Algoritmo MaiorMenor(v,n):

1. maior ← v[1]
2. menor ← v[1]
3. **PARA** i ← 2 **ATÉ** n **FAÇA:**
4. **SE** v[i] > maior
5. **ENTÃO** maior ← v[i]
6. **SENÃO SE** v[i] < menor
7. **ENTÃO** menor ← v[i]
8. **devolva** maior, menor

Pelo princípio da esperança, a esperança de um valor i de um vetor ser menor que o valor dos i-1 elementos anteriores é $\frac{1}{i}$ (vetor se inicia na posição i. Logo

a esperança da linha 6 é o complemento da esperança da linha 4.

X = variável binária [0,1] onde 0 corresponde a não menor e 1 a menor

linha 4.

$$E[X]_{\text{linha 4}} = \sum E[x_i] = E[x_1]+E[x_2]+\cdots+E[x_n] = \sum_{i=2}^{n} \frac{1}{i} = \frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots+\frac{1}{n} = \lg n = \Theta(\lg n)$$

⇒ série harmônica - n(ln(n)) ∴ E[X] = ln n

linha 6:

$$E[X]_{\text{linha 6}} = \sum_{i=2}^{n} 1 - E[x_i]_{\text{linha 4}} = 1-\frac{1}{2}+\cdots+1-\frac{1}{n} \Rightarrow \underbrace{(1+1+1+\cdots+1)}_{(n-1)} - \underbrace{\left(\frac{1}{2}+\frac{2}{3}+\cdots+\frac{n-1}{n}\right)}_{(n-1)}$$

$$\sum_{i=2}^{n} 1 \quad - \sum_{i=2}^{n} \frac{i-1}{i}$$

⇒ $n-1-\lg n = \Theta(n)$

linha 7: como já sabemos que v[i] não é o maior de todos temos

$$E[X] = \sum_{i=2}^{n} \frac{1}{i-1} \rightarrow \text{multiplicando: } \frac{1}{i-1}\cdot\frac{i-1}{i} = \frac{1}{i}.$$ a esperança terá essa $\rightarrow \Theta(\lg n)$

exercício (2)

```
PARTICIONE(A, p, r):
1.  indice ← FIND_meio // o pivo escolhido é o central
2.  pivo ← A[indice]
3.  aux ← A[indice] // Trocamos o pivô com o último valor do vetor
4.  A[indice] ← A[r]
5.  A[r] ← aux
6.  inicio ← p
7.  PARA j de p até r-1 FAÇA // percorre o vetor
8.      SE A[j] < pivo
9.          aux ← A[inicio]          // troca de posição de modo que
10.         A[inicio] ← A[j]         // o lado esquerdo da peça do
11.         A[j] ← aux               // pivô vem menos que ele e
                                     // o lado direito maiores
12. aux ← A[i]
13. A[i] ← A[r]
14. [A]r ← aux
15. RETORNE i
```

```
K-esimo_menor(A, p, r, k)
    Se p = r
        devolva A[1]
    q ← PARTICIONE(A, p, r)
    prox_pivo ← 1 + q - p
    Se k < prox_pivo
        devolva K-esimo(A, p, q-1, k)
    SENÃO Se k = prox_pivo
        devolva A[q]
    SENÃO
        devolva K-esimo(A, q+1, r, k - prox_pivo)
```

FUNCIONAMENTO:

a linha 1 e 2 tratam do caso base do algoritmo, garantindo que, caso o vetor possua apenas um elemento, retorne o mesmo.

O algoritmo utiliza o particione para dividir o vetor em duas partes e garante que o lado esquerdo do vetor contenha apenas elementos menores que q e que o lado direito contenha apenas elementos maiores que q. (k-esimo_menor, linha 3)

O próximo pivô é calculado como 1 + q - p. através da comparação de k com pivô ele determinará em qual parte do A o k-esimo menor está para chamar recursivamente o k-esimo.

Análise do consumo de tempo:

1. $O(n)$
2-6: $O(1)$
7. $O(n)$
8-11: $O(1)$
12-15: $O(1)$

soma total: $O(n)$

K-ésimo_menor:
1-2: $O(1)$
3. $O(n)$
4-7: $O(1)$
8. $O(\log n)$
9. $O(1)$
10. $O(\log n)$

soma total: $O(n) \cdot O(\log n) = n \log n$

análise de corretude

RECORRÊNCIA

$$T(n) = T(n/2) + cn$$
$$= T(n/2^2) + \frac{cn}{2} + cn$$
$$= T(n/2^2) + \frac{cn}{4} + \frac{cn}{2} + c\,n$$
$$= T(n/2^k) + cn \sum_{i=0}^{k-1} (1/2) \rightarrow \frac{-2}{n} + 2$$
$$= \text{para } k = \lg n, \; = T(n/n) + cn\left(\frac{-2}{n} + 2\right) = -2c + 2cn = \theta(n)$$

$T(n/n) = T(1)$

INDUÇÃO:

POR INDUÇÃO:

① $T(n) = T(n/2) + cn$
② $T(n) = 2cn - 2c$

⊕ para a base $k=0 \Rightarrow T(2^0) = 2^0 = 1$ ✓

1.1 para $n = 2^k$: $T(\frac{2^k}{2}) + 2^k c = T(2^{k-1}) + 2^k c$ (recorrencia) +
2.1 para $n = 2^k$: $2c\,2^k + 2c = 2^{k+1}c + 2c$
2.2 para 2^{k-1}: $2c(2^{n-1}) + 2c = 2^k c + 2c$

$T(2^k) = 2c(2^k) - 2c = 2^{k+1}c - 2c$

⑨ **mediana** (A_1, A_2, n)

```
PARA i de 0 até n:
    Se i=n:
        av = b
        b = A[1]
    Se j = n^2
        a = b
        M = A[i]
        i = i+1
devolva (a+b)/2
```

complexidade: $O(n)$

RECORRENCIA:
$T(n) = cn$