MAC 338 - Análise de Algoritmos

Departamento de Ciência da Computação

Primeiro semestre de 2001

Prova de Recuperação

1. (Valor: 2.5 pontos) Suponha dado um conjunto de livros numerados de 1 a n. Suponha que o livro i tem peso p_i e que $0 < p_i < 1$ para cada i.

   Problema: acondicionar os livros no menor número possível de envelopes de modo que cada envelope tenha no máximo dois livros e o peso do conteúdo de cada envelope seja no máximo 1.

   Descreva um algoritmo que resolva o problema em tempo $O(n \log n)$ no pior caso. Mostre que seu algoritmo está correto e justifique que ele de fato consome tempo $O(n \log n)$.

2. (Valor: 2.5 pontos) Diga qual é a complexidade de pior caso de cada um dos algoritmo abaixo. Explique quem são os parâmetros que aparecem na complexidade de cada algoritmo.

   (a) Busca em largura em um grafo;
   (b) Countingsort;
   (c) Heapsort;
   (d) KMP;
   (e) Mergesort;
   (f) Quicksort;
   (g) Strassen.

   Descreva sucintamente o que fazem/para que servem os algoritmos KMP, Countingsort e Strassen. Seja preciso.

3. (Valor: 2.5 pontos) O que significa um problema Q ser NP-completo? Quais as implicações? Escreva o que você sabe e considera importante sobre isso. Evite definições técnicas.

4. (Valor: 2.5 pontos) Um intervalo é um par ordenado de números. O primeiro número do par é o início do intervalo. O segundo número é o término do intervalo. O início de um intervalo i será denotado por s_i e o término será denotado por f_i. Suporemos sempre que $s_i < f_i$.

   Problema: dada uma coleção de intervalos, digamos S, determinar uma subcoleção disjunta máxima de S.

   Uma coleção de intervalos é disjunta se os intervalos da coleção são disjuntos dois a dois. Dois intervalos x e y são disjuntos se $f_x \leq s_y$ ou $f_y \leq s_x$.

   Descreva um algoritmo que resolva o problema em tempo $O(n \log n)$ no pior caso. Mostre que seu algoritmo está correto e justifique que ele de fato consome tempo $O(n \log n)$.

5. (Valor: 2.5 pontos) Sabe-se que, para $m > 1$ e $n > 1$,

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

   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 complexidade de cada uma das funções? Justifique a sua resposta. Qual é a mais eficiente? Por quê?