5.10 / 20.00 5.1

/ /

devolve "circuito verde-amarelo"

2 Para cada vértice K em Adj [u]

aresta ← (u, K)

Enquanto aresta ≠ NULL ou aresta ≠ uv

Se aresta = vermelho

vermelho ← 1 amarelo ← 0

, senão se aresta = amarelo

amarelo ← 1 vermelho ← 0

Se vermelho = 1 e amarelo = 0 e aresta = NULL

devolve vermelho-verde

Se vermelho = 0 e amarelo = 1 e aresta = uv

devolve circuito verde-amarelo

aresta ← Apos (aresta)

Consumo

$O(|E|) + O(|E| \cdot |E|)$

Questão 8 - Pg. 64

Prova 2001/2

Questão 1 - O limitante inferior para uma ordenação baseada em comparação é $\Omega(n \lg n)$, baseado nisso temos como afirmativa $O(n \lg \sqrt{n})$ como um suposto limite para um algoritmo de ordenação.

O professor Progresso está certo se $n \lg \sqrt{n} = \Omega(n \lg n)$

Vamos provar que $n \lg \sqrt{n} \geq \frac{1}{4} n \lg n$ para $n \geq 1$

$n \lg \sqrt{n} = n \lg n^{1/2}$

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

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

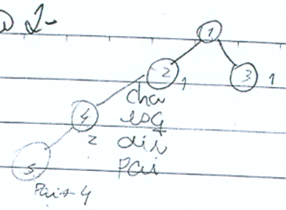
$\geq \frac{1}{4} n \lg n$

Então o professor está certo.

252

/ /

Questão 2 -



Algoritmo (x, y)

1 se x ≠ NULL

2 Algoritmo (esq (x), x)

3 Algoritmo (dir (x), x)

4 pai (x) ← y

O algoritmo recebe uma raiz e um pai = NULL, e devolve a árvore com o campo pai preenchido.

Consumo de tempo. Tamanho instância: n

Linha	T(n)
1	O(1)
2	T(K)
3	T(n-K-1)
4	O(1)

Teorema: Se x é raiz de subárvore de n nós então Algoritmo consome O(n).

Prova: Seja x a raiz de uma árvore vazia então o consumo é zero $T(n) = 0$

Seja a raiz diferente de nulo, a subárvore esquerda com K nós e a subárvore direita com (n-K-1) nós então $T(n) = T(K) + T(n-K-1) + \Theta(1)$.

Vamos provar que $T(n) \leq c_1 . n$ para $n \geq 0, c_1 \geq 1$

$T(n) = T(K) + T(n-K-1) + 1$

$\leq c_1 K + c_1 (n-K-1) + 1$

$= c_1 K + c_1 n - c_1 K - c_1 + 1$

$= c_1 n - c_1 + 1$

$\leq c_1 n$ //

$n^{n/2} \cdot n^{n/2} = n^{\frac{n}{2} + \frac{n}{2}}$

$n.n | n.n$

Questão 3 - EXPOENTE(n, n_0)

se $n = 0$

então devolva 1

se $n = 1$

então devolva n_0

se $n > 1$

então exp ← EXPOENTE($\lfloor n/2 \rfloor$, n_0).

se $(n/2 \% \lfloor n/2 \rfloor) = 0$

então exp ← exp * exp

senão exp ← exp * exp * n_0

devolve exp

3^3

$\frac{3}{2}$ 1

1,5 % 1

3 * 3 * 3

PanAmericana

Questão 4

a) Se $K = \Theta(2^n)$

$T(n)$ = consumo do radix para $K = \Theta(2^n)$

$T(n) = \Theta(n \cdot \log 2^n / \log n)$

$= \Theta(n \cdot n \log 2 / \log n)$ sejam a base do log: 2

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

$S(n)$ = consumo do Mergesort

$S(n) = n\, T(n)$?

Vou provar que:

$n \lg n \geq \frac{1}{4} \cdot n^2 / \lg n$ para $n \geq 2$

$n \lg n \geq \frac{1}{2} n \lg n$

$\geq \frac{1}{2} n \cdot n$

$\geq \frac{1}{4} n^2 / \lg n$

O radix é mais eficiente.

b) $T(n)$ = consumo do radix para $K = \Theta(2^{\log_2^3 n})$

$T(n) = \Theta(n \cdot \log 2^{\log_2^3 n} / \log n)$

$= \Theta(n \cdot \log^3 n \cdot \log 2 / \log n)$ seja a base do log 2

$= \Theta(n \lg^3 n / \lg n)$

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

$S(n)$ = consumo do Mergesort

$T(n) = n\, S(n)$?

Vou provar que $n \lg^2 n \geq \frac{1}{2} n \lg n$, para $n \geq 1$

$n \lg^2 n = n \lg n \cdot \lg n$

$\leq n \lg n$

$\leq \frac{1}{2} n \lg n$ //

O Mergesort é mais eficiente.

c) Se $K = \Theta(n^5)$

Para $K = \Theta(n^5)$ radix sort é $\Theta(n \cdot \log n^5 / \log n)$

$T(n)$ = consumo do radix

$T(n) = n \log n^5 / \lg n$

$= n \cdot 5 \log n / \log n$

$= 5n$

Seja $S(n)$ o consumo do Mergesort

Vou provar que $S(n) = n\, T(n)$ ou seja que

$n \log n \geq \frac{1}{10} \cdot 5n$ para $n \geq 2$

$n \log n \geq n$

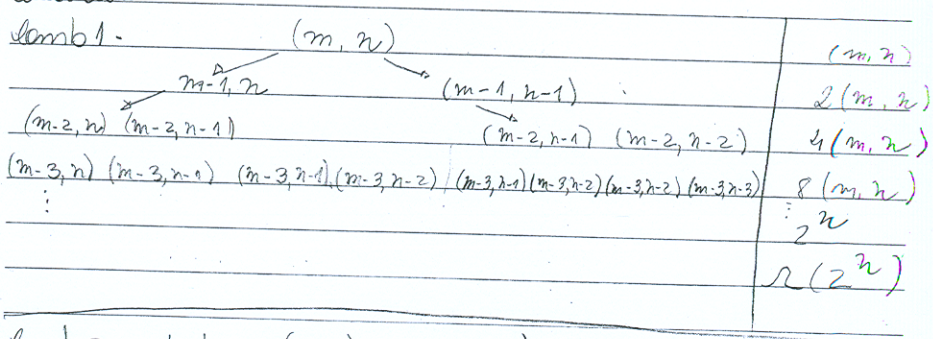
$> \frac{1}{2} n$

$= \frac{5}{10} n$ //

O radix é mais eficiente que o Mergesort.

Questão 5 -

lamb1 -



lamb2 - linha $T(m,n)$

1 $O(1)$

2-3 $O(m)$

4 $O(m)$

5-6 $O(m^2)$

Total $O(m^2)$

O algoritmo lamb2 é mais eficiente pois utiliza P.D. e tem consumo polinomial já lamb1 tem consumo exponencial.

Questão 6 - $T(1) = 1$

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

Mostre que $T(n)$ não é $O(n \lg n)$

Observação recursão: Para simplificar vamos considerar $T(1) = 1$

$T(n) = 2T(n/2) + n \lg n$ para $n \geq 2^1, 2^2 \ldots$

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

$= 2^3 T(n/8) + 2^2 n/4 \lg n/4 + 2 \cdot n/2 \lg n/2 + n \lg n$

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

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

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

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

$= n + n \lg^2 n - n \cdot 1/2 \cdot (\lg n \cdot (0 + \lg n - 1))$

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

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

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

Vou provar que $T(n) = \frac{1}{2} n \lg^2 n + \frac{1}{2} n \lg n + n$

Prova por indução em n:

base: $n = 1$

$T(1) = 1 = \frac{1}{2} \cdot 1 \lg^2 1 + \frac{1}{2} \cdot 1 \cdot \lg 1 + 1 = 1$ //

passo: $n \geq 2^1, 2^2 \ldots$

Hipótese $T(n/2) = \frac{1}{2} \cdot \frac{n}{2} \cdot \lg^2 \frac{n}{2} + \frac{1}{2} \cdot \frac{n}{2} \cdot \lg \frac{n}{2} + n/2$

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

$= 2 \cdot \left(\frac{n}{4} \lg^2 \frac{n}{2} + \frac{n}{4} \lg n/2 + n/2 \right) + n \lg n$

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

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

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

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

$= T(n)$

É fácil provar que $\frac{n}{2} \lg^2 n + \frac{n}{2} \lg n + n \leq 2 \cdot n \lg^2 n$

$\frac{n}{2} \lg^2 n + \frac{n}{2} \lg n + n \leq \frac{n}{2} \lg^2 n + \frac{n}{2} \lg^2 n + n$

$= n \lg^2 n + n$

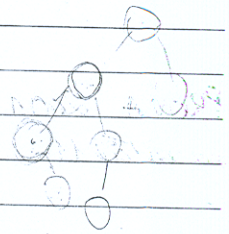
$\leq n \lg^2 n + n \lg^2 n$

$= 2 n \lg^2 n$ //

$T(n) = O(n \lg^2 n)$ e não é $O(n \lg n)$

Questão 7 - EMBUTIVEL (A, B)

```
1  Se raiz(A) = NULL então devolva 'S'
2  raizB ← BUSCA(B, raiz(A))
3  se raizB = NULL então devolva 'N'
4  sont ← EH-EMBUTIVEL(raiz(A), raiz(B))
5  se sont = 0
6      então devolva 'S'
7      senão devolva 'N'
```



```
EH_EMBUTIVEL(raizA, raizB)
1  se raizA ≠ NULL e raizB = NULL
2      então devolva 1
3  se raizA ≠ NULL e raizB ≠ NULL
4      então sont ← EH_EMBUTIVEL(esq(raizA), esq(raizB))
5          se sont ≠ 0 então devolva sont
6          se raizA ≠ raizB então devolva 1
7          sont ← EH_EMBUTIVEL(dir(raizA), dir(raizB))
8          se sont ≠ 0 então devolva sont
9  devolva 0
```

Tamanho instância: m - nº de nós em B

Consumo do EMBUTÍVEL $T(m,n)$	Total $(n + \log m)$
Linha 2 | $\log m$	
Linha 4 | n	

Questão 8 -

Custo Insere - $O(1)$

Custo remove - $O(k)$ sendo k o nº de elementos em D.

Como na sequência as operações não se pode remover um elemento da lista vazia a operação remove só poderá remover o que foi inserido.

Ou seja n elementos.

Seja $T(n)$ o custo da sequência acima

$T(n) = n \cdot O(1) + O(n)$ = n inserção + n remoção

$= O(n) + O(n)$

$= 2 \cdot O(n)$

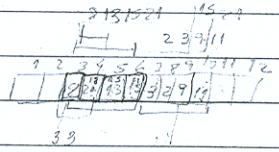
$= O(n)$.

Assim o custo amortizado para a sequência é $O(n)$

Logo custo amortizado por operação é $\frac{c(n)}{n} = O(1)$ //

PROVA 2001 | 1

Questão 11 -



$i = 3$ $j = 10$

$m = 6$ Não Sei

Tamanho da instância: $n = j - i + 1$

Seja $T(n)$ = tempo do algoritmo no melhor caso

linha	$T(n)$	Recorrência:
1-2	$\Omega(1)$	
3	$T(\lfloor n/2 \rfloor)$	
4	$T(\lceil n/2 \rceil)$	
5-6	$\Omega(1)$	$T(n) = 1$
7	$T(n-1)$	$T(n) = 2T(n/2) + T(n-1)$

Vou mostrar que $T(n) \geq c \cdot n^7$ $\forall n \geq n_0$ para c e n_0 constantes positivas.

Prova por indução em n:

Suponha que n é constante, então $T(n) \geq c\,n^7$ para c suficientemente pequeno.

Portanto tome $c = 1$ e $n_0 = 1$ e $n = 1$

base: $n = n_0$ ✓

passo: $n > n_0$

H.I. $T(n/2) \geq 2\left(c \cdot \frac{n}{2}\right)^7$

$T(n-1) \geq (c \cdot (n-1))^7$

$T(n) \geq 2\left(c \cdot \frac{n}{2}\right)^7 + (cn - c)^7 +$

$T(n) = 2\left(c\frac{n}{2}\right)^7 + cn^7 - \Theta(n^6)$

$\geq 2\left(\frac{c \cdot n}{2}\right)^7 + cn^7 - d \cdot n^6$

$= c \cdot n^7 + \underbrace{2\left(\frac{cn}{2}\right)^7 - dn^6}_{\geq 0}$

$\geq c \cdot n^7$

Corretude

Invariante: linha 5: o vetor $A[i..m]$ e $A[m+1..j]$ estão ordenados

Linha 7: o vetor $A[k, j]$ está ordenado para $i \leq k \leq j$

Prova do invariante $A[k, j]$

Início: $k = i$ e $j = i$, as linhas 2 e 4 ordenam os subvetores $A[i..i]$ e $A[i..i]$ com 1 elemento e linhas 5 e 6 trocam o vetor $A[i]$ por $A[i]$

Assim no início da linha 7 o vetor $A[i..i]$ está ordenado de forma trivial.

Manutenção: Para ver que cada recursão mantém o invariante, observe que os subvetores $A[i..m]$ e $A[m+1..j]$ são ordenados

e a troca entre A[m] e A[j] se A[m] ≥ A[j] garante que A[j] é o maior valor do subvetor A[i..j]

término: Após n-1 chamadas recursivas da linha 7 Slowsort(i, j-1) temos k = i e o vetor A[i, j] ordenado

Vou provar que $T(n) \geq \frac{1}{2^{10}} n^7$ para $n \geq 2$

Prova por indução em n

base n = 2

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

$= 2 \cdot 0 + 0 + 1$

$= 1$

$\geq \frac{1}{2^{10}} \cdot 2^7$

$= \frac{1}{2^3}$

passo $n \geq 2^2, 2^3 \ldots$

Hipótese: $T(n/2) = \frac{1}{2^{10}} \cdot (\frac{n}{2})^7 = \frac{n^7}{2^{17}}$

$T(n-1) = \frac{1}{2^{10}} \cdot (n-1)^7$

$T(n) = 2 \cdot T(n/2) + T(n-1) + 1$

$\overset{H.I.}{\geq} 2 \cdot (\frac{1}{2^{10}} \cdot (n/2)^7) + \frac{1}{2^{10}} (n-1)^7 + 1$

$= \frac{1}{2^6} \cdot \frac{n^7}{2^{10}} + \frac{(n-1)^7}{2^{10}} + 1$

$> \frac{n^7}{2^{16}} + \frac{1}{2^{10}} (n-1)^7$

$\geq \frac{n^7}{2^{16}} +$

Questão 1 -

$f(1) = 0$

$f(2) = f(1) + \log n$

$f(3) = f(1) + \log n$

$f(2n) = f(n) + \log n$

$f(2n+1) = f(n) + \log n$

$f(2) = \log n$

$f(3) = \log n$

$f(4) = \log n + \log n$

$f(5) = \log n + \log n$

$f(6) = \log n + \log n$

$f(7) = \log n + \log n$

$f(8) = 2 \log n + \log n$

$\vdots$

$f(2^i) = f(2^{i-1}) + \log n = f(2 \cdot n)$

$f(2^i + 1) = i \log n = f(2n+1)$

$f(2n) = \log 2n \cdot \log n$

$f(2n+1) = \log 2n \cdot \log n$

$= (\log 2 + \log n) \cdot \log n$

$= \log n + \log^2 n$

$2^i = 2n$

$i = \log 2n$

$2^i + 1 = 2n + 1$

$2^i = 2n$

$i = \log 2n$

Vou provar que $f(2n) = \log n + \log^2 n$ e $f(2n+1) = \log 2n \cdot \log n$

Prova por indução em n:

base: $n = 1$

$f(1) = 0 = \log 1 + \log^2 1 = 0$

Passo: $n > 1$

$f(2n) = f(n) + \log n$ s/a $2n = x$ $f(x) = \log x/2 + \log^2 x/2$

$f(x) = f(x/2) + \log x/2$

Hipótese: $f(x/2) = \log x/4 + \log^2 x/4$

$f(x) = f(x/2) + \log x/2$

$\overset{H.I.}{\leq} \log x/4 + \log^2 x/4 + \log x/2$

$= \log x - 2 + (\log^2 x - 4 \log x + 4) + \log x/2$

$= \log x - 1 - 1 + \log^2 x - 4 \log x + 4 + \log x/2$

$\log x/2 + \log^2 x - 2\log x + 1 + 2 - 2\log x + \log x/2$

$= \log x/2 + \log^2 x - 2\log x + 1 + 2 - 2\log x + \log x/2$

$= \log x/2 + \log^2 x/2 + 2 - 2\log x + \log x/2$

$= \log x - 1 + \log^2 x/2 + 2 - 2\log x + \log x/2$

$= -\log x + 1 + \log^2 x/2 + \log x/2$

$= \log^2 x/2 + \log x/2 - \log x/2$

$\leq \log^2 x/2 + \log x/2 = \log^2 n + \log n$

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

$2n+1 = x$

$n = \frac{x-1}{2}$

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

hipótese: $f\left(\frac{x-1}{2}\right) = \log\left(\frac{x-3}{4}\right) + \log^2\left(\frac{x-3}{4}\right)$

$f(x) \overset{H.I.}{=} f\left(\frac{x-1}{2}\right) + \log\left(\frac{x-1}{2}\right)$

$\leq \log\left(\frac{x-3}{4}\right) + \log^2\left(\frac{x-3}{4}\right) + \log\left(\frac{x-1}{2}\right)$

$= \log\left(\frac{x-3}{4}\right) + (\log^2 x - 3 - 4 \cdot \log x - 3 + 4) + \log\left(\frac{x-1}{2}\right)$

$= \log(x-3/4) - \log(x-3) + 2 + \log^2(x-3) - 3\log(x-3) + 2 + \log\left(\frac{x-1}{2}\right)$

$= \log\left(\frac{x-3}{4}\right) - \log\left(\frac{x-3}{4}\right) + \log^2(x-3) - 2\log(x-3) + 2 + 1 + \log\left(\frac{x-1}{2}\right) - \log(x-3)$

$= \log^2\left(\frac{x-3}{2}\right) - \log(x-3) + 1 + \log\left(\frac{x-1}{2}\right)$

$= \log^2\left(\frac{x-3}{2}\right) - \log\left(\frac{x-3}{2}\right) + \log\left(\frac{x-1}{2}\right)$

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

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

$f(n) = O(\log^2 n)$

letra a

Questão 2 -

a) Um vetor ordenado
Inserção : $O(n)$
Remoção : $O(n)$
Acha : $O(1)$

b) Um vetor não ordenado
Inserção : $O(1)$
Remoção : $O(n)$
Acha : $O(n)$

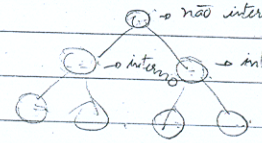
c) Heap
Inserção : $O(\lg n)$
Remoção : $O(\lg n)$
Acha : $O(n \lg n)$

a) Uma árvore balanceada
Inserção : $O(\lg n)$
Remoção : $O(\lg n)$
Acha : $O(\lg n)$

d) Uma tabela hashing
Inserção : $O(n)$ Remoção : $O(n)$
Acha : $O(1)$

O melhor seria utilizar uma árvore balanceada.

Questão 3 - Qual é o maior número possível de nós internos em uma árvore rubro-negra de altura k?



$n \leq 2^{k+1} - 2$, $2^{0+1} - 2 = 0$

já que ela começa em 0

Questão 4 - Não. Seja $f(n) = O(n^2)$ e $g(n) = \Omega(n)$

Vou provar que $f(n) \neq \Omega(n)$

$f(n) = n^2$ e $g(n) = n$

$n^2 \geq 1 \cdot n$ para $n \geq 1$

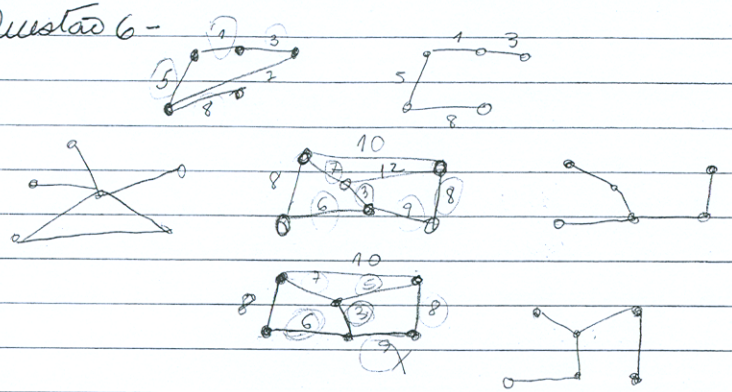
$n^2 = n \cdot n$

$\geq n$ //

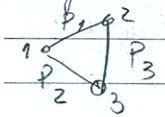
Assim é possível que o algoritmo do amigo tenha o mesmo consumo do seu.

Questão 5 - Errado, pois o conjunto de problemas P que possui solução polinomial também faz parte da classe NP.

Questão 6 -



Este algoritmo guloso não garante encontrar uma árvore geradora de peso mínimo. Veja o seguinte exemplo



$P_1 < P_2 < P_3$

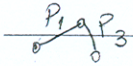
Escolha os vértices na ordem crescente (1,2,3)

1 → aresta P_1 3 → já foi cobrado

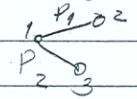
2 → aresta P_3, pois P_1 já foi escolhida

árvore obtida

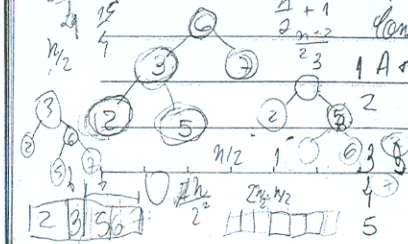
P_1, P_3



árvore mínima



Questão 4 - Construção de uma árvore de busca binária



Constrói (n)

1 A ← ordena n chaves por radixsort

2 raiz ← A[⌈n/2⌉]

3 completa(raiz, 1, ⌈n/2⌉ - 1)

4 completa(raiz, ⌊n/2⌋ + 1, n)

completa (raiz, p, r)

se p <= r

q ← ⌊(r+p)/2⌋

se raiz > A[q]

então esq[raiz] ← A[q]

completa(esq[raiz], p, q-1)

completa(esq[raiz], q+1, r)

senão dir[raiz] ← A[q]

completa(dir[raiz], p, q-1)

completa(dir[raiz], q+1, r)

Consumo

Linha	$T(n)$
1	$O(n)$
2	$O(1)$
3	$T(n/2)$
4	$T(n/2)$

$T(n) = 1$ para $n = 1$

$T(n) = 2T(n/2) + 1$

$= 2^2 T(n/4) + 2 + 1$

$= 2^3 T(n/2^3) + 2^2 + 2 + 1$

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

$= 2^{\lg n} \cdot 1 + 2^{\lg n} - 1 \,//\, 1$

$= n + n - 1$

$= 2n - 1$ // $T(n) = O(n)$

Questão 8 - $A[i] = \lg\left(\frac{i}{i+1}\right) = \lg i - \lg(i+1)$

O Heap máximo obedece a seguinte regra $A[i] \leq A[\lfloor i/2 \rfloor]$

Vou provar que para $A[i] = \lg\left(\frac{i}{i+1}\right)$ vale que $A[i] \leq A[\lfloor i/2 \rfloor]$

$\lg\left(\frac{i}{i+1}\right) \leq \lg\left(\frac{\lfloor i/2 \rfloor}{\lfloor i/2 \rfloor + 1}\right)$

$\lg\left(\frac{i}{i+1}\right) = \lg i - \lg(i+1)$

$\geq \lg i - \log(i+2)$

$= \lg i - \lg 2 - \log(i+2) + \log 2$

$= \log i/2 - \log\left(\frac{i+2}{2}\right)$

$= \log i/2 - \log_2(i/2 + 1)$

-3 -5

Etapa mínimo vim a propriedade $A[i] \geq A[\lfloor i/2 \rfloor]$

Era provar para $A[i] = \lg(\frac{i}{i+1})$ que $A[i] \geq A[\lfloor i/2 \rfloor]$

$\lg(\frac{i}{i+1}) \geq \lg(\frac{i/2}{i/2+1}) \geq \lg(\frac{\lfloor i/2 \rfloor}{\lfloor i/2 \rfloor + 1})$

$\lg(\frac{i}{i+1}) \geq \lg i - \lg(i+1)$

$= \lg i - \lg 2 - \lg(i+1) + \lg 2$

$\geq \lg i/2 - \lg(i+1) + \lg 2$

$\geq \lg i/2 - \lg(i+2) + \lg 2$

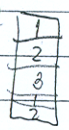
$= \lg i/2 - \lg(\frac{i+2}{2})$

$= \lg i/2 - \lg(i/2 + 1)$

$\geq \lg\lfloor i/2 \rfloor - \lg(\lfloor i/2 \rfloor + 1)$

Questão 9-

1-
```
Segmento(n)
1  se n = 1
2      C[n] ← 1
3      devolva 1
4  senão max ← Segmento(n-1)
5      se A[n] ≥ A[n-1]
6      então c[n] ← c[n-1] + 1
7      senão c[n] ← 1
8      se max < c[n]
9      então max ← c[n]
10     devolva max
```



2- $T(1) = 1 \quad n = 1$

$T(n) = T(n-1) + 5 \quad n \geq 2$

$n-1 = 1$

$i = n-1$

3- $T(n) = T(n-1) + 5$

$= T(n-2) + 5 + 5$

$= T(n-3) + 5 + 5 + 5$

$= T(n-i) + i.5$

$= T(i) + (n-1).5 = 1 + 5n - 5 = 5n - 4$ //

prova por indução em n:

Base: $n = 1$

$T(1) = 1 = 5.1 - 4 = 1$ //

passo: $n \geq 2$

Hipótese: $T(n-1) = 5.(n-1) + 4$

$T(n) \overset{H.I}{=} T(n-1) + 5$

$\leq 5(n-1) + 4 + 5$

$= 5n - 5 + 4 + 5$

$= 5n + 4$

$\leq$

Como $5n + 4 \leq 10.n$ para $n \geq 1$ então podemos provar que $T(n) = O(n)$ =

Questão 10- Algoritmo guloso

Perguntar sobre a prova da corretude!

m pares de esquis

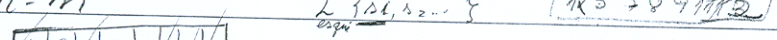
s_i - altura do i-ésimo par

n - esquiadores

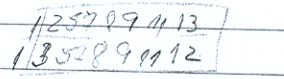
h_i - altura do i-ésimo esquiador

$n = m$

$L \{s_1, s_2 \ldots\}$ esqui



esquiadores



```
Esqui(H, S, n)
H ← ordena em ordem decrescente
S ← ordena em ordem crescente
Para i ← 1 até n
    Soma = |H[i] - S[i]| + Soma
devolva Soma
```

consumo

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

$= O(n \lg n)$ //

Escolha gulosa

Teorema: Seja $s_i = \min\{s_i \ldots s_n\}$ e $h_i = \min\{h_i \ldots h_n\}$ então a combinação $\langle s_i, h_i \rangle$ pertence a alguma alocação ótima.

(14)

Prova:

Seja $s_i = \min\{s_i, \ldots, s_n\}$ e $h_i = \max\{h_i, \ldots, h_n\}$

Seja A uma alocação ótima de S e H

Suponha que A não contém a combinação $\langle s_i, h_i \rangle$

A contém $\langle s_i, h_{i+1} \rangle$

Muito difícil! Não consegui, mas o Alexandre Freire tem uma solução para ela.

PROVA 2000/02

Questão 1 -

$T(n) \leq T(n-1) + O(n)$ significa dizer que $T(n) \leq T(n-1) + f(n)$ onde $f(n) \leq 1.n$ para $n \geq 1$

Dessa forma podemos escrever $T(n)$ da seguinte forma:

$T(n) \leq T(n-1) + n$ para $n \geq 1$

Vamos supor a recorrência

$T(n) \leq a$ para $n = 1$

$T(n) \leq T(n-1) + n$ para $n > 1$

Desenvolvendo a recorrência temos que:

$T(n) \leq T(n-1) + n$

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

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

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

$= a + (n-1).n - \frac{1}{2}.(n-1).(n-1-1)$

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

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

$(n-1).(n-2)$

$n^2 - 2n - n + 2$

$n - i = 1$

$n - 1 = i$

Vou provar que $T(n) \leq a + \frac{1}{2}n^2 + \frac{n}{2} - 1$

base $n = 1$

$T(1) \leq a \leq a + \frac{1}{2} + \frac{1}{2} - 1 = a$

passo $n > 1$

Hipótese: $T(n-1) \leq a + \frac{1}{2}(n-1)^2 + \frac{1}{2}(n-1) - 1$

$T(n) \leq T(n-1) + n$

$\leq a + \frac{1}{2}(n-1)^2 + \frac{1}{2}(n-1) - 1 + n$

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

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

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

$T(n) \leq a + \frac{1}{2}n^2 + \frac{n}{2} - 1 \leq 1.n^2$ para $a = +1$ e $n \geq 1$

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

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