Lista 1 - IAA Mirela Mei (11208392)

1. Calcule o tempo de processamento em função do tamanho n da entrada

* Análise de pior caso

• A atribuição na linha 1 toma tempo constante C_2 e é executada apenas uma vez.

• O retorno na linha 7 toma tempo constante C_4 e é executado apenas uma vez.

• A linha 2 será executado enquanto $i \leq n$. A linha 3 enquanto $j \leq n$ e a linha 4 enquanto $k \leq n$.
Cada uma irá se repetir n vezes, levando tempo C_1.

• A comparação na linha 5 toma tempo C_5 e será executada n vezes, + 2 somas por iteração.

• A linha 6 será executado apenas em alguns casos, e toma tempo const. C_2

→ Função de tempo de processamento em função de n:

$$t(n) = C_2 + C_4 + (C_1 + C_3 + C_5) \cdot n + C_5 \cdot n + C_2 + (C_3 + C_3) \cdot n$$

$$t(n) = 3C_1 \cdot n + 2C_2 + 2C_3 \cdot n + C_4 + C_5 \cdot n$$

$$t(n) = (3C_1 + 2C_3 + C_5) \cdot n + 2C_2 + C_4$$

2. Mostre que o tempo de processamento do algoritmo 3soma é $\Theta(n^3)$ no pior caso.

No pior caso desse algoritmo, a soma não será nulidade nenhuma vez, portanto o algoritmo irá realizar todas as iterações. Então, o maior gasto de processamento será devido aos laços (linhas 2, 3, 4).

$$T(n) = \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{k=1}^{n} 1 = \sum_{i=1}^{n} \sum_{j=1}^{i} n$$

$$T(n) = \sum_{i=1}^{n} \frac{n(n+1)}{2}$$

$$T(n) = \sum_{i=1}^{n} 1/2 \cdot i(i+1) = 1/2 \sum_{i=1}^{n} i(i+1)$$

$$T(n) = 1/2 \cdot \sum_{i=1}^{n} i^2 + 1 = 1/2 \left(\sum_{i=1}^{n} i^2 + \sum_{i=1}^{n} i \right)$$

$$T(n) = 1/2 \left(\sum_{i=1}^{n} i^2 + \frac{n(n+1)}{2} \right)$$

$$T(n) = 1/2 \left(\frac{1}{2+1} \cdot n^3 + \frac{n(n+1)}{2} \right)$$

$$T(n) = 1/2 \left(\frac{n^3}{3} + \frac{n^2+n}{2} \right) = \frac{n^3}{6} + \frac{n^2+n}{4}$$

$$T(n) = \Theta(n^3)$$

3. Descreva em pseudo-código um algoritmo cujo tempo de processamento no pior caso é $\Theta(n^2 \log(n))$.

Algoritmo 3soma

```
para j ← 2 até n
    chave ← aj
    para i ← j-1 até 0
        se ai ≤ chave
            break.
        ai+1 ← ai
    ai+1 ← chave
```

Busca Binária (chave, A)

```
esq = 1
dir = n
enquanto esq ≤ dir
    faça meio ← ⌊(esq + dir)/2⌋
        se (A[meio] == A[k])
            devolva k
        senão (A[meio] < A[k])
            esq = meio + 1
        senão
            dir = meio - 1
    devolva 0

m ← 0
para i ← 1 até n
    para j ← i+1 até n
        k = (BuscaBinaria (-(A[i] * A[j]), A))
        se k ≠ 0 e k > j
            m ← m + 1
```

4. Mostre que o algoritmo é correto, ou seja, que ele resolve o problema de ordenação.

I. INVARIANTE DE LAÇO

Ao início de cada iteração do for na linha 3, os elementos da sequência $A(1 \dots i-1)$ são menores que os elementos do sequência $A(i \dots n)$.

II. INICIALIZAÇÃO

A sequência $A[1 \dots i-1]$ é um vetor vazio, então é o menor elemento do subvetor.

III. MANUTENÇÃO

Após a execução do laço do linha 4, $A[i]$ será o menor elemento do subvetor $A[i..n]$.

No início do linha 3, o subvetor que vai de $1 \dots i-1$ é composto por elementos que são menores que os elementos de $A[i \dots n]$, em sequência ordenada.

Ao fim da execução desse laço, o subvetor $A[1 \dots i]$ será composto por elementos menores que os elementos $(i+1 \dots n]$.

IV. FINALIZAÇÃO

Quando o laço é finalizado, em $i=n$, a permutação será composta apenas por números ordenados.