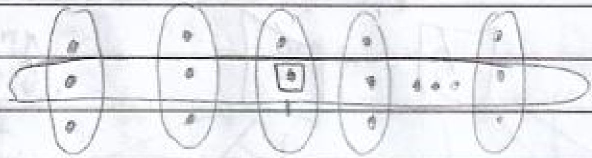
data
fecha

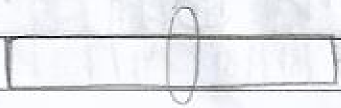
D S T Q Q S S
D L M M J V S

Lista 11
CANA

① Grupinhos de 3: 

$\Downarrow$

$$\frac{n}{3} + \frac{n}{3} = \frac{\frac{2n}{3}}{2} = \frac{n}{3}$$

$n/3$  $\frac{2n}{3}$

No pior caso o pivô será maior que $n/3$ elementos e menor que $\frac{2n}{3}$.

No pior caso, $T(n) = T\left(\frac{2n}{3}\right) + T\left(\frac{n}{3}\right) + t_q(n)$
temos que:

$t_q(n) \leq a \cdot n$, então $Tn \leq T\left(\frac{2n}{3}\right) + T\left(\frac{n}{3}\right) + a \cdot n$

Supondo $T(n) = O(n)$, se $T(n) \leq c \cdot n$, logo,

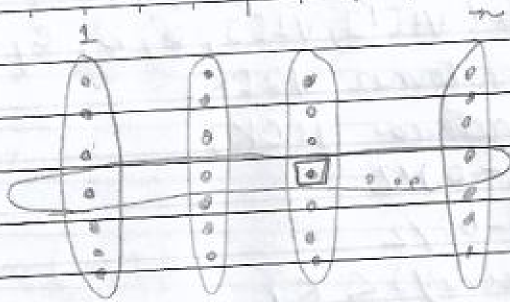
$c \cdot n \leq c \cdot \left(\frac{2n}{3}\right) + c\left(\frac{n}{3}\right) + a \cdot n \iff$

$a \cdot n \geq c \cdot n - c \cdot \frac{2n}{3} - c \cdot \frac{n}{3}$

$a \geq c - c \cdot \frac{2}{3} - c \cdot \frac{1}{3} \iff a \geq c\left(1 - \frac{2}{3} - \frac{1}{3}\right)$

$a > c$, o que me diz que ainda é $O(n)$.

• grupinhos de 7:



$$\frac{n}{7} + \frac{3n}{7} = \frac{4n}{7}$$

$$= \frac{\frac{4n}{7}}{2} = \frac{4n}{14} = \frac{2n}{7}$$

$$\frac{2n}{7} \Big] \quad \bigcirc \quad \Big[\frac{5n}{7}$$

No pior caso o pivô será maior que $\frac{2n}{7}$ e menor que $\frac{5n}{7}$.

Assim,

$$T(n) = TDC\left(\frac{5n}{7}\right) + TDC\left(\frac{n}{7}\right) + TQ(n)$$

temos que, $TQ(n) \leq a \cdot n$, então $T(n) \leq TDC\left(\frac{5n}{7}\right) + TDC\left(\frac{n}{7}\right) + a \cdot n$

Supondo $T(n) = O(n)$, se $T(n) \leq c \cdot n$, logo,

$$c \cdot n \leq c \cdot \left(\frac{5n}{7}\right) + c \cdot \left(\frac{n}{7}\right) + a \cdot n \iff$$

$$a \cdot n \geq c \cdot n - c \cdot \left(\frac{5n}{7}\right) - c \cdot \left(\frac{n}{7}\right)$$

$$a \geq c - c \cdot \frac{5}{7} - c \cdot \frac{1}{7} \iff a \geq c\left(1 - \frac{5}{7} - \frac{1}{7}\right)$$

$a > c$, ainda é $O(n)$.

$O(n)$ para $\frac{1}{7}c \leq a$.

D S T Q Q S S
D L M M J V S

(2) Selecao_Modificada(V1[ ], V2[ ], a, b, c, d, K)
se (a = c) retorna V2[K]
se (b = d) retorna V1[K]
meiodeX ← (c - a)/2
meiodeY ← (d - b)/2
se ((meiodeX + meiodeY) < K)
se (V1[meiodeX] > V2[meiodeY])
retorne Selecao_Modificada(V1[ ], V[2], a, b + meiodeY + 1, ...
... c, d - (meiodeY - 1)
senão
retorna Selecao_Modificada(V1[ ], V2[ ], (a + (meiodeX + 1)), b, c...
... d, (K - (meiodeX - 1))
senão
se V1[meiodeX] > V2 [meiodeY]
retorna Selecao_Modificada (V1[ ], V2[ ], a, b, (a + meiodeX), d, K)
senão retorne Selecao_Modificada(V1[ ], V2[ ], a, b, c, (b + meiodeY, K))

O(log n + log m)

Lista 09 - Cana

① Procedimento maiorDif (array [1..n])
    se (i >= j) : retorna - ∞
    se (j == i+1) : retorna valorabsoluto(array [i] - array [j])
    K = j / 2
    retorn max (maiorDif (array, i, K), maiorDif (array, K+1, j))

Procedimento CompFinal (array, i, j):
    dif = maiorDif (array, i, j)
    K = j / 2
    menor = array [1]
    maior = array [K]

    Para m de 1 até K+1 :
        se (array [m] < menor):
            menor = array [m]
    Para m de K+1 até j:
        se (array [m] > maior):
            maior ← array [m]
    retorna max (dif, valorabsoluto (maior - menor))

data
fecha

D S T Q Q S S
D L M M J V S

Lista 12 - Caso ~

Procedimento mdc Euclides (m, n)
se (n == 0) retorna m
retorna mdc Euclides (n, m % n)

Complexidade: O (log n)

No pior caso, esse algoritmo é aproximadamente O(log n)

data
fecha

D S T Q Q S S
D L M M J V S

Lista 14 - Lana

①

ⓐ Bolha recursivo.

```
1  Algoritmo troca (a, b)
2      aux ← a
3      a ← b
4      b ← aux

5  Procedimento bubbleSort (V[1..n])
6      Se (n < 1) retorna
7      Para i de 1 até n
8          se (V[i] > V[i+1])
9              troca (V[i], V[i+1]
10     BubbleSort (V[1.. n-1])
```

ⓑ

O procedimento verdadeira na linha 7 :
No início (27) o valor de V[i] é o menor, uma vez que inicia na posição 1. Se tiver apenas um elemento, a solução é trivial e vai retorna o menor valor.
O se (L8) compara os elementos em V[i] e V[i+1], trocando V[i] por V[i+1] se for o valor menor e deixando no lugar se não for.
No início o valor em V[1] era o menor, isso significa que o valor em V[i-1] é agora o menor valor em V[i-1] até n.
Isso também significa que cada valor em V[i] até n é maior do que o valor em V[i-1].
Como a chamada recursiva retorna V[1.. n-1] a cada chamada um procedimento se repete, e (ie) significa que o valor em A[i] (que é A[i-1] será o menor.