MAC 5711 – Análise de Algoritmos

SEGUNDO SEMESTRE DE 2005

Terceira Prova – 24 de novembro

Nome do aluno: ____ **Curso**:____________

Assinatura: ____

No. USP: ______________________ **Professor**:________________________________

Instruções

1. Não destaque as folhas deste caderno.
2. A prova pode ser feita a lápis.
3. A legibilidade também faz parte da nota!
4. A duração da prova é de **3 horas**.
5. A prova consta de 4 questões. Verifique antes de começar a prova se o seu caderno de questões está completo.
6. Não é permitido o uso de folhas avulsas para rascunho.
7. Não é necessário apagar rascunhos no caderno de questão mas especifique qual é a resposta e qual é o rascunho.
8. A prova é sem consulta.

Não escrever nesta parte da folha

Questão	Nota	Observação
1		
2		
3		
4		
Total		

Boa Sorte !

1. [3,0 pontos]
Considere o algoritmo LCS-LENGTH visto em aula:

```
LCS-LENGTH(X, n, Y, m)
para i ← 1 até n faça
   c[i, 0] ← 0
para j ← 0 até m faça
   c[0, j] ← 0
para i ← 1 até n faça
   para j ← 1 até m faça
      se X[i] = Y[j]
         então c[i, j] ← c[i − 1, j − 1] + 1
               b[i, j] ← "↖"
         senão se c[i − 1, j] ≥ c[i, j − 1]
                  então c[i, j] ← c[i − 1, j]
                        b[i, j] ← "↑"
                  senão c[i, j] ← c[i, j − 1]
                        b[i, j] ← "←"
devolva c e b
```

(a) Escreva uma versão recursiva com memoização deste algoritmo que devolva o valor de $c[n, m]$. (Essa versão não precisa construir a matriz b.) Analise a complexidade de tempo do seu algoritmo.

(b) Escreva um algoritmo *recursivo* que recebe a tabela b gerada pelo algoritmo LCS-LENGTH acima, bem como as seqüências $X[1\,..\,n]$ e $Y[1\,..\,m]$, e imprime uma subseqüência de comprimento máximo das seqüências $X[1\,..\,n]$ e $Y[1\,..\,m]$. Seu algoritmo deve consumir tempo $O(n+m)$. Analise a complexidade de tempo do seu algoritmo.

2. [3,0 pontos]
Considere o problema de imprimir nitidamente um parágrafo em uma impressora. O texto de entrada é uma seqüência de n palavras de comprimentos $l_1, l_2, \ldots, l_n$, medidos pelo número de caracteres. Queremos imprimir esse parágrafo com nitidez em uma série de linhas que contêm no máximo M caracteres cada uma. Nosso critério de "nitidez" é dado a seguir. Se uma determinada linha contém palavras de i até j, onde $i \leq j$, e deixamos exatamente um espaço entre as palavras, o número de espaços extras no final da linha é $M - j + i - \sum_{k=i}^{j} l_k$, que deve ser não-negativo para que as palavras caibam na linha. Desejamos minimizar a soma, sobre todas as linhas exceto a última, do cubo do número de espaços extras no final das linhas. Escreva um algoritmo de programação dinâmica que, dado n e a seqüência de comprimentos $l_1, l_2, \ldots, l_n$, **calcula o valor desta soma mínima**. Explique sucintamente porque o seu algoritmo funciona e analise o seu tempo de execução e os seus requisitos de espaço. Seu algoritmo deve consumir tempo polinomial em n.

3. [1,0 pontos]
   Suponha que desejamos percorrer uma lista ligada de comprimento n onde cada elemento contém uma chave k junto com um valor de hash $h(k)$. Cada chave é uma longa cadeia de caracteres. Como podemos tirar vantagem dos valores de hash quando fazemos uma busca por um elemento com uma dada chave?

4. [3,0 pontos]

   (a) Defina

   - algoritmo polinomial;
   - problema de decisão;
   - certificado curto para SIM e para NÃO;
   - verificador polinomial para SIM e para NÃO;
   - as classes P, NP e coNP.

(b) Seja $G = (V, E)$ um grafo. Um conjunto $S \subseteq V$ é *independente* se quaisquer dois vértices de S não são adjacentes. Ou seja, não há nenhuma aresta do grafo com as duas pontas em S. O problema IS consiste no seguinte: dado um grafo G e um inteiro $k \geq 0$, existe um conjunto independente em G com k vértices? Mostre que IS é NP-completo. (*Dica:* Use o problema do CLIQUE

> Problema CLIQUE (G, k): dado um grafo G e um inteiro $k \geq 0$, existe um clique em G com k vértices?

que, como vimos em aula, é NP-completo.)