Exame Preliminar para o Doutorado

IME–USP, Agosto, 2001

Prova de Análise de Algoritmos

Instruções:

(i) O candidato pode resolver todas as questões.

(ii) A banca considerará questões cujos valores somem até 10 pontos de modo que a nota seja máxima.

(iii) Mencione os teoremas e propriedades usados para justificar suas afirmações.

Questão 1 [1 ponto]

O Professor Progresso encontrou um algoritmo de ordenação baseado em comparações que gasta tempo $O(n \lg \sqrt{n})$. Considerando-se o limitante inferior para o tempo de ordenação, isso é possível? Justifique.

Questão 2 [1 ponto]

Considere árvores binárias em que cada nó tem um campo `chave` para a chave, dois campos `esq` e `dir` que apontam para os filhos esquerdo e direito, respectivamente, e um campo `pai` que deve apontar para o pai.

(a) Descreva um algoritmo que, recebendo um apontador para a raiz de uma tal árvore (em que o campo `pai` não foi preenchido), preenche o campo `pai` de todos os nós corretamente.

(b) Qual é a complexidade do seu algoritmo? Ele é ótimo? Justifique.

Questão 3 [1 ponto]

Descreva um algoritmo que, dado um inteiro positivo n, devolve n^n fazendo no máximo $2 \lg n$ multiplicações.

Questão 4 [2 pontos]

O mergesort gasta tempo $\Theta(n \log n)$ para ordenar n valores, enquanto que, se os valores forem inteiros positivos menores ou iguais a k o radix sort gasta tempo $\Theta(n \log k / \log n)$.

(a) Se $k = \Theta(2^n)$, qual é a complexidade de tempo do radix sort (expressa como uma função que depende apenas de n). Neste caso, qual dos dois algoritmos é mais eficiente?

(b) Se $k = \Theta(2^{\log^3 n})$, qual é a complexidade de tempo do radix sort (expressa como uma função que depende apenas de n). Neste caso, qual dos dois algoritmos é mais eficiente?

(c) Se $k = \Theta(n^5)$, qual é a complexidade de tempo do radix sort (expressa como uma função que depende apenas de n). Neste caso, qual dos dois algoritmos é mais eficiente?

Questão 5 [2 pontos]
Sabe-se que, para $n \geq 0$ e $0 \leq m \leq n$,

$$\binom{m}{n} = \begin{cases} \binom{m-1}{n} + \binom{m-1}{n-1} & \text{se } m \geq 2 \text{ e } 0 < n < m, \\ 1 & \text{se } n = 0 \text{ ou } m = n. \end{cases}$$

Considere as duas maneiras de implementar uma função que calcula, dados m e n, o valor de $\binom{m}{n}$.

```
int comb1 (int m, int n) {
  if (n == 0 || m == n)
    return 1;
  else
    return comb1(m-1,n) + comb1(m-1,n-1);
}

int comb2 (int m, int n) {
  int i, j, t[max,max];

  for (i = 0; i <= m; i++)
    t[i][0] = t[i][i] = 1;

  for (i = 2; i <= m; i++)
    for (j = 1; j < i; j++)
      t[i][j] = t[i-1][j] + t[i-1][j-1];
  return t[m,n];
}
```

Qual é a mais eficiente? Justifique a sua resposta com estimativas adequadas de complexidade.

Questão 6 [2 pontos]
Considere a sequência $T(n)$ tal que:

$T(1) = 1$

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

Mostre que $T(n)$ **não** é $O(n \lg n)$.

Questão 7 [3 pontos]
Para uma árvore de busca binária (ABB), e um ítem x armazenado nela, seja $N(x)$ o nó da árvore que contêm x. Dizemos que uma ABB A é *embutível* na ABB B se:

- Todo ítem armazenado em A também está em B, e
- Para todos pares x, y de ítens, se $N(x)$ é descendente de $N(y)$ em A, então $N(x)$ é descendente de $N(y)$ em B.

Dê um algoritmo que, dadas A e B, decide em tempo linear se A é embutível em B.

Questão 8 [3 pontos]

O código abaixo descreve a implementação de uma fila usando duas pilhas, `E` e `D`. As funções **`push`**, **`pop`**, **`empty`**, respectivamente, empilham, desempilham e testam se a pilha é vazia, e são consideradas operações elementares. A fila vazia é representada com ambas as pilhas vazias; o teste para fila vazia não está descrito.

```
void insere (int x) {
  push(D, x)
}

int remove {
  if (empty(E)) {
    if (empty(D)) {if (windows) { show blue screen of death } else { exit gracefully }}
    while (!empty(D))
      push(E, pop(D))
  }
  return pop(E)
}
```

Analise o custo amortizado para uma sequência de inserções e remoções de n elementos. Note que a sequência pode alternar operações à vontade, desde que não tente remover um elemento da fila vazia.