(14)

Prova:

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

Seja A uma alocação ótima de S e H

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

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

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 $n > 1$

Desenrolando 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 $n \geq 1$

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

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

Questão 2 -

Heap		Árvore	
Construção	$O(n)$	Construção	$O(\lg n)$
Inserção	$O(\lg n)$	Inserção	$O(\lg n)$
Remoção	$O(\lg n)$	Remoção	$O(\lg n)$
Busca	$O(\lg n)$	Busca	$O(\lg n)$

Situação 1.
Consultar maior valor.
Para um heap-máximo custa $O(1)$
Para uma ABB custa $O(\lg n)$

Situação 2
Listar todos os elementos em ordem crescente
Para um heap - custa $O(n \lg n)$
Para uma ABB - custa $O(n)$

Questão 3 - 1 se i é uma potência de 2
1 em caso contrário.

Método Agregado

$$\sum_{i=1}^{n} c_i = \sum_{k=0}^{\lfloor \lg n \rfloor} 2^k + n - \lfloor \lg n \rfloor - 1$$
$$= 2^{\lfloor \lg n \rfloor + 1} - 1 + n - \lfloor \lg n \rfloor - 1$$
$$\leq 2^{\lg n + 1} - 2 + n - \lg n = 2^1 \cdot n - 2 + n - \lg n$$
$$= 3n - (\lg n + 2)$$
$$\leq 3n$$

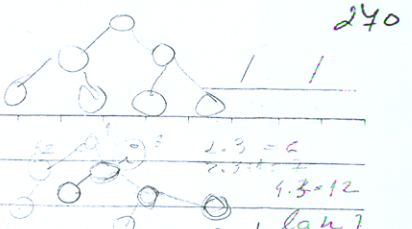
$T(n) \leq 3n$

Custo amortizado por operação $= \frac{3n}{n} = 3 = O(1)$.

1	2	3	4	5	6	7	8	9
2^0	2^1	1	2^2	1	1	1	2^3	1

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

2 - Pg 88



Questão 4 - Sim.
Vou provar que a altura de um nó i é $\lfloor \lg \frac{n}{i} \rfloor$
Suponha que h_i é a altura do nó i
h_i vai ser o comprimento da sequência $\langle 2.i, 4.i, 8.i \ldots 2^{h_i}.i \rangle$
onde $2^{h_i} \cdot i \leq n \leq 2^{h_i+1} \cdot i + 2^{h_i} - 1$ $\leftarrow$ n° de nós
$2^{h_i} \cdot i \leq n \leq 2^{h_i}(i+1) - 1$
$2^{h_i} \cdot i \leq n < 2^{h_i}(i+1)$
$\lg 2^{h_i} \cdot i \leq \lg n < \lg 2^{h_i}(i+1)$
$h_i + \lg i \leq \lg n < h_i + \lg(i+1)$ $\quad \lg(i+1) \leq \lg i + 1$
$h_i + \lg i \leq \lg n < h_i + \lg i + 1$
$h_i \leq \lg n - \lg i < h_i + 1$
$h_i = \lfloor \frac{\lg n}{i} \rfloor$

Questão 5 - Não sei

Questão 6 - Seja X a variável aleatória que representa o n-ésimo execução das linhas 2 e 5.

$X_i = \begin{cases} 1 & \text{se as linhas 2 e 5 executam pelo menos } i \text{ vezes} \\ 0 & \text{se caso contrário} \end{cases}$

$X_i = Pr\{X_i = 1\} = 1/i$

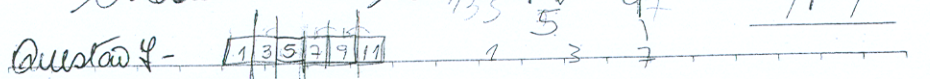
Seja $E[X]$ o número esperado de vezes as linhas 2 e 5 executam

então $E[X] = \sum_{i=1}^{n} E[X_i]$
$= E[\sum_{i=1}^{n} X_i]$
$= \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n}$
$\leq \lg n + 1$

Considerando $E[X] = T(n)$. Temos que $T(n) = O(\lg n)$.

Errado é só usar Max-Heap-Delete mas para MinHeap.

Questão 7 -




```
BUILDHEAP1(A, i)
1  x ← A[i]
2  H ← [1..n] ← 0        Eu posso apenas trocar os elementos em A
3  BUILDMAX(A, H, 1, n, 1)
4  REMOVEDEHEAP(H, x)
```

```
BUILDMAX(A, H, p, r, j)
1 se p <= r
2    então H[j] ← A[r]
3          q ← ⌊r-1+p/2⌋
4          BUILDMAX(A, H, p, q, 2j)
5          BUILDMAX(A, H, q+1, r-1, 2j+1)
```

Questão 7 errada! Veja solução do Jesús.

Consumo

BuildHeap1

Linha

1, O(1)
3 O(lg n)
4 O(lg n)

BuildMax

Linhas

1, 2, 3 O(1)
4 T(⌈n/2⌉)
5 T(⌊n/2⌋)

$T(n) = 0 \quad n = 1$

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

$= \log n$

O algoritmo BuildHeap1 cria um Max-heap A a partir de um vetor ordenado crescentemente e depois remove o elemento A[i].

A rotina Remove de heap é uma operação básica de Heap que remove o elemento e restabelece o heap tendo um consumo de O(lg n)

26/10

Corretude: Invariante H[1..j] é um max-heap.

Início - Na 1ª chamada H[1] recebe o maior elemento de A[1..n] sendo trivialmente um max-heap com 1 elemento.

Manutenção: Vou provar que o invariante se mantém na recursão.

H[j] que é próximo índice vazio recebe o maior valor do vetor A[p..r] que é o último já que A está crescentemente ordenado.

As linhas 4 e 5 subdividem o vetor em A[p..q] e A[q+1, r-1] e muda o próximo índice a ser preenchido 2j ou 2j+1 que restabelece o invariante.

Término: A última chamada da recursão terá j = n e H[j] receberá elemento de menor valor do vetor A[⌈n/2⌉+1, n-1] caracterizando assim que todos os elementos de H segue a propriedade $H[i] \leq H[\lfloor i/2 \rfloor]$

Portanto, um max-heap.

Prova 2000/01

Questão 1 - São árvores de busca binária que tem um bit extra que armazena a cor: que pode ser vermelho ou preto. As folhas NIL tem cor preta e a raiz também. Se a raiz for rubra então não é uma árvore rubro-negra.