Universidade Estadual da Paraíba
CAMPUS VII - Patos

Professor:	**Allan Vilar de Carvalho**
Curso:	**Bacharelado em Computação**
Componente:	**Técnicas e Análise de Algoritmos**
Nome do aluno:	**Harlem Alves do Nascimento**

Prova 1

Questões:

1) Desenvolver um algoritmo para transpor uma matriz quadrada M. Os parâmetros do algoritmo são a matriz M, de tamanho n × n. Não utilize matriz ou vetor auxiliar na solução. Em seguida realizar a análise do algoritmo utilizando a abordagem matemática. Identificar as primitivas, identificar a quantidade de vezes que cada uma das primitivas é executada e somar o custo total.
2) Dado um conjunto de valores previamente armazenados em um vetor A, nas posições A[i], A[i + 1], ...A[n], verificar se um número v está entre este conjunto de valores. Se o elemento procurado v não for encontrado a função deve retornar −1. Caso contrário deve retornar o índice do vetor A que contém o elemento v. Desenvolver o algoritmo, e em seguida realizar a análise do algoritmo utilizando a abordagem matemática. Identificar as primitivas, identificar a quantidade de vezes que cada uma das primitivas é executada e somar o custo total.

Respostas:

Coloque sua(s) resposta(s) aqui!

1)

RESPOSTA DA QUESTÃO 01:

```
void transpose(int **matriz, int rows, int cols){
    for(int i = 0; i < rows; i++){
        for(int j = 0; j < cols; j++){
            cout << to_string(matriz[j][i]) + " ";
        }
        cout << "\n";
    }
}
```

ANÁLISE DE ALGORITMO:

1 -> Atribuição de valor (int i = 0) | 1 vez
2 -> Verificação booleana (i < rows) | (N+1) vezes
3 -> Incremento (i++) | N vezes
4 -> Atribuição de valor (int j = 0) | N vezes
5 -> Verificação booleana (j < cols) | (N^2 + N) vezes
6 -> Incremento (j++) | N^2 vezes
7 -> Impressão de Valor da matriz transposta (cout <<) | N^2 vezes
8 -> Impressão de quebra de linha (cout <<) | N vezes

Resultado:

$f(N) = 1 + (N + 1) + N + N + (N^2 + N) + N^2 + N^2 + N$
$f(N) = 2 + 5N + 3N^2$
$f(N) = 3N^2 + 5N + 2$

Cortando os custos constantes ficamos com:

$f(N) = N^2$

Temos um algoritmo quadrático!

TESTE DO ALGORITMO:

```
int main()
{
    const int N = 3; // TAMANHO DO ARRAY
    int **matriz = new int*[N]{
        new int[N]{1,2,3},
```

```
        new int[N]{4,5,6},
        new int[N]{7,8,9}
    };

    cout << "Matriz Transposta:\n";
    transpose(matriz, N, N);

    return 1;
}
```

2)

RESPOSTA DA QUESTÃO 02:

```
int findFirst(int array[], int size, int value){
    for(int i = 0; i < size; i++){
        if(array[i] == value) return i;
    }
    return -1;
}
```

ANÁLISE DE ALGORITMO:

1 -> Atribuição de valor (int i = 0) | 1 vez
2 -> Verificação booleana (i < rows) | (N+1) vezes
3 -> Incremento (i++) | N vezes
4 -> Verificação booleana (array[i] == value) | N vezes
5 -> Retorno da posição do vetor com o valor encontrado (return i) | 1 vezes
6 -> Retorno da posição do vetor com o valor não encontrado (return -1) | 1 vezes

- Não será usado o numero 6 pois estamos escolhendo o pior caso!

Resultado:

f(N) = 1 + (N + 1) + N + N + 1
f(N) = 3N + 3

Cortando os custos constantes ficamos com:

f(N) = N

Temos um algoritmo linear!

TESTE DO ALGORITMO:

```
int main()
{
    const int N = 10; // TAMANHO DO ARRAY
    const int VALOR_PARA_BUSCAR = 4;
    int vetor[N] = {0,3,5,1,7,2,9,4,8,6};

    cout << "Buscando valor:\n";
    int found = findFirst(vetor, N, VALOR_PARA_BUSCAR);
    if(found != (-1)) {
        cout << "O valor " + to_string(VALOR_PARA_BUSCAR)
                + " foi encontrado na posicao "
                + to_string(found);
    }else{
        cout << "Valor não encontrado!";
    }

    return 1;
}
```