Universidade Federal de Goiás
Goiânia 07 de Abril de 2018
Edson Júnior Frota Silva – 201515412
Engenharia de Computação

Atividade de Estrutura de Dados II

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
```

c) Insertion sort.

QUESTÃO 02: Qual é o método de ordenação mais eficiente entre os listados a seguir?

b) O(n²)

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).

```c
#include <stdio.h>
void ordenaInsercao (int Vetor[], int tamanho)
{
    int j, k, aux;
    for (k=1; k<tamanho; k++)
    {
        aux = Vetor[k];
        for ( j=k-1; j>=0 && Vetor[j]>aux; j--)
            Vetor[j+1] = Vetor[j];
        Vetor[j+1] = aux;
    }
}

int main()
{
    int Vetor[6] = {12,8,6,10,23,14}, tamanho, i;
    tamanho = 6;
    ordenaInsercao(Vetor,tamanho);
    printf(" Vetor Ordenado: \n");
    for (i=0; i<tamanho; i++)
    {
        printf(" \n%i ", Vetor[i]);
    }
    return(0);
}
```

b) Reescreva o algoritmo da ordenação por inserção de forma a que a ordenação se faça por ordem decrescente.

```c
#include <stdio.h>
void ordenaInsercao (int Vetor[], int tamanho)
{
    int j, k, aux;
    for (k=1; k<tamanho; k++)
    {
        j = k;
        while (j > 0 && Vetor[j - 1] < Vetor[j])
            {
            aux = Vetor[j];
            Vetor[j] = Vetor[j - 1];
            Vetor[j - 1] = aux;
            j--;
        }
    }
}
```

```
int main()
{
    int Vetor[6] = {12,8,6,10,23,14}, tamanho, i;
    tamanho = 6;
    ordenaInsercao(Vetor,tamanho);
    printf(" Vetor Ordenado: \n");
    for (i=0; i<tamanho; i++)
  {
        printf(" \n%i ", Vetor[i]);
    }
    return(0);
}
```

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.

Bubble Sort é um algoritmo de ordenação que pode ser aplicado em Arrays e Listas dinâmicas. Se o objetivo é ordenar os valores em forma decrescente, então, a posição atual é comparada com a próxima posição e, se a posição atual for maior que a posição posterior, é realizada a troca dos valores nessa posição. Caso contrário, não é realizada a troca, apenas passa-se para o próximo par de comparações.

Se o objetivo é ordenar os valores em forma crescente, então, a posição atual é comparada com a próxima posição e, se a posição atual for menor que a posição posterior, é realizada a troca. Caso contrário, a troca não é feita e passa-se para o próximo par de comparação.No melhor caso, o algoritmo executa n operações relevantes, onde n representa o número de elementos do vector. No pior caso, são feitas $n^2$ operações. A complexidade desse algoritmo é de ordem quadrática. Por isso, ele não é recomendado para

programas que precisem de velocidade e operem com quantidade elevada de dados.

Insertion Sort ou ordenação por inserção é o método que percorre um vetor de elementos da esquerda para a direita e à medida que avança vai ordenando os elementos à esquerda. Possui complexidade C(n) =

$O(n)$ no melhor caso e $C(n) = O(n^2)$ no caso médio e pior caso. É considerado um método de ordenação estável. Um método de ordenação é estável se a ordem relativa dos itens iguais não se altera durante a ordenação. O funcionamento do algoritmo é bem simples: consiste em cada passo a partir do segundo elemento selecionar o próximo item da sequência e colocá-lo no local apropriado de acordo com o critério de ordenação.

Selection Sort ordenação por seleção consiste em selecionar o menor item e colocar na primeira posição, selecionar o segundo menor item e colocar na segunda posição, segue estes passos até que reste um único elemento. Para todos os casos (melhor, médio e pior caso) possui complexidade $C(n) = O(n^2)$ e não é um algoritmo estável

QUESTÃO 05: Vamos praticar? Implemente na linguagem C o algoritmo de ordenação insertion sort. Utilize uma função auxiliar para implementar a ordenação.

```c
#include <stdio.h>
#define maximo 10
void insertion_sort(int *posicao);
int main(int argc, char** argv)
{
    int i, vetor[maximo];
    printf("Informe os elementos do vetor:\n");
    for(i = 0; i < maximo; i++)
    {
        scanf(" %i", &vetor[i]);
    }
    insertion_sort(vetor);
    printf("\nValores ordenados:\n");
    for(i = 0; i < maximo; i++)
    {
        printf(" \n%i ", vetor[i]);
    }
```

```
    return(0);
}
void insertion_sort(int *posicao)
{
    int i, j, x;
    for(i = 1; i < maximo; i++)
    {
        x = posicao[i];
        for(j = i-1; j >= 0 && x < posicao[j]; j--)
        {
            posicao[j+1] = posicao[j];
        }
        posicao[j+1] = x;
    }
}
```

QUESTÃO 06: Implemente na linguagem C o algoritmo de ordenação selection sort. Utilize uma função auxiliar para implementar a ordenação.

```
#include<stdio.h>
#define tamanho 10
void selection_sort(int v[])
{
    int i, j, aux, menor;
    for(i = 0; i < tamanho; i++)
    {
        menor = i;
        for(j = i+1; j < tamanho; j++)
        {
            if( v[j] < v[menor] )
                menor = j;
        }
        if( i != menor )
        {
            aux = v[i];
            v[i] = v[menor];
            v[menor] = aux;
        }
    }
}
```

```c
int main()
{
    int i, vetor[tamanho] = {25,11,97,96,1,5,2,3,4,12};
    printf("Vetor atual: ");
    for(i = 0; i < tamanho; i++)
        printf( "%i ", vetor[i]);
    printf(" \n ");

    selection_sort(vetor);

    printf(" \n\nVetor ordenado: ");
    for(i = 0; i < tamanho; i++)
        printf( "%i ", vetor[i]);
    printf( "\n" );

    return (0);
}
```

QUESTÃO 07: Vamos praticar? Implemente na linguagem C o algoritmo de ordenação bubble sort. Utilize uma função auxiliar para implementar a ordenação.

```c
void BubbleSort(int v[], int tam)
{
    int i, j, aux;
    for (i = tam - 1; i > 0; i--)
    {
         for(j = 0; j < i; i++)
         {
             if(v[j] > v[j+1])
             {
                aux = v[j];
                v[j] = v[j+1];
                v[j+1] = aux;
             }
         }
    }
}
```