Lista 5 - Projeto e Análise de Algoritmos
Matrícula: 201700030154
Aluno: Edilson leite Santos Junior
Turma: 02

1)

```
função interseção(A, B): array
    inicio
        heapsort(A) --tamanho n
        heapsort(B) --tamanho m
        interseção := []
        j := 0

        para i de 0 até (tamanho(A)-1)
        faça:
            enquanto(j < tamanho(B) & A[i] > B[j])
            faça:
                j = j + 1
            fim enquanto
            se(j < tamanho(B) & A[i] = B[j])
            então insira A[i] em interseção
            fim se
        fim para

        retorne interseção
    fim


    Cálculo de complexidade:

        complexidade de heapsort: O(nlog(n))

        complexidade do para: Σ(i = 0; n) c
        complexidade do enquanto: Σ(j = 0; m) d
        c = constante; d = constante

        T(n + m) = O(nlog(n)) + O(mlog(m)) + Σ(i = 0; n) c + Σ(j = 0; m) d
        T(n + m) = O(nlog(n)) + O(mlog(m)) + n + m
        T(n + m) = O(nlog(n) + mlog(m))

        n ⩽ m, ou seja, o tamanho máximo do heap(A) é log(m)
        no pior caso:

                    T(n + m) = O( (n + m) log(m) )
```

2)

---

3) a)

```
função consecutivos(intervalos): array
    inicio
        array = []
        para i de 1 até tamanho(intervalos)
        faça:
            para j de intervalos[i][0] até intervalos[i][-1]
            faça:
                insira j em array
            fim para
        fim para

        heapsort(array)
        retorne array
    fim

Cálculo de complexidade:

    complexidade de heapsort: O(nlog(n))
        complexidade do para:  Σ(i = 0; k) Σ(i = 0; l) c = k * l = n

        T(n) = O(nlog(n)) + n = O(nlog(n))
```

b)

4)

```
função contador(Array): array
inicio
    maior = Array[0]

    para i de 0 até tamanho(Array)
    faça
        se(Array[i] > maior)
        então maior = Array[i]
    fim para

    contador[maior + 1] -- declarar os vetores contador
    posições[maior + 1] -- e posições, ambos com todos os valores iguais a 0
    intervalos = []

    para i de 0 até tamanho(Array)
    faça:
        se contador[Array[i]] <> 0)
        então insira (posições[Array[i]], i) em intervalos

        contador[Array[i]] += 1
        posições[Array[i]] = inicio
    fim para

    retorne intervalos
fim

Cálculo de complexidade:

        complexidade do primeiro para: Σ(i = 0; n) c = n
        complexidade do segundo para Σ(i = 0; n) c = n

        T(n) = Σ(i = 0; n) c + Σ(i = 0; n) c = n + n = O(n)
```