Lista 3
Análise de Algoritmos
MAC5711

1 .

```
Algoritmo MaiorMenor (v, n)
    maior <- v[1]
    menor <- v[1]
    para i <- 2 ate n faca
        se v[i] > maior
            entao maior <- v[i]
            senao se v[i] < menor
                entao menor <- v[i]
    devolva maior, menor
```

(a) Qual é o número esperado de comparações executadas na linha 6 do algoritmo?

A linha `6` somente é executada se a primeira condição da linha `4` não for verdadeira. Portanto, para saber quantas vezes a linha `6` é executada, deve ser calculada a quantidade de vezes que a linha `5` é executada primemiro.

Considere que: $X(v, n)$ = número de execuções da linha 5.

$$E[X] = \Sigma_{s \in S} X(s) \cdot Pr(s)$$

$$X_i = \begin{cases} 1 & \text{se } v[i] > \text{maior} \\ 0 & \text{caso contrário} \end{cases}$$

$$\therefore E[X_i] = \text{probabilidade que } A[i] \text{ seja o valor máximo em } A[1...i] = \frac{1}{i}$$

$$\begin{aligned} E[X] &= E[X_i + ... + X_n] = E[X_1] + .. + E[X_n] = \frac{1}{1} + ... + \frac{1}{n} \\ &< 1 + \ln n = \Theta(\lg n) \end{aligned}$$

Achado o número de execuções da linha `5`, para achar o da linha `6` é o número de vezes que o laço é executado menos o número de execuções da `5`. Se o laço executa n vezes e a linha `5` executa $\Theta(\lg n)$ vezes, pode-se concluir que o número de execuções da linha `6` é:

$$\Theta(n) - \Theta(\lg n) = \Theta(n - \lg n)$$

(b) Qual é o número esperado de atribuições efetuadas na linha 7 do algoritmo?

$X(v, n) =$ número de execuções da linha 7

$$E[X] = \Sigma_{s \in S} X(s) \cdot Pr(s)$$

Dividindo o problema em X_i:

$$X_i = \begin{cases} 1 & \text{se } v[i] < \mathit{maior} \text{ e } v[i] < \mathit{menor} \\ 0 & \text{caso não.} \end{cases}$$

O $v[i] < \mathit{maior}$ é o inverso do cálculo da linha 5. Pois para X_i ser executado, é necessário que não entre na condição da linha 4 e que a condição da linha 6 seja verdadeira. Portanto:

$$X_i = \frac{i-1}{i} \cdot \frac{1}{i} = \frac{i-1}{i^2} = \frac{1}{i} - \frac{1}{i^2}$$

$$\begin{aligned} E[X] &= (\frac{1}{1} - \frac{1}{1}) + (\frac{1}{2} - \frac{1}{4}) + (\frac{1}{3} - \frac{1}{9}) + ... + (\frac{1}{n} - \frac{1}{n^2}) \\ &= \Sigma_{i=1}^{n} \frac{1}{i} + \Sigma_{i=1}^{n} \frac{-1}{i^2} \end{aligned}$$

As duas somatórias podem ser consideradas séries harmônicas. Resolvendo a primeira, como visto anteriormente:

$$\Sigma_{i=1}^{n} \frac{1}{i} < \ln +1 = \Theta(\lg n)$$

A segunda somatória é uma série que pode ser simplificada por:

$$\begin{aligned} \Sigma_{i=1}^{n} \frac{-1}{i^2} &= \frac{-1}{1} + \frac{-1}{4} - \frac{-1}{9} + \frac{-1}{16} + .. + \frac{-1}{n^2} \\ &< \underbrace{1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} + ...}_{\text{Série Hamônica alternada}} < \ln 2 \end{aligned}$$

A série harmônica alternada converge em um valor, portanto:

$$\Sigma_{i=1}^{n} \frac{1}{i} + \Sigma_{i=1}^{n} \frac{-1}{i^2} = \theta(\lg n)$$

Então, o número esperado de atribuições é $\Theta(\lg n)$ para a linha 7.

3 . Para esta questão, vamos dizer que a mediana de um vetor A[p . . r] com número inteiros é o valor que ficaria na posição $A[\lfloor (p+r)/2 \rfloor]$ depois que o vetor $A[p..r]$ fosse ordenado. Dado um algoritmo linear "caixa-preta" que devolve a mediana de um vetor, descreva um algoritmos, linear, que, dado um vetor $A[p..r]$ de inteiros distintos e um inteiro k, devolve o k-ésimo menor do vetor. (O k-ésimo menor de um vetor de inteiros distintos é o elemento que estaria na k-ésima posição do vetor se ele fosse ordenado). Você pode assumir que o vetor tem tamanho igual uma potência de 2.

```
kEsimoTermo(A, p, r, k):
    valor_mediana = mediana(A, p, r)
    esquerda = [elementos de A < valor_mediana]
    direita = [elementos de A > valor_mediana]
    k_Atual = len(esquerda)
    se k <= k_Atual:
        retorna kEsimoTermo(esquerda, 0, k_Atual - 1, k)
        senao se k >= k_Atual:
            retorna kEsimoTermo(direita, 0, len(direita), k - k_Atual)
    retorna valor_mediana
```

O algoritmo funciona da seguinte maneira: obtem o valor da mediana. A partir dela, cria-se uma lista com todos os valores menores que a mediana e uma outra lista com todos os valores maiores. Com isso, é possível saber a posição exata da mediana e comparar o valor de k recebido na função com o tamanho das duas listas.

Se `k` for menor que o índice atual da mediana significa que o *k-Ésimo* elemento está na lista da esquerda. Chama e retorna recursivamente a função passando a lista dos valores menores que a mediana.

Se `k` for maior que o índice atual da mediana significa que o seu elemento está na lista dos elementos maiores que a mediana, portanto retorna o valor obtido da chamada da função recursivamente com o vetor dos valores maiores. No entanto, temos que atualizar o valor do arguemtno `k`, subtraindo o número total de elementos que serão ignorados da lista `esquerda`. Por fim, se `k` não é nem maior nem menor que `k_Atual` significa que encontramos o *k-ésimo* valor, portanto o valor da mediana do vetor atual é retornado.

Considerando $T(n)$ como a função de tempo do algoritmo, temos:

1 $\Theta(N)$
2 $\Theta(N)$
3 $\Theta(N)$
4 $\Theta(1)$
5 $\Theta(1)$
6 $T(\frac{n}{2})$
7 $T(\frac{n}{2})$
8 $\Theta(1)$

Portanto, pode-se definir a função $T(n)$ como:

$$T(n) = T(\frac{n}{2}) + n$$

$$\begin{aligned}
T(n) &= T(\frac{n}{2}) + n \\
&= T(\frac{n}{4}) + n(1 + \frac{1}{2}) \\
&= T(\frac{n}{8}) + n(1 + \frac{1}{2} + \frac{1}{4}) \\
&\vdots \\
&= T(\frac{n}{2^k}) + n\underbrace{(1 + \frac{1}{2} + \frac{1}{4} + ... + \frac{1}{2^{k-1}})}_{\text{Soma de P.G.}}
\end{aligned}$$

Igualando $k = \lg n$, temos:

$$\begin{aligned}
T(n) &= T(1) + n(1 + \frac{1}{2} + \frac{1}{4} + ... + \frac{1}{2^{\lg n - 1}}) \\
&= 1 + 2n - 2 \\
&= \Theta(n)
\end{aligned}$$