Instituto de Informática UFG

Projeto e Analise de Algoritmos
Prova 2 – 17/04/2006

PROFESSORA: Diane Castonguay

JUSTIFIQUE TODAS AS SUAS RESPOSTAS

Questão 1 **[3pt]**
Considere o algoritmo ALGO de dividir e conquistar em anexo.

(a) Exiba a equação de recorrências do tempo de execução de ALGO.

(b) Usando o teorema mestre, resolva a recorrência obtida em (a) sabendo que:
- d é sempre igual a n/8
- o tempo de execução de DIVIDIR é n lg n
- o tempo de execução de COMBINAR é n+d

Questão 2 **[1pt]**
Considere o vetor A dado por < 23, 1, 16, 33, 8, 23, 37, 35, 8, 13 >.
Ilustre o vetor em forma de heap antes e depois da chamada de BUILD-MAX-HEAP(A).
Explicita cada chamada de MAX-HEAPIFY no decorrer de BUILD-MAX-HEAP(A) e para cada, ilustra a parte do heap onde teve modificações.

Questão 3 **[3pt]**
Considere o algoritmo NOTCOUNTINGSORT em anexo.

(a) Qual é o loop invariante do segundo loop para do algoritmo NOTCOUNTINGSORT?

(b) Qual as diferências e as similitudes entre o algoritmo NOTCOUNTINGSORT e COUNTINGSORT. O algoritmo NOTCOUNTINGSORT é estável? Explique.

(c) Escreve o algoritmo EXISTENCIA(A, ?,v) de tempo $\Theta(1)$ que utilisa um dos vetores auxiliares do NOTCOUNTINGSORT. A saída do algoritmo é verdadeiro se o valor aparece em A e falso senão. **Obs.:** O valor v é um inteiro de 1 a k.

Questão 4 **[3pt]**
Considere o algoritmo PESQUISAORDENADA em anexo.
(a) Prove que o seguinte loop invariante do enquanto do algoritmo PESQUISAORDENADA é valido.

Se o valor v aparece no vetor A na posição i (i.e A[i]=v) então $p \leq i \leq r$.

Obs.: Sabemos que se $p \leqslant r$ então $p \leq \lfloor (p + r)/2 \rfloor \leq r$.

(b) Use o item (a) para concluir sobre a correteza do algoritmo PESQUISAORDENADA, es decir, retorna um índuce i tal que v = A[i] ou o valor especial NIL, se v não aparecer em A.

ALGORITMOS

Questão 1

```
ALGO(A, n)
1. se n > 1
2.       então   d ← DIVIDIR(A, n)
3.               para i ← 1 até 4 faça
4.                       ALGO(A, d)
5.               fim-para
6.               retorna COMBINAR(A, n, d)
7.       senão retorna NIL
8. fim-se
```

Questão 2

```
MAX-HEAPIFY(A, i)
1.       l ← LEFT(i)
2.       r ← RIGHT(i)
3.       se l ≤ tamanho-do-heap[A] e A[l] > A[i]
4.               então maior ← l
5.               senão maior ← i
6.       fim-se
7.       se r ≤ tamanho-do-heap[A] e A[r] > A[maior]
8.               então maior ← r
9.       fim-se
10.      se maior ≠ i
11.              então A[i] ↔ A[maior]
12.                      MAX-HEAPIFY(A, maior)
13.      fim-se
```

```
BUILD-MAX-HEAP(A)
1. tamanho-do-heap[A]  ← comprimento[A]
2. para i ← ⌊comprimento[A]/2⌋ até 1 passo -1 faça
3.               MAX-HEAPIFY(A, i)
4. fim-para
```

Questão 3

```
/* Os valores de A são inteiros de 1 a k. B é o vetor ordenado.*/
NOTCOUNTINGSORT(A, k)
1. para i ← 1 até k faça
2.        C[i] ← 0
3. fim-para

4. para j ← 1 até comprimento[A] faça
5.        C[A[j]] ← C[A[j]] + 1
6. fim-para

7. B[1] ← C[1]
8. para i ← 2 até k faça
9.        B[i] ← B[i – 1] + C[i]
10. fim-para

11. para i ← 1 até k faça
12.       B[i] ← B[i] – C[i] + 1
13 fim-para

14.para j ← 1 até comprimento[A] faça
15.               D[B[A[j]]] ← A[j]
16.               B[A[j]] ← B[A[j]] + 1
17.fim-para

18.para i ← 1 até comprimento[A] faça
19.               A[i] ← D[i]
20.fim-para
```

Questão 4

```
/* A é um vetor ordenado */
PESQUISAORDENADA(A, v)
1. p ← 1
2. r ← comprimento[A]
3. medio ← ⌊(p + r)/2⌋
4. enquanto p ≤ r e A[medio] ≠  v faça
5.        se A[medio] > v
6.                 então   r ← medio – 1
7.                 senão   p ← medio + 1
8.        fim-se
9.        medio ← ⌊(p + r)/2⌋
10. fim-equanto
11. se  p ≤ r
12.       então   retorna medio
13.       senão   retorna NIL
14. fim-se
```