Nome: Sabrina Araujo da Silva   nºUSP: 12566182

MAC0338 - ANÁLISE DE ALGORITMOS

LISTA 2

exercícios 1(a), 1(d), 3 e 4

1. Resolva as recorrências abaixo.

(a) $T(n) = 2T(\lfloor n/2 \rfloor) + \Theta(n^2)$

$T(1) = 1$ e $T(n) = 2T(n/2) + n^2$ para $n \geq 2$ potência de 2

por expansão

$$T(n) = 2T(n/2) + n^2$$

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

$$= 2^2\left(2T\left(\frac{n}{2^3}\right) + \frac{n^2}{2^4}\right) + \frac{3n^2}{2} = 2^3T\left(\frac{n}{2^3}\right) + 2^2\frac{n^2}{2^4} + \frac{3n^2}{2} = 2^3T\left(\frac{n}{2^3}\right) + \frac{7n^2}{2^2}$$

$$= 2^3\left(2T\left(\frac{n}{2^4}\right) + \frac{n^2}{2^6}\right) + \frac{7n^2}{2^2} = 2^4T\left(\frac{n}{2^4}\right) + 2^3\frac{n^2}{2^6} + \frac{7n^2}{2^2} = 2^4T\left(\frac{n}{2^4}\right) + \frac{15n^2}{2^3}$$

$$= 2^4\left(2T\left(\frac{n}{2^5}\right) + \frac{n^2}{2^8}\right) + \frac{15n^2}{2^3} = 2^5T\left(\frac{n}{2^5}\right) + 2^4\frac{n^2}{2^8} + \frac{15n^2}{2^3} = 2^5T\left(\frac{n}{2^5}\right) + \frac{31n^2}{2^4}$$

$$= \dots = 2^kT\left(\frac{n}{2^k}\right) + (2^k - 1)\frac{n^2}{2^{k-1}} \quad \text{para } k = \lg n$$

$$= 2^{\lg n}T\left(\frac{n}{2^{\lg n}}\right) + (2^{\lg n} - 1)\frac{n^2}{2^{\lg n - 1}}$$

$$= nT\left(\frac{n}{n}\right) + (n-1)\frac{n^2}{\frac{1}{2}n} = n + (n-1)\frac{2}{n}n^2$$

$$= n + (n-1)2n = n + 2n^2 - 2n$$

$$= 2n^2 - n$$

$$= \Theta(n^2)$$

verificação

$$T(n) = 2T(n/2) + n^2 = n + n^2$$

por hipótese de indução

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

$$= 2\left(2\frac{n^2}{2^2} - \frac{n}{2}\right) + n^2 = n^2 - n + n^2$$

$$= 2n^2 - n$$

logo, $T(n) = 2T\left(\frac{n}{2}\right) + n^2 = n + n^2 = \Theta(n^2)$

(d) $T(n) = 7T(\lfloor n/3 \rfloor) + \Theta(n^2)$

$T(1) = 1$ e $T(n) = 7T\left(\frac{n}{3}\right) + n^2$ para $n \geq 3$ potência de 3

por expansão

$$T(n) = 7T\left(\frac{n}{3}\right) + n^2$$

$$= 7\left(7T\left(\frac{n}{3^2}\right) + \frac{n^2}{3^2}\right) + n^2 = 7^2T\left(\frac{n}{3^2}\right) + 7\frac{n^2}{3^2} + n^2$$

$$= 7^2\left(7T\left(\frac{n}{3^3}\right) + \frac{n^2}{3^4}\right) + 7\frac{n^2}{3^2} + n^2 = 7^3T\left(\frac{n}{3^3}\right) + 7^2\frac{n^2}{3^4} + 7\frac{n^2}{3^2} + n^2$$

$$= 7^3\left(7T\left(\frac{n}{3^4}\right) + \frac{n^2}{3^6}\right) + 7^2\frac{n^2}{3^4} + 7\frac{n^2}{3^2} + n^2 = 7^4T\left(\frac{n}{3^4}\right) + 7^3\frac{n^2}{3^6} + 7^2\frac{n^2}{3^4} + 7\frac{n^2}{3^2} + n^2$$

$$= 7^4\left(7T\left(\frac{n}{3^5}\right) + \frac{n^2}{3^8}\right) + 7^3\frac{n^2}{3^6} + 7^2\frac{n^2}{3^4} + 7\frac{n^2}{3^2} + n^2 = 7^5T\left(\frac{n}{3^5}\right) + 7^4\frac{n^2}{3^8} + 7^3\frac{n^2}{3^6} + 7^2\frac{n^2}{3^4} + 7\frac{n^2}{3^2} + n^2$$

$$= \cdots = 7^kT\left(\frac{n}{3^k}\right) + \left(\sum_{i=0}^{k}\frac{7^i}{3^{2i}}\right)n^2 \quad \text{para } k = \log_3 n$$

$$= 7^{\log_3 n}T\left(\frac{n}{n}\right) + \left(\sum_{i=0}^{\log_3 n}\frac{7^i}{3^{2i}}\right)n^2$$

$$= 7^{\log_3 n} + \left(\frac{\left(\frac{7}{3^2}\right)^{\log_3 n} - 1}{\frac{7}{3^2} - 1}\right)n^2 = 7^{\log_3 n} + \left(\frac{7^{\log_3 n}}{n^2} - 1\right)\cdot\left(-\frac{9}{2}\right)n^2$$

$$= 7^{\log_3 n} + \left(7^{\log_3 n} - n^2\right)\cdot\left(-\frac{3^2}{2}\right)$$

$$= \Theta(n^2)$$

verificação

$$T(n) = 7T\left(\frac{n}{3}\right) + n^2$$

por hipótese de indução

$$T(n) = 7T\left(\frac{n}{3}\right) + n^2$$

$$= 7\left(7^{\log_3 n} + \left(7^{\log_3 \frac{n}{3}} - \frac{n^2}{3^2}\right)\left(-\frac{3^2}{2}\right)\right) + n^2$$

$$= 7\left(7^{\log_3 n} + -\frac{3^2}{2}7^{\log_3 \frac{n}{3}} + \frac{3^2}{2}\frac{n^2}{3^2}\right) + n^2$$

$$= 7\left(7^{\log_3 n} - \frac{3^2}{2}7^{\log_3 \frac{n}{3}} + \frac{n^2}{2}\right) + n^2$$

$$= 7.7^{\log_3 n} - 7.\frac{3^2}{2}7^{\log_3 \frac{n}{3}} + \frac{3^2}{2}n^2$$

$$= 7^{1+\log_3 n - 1} - \frac{3^2}{2}.7^{1+\log_3 n - 1} + \frac{3^2}{2}n^2$$

$$= 7^{\log_3 n} - \frac{3^2}{2}7^{\log_3 n} + \frac{3^2}{2}n^2 = 7^{\log_3 n} + \left(7^{\log_3 n} - n^2\right)\left(-\frac{3^2}{2}\right)$$

logo, $T(n) = 7T\left(\frac{n}{3}\right) + n^2 = 7^{\log_3 n} + \left(7^{\log_3 n} - n^2\right)\left(-\frac{3^2}{2}\right) = \Theta(n^2)$

3. Seja $X[1\,..\,n]$ um vetor de inteiros e i e j dois índices distintos de X, ou seja, i e j são inteiros entre 1 e n. Dizemos que o par (i, j) é uma *inversão* de X se $i < j$ e $X[i] > X[j]$. Escreva um algoritmo $\mathrm{O}(n \lg n)$ que devolva o número de inversões em um vetor X, onde n é o número de elementos em X.

Podemos construir o algoritmo adaptando o algoritmo de ordenação mergesort

seja um vetor X[a...b]

```
INVERSOES (X, a, b)

    se a < b
        então q ← ⌊(a+b)/2⌋          q é o pivô
              c ← INVERSOES (X, a, q) +
                  INVERSOES (X, q+1, b) +
                  INTERCALA (X, a, q, b)
              retorna c
    senão retorna 0

INTERCALA (X, a, q, b)

    B[a...b] é um vetor auxiliar
    para i ← a até q faça
        B[i] ← X[i]
    para j ← q+1 até b faça
        B[b+q+1-j] ← X[j]
    i ← a
    j ← b
    contador ← 0
    para K ← a até b faça
        se B[i] ≤ B[j]
            então X[K] ← B[i]
                  i ← i + 1
            senão X[K] ← B[j]
                  j ← j - 1
                  contador ← contador + (q - i + 1)
    retorna contador
```

4. Descreva um algoritmo que, dados inteiros n e k, juntamente com k listas ordenadas que em conjunto tenham n registros, produza uma única lista ordenada contendo todos os registros dessas listas (isto é, faça uma *intercalação*). O seu algoritmo deve ter complexidade $O(n \lg k)$. Note que isto se transforma em $O(n \lg n)$ no caso de n listas de 1 elemento, e em $O(n)$ se só houver duas listas (no total com n elementos).

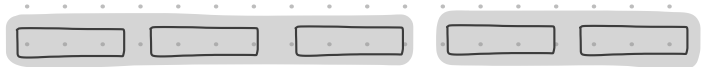
Esse algoritmo pode ser construído de forma que recursivamente o número de vetores é dividido por dois e em cada recursão dois vetores são intercalados. Como o número de vetores é dividido por dois serão $\Theta(\log k)$ iterações. É possível adaptar o algoritmo mergesort e utilizar a função intercala (mostrada em aula) também adaptada.

```
INTERCALA (A, p, q, r)
▷ B[p . . r] é um vetor auxiliar
para i ← p até q faça
    B[i] ← A[i]
para j ← q + 1 até r faça
    B[r + q + 1 − j] ← A[j]
i ← p
j ← r
para k ← p até r faça
    se B[i] ≤ B[j] e i ≤ q          ▷ alteração
        então A[k] ← B[i]
              i ← i + 1
        senão A[k] ← B[j]
              j ← j − 1
```

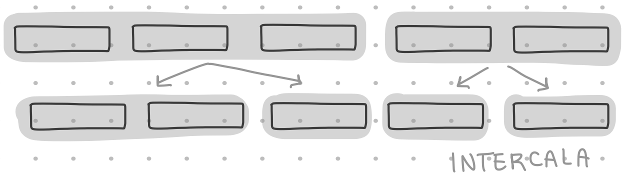
Assim, como a função intercala consome tempo $\Theta(n)$ e as recursões consomem $\Theta(\lg k)$, o algoritmo completo terá complexidade $O(n \lg k)$.

Simulação do algoritmo proposto para $k = 5$

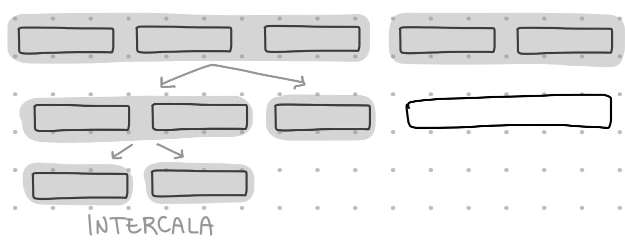
1. Inicialmente dividimos as 5 listas com o pivô sendo igual a $\lceil k/2 \rceil$



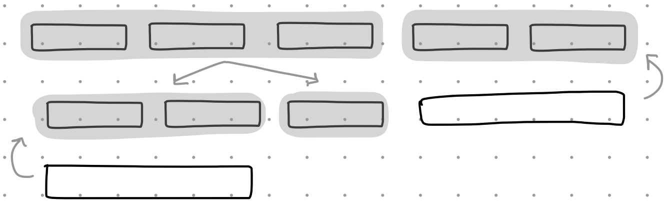
2. recursivamente temos



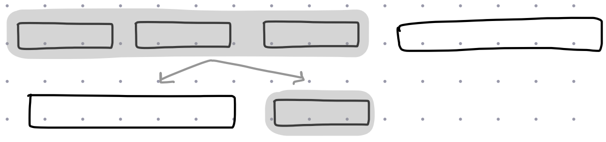
3. como na esquerda chegamos no momento da recursão no qual não há mais que 1 lista no grupo a função intercala é chamada, e na direita a recursão continua.



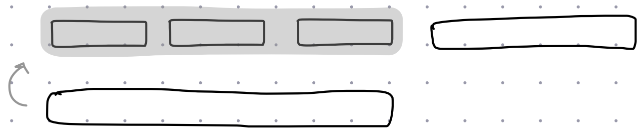
4. seguindo a lógica apresentada



5.



6.



7.

8. e recursivamente temos o vetor final

Pseudo código:

```
INTERCALACAO (n, K, L1,..., Li) (lista Li com i sendo o nº de listas em cada recursão indo de 1 a K)
se o número de listas for maior que 1
        K ← nº de listas atual
        então pivô ← ⌈K/2⌉
            (divide-se a listas em dois grupos de acordo com um pivô arredondando para cima)
            A ← INTERCALACAO (n, K, L1, ..., Lpivô)
            B ← INTERCALACAO (n,   Lpivô+1, ..., Li)
            D ← INTERCALA (n, A, B)
retorna D
```

```
INTERCALA (n, A, B)
    para i ← 0 até n-1 faça
        C[i] ← A[i]
        C[2n-1-i] ← B[i]        o vetor C é junção do A com o inverso do B
    i ← 0
    j ← 2n-1
    para K ← 0 até 2n-1 faça
        se C[i] ≤ C[j]
                D[0,...,2n-1] é um vetor auxiliar
            então D[K] ← C[i]
                i ← i+1
            senão D[K] ← C[j]
                j ← j-1
    retorna D
```