > V(j); j–)
                        trocar(&V(j), &V(j - 1));
}
```



Ordenação por Inserção

- Implementação em C

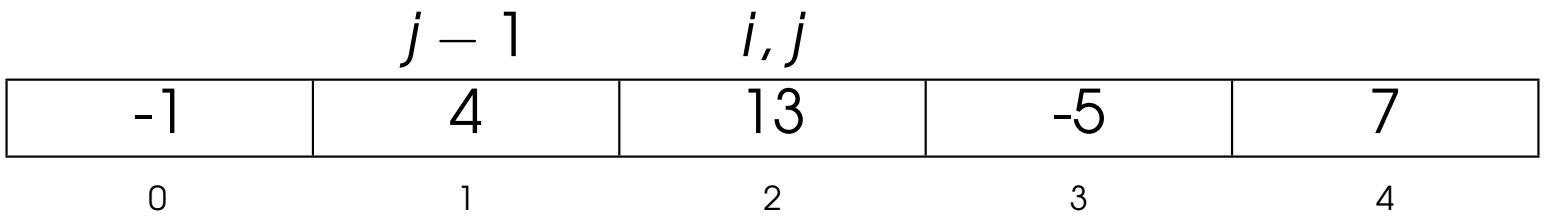
```
void ordenacao_por_insercao(int V(), int n) {
        int i, j;
        for(i = 1; i < n; i = i + 1)
                for(j = i; j > 0 && V(j - 1) > V(j); j–)
                        trocar(&V(j), &V(j - 1));
}
```



Ordenação por Inserção

- Implementação em C

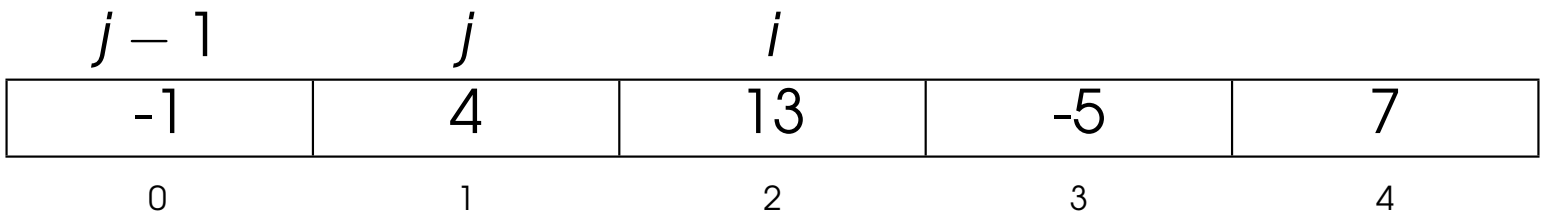
```
void ordenacao_por_insercao(int V(), int n) {
        int i, j;
        for(i = 1; i < n; i = i + 1)
                for(j = i; j > 0 && V(j - 1) > V(j); j–)
                        trocar(&V(j), &V(j - 1));
}
```



Ordenação por Inserção

- Implementação em C

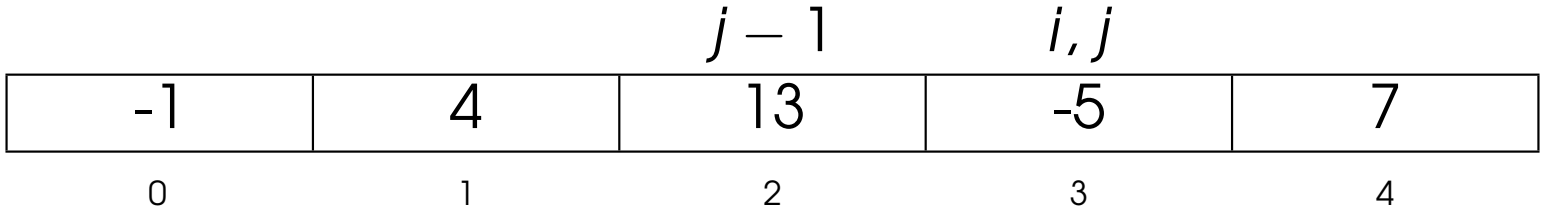
```
void ordenacao_por_insercao(int V(), int n) {
        int i, j;
        for(i = 1; i < n; i = i + 1)
                for(j = i; j > 0 && V(j - 1) > V(j); j–)
                        trocar(&V(j), &V(j - 1));
}
```



Ordenação por Inserção

- Implementação em C

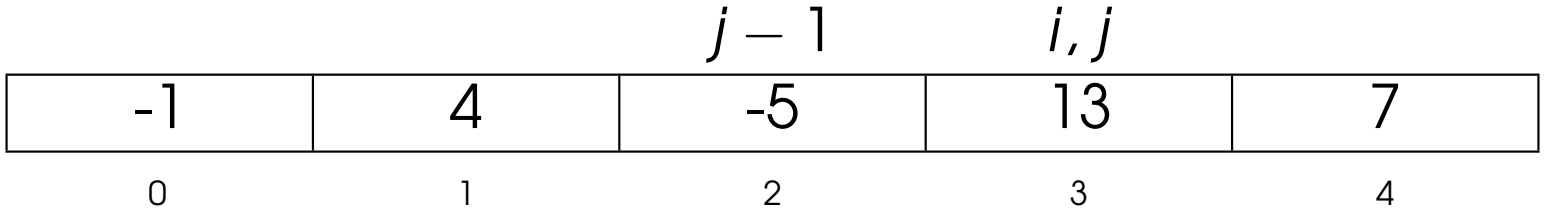
```
void ordenacao_por_insercao(int V(), int n) {
        int i, j;
        for(i = 1; i < n; i = i + 1)
                for(j = i; j > 0 && V(j - 1) > V(j); j–)
                        trocar(&V(j), &V(j - 1));
}
```



Ordenação por Inserção

- Implementação em C

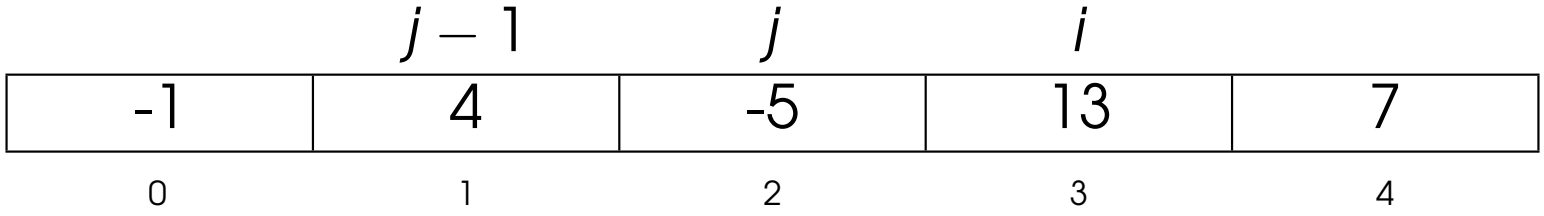
```
void ordenacao_por_insercao(int V(), int n) {
        int i, j;
        for(i = 1; i < n; i = i + 1)
                for(j = i; j > 0 && V(j - 1) > V(j); j–)
                        trocar(&V(j), &V(j - 1));
}
```



Ordenação por Inserção

- Implementação em C

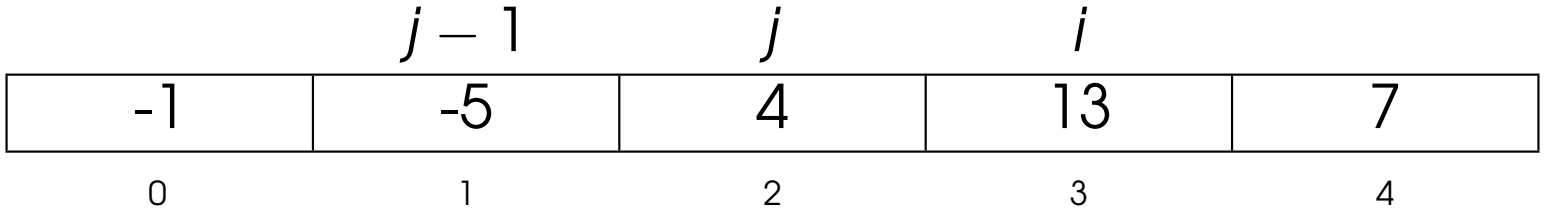
```
void ordenacao_por_insercao(int V(), int n) {
        int i, j;
        for(i = 1; i < n; i = i + 1)
                for(j = i; j > 0 && V(j - 1) > V(j); j–)
                        trocar(&V(j), &V(j - 1));
}
```



Ordenação por Inserção

- Implementação em C

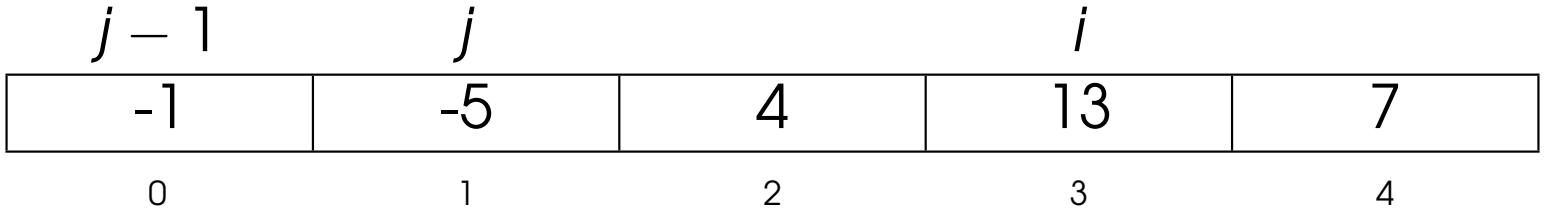
```
void ordenacao_por_insercao(int V(), int n) {
        int i, j;
        for(i = 1; i < n; i = i + 1)
                for(j = i; j > 0 && V(j - 1) > V(j); j–)
                        trocar(&V(j), &V(j - 1));
}
```



Ordenação por Inserção

- Implementação em C

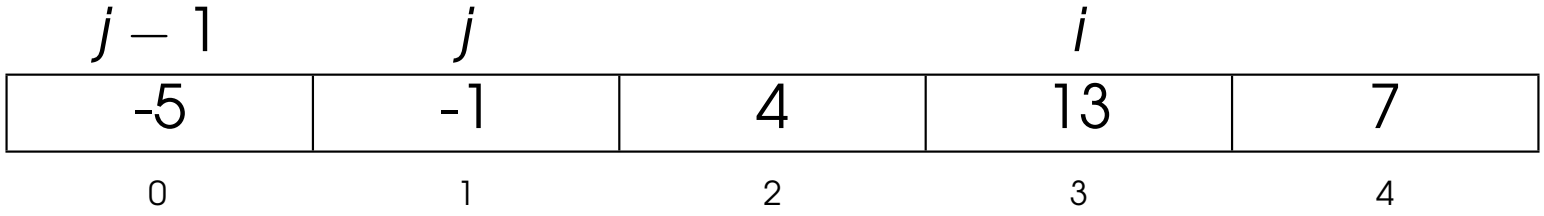
```
void ordenacao_por_insercao(int V(), int n) {
        int i, j;
        for(i = 1; i < n; i = i + 1)
                for(j = i; j > 0 && V(j - 1) > V(j); j–)
                        trocar(&V(j), &V(j - 1));
}
```



Ordenação por Inserção

- Implementação em C

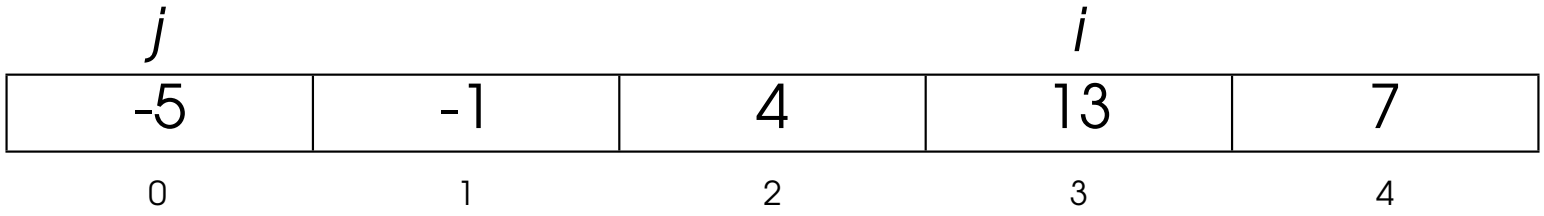
```
void ordenacao_por_insercao(int V(), int n) {
        int i, j;
        for(i = 1; i < n; i = i + 1)
                for(j = i; j > 0 && V(j - 1) > V(j); j–)
                        trocar(&V(j), &V(j - 1));
}
```



Ordenação por Inserção

- Implementação em C

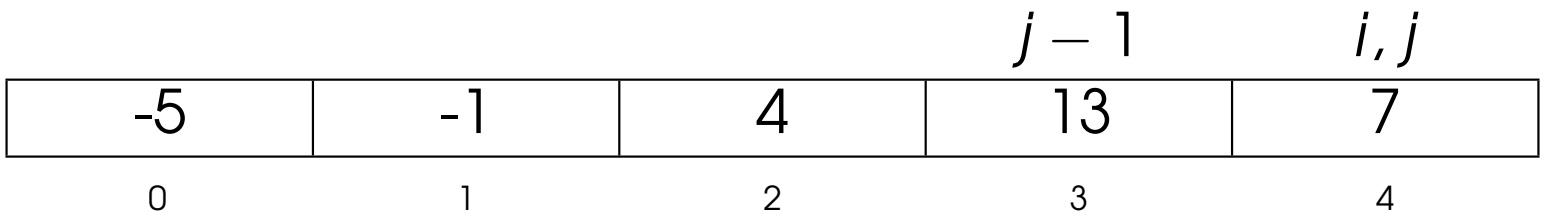
```
void ordenacao_por_insercao(int V(), int n) {
        int i, j;
        for(i = 1; i < n; i = i + 1)
                for(j = i; j > 0 && V(j - 1) > V(j); j–)
                        trocar(&V(j), &V(j - 1));
}
```



Ordenação por Inserção

- Implementação em C

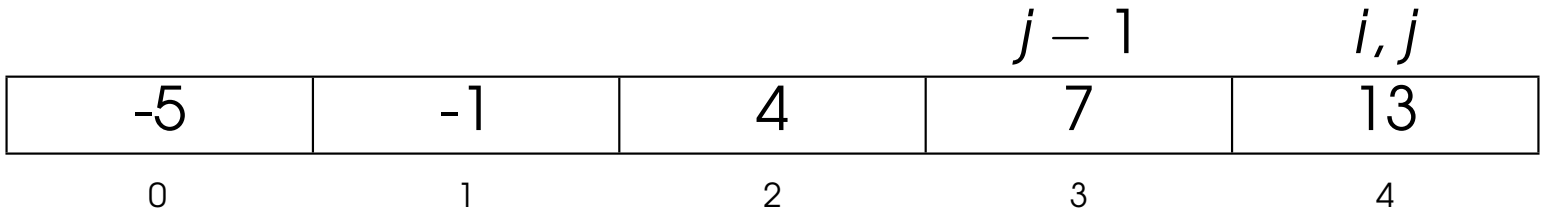
```
void ordenacao_por_insercao(int V(), int n) {
        int i, j;
        for(i = 1; i < n; i = i + 1)
                for(j = i; j > 0 && V(j - 1) > V(j); j–)
                        trocar(&V(j), &V(j - 1));
}
```



Ordenação por Inserção

- Implementação em C

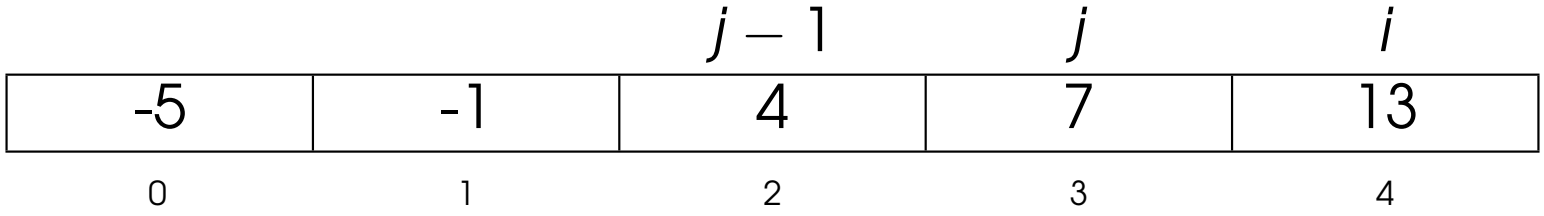
```
void ordenacao_por_insercao(int V(), int n) {
        int i, j;
        for(i = 1; i < n; i = i + 1)
                for(j = i; j > 0 && V(j - 1) > V(j); j–)
                        trocar(&V(j), &V(j - 1));
}
```



Ordenação por Inserção

- Implementação em C

```
void ordenacao_por_insercao(int V(), int n) {
        int i, j;
        for(i = 1; i < n; i = i + 1)
                for(j = i; j > 0 && V(j - 1) > V(j); j–)
                        trocar(&V(j), &V(j - 1));
}
```



Ordenação por Inserção

- Implementação em C

```
void ordenacao_por_insercao(int V(), int n) {
        int i, j;
        for(i = 1; i < n; i = i + 1)
                for(j = i; j > 0 && V(j - 1) > V(j); j–)
                        trocar(&V(j), &V(j - 1));
}
```

-5	-1	4	7	13
0	1	2	3	4

Ordenação por Inserção

- Análise de complexidade
  - Espaço $O(1)$
  - Tempo
    - Melhor caso $\Omega(n)$ com a sequência de números parcialmente ou totalmente ordenada
    - Pior caso $O(n^2)$ que ocorre para um conjunto de dados sem ordenação

Ordenação por Inserção

- Vantagens
  - Implementação simples e in-place
  - Eficiente para conjuntos de dados pequenos
  - Permite a ordenação em tempo de execução
  - É estável

Exemplo

- Considerando a sequência de elementos, realize a ordenação crescente e decrescente usando o algoritmo de ordenação por inserção
  - Execute passo a passo
  - Indique os índices *i* e *j* sendo atualizados

14	2	-3	55	-7
0	1	2	3	4