UNIVERSIDADE FEDERAL DE GOIÁS – UFG
INSTITUTO DE INFORMÁTICA
SEMESTRE SELETIVO 2018/1

CURSO:	Engenharia da Computação
DISCIPLINA:	ESTRUTURA DE DADOS II
PROFESSORA:	Ma. Renata Dutra Braga (renata@inf.ufg.br ou professorarenatabraga@gmail.com)
TEMA DA AULA:	Algoritmos de ordenação: insertion sort, bubble sort e selection sort.
DIA:	02/04/2018

ALGORITMOS DE ORDENAÇÃO

Algoritmos: considerações gerais

- Servem para ordenar/organizar uma lista de números ou palavras de acordo com a sua necessidade.
- Os mais populares algoritmos de ordenação são: Insertion sort, Selection sort, Bubble sort, Quick sort, Merge sort, Heap sort e Shell sort, estando assim estruturados:
    - Ordenação por inserção
        - Insertion Sort (direta)
        - Shell sort
    - Ordenação por troca
        - Bubble sort
        - Quick sort
    - Ordenação por seleção
        - Selection sort
        - Heap sort
    - Ordenação por intercalação
        - Merge sort
    - Ordenação por distribuição

Um algoritmo descreve o passo a passo dos procedimentos necessários para chegar a solução de um problema. Eles podem ser representados de três formas:

- A forma de **descrição narrativa**, na qual se usa a linguagem nativa de quem escreve

```
Início
        Passo 1: Obter os valores de n1,n2,n3;
        Passo 2: Somar os valores do passo 1;
        Passo 3: Dividir o resultado obtido no Passo 2 por 3;
        Passo 4: Se o resultado do Passo 3 for maior ou igual a 6 então escre
        "Parabéns você foi aprovado", senão, escreva "Infelizmente você ficou
        e vá para o fim do programa
Fim
```

- Elaborando um **fluxograma**, que é uma representação visual que utiliza símbolos que são figuras geométricas, cada uma com sua função específica

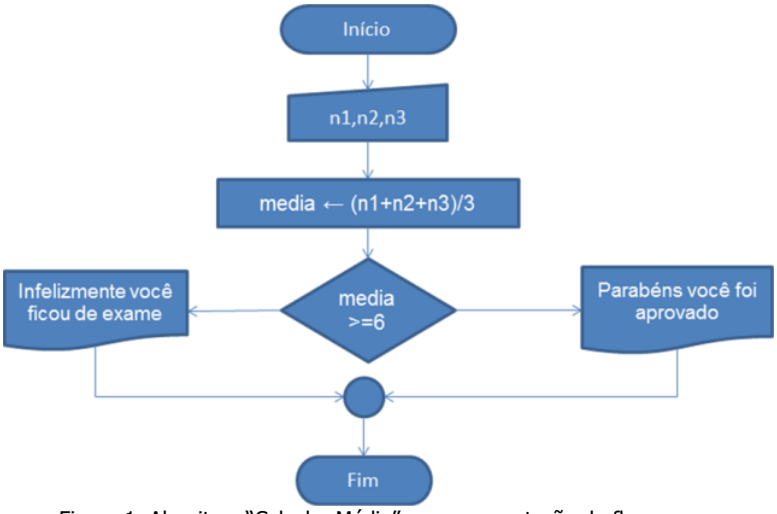


Figura 1. Algoritmo "Calcular Média" em representação de fluxograma

- A **linguagem algoritma** (Pseudocódigo ou Portugol) que é a que mais se aproxima da estrutura de uma linguagem estruturada

```
 algoritmo "CalcularMedia"
var
        n1,n2,n3,media :real;
inicio
        leia(n1,n2,n3);
                media← (n1+n2+n3)/3;
        se  media>=6 entao
                 escreva("Parabéns você foi aprovado");
        senão
                escreva("Infelizmente você ficou de exame");
        Fimse
fimalgoritmo
```

Listagem 2. Algoritmo "Calcular Média" em linguagem algoritma

Algoritmos de ordenação: considerações gerais

- Algoritmo de ordenação é um algoritmo que coloca os elementos de uma dada sequência em uma certa ordem.
- Isto é, rearranja os itens de um vetor ou lista de modo que suas chaves estejam ordenadas de acordo com
- alguma regra (podendo a ordenação ser completa ou parcial).
- O objetivo da ordenação é facilitar a recuperação dos dados de uma lista.
- Exemplo de aplicação: A realizar a consulta por produtos, ordenados pela data de cadastro, em um banco de dados.

Alguns conceitos importantes:

- **Ordenação interna**: todas os dados estão em memória principal (RAM). Portanto, todos os elementos a serem ordenados cabem na memória principal e qualquer registro pode ser imediatamente acessado.
- **Ordenação externa**: memória principal não cabe todos os dados. Portanto, devem estar armazenados em memória secundária (disco) e os registros são acessados sequencialmente ou em grandes blocos.
- **Estabilidade**: um algoritmo é estável se a ordem relativa dos registros com a mesma chave não se altera após a ordenação (mantém a ordem de inserção).
- **Adaptabilidade**: um algoritmo é adaptável quando a sequência de operações realizadas depende da entrada. Inversamente, quando um algoritmo que sempre realiza as mesmas operações, independente da entrada, é não adaptável.
- **In-place**: é quando necessita de uma estrutura auxiliar para realizar a ordenação (comparações e movimentos).
- A quantidade de comparação e a troca realizadas por um algoritmo são os fatores que determinam a eficiência (ou ineficiência) do mesmo.

- **Limite inferior**: é a melhor complexidade já encontrada pelo homem para um determinado tipo de ordenação. Exemplo: um algoritmo de faz ordenação usando comparações faz em tempo n^2 no pior caso. Portanto, um algoritmo é ótimo quando ele alcança, no seu pior caso, o limite inferior.
- **Complexidade assintótica**: classes de grandes de algoritmos

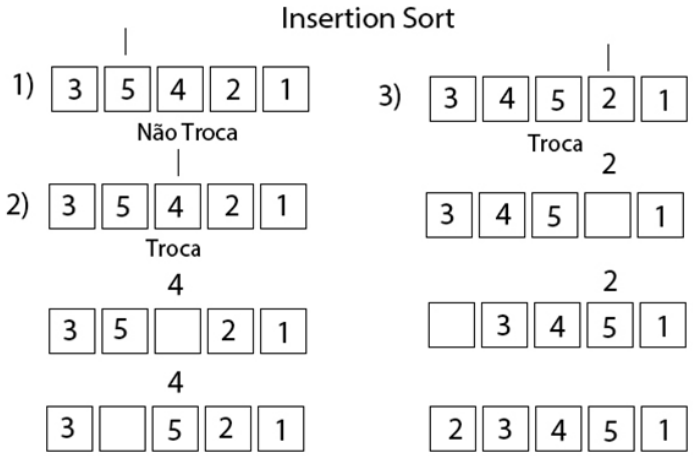
Execution times of a machine that executes 10^9 steps by second (~ 1 GHz), as a function of the algorithm cost and the size of input n:

Size	$\log_2 n$	n	$n \log_2 n$	n^2	n^3	2^n
10	3.322 ns	10 ns	33 ns	100 ns	1 μs	1 μs
20	4.322 ns	20 ns	86 ns	400 ns	8 μs	1 ms
30	4.907 ns	30 ns	147 ns	900 ns	27 μs	1 s
40	5.322 ns	40 ns	213 ns	2 μs	64 μs	18.3 min
50	5.644 ns	50 ns	282 ns	3 μs	125 μs	13 days
100	6.644 ns	100 ns	664 ns	10 μs	1 ms	$40 \cdot 10^{12}$ years
1000	10 ns	1 μs	10 μs	1 ms	1 s	
10000	13 ns	10 μs	133 μs	100 ms	16.7 min	
100000	17 ns	100 μs	2 ms	10 s	11.6 days	
1000000	20 ns	1 ms	20 ms	16.7 min	31.7 years	

Ordenação por inserção

Insertion sort

- O Insertion sort é um algoritmo simples e eficiente quando aplicado em pequenas listas.
- O método é estável, ou seja, a ordem relativa dos registros com a mesma chave não se altera após a ordenação.
- Por ser um algoritmo de ordenação quadrática – $O(n^2)$ – é bastante eficiente para problemas com pequenas entradas, sendo o mais eficiente entre os algoritmos desta ordem de classificação.
- Neste algoritmo a lista é percorrida da esquerda para a direita, à medida que avança vai deixando os elementos mais à esquerda ordenados.
- O algoritmo funciona da mesma forma que as pessoas usam para ordenar cartas em um jogo de baralho, como o pôquer. Percorra as posições do array, começando com o índice 1 (um). Cada nova posição é como a nova carta que você recebeu, e você precisa inseri-la no lugar correto no subarray ordenado à esquerda daquela posição.
    - Ideia básica: ´
        - Compare a chave (x) com os elementos à sua esquerda, deslocando para direita cada elemento maior do que a chave;
        - Insira a chave na posição correta à sua esquerda, onde os elementos já estão ordenados;
        - Repita os passos anteriores atualizando a chave para a próxima posição à direita até o fim do vetor.



- Neste passo é verificado se o 5 é menor que o 3. Como essa condição é falsa, então não há troca.
- É verificado se o quatro é menor que o 5 e o 3. Ele só é menor que o 5, então os dois trocam de posição.
- É verificado se o 2 é menor que o 5, 4 e o 3. Como ele é menor que 3, então o 5 passa a ocupar a posição do 2, o 4 ocupa a posição do 5 e o 3 ocupa a posição do 4, assim a posição do 3 fica vazia e o 2 passa para essa posição.

```
DECLARE i, j, x, n, v[n] NUMERICO;
PARA i = 1 ATÉ i < n FAÇA
INICIO
  x = v[i];
  j = i - 1;
  ENQUANTO j >= 0 e v[j] > x FAÇA
  INÍCIO
    v[j+1] = v[j];
    j = j-1;
  FIM
  v[j+1] = x;
FIM
```

Características do Insertion sort

- É de simples implementação, leitura e manutenção;
- In-place: Apenas requer uma quantidade constante de O(1) espaço de memória adicional;
- Estável: Não muda a ordem relativa de elementos com valores iguais;
- Útil para pequenas entradas;
- Muitas trocas, e menos comparações;
- Melhor caso: O(n), quando a matriz está ordenada;

- Médio caso: O(n²/4), quando a matriz tem valores aleatórios sem ordem de classificação (crescente ou decrescente);
- Pior caso: O(n²), quando a matriz está em ordem inversa, daquela que deseja ordenar.

Insertion sort	
classe	Algoritmo de ordenação
estrutura de dados	Array, Listas ligadas
complexidade pior caso	$O(n^2)$
complexidade caso médio	$O(n^2)$
complexidade melhor caso	$O(n)$
complexidade de espaços pior caso	$O(n)$ total, $O(1)$ auxiliar
estabilidade	estável

Vantagens do Insertion sort
- É o método a ser utilizado quando o arquivo está "quase" ordenado
- É um bom método quando se desejar adicionar poucos elementos em um arquivo já ordenado, pois seu custo é linear.
- O algoritmo de ordenação por inserção é estável.

Desvantagens do Insertion sort
- Alto custo de movimentação de elementos no vetor.

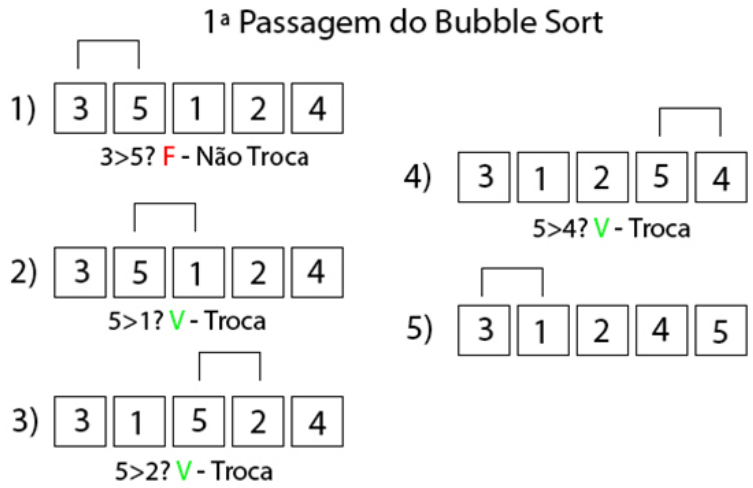
Vídeos sobre o Insertion Sort:
- https://www.youtube.com/watch?v=-hf4Pdwe8tg&feature=youtu.be
- https://www.youtube.com/watch?v=rG08wJgfydA&feature=youtu.be

Vamos praticar?
Implemente na linguagem C o algoritmo de ordenação insertion sort. Utilize uma função auxiliar para implementar a ordenação.

Ordenação por troca

Bubble sort

- É o algoritmo mais simples, mas o menos eficiente.
- Nele, cada elemento da posição i será comparado com o elemento da posição i + 1
- Caso o elemento da posição 2 for maior que o da posição 3, eles trocam de lugar e assim sucessivamente.
- Por causa dessa forma de execução, o vetor terá que ser percorrido quantas vezes que for necessária, tornando o algoritmo ineficiente para listas muito grandes.
- Ideia básica:
    - Compare o primeiro elemento com o segundo. Se estiverem desordenados, então efetue a troca de posição
    - Compare o segundo elemento com o terceiro e efetue a troca de posição, se necessário
    - Repita a operação anterior até que o penúltimo elemento seja comparado com o último.
    - Ao final desta repetição o elemento de maior valor estará em sua posição correta, a n-ésima posição do vetor
    - Continue a ordenação posicionando o segundo maior elemento, o terceiro, ..., até que todo o vetor esteja ordenado.



- É verificado se o 3 é maior que 5, por essa condição ser falsa, não há troca.
- É verificado se o 5 é maior que 1, por essa condição ser verdadeira, há uma troca.
- É verificado se o 5 é maior que 2, por essa condição ser verdadeira, há uma troca.
- É verificado se o 5 é maior que 4, por essa condição ser verdadeira, há uma troca.
- O método retorna ao início do vetor realizando os mesmos processos de comparações, isso é feito até que o vetor esteja ordenado.

```
DECLARE i, j, aux, n, v[n] NUMÉRICO;
PARA i = n-1 ATÉ i > 0 FAÇA
INÍCIO
  PARA j = 0 ATÉ j < i FAÇA
  INÍCIO
    SE v[j]> v[j+1]
      aux = v[j]; v[j] = v[j+1]; v[j+1] = aux;
  FIM
FIM
```

Características do Bubble sort

- Percorre o vetor diversas vezes, e a cada passagem flutua para o topo o maior elemento da sequência.
- Essa movimentação lembra a forma como as bolhas em um tanque de água procuram seu próprio nível. Daí o nome do algoritmo.

Bubble sort	
classe	Algoritmo de ordenação
estrutura de dados	Array, Listas ligadas
complexidade pior caso	$O(n^2)$
complexidade caso médio	$O(n^2)$
complexidade melhor caso	$O(n)$
complexidade de espaços pior caso	$O(1)$ auxiliar

Vantagens do Bubble sort

- Algoritmo simples
- Algoritmo estável

Desvantagens do Bubble sort

- Não adaptável. Em outras palavras, é um método que sempre realiza as mesmas operações, independente da entrada
- Muitas trocas de itens

Vídeos sobre o Bubble sort:

- https://www.youtube.com/watch?v=llX2SpDkQDc

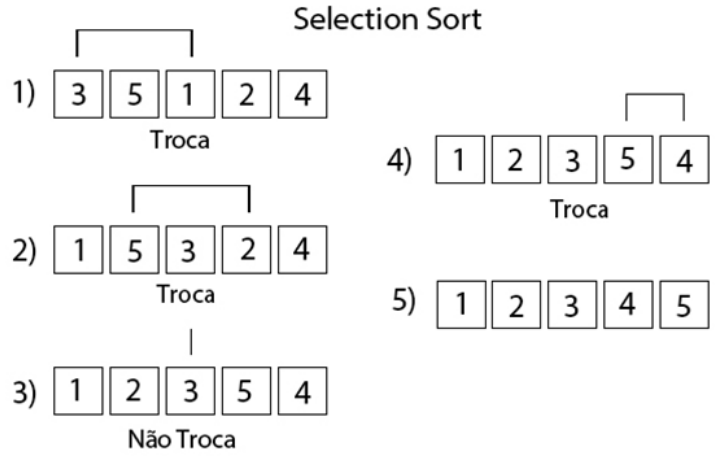
Vamos praticar?

Implemente na linguagem C o algoritmo de ordenação bubble sort. Utilize uma função auxiliar para implementar a ordenação.

Ordenação por seleção

Selection sort

- É vantajoso quanto ao número de movimentos de registros, que é O(n)
- Deve ser usado para arquivos com elementos muito grandes, desde que o número de elementos a ordenar seja pequeno
- Baseia-se em passar sempre o menor valor do vetor para a primeira posição (ou o maior dependendo da ordem requerida), depois o segundo menor valor para a segunda posição e assim sucessivamente, até os últimos dois elementos.
- Nele, é escolhido um número a partir do primeiro, este número escolhido é comparado com os números a partir da sua direita, quando encontrado um número menor, o número escolhido ocupa a posição do menor número encontrado. Este número encontrado será o próximo número escolhido, caso não for encontrado nenhum número menor que este escolhido, ele é colocado na posição do primeiro número escolhido, e o próximo número à sua direita vai ser o escolhido para fazer as comparações. É repetido esse processo até que a lista esteja ordenada.
- Ideia básica:
    - Selecione o menor elemento do vetor;
    - Troque esse elemento com o elemento da primeira posição do vetor;
    - Repita as duas operações anteriores considerando apenas os n-1 elementos restantes, em seguida repita com os n-2 elementos restantes; e assim sucessivamente até que reste apenas um elemento no vetor a ser considerado.



- Neste passo o primeiro número escolhido foi o 3, ele foi comparado com todos os números à sua direita e o menor número encontrado foi o 1, então os dois trocam de lugar.
- O mesmo processo do passo 1 acontece, o número escolhido foi o 5 e o menor número encontrado foi o 2.
- Não foi encontrado nenhum número menor que 3, então ele fica na mesma posição.
- O número 5 foi escolhido novamente e o único número menor que ele à sua direita é o 4, então eles trocam.
- Vetor já ordenado.

```
DECLARE i, j, aux, n, min, v[n] NUMERICO;
PARA i = 0 ATÉ i < n-1 FAÇA
INICIO
  min = i;
  PARA j = i+1 ATÉ j < n FAÇA
  INICIO
    SE v[j] < v[min] ENTAO
      min = j;
  FIM
  aux = v[i]; v[i] = v[min]; v[min] = aux;
FIM
```

Características do Selection sort

- É composto por dois laços, um laço externo e outro interno.

- O laço externo serve para controlar o índice inicial e o interno percorre todo o vetor.
- Na primeira iteração do laço externo o índice começa de 0 e cada iteração ele soma uma unidade até o final do vetor e o laço mais interno percorre o vetor começando desse índice externo + 1 até o final do vetor.

Selection sort	
classe	Algoritmo de ordenação
estrutura de dados	Array, Listas ligadas
complexidade pior caso	$O(n^2)$
complexidade caso médio	$O(n^2)$
complexidade melhor caso	$O(n^2)$
complexidade de espaços pior caso	$O(n)$ total, $O(1)$ auxiliar

Vantagens do Selection sort

- Ele é um algoritmo simples de ser implementado em comparação aos demais.
- Não necessita de um vetor auxiliar (in-place).
- Por não usar um vetor auxiliar para realizar a ordenação, ele ocupa menos memória.
- Ele é uns dos mais velozes na ordenação de vetores de tamanhos pequenos.

Desvantagens do Selection sort

- Ele é um dos mais lentos para vetores de tamanhos grandes.
- Ele não é estável.
- Ele faz sempre n² comparações, independente do vetor está ordenado ou não.

Vídeo sobre o Selection sort:

- https://www.youtube.com/watch?v=BSXIolKg5F8

Vamos praticar?

Implemente na linguagem C o algoritmo de ordenação selection sort. Utilize uma função auxiliar para implementar a ordenação.

EXERCÍCIOS

QUESTÃO 01

Segundo a análise do trecho de algoritmo a seguir, conclui-se que se trata de um algoritmo de ordenação do tipo:

```
1   declare X[5], i, j, eleito numérico
2   para i ← 1 até 4 faça
3     inicio
4     eleito ← X[ i ]
5     j ← i – 1
6     enquanto (j >= 0 E X[ j ] > eleito)
7       inicio
8           X [ j + 1 ] ← X [ j ]
9           j ← j – 1
10      fim
11      X[j + 1] ← eleito
12    fim-enquanto
13  fim-para
```

a) Quick sort.
b) Bubble sort.
c) Insertion sort.
d) Selection sort.
e) Merge sort.

QUESTÃO 02

Qual é o método de ordenação mais eficiente entre os listados a seguir?
a) $O(n * n^2)$
b) $O(n^2)$
c) $O(2^n)$
d) $O(n^n)$

QUESTÃO 03

A função seguinte codifica o algoritmo de ordenação (por ordem crescente) por inserção.

```
1  void OrdenaInsercao (int V[], int tam){
2      int j, k, aux;
3      for (k=1; k<tam; k++){
4          aux = V[k];
5          for (j=k-1; j>=0 && V[j]>aux; j--)
6              V[j+1] = V[j];
7          V[j+1] = aux ;
8      }
9  }
```

a) Simule o algoritmo apresentado para o seguinte exemplo: (12, 8, 6, 10, 23, 14).
b) Reescreva o algoritmo da ordenação por inserção de forma a que a ordenação se faça por ordem decrescente.

QUESTÃO 04

Faça uma comparação entre todos os métodos de ordenação estudados em aula com relação a estabilidade (preservar ordem lexicográfica), ordem de complexidade levando em consideração comparações e movimentações.

REFERÊNCIAS SUPLEMENTAR

Juliano Schimiguel. Algoritmos de ordenação: análise e comparação. Disponível em: https://www.devmedia.com.br/algoritmos-de-ordenacao-analise-e-comparacao/28261