$$\sum_{i=0}^{n-1} c_i \le n + \sum_{k=0}^{\log_2 n} \left\lfloor \log_2 k + 1 \right\rfloor + \sum_{k=0}^{\log_2 n} \left[i - \lfloor \lg i \rfloor \right]$$

$$n + \sum_{k=1}^{\log_2 n} \log_2 k + \sum_{k=1}^{\log_2 n} 1 + \sum_{k=1}^{\log_2 n} k - \sum_{k=1}^{\log_2 n} \lfloor \lg k \rfloor$$

$$n + \frac{1}{2}(\log_2 n)\cdot(\log_2 n) + \log_2 n + \frac{1}{2}\log_2 n\,(1+\log_2 n) - \frac{1}{2}\log_2 n \cdot (\log_2 n)$$

$$n + \log_2 n + \frac{1}{2}\log_2 n + \frac{1}{2}\log_2^2 n$$

$$n + \frac{1}{2}\log_2^2 n + \frac{3}{2}\log_2 n$$

$$= n + \frac{1}{4} n + \frac{3}{2} n$$

$$= n + \frac{4}{2} n$$

$$= 2n$$

Acho que certo! / errado

$$\sum_{i=0}^{n-1} c_i \le \sum_{k=1}^{\log_2 2n} \left\lfloor \log_2 k + 1 \right\rfloor + \sum_{k=1}^{n} i - \log i$$

$$= \sum_{k=1}^{\log_2 n} \log_2 k + \sum_{k=1}^{\log_2 n} 1 + \sum_{k=1}^{n} i - \sum_{k=1}^{n} \log_2 i$$

$$= \frac{1}{2}\log_2 n\,(\log_2 n) + \log_2 n + n - \frac{1}{2} n (\log_2 n)$$

$$= \frac{1}{2}\log_2^2 n + \log_2 n + n - \frac{1}{2} n \log_2 n$$

$$= n \log_2 n \qquad \tfrac{1}{2} + 1 + 5 - \tfrac{1}{2}\ 5.1 \qquad 2.5$$

$$= O(n \log_2 n) \text{ para } n \text{ operações}$$

$$d(n)/n = \frac{n \log_2 n}{n} = \log_2 n \quad \text{// custo amortizado}$$

Para n múltiplo de 3

$t(n) = O(\log_3 n)$

Para não múltiplo

$t(n) = O(n - \log_3 n)$

custo total amortizado: $O(n \log_3 n)$

custo amortizado por operação $O(\log_3 n)$ //

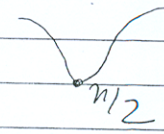
Prova - 2003/2

Questão 1 - $f(j) = (n-j)^2 + j^2$

$f'(j) = 2(n-j)\cdot(-1) + 2j = -2n + 2j + 2j = 4j - 2n$

$f'(j) = 0$

$4j - 2n = 0 \Rightarrow j = \frac{2n}{4} \Rightarrow j = n/2$



$f'(n/4) = 4\cdot\left(\frac{n}{4}\right) - 2n = -n$ negativo

$f'(n/2+1) = 4\cdot\left(\frac{n}{2}+1\right) - 2n = 2n + 4 - 2n = 4$ positivo

Para $j = n/4$

$j < n/2$ e $f'(j) < 0$ decrescente

Para $j = n/2 + 1$

$j > n/2$ e $f'(j) > 0$ crescente

Então $n/2$ é ponto mínimo, assim forma pontos próximos a 0 e a n maximizam a função

Questão 2 - Prove que $\sum_{i=1}^{n} \frac{i}{2^i} \leq 2$

$$\sum_{i=1}^{n} \frac{i}{2^i} = \sum_{i=1}^{n} \sum_{j=i}^{n} \frac{1}{2^j} \Rightarrow \sum_{i=1}^{n} \frac{\left(\frac{1}{2}\right)^{n+1} - \left(\frac{1}{2}\right)^{i}}{\frac{1}{2} - 1} = \sum_{i=1}^{n} \left[\left(\frac{1}{2}\right)^{n+1} - \left(\frac{1}{2}\right)^{i}\right] \cdot (-2)$$

$$= -2 \cdot \left[\sum_{i=1}^{n} \frac{1}{2^{n+1}} - \sum_{i=1}^{n} \left(\frac{1}{2}\right)^{i}\right]$$

$$= -2 \cdot \left[\frac{n}{2^{n+1}} - \left(\sum_{i=0}^{n} \left(\frac{1}{2}\right)^{i} - 1\right)\right]$$

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

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

$$= \frac{-2n}{2^{n+1}} - \frac{4}{2^{n+1}} + 4 - 2$$

$$= \frac{-2n-4}{2^{n+1}} + 2$$

$$= +2 - \left(\frac{2n+4}{2^{n+1}}\right)$$

$$\leq 2 \quad //$$

e) $\frac{1}{n+1} + \frac{1}{n+2} + \ldots + \frac{1}{2n} > \frac{13}{24}$

$\frac{1}{2n} \quad \frac{1}{n+n-1}$

$\sum_{i=1}^{n} \frac{1}{n+i}$

Irei mostrar que $\sum_{i=1}^{n} \frac{1}{n+i} > \frac{13}{24}$

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

$\leq \frac{1}{2} \cdot (2n-n-1+1) \cdot \left((n+1)^{-1} + (2n)^{-1}\right)$

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

$= \frac{1}{2} \cdot \frac{2n+1}{2n^2+2n} = \frac{2n^2+n}{4n^2+4n} > \frac{16}{16+8} > \frac{13}{24}$

Questão 3 -

SOMA(V, x)

1 Para i ← 1 até n
2 Se BUSCABIN(V, x − V[i]) ≠ -1
3 devolva 1
4 devolva 0

Este algoritmo leva tempo $O(n \cdot \lg n)$

SOMA2(V, x)

i ← 1 j ← n
enquanto i < j
se V[i] + V[j] = x
então devolva 1
senão se V[i] + V[j] > x
então j ← j − 1
senão i ← i + 1
devolva 0

Este algoritmo leva tempo $O(n)$

Se a soma é maior que x deve-se diminuir o valor reduzindo o maior número e se a soma for menor deve-se aumentar a soma aumentando o menor número. Aproveita-se a ordenação crescente.

Questão 4 -

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

$1 \div 1+1 =$

$\log \sum_{k=1}^{n} k + \log n$

$\sum_{i=2}^{n} \log \sum_{k=1}^{i-1} k + 1 + (\log(n) + 1 \quad + 1$

$\sum_{i=2}^{n} \log\left(\frac{1}{2}(i-1) \cdot (1+i-1)\right) + \sum_{i=2}^{n} \log n + \sum_{i=2}^{n} 2 \cdot o(n \log n)$

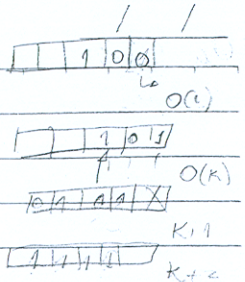
$\sum_{i=1}^{n} \log\left(\frac{1}{2}(i^2 - i)\right) + n \lg n + 2n$

$\sum_{i=2}^{n} \log\left(\frac{i^2}{2} - \frac{i}{2}\right) + \ldots$

$\frac{1}{2} n \left(\log\left(\frac{n^2}{2} - \frac{n}{2}\right)\right)$

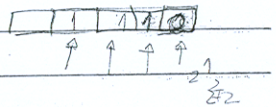
$\frac{1}{2} n \log\left(\frac{n^2 - n}{2}\right) + n \log n + 2n$

$o(n \log n)$



Questão 5 - Análise Amortizada

Incrementa1 (A)
i ← 0
enquanto i < comprimento[A] e A[i] = 1
A[i] ← 0
i ← i+1
se i < comprimento[A]
então A[i] ← 1
i ← i + 1 → 1 mais significativo



Zera-Contador (A)
i ← 0
enquanto i ≤ K — O(K)
A[i] ← 0
i ← i+1 — log n

K como var global

Uma única execução de incrementa pior tempo O(K) no pior caso é uma única execução de zera-contador poderia também levar tempo O(K) caso o 1 estivesse na posição mais significativa.
O zera-contador só zera caso o incrementa tenha sido executado
Vamos supor $n/2$ operações incrementa onde cada uma é precedida de uma operação zera-contador

$$\sum_{i=0}^{\log(n/2)} \left[\frac{n}{2^i} + i + 1\right] = \sum_{i=0}^{\log(n/2)} \frac{n}{2^i} + \sum_{i=0}^{\log n/2} i + \sum_{i=0}^{\log n/2} 1$$

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

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

$$= -2n \cdot \frac{1}{2^{\log n/2} \cdot 2} + 2n + \frac{1}{2}(\log n - 1)^2 - \frac{1}{2}(\log n - 1) + \log n - 2$$

$$= -\frac{n}{n/2} + 2n + \frac{1}{2}(\log^2 n - \log n + 1) - \frac{1}{2}\log n + \frac{1}{2} + \log n - 2$$

$$= -\frac{2n}{n} + 2n - 2 + \frac{1}{2} + \frac{1}{2}\log^2 n - \frac{1}{2}\log n + \frac{1}{2} + \log n$$

$$= -2 + 2n - 2 + 1 + \frac{1}{2}\log^2 n + \frac{1}{2}\log n$$

$$= 2n - 3 + \frac{1}{2}\log^2 n + \frac{1}{2}\log n$$

$$\leq 2n + \frac{1}{2}n + \frac{1}{2}n$$

$$\leq 3n$$

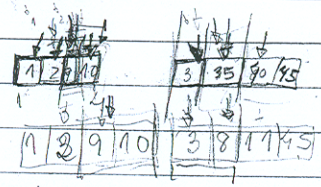
$O(n)$

Questão 6 -

1- Significa que este problema está na classe NP e qualquer problema NP pode ser reduzido polinomialmente a este problema.
O fato de se descobrir um novo problema NP-completo aumenta o conjunto de possibilidades de se reduzir polinomialmente um problema NP.

2- 1- Verdadeira. Considerando P≠NP é trivial que P ∩ NP-completo = ∅.
2- Falso, ???
3- Verdadeiro. P ⊆ NP
4- Verdadeiro. Quando A é reduzido a B então A pode ser resolvido por meio de B, sendo este último possuidor de solução polinomial então A também possui solução polinomial.
5- Falso. Para ser NP-completo não basta ser NP ele precisa necessariamente que qq NP seja reduzido a ele e não o contrário.

Questão 7 -



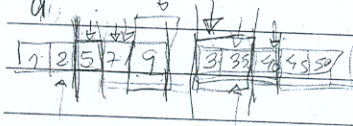

```
MEDIANA(X, Y, n)
1  q ← ⌈2n/2⌉      X[n+1] ← ∞   Y[n+1] ← ∞
2  i ← 1  j ← 1  k ← 0
3  enquanto k < q
4        se X[i] < Y[j]
5              então med ← X[i]  i ← i+1
6              senão med ← Y[j]  j ← j+1
   k ← k+1
   devolva med
```

A linha 3 executa q vezes ou seja $2n/2 = n$ vezes

O livro →

```
MEDIANA(X, Y, n)
    devolva MED(X, 1, n, Y, 1, n)

MED(X, px, rx, Y, py, ry)
1   se px ≥ rx
2         então se rx - px + 1 = 1
3               então se X[px] < Y[py]
4                     então devolva X[px]
5                     senão devolva Y[py]
6         senão q ← ⌈(rx - px + 1)/2⌉
7               se X[px+q-1] < Y[py+q-1]
8               então devolva MED(X, px+q, rx, Y, py, py+q-1)
9               senão devolva MED(X, px, px+q-1, Y, py+q, ry)
```



Só funciona para n par:

$T(n) = O(1) + T(n/2)$
$\ldots O(\lg n)$

Questão 8 - pg 205

Prova 2003/1 Não sei fazer!

* Questão 1 - $T_A(n) = 0$ para $n = 256$

$T_A(n) = T(n/2) + 3n$ para $> 2^9$

$T_B(n) = 0$ para $n = 2^8, 2^9, 2^{10}, 2^{11}$

$T_B(n) = T_B(n/2)$ + 13.107z

$T_C(n) = 0.06$ para $n = 1000000$ X

$T_C(n) = 2.(T(n-1) + T(n-2)$

Questão 2 - HEAP-MIN

KMINIMO (A, K, n)

1	B ← BUILD-HEAP (A, n)	O(n)
2	Para i ← 1 até K	O(K)
3	IMPRIME (EXTRACT-MIN(B))	O(lgn)

$O(n) + O(k . \lg n)$

A partir do Heap Mínimo criado é fácil retirar os K elementos mínimos mas a cada extração é necessário rearranjar o heap para que continue mínimo. que leva no pior caso O(lgn) a rotina Build-Heap constrói um heap mínimo em O(n).

* Questão 3 - a) READFILE (char *dname, DIR * dir)

DO READDIR(dir)

enquanto D ≠ NULL e D.dname ≠ dname

D ← D + d.off - D

se D.dname = dname

devolva D

senão devolva NULL

b) O(|d_name|)

Questão 4 - a) pJÓIA ≤p JÓIA ?

Sim.

Prova de que PJOIA ≤p JÓIA:

Suponha uma Instância $(G, \langle v, u \rangle)$ para JÓIA e uma Instância $(G, \langle i, j \rangle)$ para PJOIA.

Existe uma Instância $(G, \langle i, j \rangle)$ que é JÓIA para $\langle i, j \rangle$ uma renumeração de 1..n se e somente se existe uma Instância $(G, \langle v, u \rangle)$ JÓIA $\langle v, u \rangle = \langle i, j \rangle$

3D-Matching (Emparelhamento.

Correspondência tuerata de Grafos

3D ≤p correspondência

b) Sim, pois se pJÓIA ≤p JÓIA existe um certificado para pJÓIA.

Questão 5 -

1ºA - $\Theta(n^2)$

2ºA - $T(n) = 2T(n/2) + n \lg n$ $i = \lg n$

$T(1) = 1$

$T(n) = 2T(n/2) + n \lg n$

$= 2(2T(n/4) + n/2 \lg n/2) + n \lg n$

$= 2^2(2T(n/8) + n/4 \lg n/4) + n \lg n/2 + n \lg n$

$= 2^3 T(n/2^3) + n \log n/4 + n \lg n/2 + n \lg n$

$= 2^i T(n/2^i) + n \cdot \sum_{k=0}^{i-1} \log \frac{n}{2^k}$

$= n \cdot 1 + n \cdot \sum_{k=0}^{\lg n - 1} \lg n - \log 2^k$

$= n + n \left[\sum_{k=0}^{\lg n - 1} \lg n - \sum_{k=0}^{\lg n - 1} k \right]$

$= n + n \cdot \lg n \cdot \lg n - n \cdot \frac{1}{2} \cdot \lg n \cdot (\lg n - 1)$

$= n + n \lg^2 n - \frac{n}{2} \lg^2 n + \frac{1}{2} n \lg n$

$= n + \frac{1}{2} n \lg^2 n + \frac{1}{2} n \lg n$

$= n \lg^2 n$ //

Provar por indução...

2º A – $O(n \lg^2 n)$

O segundo algoritmo é mais eficiente

Questão 6 - a) se a = b temos

a = 2 b = 1	a = 2 b = 2
P(1,1) + P(2,0)	P(1,2) + P(2,1)
P(0,1) + P(1,0) + 1	P(0,2) + P(1,1) + P(1,1) + P(2,0)
1 + 1 + 1 = 3	1 + P(0,1) + P(1,0) + P(0,1) + P(1,0) + 1
	1 + 1 + 1 + 1 + 1 + 1 = 6

$T(a) = 2T(a-1) + 1$

$T(a) = 2 \cdot (2T(a-2) + 1) + 1$

$= 4 \cdot T(a-2) + 2 + 1$

$= 4 \cdot 1 + 2 + 1 = 7$

a = 3 b = 3
P(2,3) + P(3,2)
P(1,3) + P(2,2) + P(2,2) + P(3,1)
P(0,3) + P(1,2) + P(1,2) + P(2,1) + P(1,2) + P(2,1) + P(2,1) + P(3,0)
1 + P(0,2) + P(1,1) + P(0,2) + P(1,1) + P(1,1) + P(2,0) + P(0,2) + P(1,1)

$T(a) = 1 \quad a = 1$

$T(a) = 2T(a-1) \quad a > 1$

$T(a) = 2^2 T(a-2)$

$T = 2^3 T(a-3)$

$= 2^i T(a-i)$

$= 2^{a-1} T(1)$

$= 2^{a-1}$

$T(a) \geq 2^{a-1}$

Prova por indução

base $a = 1$

$T(1) \geq 2^{1-1} = 1$

passo $a > 1$.

$T(k) \overset{H.I.}{\geq} 2 \cdot (2^{a-1-1})$

$= 2^{a-1-1+1}$

$= 2^{a-1}$ //

$a - i = 1$

$a - 1 = i$

Assim $T(a) \geq 2^{a-1}$ $T(a) = \Omega(2^a)$

b) recorrência:

$P[0, j] = 1$

$P[i, 0] = 1$

$P[i, j] = P[i-1, j] + P[i, j-1]$

PASCAL (a, b)

1 Para i ← 0 até a

2 P[i, 0] ← 1

3 Para j ← 0 até b

4 P[0, j] ← 1

5 Para j ← 1 até b

6 Para i ← 1 até a

7 P[i, j] ← P[i-1, j] + P[i, j-1]

8 devolva P[a, b]

a = 2 b = 1

a \ b	0	1	2
0	1	1	1
1	1	2	3
2	1	3	6

$P_{a,b}$

$P(a-1, b) + P(a, b-1)$

$P(i-1) + P(i)$

$(a, b) + (a+1$

$(a-1, b)$

(a, b)

$(a-2, b)$

Consumo:

Linha	Consumo
1-2	$\Theta(a)$
3-4	$\Theta(b)$
5	$\Theta(b)$
6-7	$\Theta(ab)$
8	$\Theta(1)$

Consumo total: $\Theta(ab)$

Espaço: $\Theta(a \cdot b)$

Outra solução com espaço $\Theta(\min\{a, b\})$

PASCAL (a, b)

se a > b então a ↔ b

~~Para i ← 0 até a~~

~~P[i] ← 1~~

Para j ← 1 até b

Para i ← 1 até a

P[i] ← P[i-1] + P[i]

a = 3 b = 1

Consumo $\Theta(ab)$

Espaço $\Theta(\min(a, b))$

Prova da subestrutura ótima

Teorema: Seja $P_{a,b}$ uma solução ótima para (a,b) então $P_{a-1,b}$ é solução para $(a-1,b)$ e $P_{a,b-1}$ $(a,b-1)$

Prova:

1- Suponha que $P_{a-1,b}$ não seja solução ótima para $(a-1,b)$ e que $P_{a,b-1}$ seja solução ótima para $(a,b-1)$ então existe uma outra solução para a instância $(a-1,b)$ que somada com $P_{a,b-1}$ produz uma outra solução ótima para $P_{a,b}$, uma contradição.

2- Suponha que $P_{a,b-1}$ não seja solução ótima para $(a,b-1)$ e que $P_{a-1,b}$ seja solução ótima para $(a-1,b)$ então existe uma outra solução para a instância $(a,b-1)$ que somada com $P_{a-1,b}$ produz uma outra solução para $P_{a,b}$, uma contradição.

3- Suponha que tanto $P_{a,b-1}$ quanto $P_{a-1,b}$ não sejam solução ótima para $(a,b-1)$ e $(a-1),b$...

Questão 4 - pag 131 OK!

a) QuickSort-K(A, p, r, K)

se $p < r - K$

q ← particione(A, p, r)

QuickSort(A, p, q-1, K)

QuickSort(A, q+1, r, K)

$T(n) = T(n-1) + n$ para $n > K$ $\quad r - p < K$

$T(n) = 0$ para $n = K$ $\quad n - i = K$

$T(n) = T(n-2) + n-1 + n$ $\quad i = n - K$

$= T(n-3) + n-2 + n-1 + n$

$= T(n-i) + \sum_{j=0}^{n-K-1} n-j$

$= T(k) + \sum_{j=0}^{n-K-1} n-j$

$= 0 + \sum_{j=0}^{n-K-1} n - \sum_{j=0}^{n-K-1} j$

$= (n-K)\cdot n - \frac{1}{2}(n-K)\cdot(n-K-1)$

$= n^2 - Kn - \frac{1}{2}\left[n^2 - Kn - Kn + K^2 - n + K\right]$

$= n^2 - Kn - \frac{1}{2}n^2 + \frac{1}{2}Kn + \frac{1}{2}Kn - \frac{1}{2}K^2 + \frac{1}{2}n - \frac{1}{2}K$

$= \frac{1}{2}n^2 + \frac{1}{2}n - \frac{1}{2}K^2 - \frac{1}{2}K$ $\quad O(n^2)$

$\leq n^2 - K^2 + n^2 = 2n^2 - K^2$ $\quad T(n) = O(2n^2 - K^2)$

$T(n) = 0 \quad n = K$

$T(n) = 2T(n/2) + n \quad n > K$

: Questão 8 - Prova 2008/1

$O(n \lg n/K)$

b) Insertion Sort

```
INSERTION-SORT(A, n)
  Para i ← 2 até n faça
      j ← i-1
      enquanto A[j] > A[i] e j > 0
          A[j] ↔ A[i]
          j ← j-1
```

$O(n^2)$

Já que no vetor de entrada os elementos estão ordenados apenas pela distância k então para cada trabalho no vetor A será feita a troca de no máximo k elementos duma forma a complexidade é de $O(n \cdot K)$.

c) pag 131

Questão 8 - a) Uma fila de prioridade first in first out
A operação Insira sempre insere no fim da fila
e a operação remova sempre extrai o 1º elemento
da fila levando tempo $O(1)$

b) Um Max-Heap, onde a tarefa mais antiga
é a maior e a recém-chegada é a de menor
valor.
A operação Insira busca o elemento no heap
em $O(\log n)$, caso não encontre insere um
novo elemento no Max-Heap que leva t_b tempo
$O(\log n)$.
A operação remova retira o elemento
da folha levando tempo $O(1)$

c) A estrutura é um vetor indexado de 0 a n
onde cada elemento inteiro i será armazenado
em $v[i]$.
A operação insira 1º verifica se $v[i]$ está
ocupado em tempo $O(1)$ e caso não esteja insere
o elemento i em $v[i]$ em tempo $O(1)$.
A operação remova remove o elemento
apontado por j sendo j um apontador para o
último elemento inserido, não precisando
dessa forma procurar uma posição na fila
levando t_b tempo $O(1)$.

Prova 2002/2

Questão 1 - $p-1$ $r=2$ 1 vez $n = r-p+1$
$q-1$ $p-1$ $r=3$ 2 vezes

$$\boxed{\begin{aligned} T(n) &= 0 \quad n=1 \\ T(n) &= T(n-1)+1 \end{aligned}}$$ recorrência

$n-i=1$
$i=n-1$

$$\begin{aligned} T(n) &= T(n-1)+1 \\ &= T(n-2)+1+1 \\ &= T(n-3)+1+1+1 \\ &= T(n-i)+i \\ &= T(1)+n-1 \\ &= n-1 \end{aligned}$$

A fórmula exata é $T(n) = n-1$

Vou provar que a fórmula está certa
Prova por indução em n:
base - $n=1$
$T(1) = 0 = 1-1 = 0$ OK!
passo $n>1$

$$\begin{aligned} T(n) &= T(n-1)+1 \\ &\overset{H.I.}{=} (n-1)-1+1 \\ &= n-1 \quad // \end{aligned}$$

$q(n) = n-1 \leq 1 \cdot n$ para $n \geq 1$ e
$n-1 \geq \frac{1}{2} n$ para $n \geq 1$

Assim concluímos que $T(n) = \Theta(n)$

Questão 2 - Falsa. O algoritmo build-Heap que recebe um vetor $A[1..n]$ e devolve um Heap com n elementos consome tempo $O(n)$ fazendo uma análise cuidadosa.