UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
Bacharelado em Ciência da Computação e
Engenharia da Computação

INF 01203 – Estruturas de Dados

Aula 03 – LISTAS LINEARES – Alocação Sequencial

01. Considere os quatro trechos de código apresentados a seguir, que realizam buscas em listas lineares alocadas seqüencialmente (*arrays*).

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 7


int busca1 (int vet[], int n) {
 int i = 0;
 while(i < MAX){
 if (vet[i] == n)
 return i;
 else
 i++;
 }
 return -1;
}


int main()
{
 int v[]={2,4,5,1,6,9,3};
 int resultado;
  resultado  =  busca1(v,  ???);   printf("%d\n",
resultado);   system("pause");
}
```

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 7

int busca2(int vet[], int n)
{
 int i = 0;
 vet[MAX] = n;

 while(vet[i]!=n)
 i++;
 if (i == MAX)
 return -1;
 else
 return i;
}

int main()
{
 int v[]={2,4,5,1,6,9,3,0};
 int resultado;
 resultado = busca2(v, ???);
 printf("%d\n", resultado);
 system("pause");
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 7

int busca3(int vet[], int n)
{
 int i = 0;

 vet[MAX] = n;
 while(vet[i]<n)
 i++;
 if ((i == MAX) || (vet[i]!= n))   return -1;
 else
 return i;
}
int main()
{
 int v[]={1,2,4,5,6,7,8,0};
 int resultado;
  resultado  =  busca3(v,  ???);   printf("%d\n",
resultado);   system("pause");
}
```

```c
#include <stdio.h>
#include <stdlib.h>
#define MAX 7

int busca4(int vet[], int n)
{
 int inf = 0, sup = MAX -1, meio;

 while(inf <= sup) {
 meio = (inf + sup)/2;
 if (vet[meio] == n) {
 inf = sup + 1;
 return meio;
 }
 else if (vet[meio] < n) inf = meio +1;  else
sup = meio - 1;
 }
 if (inf >= sup) return -1;
}
int main()
{
 int v[]={1,2,4,5,6,7,8};
 int resultado;
 resultado = busca4(v, ???);
 printf("%d\n", resultado);
 system("pause"); }
```

1 – O que faz o trecho de código **busca1()?**

- Recebe um array e um inteiro n
- Percorre elemento por elemento do array e retorna a posição do elemento da lista que tiver o valor igual a **n**, caso não ache retorna -1

2 – O trecho de código **busca1()** é indicado para (marque X na(s) alternativa(s) correta(s)) :

( ) lista ordenada (ordem crescente)
( ) lista ordenada (ordem decrescente)
(X) lista desordenada

3 – O que faz o trecho de código **busca2()?**

- Recebe um array e um inteiro n
- Atribui a última posição do array o valor de **n** e percorre todo o array enquanto o valor o valor array for diferente de n
- Se não encontrou (percorreu todo arraz e i == max) retorna -1, caso contrário retorna a posição do elemento

4 – O trecho de código **busca2()** é indicado para (marque X na(s) alternativa(s) correta(s)) :

( ) lista ordenada (ordem crescente)
() lista ordenada (ordem decrescente)
(X) lista desordenada

5 – Considerando as funções **busca1()** e **busca2()**, qual delas é mais "eficiente" considerando o número de comparações executadas**?** Justifique sucintamente sua resposta.

busca2() é mais eficiente pois executa o bloco if else somente uma vez ao final da execução da função, enquanto o **busca1()** executa o bloco if else em todas execuções do while

6 – O que faz o trecho de código **busca3()?**

- Executa a busca de forma similar ao busca2(), porém a condição de vet[i] != n foi removida do while e passou para a condição ao final da execução, o que faz mais sentidos para listas ordenadas crescentes pois percorre elemento por elemento considerando que n é maior que todos os da lista
-

7 – O trecho de código **busca3()** é indicado para (marque X na(s) alternativa(s) correta(s)) :

(X) lista ordenada (ordem crescente)
( ) lista ordenada (ordem decrescente)
( ) lista desordenada

8 – O que faz o trecho de código **busca4()?**

Ele faz a busca de forma similar a uma busca binaria, onde quebra a lista ao meio para otimizar a busca, funcionando somente para listas ordenadas, listas desordenadas não faz muito sentido dividir a lista em dois para procurar elementos.

9 – O trecho de código **busca4()** é indicado para (marque X na(s) alternativa(s) correta(s)) :

() lista ordenada (ordem crescente)
(X) lista ordenada (ordem decrescente)
( ) lista desordenada

10 – Considerando as funções **busca3()** e **busca4()**, qual delas é mais "eficiente" considerando o número de comparações executadas**?** Justifique sucintamente sua resposta.

busca3() o bloco if else é executado somente uma vez, enquanto no busca4() o bloco if else é executado múltiplas vezes, porém a condição do while do busca4() tende a ser executado menos vezes, então busca4() é mais eficiente