MAC 5711 – Análise de Algoritmos

PRIMEIRO SEMESTRE DE 2004

Primeira Prova – 13 de abril

Nome: _____ **Assinatura**: _____

1. [2,5 pontos] Considere a seguinte recorrência:

$$T(n) = \begin{cases} 1 & \text{se } n = 1 \\ 3T(\lfloor \frac{n}{2} \rfloor) + n & \text{se } n > 1. \end{cases}$$

   (a) Resolva-a exatamente para n potência de 2. Não utilize o Teorema Mestre. Utilize a maneira ensinada em aula: deduza (por expansão ou algo semelhante) um "chute" de solução e prove (por indução), para n potência de 2, que o chute deduzido está correto.

   (b) Usando o que concluiu no item (a), deduza o comportamento assintótico de $T(n)$ (ou seja, conclua que $T(n) = \Theta(f(n))$ para $f(n)$ tão simples quanto possível). Não precisa *provar* que $T(n) = \Theta(f(n))$. Apenas deixe claro quem é $f(n)$.

2. [3,0 pontos] Considere uma seqüência de vetores de inteiros

$$A_k[1..2^k], A_{k-1}[1..2^{k-1}], \ldots, A_1[1..2^1], A_0[1..2^0].$$

   Suponha que os inteiros armazenados nos vetores são dois a dois distintos e que cada vetor está em ordem crescente.

   O seguinte problema foi proposto a uma classe de alunos: escreva um algoritmo que reuna, por meio de sucessivas operações de intercalação como as efetuadas pela rotina *intercala* do *mergesort*, o conteúdo dos vetores $A_0, \ldots, A_k$ em um único vetor crescente $B[1..n]$, onde $n = 2^{k+1} - 1$.

   Entre várias, duas soluções para o problema, esquematizadas abaixo, apareceram:

```
1  junta1(A_0, ..., A_k) {
2    B[1..2^k] ← A_k[1..2^k]
3    m ← 2^k
4    para i de k − 1 a 0 faça {
5      B[1..m + 2^i] ← intercala(B[1..m], A_i[1..2^i])
6      m ← m + 2^i;
7    }
8  }
```

```
1  junta2(A_0, ..., A_k) {
2    B[1..1] ← A_0[1..1]
3    m ← 1
4    para i de 1 a k faça {
5      B[1..m + 2^i] ← intercala(B[1..m], A_i[1..2^i])
6      m ← m + 2^i;
7    }
8  }
```

   Em termos de complexidade de tempo, as duas soluções são ou não são equivalentes? Se não são, qual das duas é melhor? Justifique a sua resposta deduzindo o comportamento assintótico exato (ou seja, com a notação Θ) da complexidade de tempo de cada uma, e comparando estes comportamentos. *Dica:* calcule antes de mais nada o valor de m no início de cada iteração do *para* da linha 4.

3. [1,5 pontos] Mostre como multiplicar os números complexos $a+bi$ e $c+di$ usando apenas três multiplicações reais. Especificamente, escreva uma função que recebe os números reais a, b, c e d e devolve os dois reais $ac-bd$ e $ad+bc$ (a parte real e o coeficiente da parte imaginária do produto de $a+bi$ e $c+di$). Sua função deve fazer apenas três multiplicações reais.

4. [3,0 pontos] Um aluno, inspirado no algoritmo *heapsort*, pensou na seguinte idéia para ordenar um vetor $v[i..j]$:

   - determinar um elemento máximo em $v[i..j]$ e rearranjar o vetor $v[i..j]$ colocando tal máximo na posição j;
   - determinar um elemento mínimo em $v[i..j-1]$ e rearranjar o vetor $v[i..j-1]$ colocando tal mínimo na posição i;
   - repetir para o trecho $v[i+1..j-1]$ enquanto houver pelo menos dois elementos nesse trecho.

   Usando dois heaps, tal idéia pode ser implementada de forma que o algoritmo resultante consuma $O(n \log n)$, onde $n = j - i + 1$ é o número de elementos em $v[i..j]$. Mostre como. Caso você opte por armazenar informação extra nos heaps, deixe claro qual seria a chave de cada heap, como estes seriam inicializados, se e/ou como as rotinas de manipulação de heap seriam adaptadas e como ficaria o algoritmo resultante. Escreva pseudo-código das funções envolvidas que tenham sofrido alteração, bem como do algoritmo resultante.