Grupo:
- Felipe da Silva Braga
- Ícaro Santos Pereira
- Yasser Schuck Antonio Tuma

Algoritmos de ordenação

—

Bubble sort e Selection sort

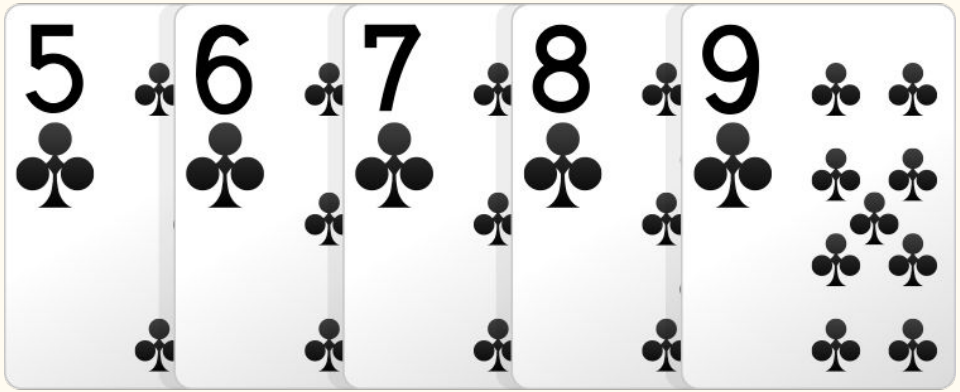
Conteúdo

- Ordenação
- Bubble Sort
  - Características
  - Algoritmo (versão melhorada)
  - Complexidade
  - Vantagem e Desvantagem
- Selection Sort
  - Características
  - Algoritmo
  - Complexidade
  - Vantagem e Desvantagem
- Avaliação de 100000 números

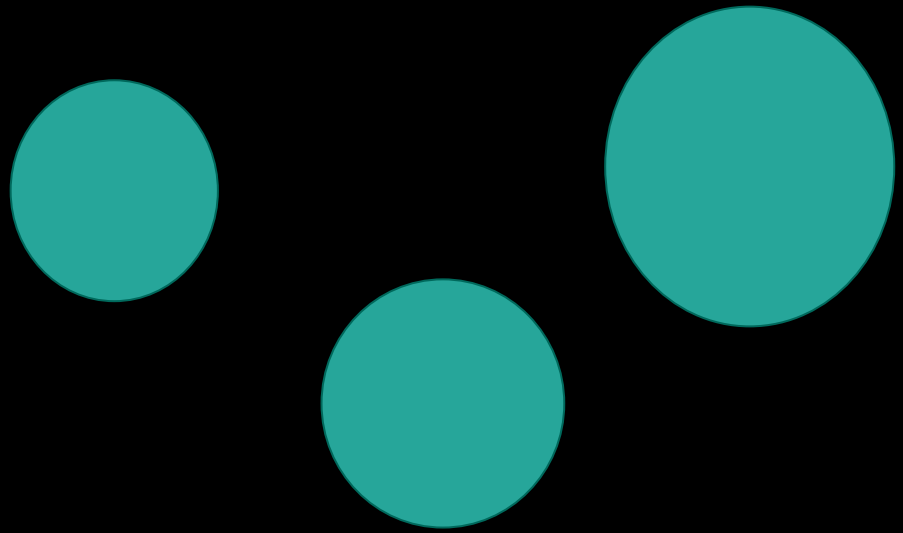
Ordenação

Corresponde ao processo de rearranjar um conjunto de objetos em ordem crescente ou decrescente.

É um processo bastante utilizado na computação. Visto que dados ordenados garantem uma melhor performance de pesquisa a uma Estrutura de Dados.



Bubble Sort



Bubble Sort.

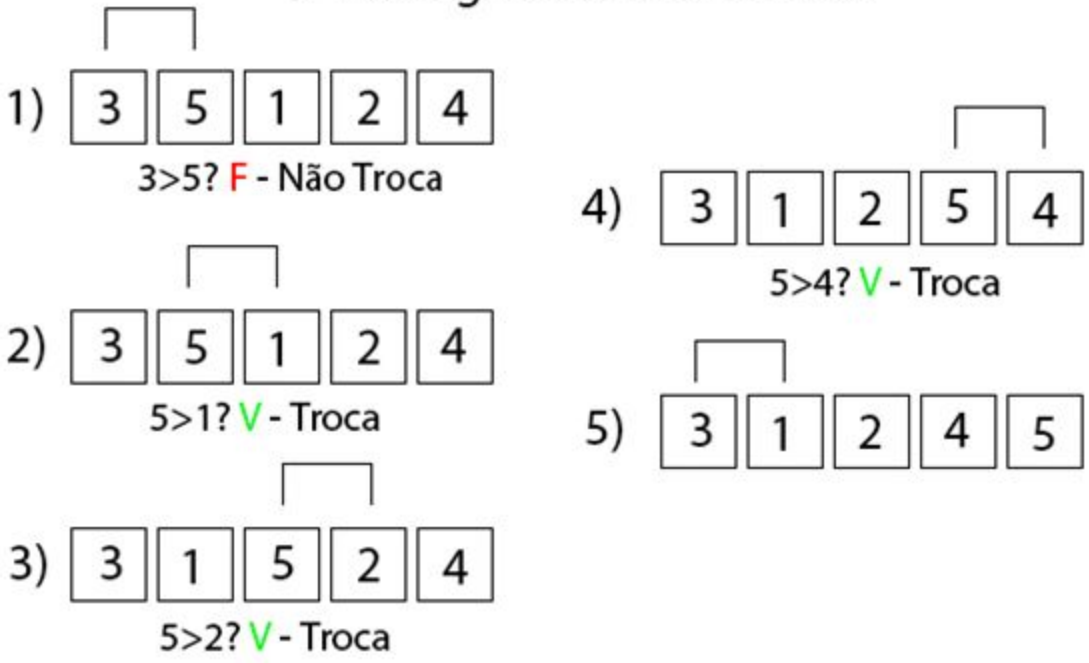
Recebeu este nome pela imagem pitoresca usada para descrevê-lo: os elementos maiores são mais leves, e sobem como bolhas até suas posições corretas.

- Ordena através de sucessivas trocas entre pares de elementos do vetor

Características:

- Fácil implementação;
- alta complexidade;
- Comparação entre elementos adjacentes;
- Ordenação completa;
- Após a primeira varredura o maior elemento da estrutura já está na sua respectiva posição.

1ª Passagem do Bubble Sort

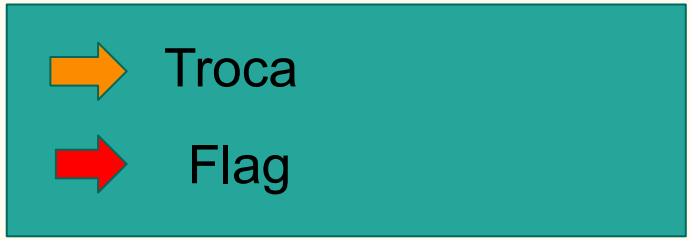


Algoritmo Bubble Sort (melhorado)

```
void bubbleSort( int vetor[], int tam){

     int lacoInterno, trocou=1, aux, novoTam=tam-1, guarda;

     while( trocou == 1){
          trocou=0;
          lacoInterno=0;
          while(lacoInterno < novoTam){
               if(vetor[lacoInterno] > vetor[lacoInterno+1]){
                    aux=vetor[lacoInterno];
                    vetor[lacoInterno]=vetor[lacoInterno+1];
                    vetor[lacoInterno+1]=aux;
                    trocou=1;
                    guarda=lacoInterno;
               }
               lacoInterno= lacoInterno+1;
          }
          novoTam=guardou
     }
}
```



Complexidade

O algoritmo realizará n-1 trocas para o primeiro passo, depois n-2 trocas para o segundo elemento e assim sucessivamente. Trocas = n-1+n-2+n-3+...+2+1 aproximadamente N^2 trocas.

$C(n)=O(n^2)$

Atividades mais custosas:

- Comparações
- Troca de posições

Algoritmo	Pior Caso	Caso médio	Melhor Caso
BubbleSort	$O(N^2)$	$O(N^2)$	O(N)*

*Quando for a versão melhorada do algoritmo, com flag e redução no número de verificação.

Bubble Sort

Vantagens:

- Fácil de implementar
- Fácil de entender
- É mais eficiente quando o vetor já está ordenado
- Não precisa de memória externa

Desvantagens:

- Trocas excessivas
- Extremamente lento

Selection Sort.

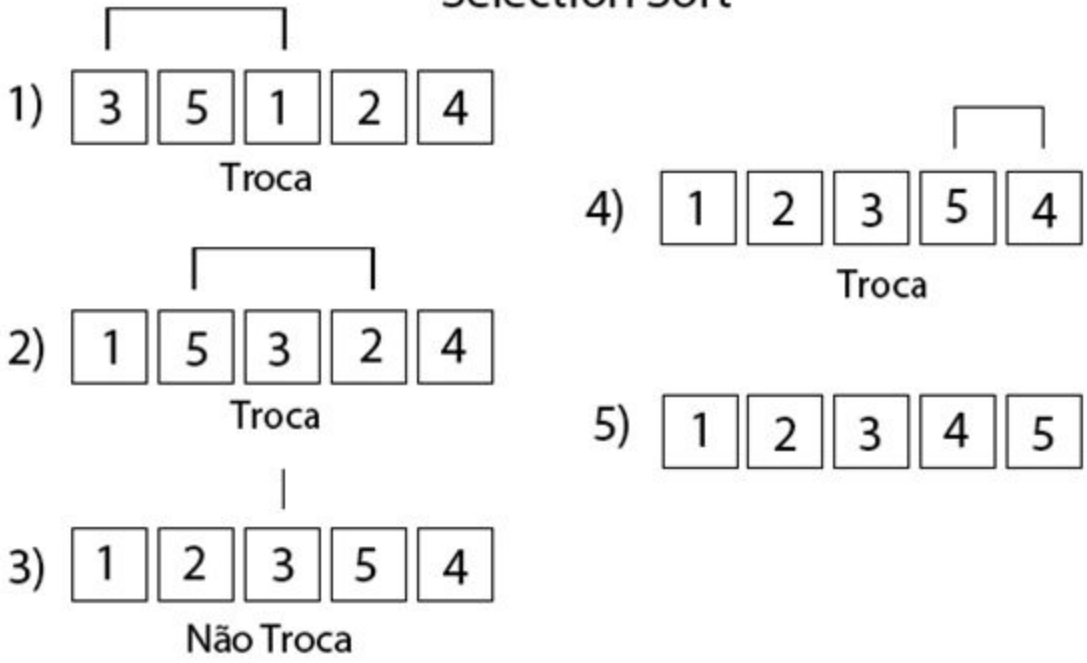
Selection Sort.

- Seleciona o menor elemento do conjunto
- Troque com o primeiro elemento

Características:

- Fácil implementação;
- alta complexidade;
- Após a primeira varredura o menor elemento da estrutura já está na sua respectiva posição.

Selection Sort



Algoritmo Selection Sort

```
void selectionSort( int vetor[], int tam){

    int lacoInterno, lacoExterno, aux, indiceMenor;

    for( lacoExterno=0; lacoExterno < tam-1; lacoExterno++){
        indiceMenor=lacoExterno;
        for(lacoInterno=lacoExterno+1; lacoInterno < tam; lacoInterno++){
            if(vetor[lacoInterno] < vetor[lacoExterno]){
                indiceMenor=lacoInterno;
            }
        }
        aux=vetor[lacoExterno];
        vetor[lacoExterno]=vetor[indiceMenor];
        vetor[indiceMenor]=aux;
    }
}
```

Troca

Complexidade

Para fazer a análise levamos em consideração 3 quesitos:

1. O tempo de execução para todas as chamadas do indiceMenor
2. O tempo de execução para todas as chamas de troca
3. O tempo de execução para a fazer a busca.

Os quesitos 2 e 3 são O(n) no pior dos casos, mas o 1 quesito tem uma interação de 1+2+...+n-1+n é uma série aritmética e ela resulta em $(n^2)/2 + (n/2)$ o que resulta em $O(n^2)$ na notação do Big O. $C(n)=O(n^2)$

Algoritmo	Pior Caso	Caso médio	Melhor Caso
Selection Sort	$O(N^2)$	$O(N^2)$	$O(N^2)$

Selection Sort

Vantagens:

- Custo linear no tamanho da entrada para o número de movimentos de registros
- É o algoritmo a ser utilizado para arquivos com registros muito grandes

Desvantagens:

- O fato de o algoritmo já estar ordenado não ajuda em nada, pois o custo continua quadrático

Avaliação com 100000 números

Avaliação com 100000 números

Ordem 1: lista em ordem decrescentes

Ordem 2: lista em ordem crescentes

Ordem 3: primeira metade da lista em ordem crescente e a segunda metade em ordem decrescente

Ordem 4: primeira metade da lista em ordem decrescente e a segunda metade em ordem crescente

Ordem 5: lista totalmente desordenada

	Tempo em segundos				
Algoritmo	Odem 1	Ordem 2	Ordem 3	Ordem 4	Ordem 5
BubbleSort	32,550	0,210	14,276	14,310	39,785
SelectionSort	20,596	12,621	14,550	12,505	12,623

Obrigado!

Link do github: https://github.com/Oyaho/AED-II

Referência

-SOUZA, Jairo. Método Bubble Sort. Minas Gerais, Brasil. Disponível em: < http://www.ufjf.br/jairo_souza/files/2009/12/2-Ordenação-BubbleSort.pdf>, Acesso em: 17 agosto, 2019, 00:13.

-DROZDEK, Adam. Estrutura de dados e algoritmos em C++.3.ed.rev.Pensilvana: EUA, 205.

-LAFORE, Robert. Estrutura de dados e algoritmos em Java. 2. ed. rev. Califórnia: EUA, 2003.

-TENEMBAUM, LANGSAM, AUGENSTEIN. Estrutura de dados usando C. Nova Jersey: EUA, 1985. Disponível em:<http://www.cin.ufpe.br/~garme/public(ebook)Estrutura%20de%20Dados%20Usando%20C%20C(Tenenmbaum).pdf>. Acesso em: 17 de agosto, 2019. 02:27.

-Cid Carvalho de Souza. Complexidade de Algoritmos I, São Paulo, Brasil. Disponível em:<http://www.ic.unicamp.br/~zanoni/mo417/2011/aulas/handout/04-ordenacao.pdf>. Acesso em: 17 de agosto, 2019. 03:59

-Imagens. Disponível em:<http://www.devmedia.com.br/algoritmos-de-ordenacao-analise-e-comparacao/28161>. Acesso em: 18 de agosto, 2019. 17:00