| Diogo Araujo Miranda |
| Matricula: 705657 |

Questão 1

Função de complexidade para o num de mult :

Melhor caso:

1° for: O melhor caso acontece se o número gerado aleatoriamente for maior do que 5, ou seja, ultima condição de else if. A execução de algum IF sempre será necessariamente verdadeira, pois ela abrange todo o intervalo de números inteiros.

1 mult * (n-4) repetições = 1*(n-4)

2° for: sempre irá executar no pior e no melhor caso.

- Nesse for ocorre um deslocamento de bits para a direita, ou seja, dividindo o número por 2, tendo assim um custo logarítimico

piso(lg(n)) + 1 * (1 multiplicação) = piso(lg(n)) + 1

3° for: sempre irá executar no pior e no melhor caso, tendo o mesmo custo.

- Nesse for, ele vai de n-2 até i > 5, decrementando de 1 em 1. Portanto o seu cursto será:

n-2-5 = (n-7) * 1 multiplicação = (n-7)

4° for: sempre irá executar no pior e no melhor caso o mesmo custo

- Nesse for, vai de n-1 até i >= 1, dividindo o i por 2. Portante terá um custo logarítimico.

piso(lg(n-1)) + 1 * 1 multiplicação = piso(lg(n-1)) + 1

if : O melhor caso acontece se cair na condição else, ou seja, for maior que a - 3 e n for maior que b + 4

custo = + 1;

5° e 6 ° for : Sempre será executado em todos os casos com o mesmo custo e terá o custo de:

(n) - for externo; vezes

n-1 - for interno; vezes

2 multiplicações, com o custo de:

n * (n-1) * 2 = (n^2-n) * 1 = 2n^2-2n

O Custo total será:

(n-4) + piso(lg(n)) + 1 + (n-7) + piso(lg(n-1)) + 1 + 1 + 2n^2-2n, que é da ordem de complexidade de $\theta(n^2)$

Sendo assim, esse algoritmo também é da ordem de $O(n^2 \times \lg n)$, pois seria justamente uma constante c1 multiplicando a função e definindo a assintota superior, o que define em parte que uma função pertence a teta. E também esse algoritmo é da ordem $\Omega(\lg n)$, pois seria justamente a constante c2 multiplicada

a uma função f(n) que define a assintota inferior.

Pior caso:

1° for: irá executar n-4 vezes, e o pior caso seria se o resto da divisão do número gerado aleatoriamente divido por 9 fosse menor do que 4, definindo assim 3 multiplicações

- (n-4)*3

2° for: irá executar no melhor e no pior caso com o mesmo custo

- piso(lg(n)) + 1

3° for: irá executar sempre com o mesmo custo no pior e no melhor caso

- (n-7)

4° for: irá executar sempre com o mesmo custo no pior e no melhor caso

- piso(lg(n-1)) + 1

if : o pior caso irá executar se n for menor que a - 3 OU n for maior que b + 4, descrevendo assim 3 multiplicações

- + 3

5° e 6° for :

O for externo irá executar n vezes; vezes

O for interno que irá executar n-1 vezes; vezes

2 multiplicações descrevidas dentro do for interno

(n) * (n-1) * 2 = (n^2+n)*2 = $2n^2$+2n

Sendo assim a função de complexidade do pior caso será:

(n-4)*3 + piso(lg(n)) + 1 + (n-7) + piso(lg(n-1)) + 1 + 3 + $2n^2$+2n, que é da ordem de complexidade de 0(n^2).

Nesse caso, também o custo será O(n^2 × lg n) e Ω(lg n), pois essas outras duas funções de complexidade definem a assintota superior e infernor que definem que o problema é da ordem de complexidade de 0(n^2)

Questao 2 :

```
    public static int[] removeDir(int items[])
    {
        int tam = items.length;
        int auxTam = 0;
        if(tam == 0)
        {
            auxTam = 1;
        } else {
            auxTam = tam;
        }

        int items_new[] = new int[auxTam-1];

        if(tam > 0)
        {
            for(int i = 0; i < items_new.length; i++)
```

```
            items_new[i] = items[i];
        }
        return tam > 0 ? items_new : items;
    }

    public static int[] removeEsq(int items[])
    {
        int tam = items.length;
        int auxTam = 0;
        if(tam == 0)
            auxTam = 1;
        else
            auxTam = tam;

        int items_new[] = new int[auxTam-1];

        if(tam > 0)
        {
            for(int i = 1; i < tam; i++)
            {
                items_new[i-1] = items[i];
            }
        }

        return tam > 0 ? items_new : items;
    }

    public static int[] insereEsq(int items[])
    {
        int tam = items.length;
        int auxTam = 0;

        int newNumber = 5;

        if(tam == 0)
            auxTam = 1;
        else
            auxTam = tam;

        int items_new[] = new int[auxTam + 1];

        if(tam > 0)
        {
            items_new[0] = newNumber;
            for(int i = 1; i < items_new.length; i++)
                items_new[i] = items[i-1];
        }

        return tam > 0 ? items_new : items;
    }

    public static int[] insereDir(int items[])
    {
        int tam = items.length;
        int auxTam = 0;

        int newNumber = 6;

        if(tam == 0)
            auxTam = 1;
        else
            auxTam = tam;

        int items_new[] = new int[auxTam + 1];
```

```
        if(tam > 0)
        {
            items_new[items_new.length-1] = newNumber;
            for(int i = 0; i < items_new.length - 1; i++)
                items_new[i] = items[i];
        }

        return tam > 0 ? items_new : items;
    }
```

Questao 4 :

```
class Time {
    int gols;
    String nome;

    public Time(int gols, String nome)
    {
        this.gols = gols;
        this.nome = nome;
    }
}

public class Questa3 {

    public static int compareTime(Time time1, Time time2, int i)
    {
        int flag = 0;

        if(time1.nome.charAt(i) < time2.nome.charAt(i))
        {
            flag = 1;
        } else if (time2.nome.charAt(i) < time1.nome.charAt(i))
        {
            flag = 2;
        } else {
            flag = compareTime(time1, time2, i+1);
        }

        return flag;
    }

    public static int compareTime(Time time1, Time time2)
    {
        return compareTime(time1, time2, 0);
    }
```

Questão 3) Não sei fazer ainda.Tentei buscar por conta própria e não consegui digerir o conteúdo para resolver a questão. Em um próximo teórico que a matéria for apresentada eu irei enviar essa questão juntamente.