Universidade Estadual do Norte do Paraná

Centro de Ciências Tecnológicas

Disciplina: Estrutura de Dados – Ciência da Computação.

Prof. Bruno Miguel

Exercícios – Aula e Revisão – 07 de agosto de 2019

# GABARITO

1) A partir do vetor não ordenado a seguir, realize as ordenações utilizando os seguintes algoritmos: Buble Sort, Insertion Sort e Selection Sort. Mostre somente o laço externo do algoritmo.

Vetor não ordenado:

| 21 | 10 | 7 | 11 | 3 | 2 | 15 | 18 |
|---|---|---|---|---|---|---|---|

Buble Sort

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1a | 10 | 7 | 11 | 3 | 2 | 15 | 18 | 21 |
| 2a | 7 | 10 | 3 | 2 | 11 | 15 | 18 | 21 |
| 3a | 7 | 3 | 2 | 10 | 11 | 15 | 18 | 21 |
| 4a | 3 | 2 | 7 | 10 | 11 | 15 | 18 | 21 |
| 5a | 2 | 3 | 7 | 10 | 11 | 15 | 18 | 21 |
| 6a | 2 | 3 | 7 | 10 | 11 | 15 | 18 | 21 |
| 7a | 2 | 3 | 7 | 10 | 11 | 15 | 18 | 21 |

Insertion Sort

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1a | 10 | 21 | 7 | 11 | 3 | 2 | 15 | 18 |
| 2a | 7 | 10 | 21 | 11 | 3 | 2 | 15 | 18 |
| 3a | 7 | 10 | 11 | 21 | 3 | 2 | 15 | 18 |
| 4a | 3 | 7 | 10 | 11 | 21 | 2 | 15 | 18 |
| 5a | 2 | 3 | 7 | 10 | 11 | 21 | 15 | 18 |
| 6a | 2 | 3 | 7 | 10 | 11 | 15 | 21 | 18 |
| 7a | 2 | 3 | 7 | 10 | 11 | 15 | 18 | 21 |

Selection Sort

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1a | 2 | 10 | 7 | 11 | 3 | 21 | 15 | 18 |
| 2a | 2 | 3 | 7 | 11 | 10 | 21 | 15 | 18 |
| 3a | 2 | 3 | 7 | 11 | 10 | 21 | 15 | 18 |
| 4a | 2 | 3 | 7 | 10 | 11 | 21 | 15 | 18 |
| 5a | 2 | 3 | 7 | 10 | 11 | 21 | 15 | 18 |
| 6a | 2 | 3 | 7 | 10 | 11 | 15 | 21 | 18 |
| 7a | 2 | 3 | 7 | 10 | 11 | 15 | 18 | 21 |

2) Construa um algoritmo recursivo para encontrar o maior elemento das entradas de um vetor v não ordenado.

```
int maior (int x, int maior, int vetor[]){
      if(x>=vetor.lenght)
            return maior;
      else if (vetor[x]>maior)
            return maior(x+1,vetor[x],vetor);
      else
            return maior(x+1,maior,vetor);
```

3) Determine o que faz o algoritmo recursivo a seguir. Mostre a pilha de execução deste algoritmo para as chamadas: recursiva(5) e recursiva(7).

```
public static int recursiva(int n){
      If (n<=0)
            return 1;
      Else
            return recursiva(n-1) + recursiva(n-1);
}
```

1) recursiva (5) -> recursiva(4) + recursiva(4)
2) recursiva(4) -> recursiva(3) + recursiva(3)
3) recursiva(3) -> recursiva(2) + recursiva(2)
4) recursiva(3) -> recursiva(2) + recursiva(2)
5) recursiva(2) -> recursiva(1) + recursiva(1)
6) recursiva(1) -> recursiva(0) + recursiva(0)
7) recursiva (0) = 1
8) recursiva(1) = 1 + 1 ➔ 2;
9) recursiva(2) = 2 + 2 ➔ 4;
10) recursiva(3) = 4+4 ➔8;
11) recursiva(4) = 8+8➔16;
12) recursiva(5) = 16+16➔32;

1) recursiva (7) -> recursiva(6) + recursiva(6)
2) recursiva (6) -> recursiva(5) + recursiva(5)
3) recursiva (5) -> recursiva(4) + recursiva(4)
4) recursiva(4) -> recursiva(3) + recursiva(3)
5) recursiva(3) -> recursiva(2) + recursiva(2)
6) recursiva(3) -> recursiva(2) + recursiva(2)
7) recursiva(2) -> recursiva(1) + recursiva(1)
8) recursiva(1) -> recursiva(0) + recursiva(0)
9) recursiva (0) = 1
10) recursiva(1) = 1 + 1 ➔ 2;
11) recursiva(2) = 2 + 2 ➔ 4;
12) recursiva(3) = 4+4 ➔8;
13) recursiva(4) = 8+8➔16;
14) recursiva(5) = 16+16➔32;
15) recursiva(6) = 32+32➔64;
16) recursiva(7) = 64+64➔128;

O Algoritmo calcula os valores de $2^n$, onde n é o valor de entrada para a função recursiva.