Prova 1 - Gabarito

Diane Castonguay
Instituto de Informática
Universidade Federal de Goiás
Projeto e Analise de Algoritmos

September 12, 2005

Questão 1 - Notação O

$$\begin{aligned} f(n) &= 5n^3 + n^2 + 7n + 3 \\ &\leq 5n^3 + n^3 + 7n^3 + 3n^3 \quad \forall n \geq 1 \\ &\leq 16n^3 \end{aligned}$$

Na definição, usamos $c = 16$ et $n_0 = 1$.

Questão 2

(a) Equação de reorrência

$$T(n) = 7\ T(d) + f(n) \text{ com } f(n) = D(n) + C(n)$$

onde $D(n)$ é o tempo de execução da linha 2, quer dizer do algoritmo DIVIDIR e da assinação (constante) e $C(n)$ é o tempo de execução da linha 10, quer dizer do algortimo COMBINAR.

(b) Teorema Mestre

Por hipotesis, $D(n) \in \mathcal{O}(n)$ e $C(n) \in \mathcal{O}(\lg n)$. Como $\lg n \in \mathcal{O}(n)$, temos que $C(n) \in \mathcal{O}(n)$ e por tanto, $f(n) \in \mathcal{O}(n)$.

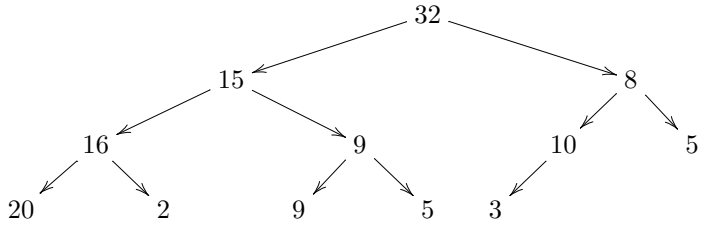
A relação de recorrência obtida em (a) se reescreve da forma:

$$T(n) = 7T(n/3) + \mathcal{O}(n)$$

Estamos no caso 1 do Teorema Mestre, pois $\varepsilon = \log_3 7 - 1 > 0$ e $f(n) \in \mathcal{O}(n) = \mathcal{O}(n^{\log_3 7 - \varepsilon})$. Por tanto, $T(n) \in \mathcal{O}(n^{\log_3 7})$.

Questão 3

(a) Heap máximo



A propriedade do heap máximo não está satisfeita.
Os nós que não satisfazem a propriedade são :
2 porque $A[2] = 15 < 16 = A[4] = A[LEFT(2)]$
3 porque $A[3] = 8 < 10 = A[6] = A[LEFT(3)]$
4 porque $A[4] = 16 < 20 = A[8] = A[LEFT(4)]$

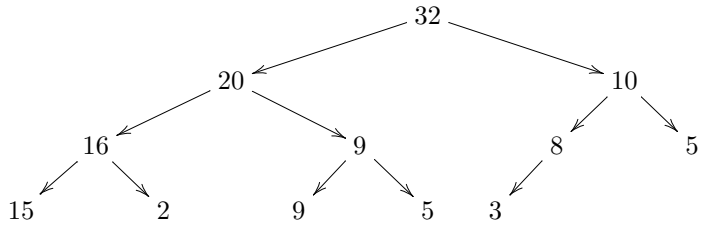
(b) MAX-HEAPIFY e BUILD-MAX-HEAP

O mínimo de chamadas é 3 es decir:
MAX-HEAPIFY(A, 4), MAX-HEAPIFY(A, 3) e MAX-HEAPIFY(A, 2)

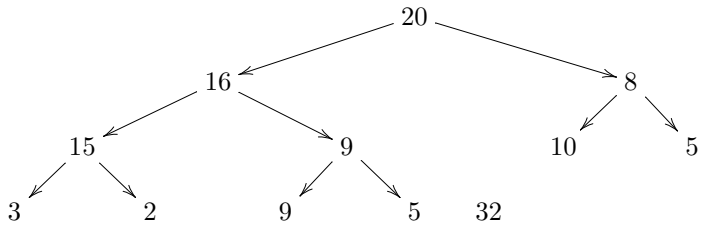
BUILD-MAX-HEAP produz o mesmo heap. De fato, as chamadas a MAX-HEAPIFY no BUILD-MAX-HEAP sÃ£o:
MAX-HEAPIFY(A, 6) e MAX-HEAPIFY(A, 5)
(não fazem nada pois o heap satisfaz a propriedade nos nós 5 e 6)
MAX-HEAPIFY(A, 4), MAX-HEAPIFY(A, 3) e MAX-HEAPIFY(A, 2)
(essas sÃ£o as mesmas chamadas que fizemos na primeira parte)
MAX-HEAPIFY(A, 1)
(não faz nada pois o heap satisfaz a propriedade no nó 1).

(c) HEAPSORT(A)

Antes da primeira iteração do loop para:



Depois da primeira iteração do loop para:



Questão 4

(a) Loop Invariante

$j - i = r - k + 1$ (isto implica que $j > i$ quando $k \leq r$)

$B[p..i-1] \leq chave$

$B[i..j]$ não tem valores assinadas

$B[j+1..r] > chave$

(b) Prova

BASE $k = p + 1, i = pej = r$

$j - i = r - p = r - (p+1) + 1 = r - k + 1$
$B[p..i-1] = B[p..p-1]$ é o vetor vazio.
$B[i..j] = B[p..r]$ não tem valores assinadas
$B[j+1..r] = B[r+1..r]$ é o vetor vazio.

HIP Suponhamos que no ínicio da k-esima iteração temos que:

$j - i = r - k + 1$
$B[p..i-1] \leq chave$
$B[i..j]$ não tem valores assinadas
$B[j+1..r] > chave$

PASSO Vamos denotar por i' e j' os valores respectivos de i e j no final da iteração k. Temos que provar que

$j' - i' = r - (k+1) + 1 = r - k$
$B[p..i'-1] \leq chave$
$B[i'..j']$ não tem valores assinadas
$B[j'+1..r] > chave$

Vamos dividir em dois casos: $A[k] \leq chave$ e $A[k] > chave$.

$A[k] \leq chave$: Neste caso, temos que $i' = i + 1$, $j' = j$ e $B[i] = A[k]$.

$j' - i' = j - (i+1) = j - i - 1 = *r - k + 1 - 1 = r - k$ ($= *$ hipotesis de indução).

Por hipotesis de indução, $B[p..i-1] \leq chave$. Por outro lado, $B[i] = A[k] \leq chave$. Por tanto, $B[p..i'-1] = B[p..i] \leq chave$.

Segue da hipotesis de indução que $B[i'..j'] = B[i+1..j]$ não tem valores assinadas.

$B[j'+1..r] = B[j..r] > chave$ por hipotesis de indução.

$A[k] > chave$: Neste caso, temos que $i' = i$, $j' = j - 1$ e $B[j] = A[k]$.

$j' - i' = j - 1 - i = *r - k + 1 - 1 = r - k$ ($= *$ hipotesis de indução).

$B[p..i'-1] = B[p..i-1] \leq chave$ por hipotesis de indução.

Segue da hipotesis de indução que $B[i'..j'] = B[i..j-1]$ não tem valores assinadas.

Por hipotesis de indução, $B[j+1..r] > chave$. Por outro lado, $B[j] = A[k] > chave$. Por tanto, $B[j'+1..r] = B[j..r] > chave$.

(c) Correteza

Pela propriedade do loop invariante, quando sai do loop para temos que ($k = r + 1$):

$j - i = r - (r+1) + 1 = 0$, quer dizer que $i = j$.

$B[p..i-1] \leq chave$

$B[i]$ não tem valor assinada

$B[i+1..r] > chave$

Depois da linha 12, temos que $B[i] = chave$. As linhas 13 a 15 identifica o vetor B com o vetor A. Por tanto, no final do algoritmo temos o desejado i.e:

$A[p..i-1] \leq chave$

$A[i] = chave$

$A[i+1..r] > chave$

(d) Eficiência

SEPARA é $\mathcal{O}(n)$ onde $n = r - p + 1$. O algoritmo PARTITION é da mesma ordem de complexidade porém ele não usa vetor auxiliar. Por tanto, sugerimos o uso de PARTITION.

Questão 5

(a) Tempo por linha

1. $n+1$

2. n

3. $0 \leq \sum_{i=1}^{n} t_i \leq n$ onde ($t_i = 1$ se $A[i] \geq 0$) e ($t_i = 0$ senão).

4. $0 \leq \sum_{i=1}^{n} t_i * A[i] \leq nk$

5. $0 \leq \sum_{i=1}^{n} t_i * (A[i] - 1) \leq n(k-1)$

6. 0

7. $0 \leq \sum_{i=1}^{n} t_i \leq n$

8. 0

9. 0

Obs. Na linha 4 e 5, se $A[i] = 0$, então a quantidade de vezes é a mesma que quando $A[i] = 1$.

(b) Tempo total

MELHOR CASO: Quando o vetor é somente formado de entrada negativa, i.e $A[i] < 0 \forall i$.

Neste caso, $t_i = 0$ para todo i. O algoritmo tem um tempo de execução $\mathcal{O}(n)$.

PIOR CASO: Quando o vetor possui somente entrada igual a k. i.e $A[i] = k \forall i$.

Neste caso, $t_i = 1$ para todo i e o algoritmo tem um tempo de execução $\mathcal{O}(nk)$. Como $k \in \mathcal{O}(n)$, temos que o algoritmo tem um tempo de execução $\mathcal{O}(n^2)$.