Linguagem de Programação

Ordenação

ECT2303

helton.maia@ect.ufrn.br

Ordenação de *Arrays*

Qual a importância?

Ordenação de *Arrays* - Introdução

Imagine como seria a busca de um endereço ou dados pessoas em uma base de dados com milhões de registros não organizados. Por exemplo, considere buscar um número de telefone de um usuário em um banco de dados onde estes nomes não estão organizados em ordem alfabética.
Como essa busca poderia ser otimizada?

A ordenação permite facilitar a recuperação desses dados, possibilitando buscas e pesquisas de ocorrências de determinado elemento(chave) em um conjunto de dados ordenado.

Algoritmos de ordenação

Alguns exemplos bastante utilizados:

- Bolha - *Bubble Sort*: $O(n^2)$
- Seleção: *Selection Sort* $O(n^2)$
- Inserção*: Insertion Sort*: $O(n^2)$
- *Quick Sort* $\theta(n^2)$

Algoritmos de ordenação

Bubble Sort (método da bolha): Deve-se percorrer o *array* diversas vezes e, a cada iteração o elemento de maior valor deve ser levado para o final do array (ou o de menor valor para o início).

Considerações:

- É um método de fácil entendimento e implementação;
- Para grandes *arrays,* pode ser um método computacionalmente custoso.

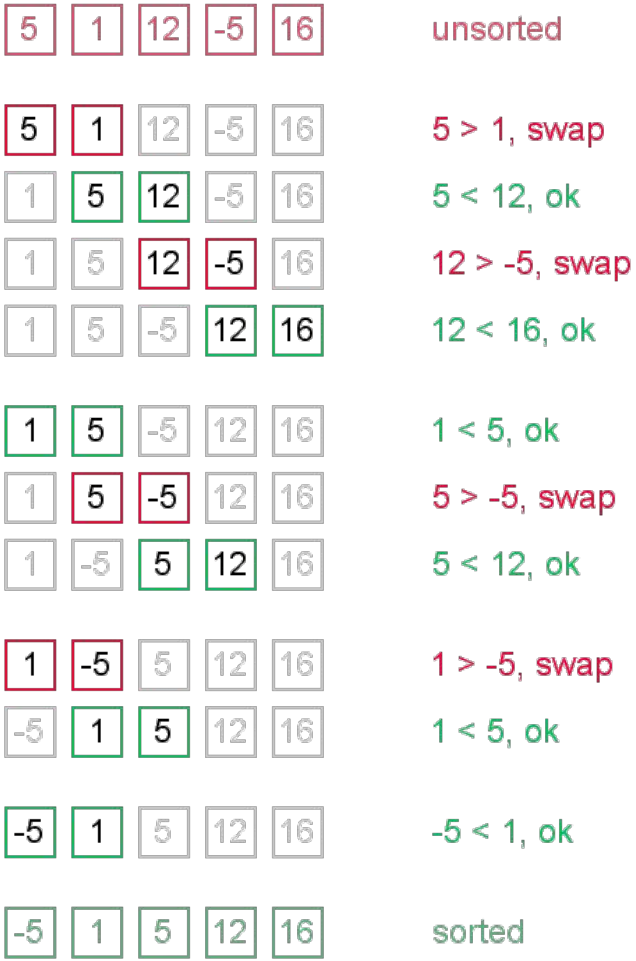
Algoritmos de ordenação

Seria útil verificar se o array está ordenado?

Verificando se os dados estão ordenados!

```cpp
#include <iostream>
#define TAM 100
using namespace std;

int main(){
    //int valores[TAM] = {11, 4, 2, 6, 3, 7, 8, 9, 3,  6};
    int valores[TAM] = { 2, 3, 3, 4, 6, 6, 7, 8, 9, 11};
    const int n = 10;
    bool flag = true;

    for(int i = 0; i < n-1; i++){
        if(valores[i] > valores[i+1]){
            flag = false;
            break;
        }
    }

    cout << "ARRAY: ";
    for(int i=0;i<n;i++){
        cout << valores[i] << " ";
    }
    cout << endl;

    if(flag == true){
        cout << "Os valores do array estao ordenados." << endl;
    } else{
        cout << "Os valores do array n estao ordenados." << endl;
    }

    return 0;
}
```

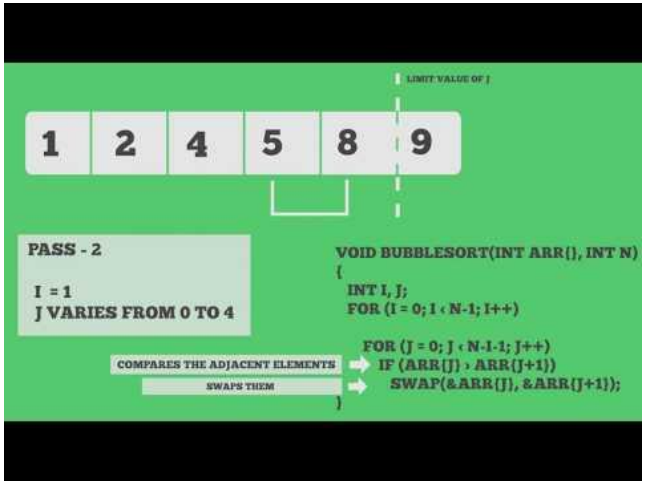
Bubble Sort




```
26  void bubbleSort(int v[MAX], int n){
27      for (int i=0; i<n-1; i++){
28          for(int j=0; j<n-1-i; j++){
29              if(v[j] > v[j+1]){
30                  int aux = v[j];
31                  v[j]    = v[j+1];
32                  v[j+1]  = aux;
33              } //troca()
34          }
35      }
36  }
```

Algoritmos de ordenação: *Bubble Sort*

vídeo: https://youtu.be/lyZQPjUT5B4

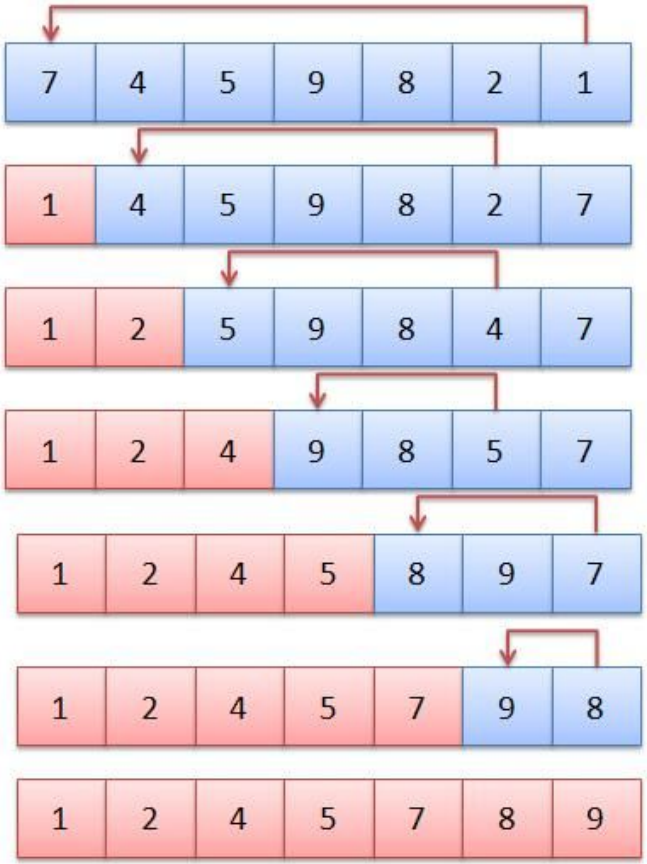


vídeo: https://youtu.be/nmhjrI-aW5o

Selection Sort

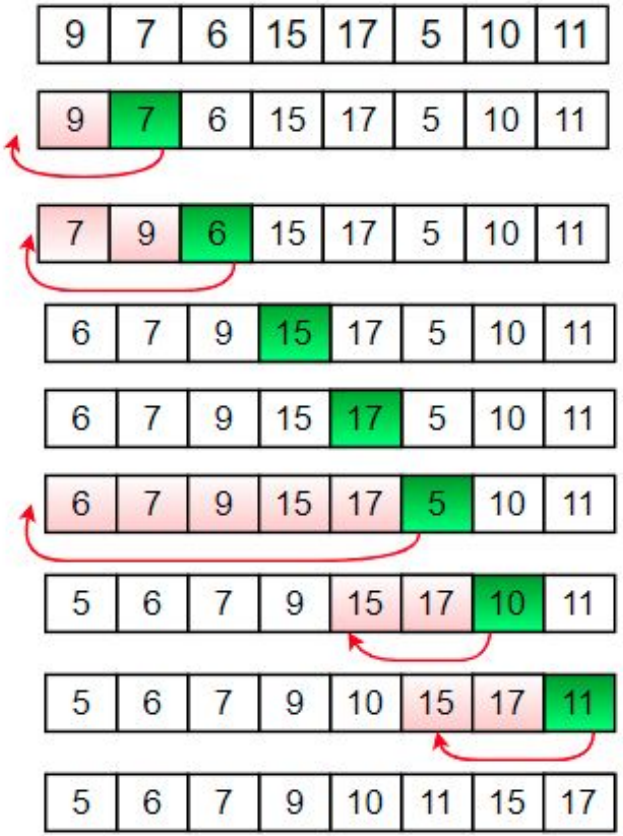
- Para cada iteração, move-se o menor elemento do vetor para a primeira posição.

```
25  void troca(int v[], int i, int j){
26     int aux = v[i];
27     v[i] = v[j];
28     v[j] = aux;
29  }
30
31  void selection(int v[], int n){
32     int i, j, menor;
33     for(i=0;i<n-1;i++){
34        menor = i;
35        for(j=i+1;j<n;j++){
36           if(v[j]< v[menor])
37              menor = j;
38        }
39        troca(v,i,menor);
40        print(v,TAM);
41     }
42  }
```



Vídeo: https://youtu.be/xWBP4lzkoyM

InsertionSort



vídeo: https://youtu.be/OGzPmgsI-pQ

Cada iteração separa os elementos em ordenados (à esquerda) e não ordenados (à direita) até que todo o *array* esteja ordenado.

```
25  void troca(int v[], int i, int j){
26    int aux = v[i];
27    v[i] = v[j];
28    v[j] = aux;
29  }
30
31  void insertion(int v[], int n){
32    int i, j;
33    for(i=1;i<n;i++){
34      for(j=i;j>0;j--){
35        if(v[j]< v[j-1])
36          troca(v,j,j-1);
37        else
38          break;
39      }
40      print(v,TAM);
41    }
42  }
```

Algoritmos de ordenação: *QuickSort*

O acham de uma pesquisa ?

Algoritmos de ordenação: Exercícios

Escreva uma função que recebe um vetor *v* de números inteiros de tamanho ímpar, ordena-o em ordem decrescente e retorna sua mediana. A entrada e saída de dados precisa ser realizada na função principal.

Entrada: v = {1, 1 ,6 ,3 ,7 ,9 ,8}
Saída: v = { 9, 8, 7, 6, 3, 1, 1}, Mediana = 6

Perguntas ?