



## Terceira Avaliação

22 de Junho de 2006

### OBSERVAÇÕES IMPORTANTES

- Leia com atenção o enunciado das questões e responda de maneira adequada.
- Para serem consideradas, as respostas deverão estar completas e legíveis. Apresente sempre os passos intermediários e não somente a resposta.
- A prova terá duração de três aulas (150 minutos) iniciando as 12:30 e terminando as 15:00.
- É permitido consultar somente uma folha A4 com anotações. Estas anotações não podem conter exercícios resolvidos, somente teoria. Não será permitido o empréstimo ou troca desta folha de anotações durante a prova.

**QUESTÃO 1** [3 pontos] Resolva a seguinte instância do problema da mochila não fracionária considerando que a capacidade da mochila é  $W = 6\text{kg}$  usando:

a) **Estratégia Gulosa.** Apresente o desenvolvimento passo a passo indicando qual item é adicionado à solução em cada passo e qual o peso e valor acumulado na mochila até a obtenção da solução final [1 ponto].

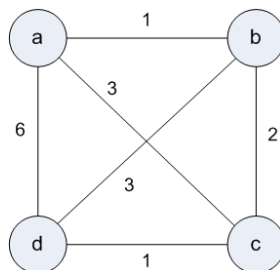
b) **Algoritmo Branch and Bound.** Construa a árvore estado-espaço e indique por números sobre os nós da árvore a ordem em que eles são gerados. Dentro de cada nó indique o peso parcial inserido na mochila ( $w$ ), o valor parcial inserido na mochila ( $v$ ) assim como o valor do limite. Indique também qual o critério de poda/corte de um nó da árvore quando pertinente. Note que esta questão será avaliada pela construção da árvore estado-espaço. Dica:  $ub = v_i + (W-w) v_{i+1}/w_{i+1}$  [2 pontos].

Item (i)	Valor (v)	Peso (w)	Relação Valor / Peso
1	\$25	3	8,33
2	\$20	2	10
3	\$15	1	15
4	\$40	3	13,33
5	\$50	4	12,50

**Obs:** O problema da mochila 0-1: Dados  $n$  itens cujo valor ( $v$ ) e peso ( $w$ ) são conhecidos (e.g. o item  $i$  vale  $\$v_i$  e pesa  $w_i$  quilos), encontrar o subconjunto mais valioso de itens com peso total  $\leq W$ . Temos que ou pegar ou não pegar um item. Não podemos pegar somente uma parte dele.

**QUESTÃO 2** [2 pontos] Resolva a seguinte instância do problema do caixeiro viajante utilizando um algoritmo aproximado.

- Proponha um algoritmo aproximado para resolver este problema. Explique quais são os passos executados por ele [0.5 pontos].
- Mostre a resolução do problema através do seu algoritmo proposto no item (a) [1 ponto].
- Qual a heurística utilizada em seu algoritmo? [0.25 pontos]
- Seu algoritmo leva a uma solução exata para o problema? [0.25 pontos]



**QUESTÃO 3** [1 ponto] Responda de maneira clara e objetiva as seguintes questões:

- O que é o limite inferior de um problema? Explique. [0.2 pontos]
- Qual a importância de se conhecer o limite inferior de um problema? Explique. [0.2 pontos]
- Qual a implicação para a ciência da computação se fosse provado que  $P=NP$ ? Explique. [0.2 pontos]
- O que é uma heurística? Explique. [0.2 pontos]
- Como podemos saber se um algoritmo aproximado leva a uma solução exata de um problema? Explique. [0.2 pontos]



**QUESTÃO 4** [2 pontos] Determine a subseqüência comum mais longa de  $X = X_1^m = \langle 1, 0, 0, 1 \rangle$  e  $Y = Y_1^n = \langle 0, 1, 0, 1, 1 \rangle$  utilizando programação dinâmica. Construa a tabela de custos  $c[0...m, 0...n]$  cujas entradas são calculadas em ordem de linha principal. Construa também a tabela  $b[1...m, 1...n]$  onde os elementos apontam para a entrada na tabela correspondente à solução ótima do subproblema ao se calcular  $c[i, j]$ .

Dica: Construa somente uma tabela incluindo tanto os valores de custos quanto a “direção da solução ótima” (em anexo).  
Dica: Utilize os algoritmos dados a seguir.

LCS-LENGTH( $X, Y$ )

```

1  m ← length[X]
2  n ← length[Y]
3  for i ← 1 to m
4      do c[i, 0] ← 0
5  for j ← 0 to n
6      do c[0, j] ← 0
7  for i ← 1 to m
8      do for j ← 1 to n
9          do if  $x_i = y_j$ 
10             then  $c[i, j] \leftarrow c[i - 1, j - 1] + 1$ 
11                  $b[i, j] \leftarrow \text{“}\searrow\text{”}$ 
12             else if  $c[i - 1, j] \geq c[i, j - 1]$ 
13                 then  $c[i, j] \leftarrow c[i - 1, j]$ 
14                      $b[i, j] \leftarrow \text{“}\uparrow\text{”}$ 
15             else  $c[i, j] \leftarrow c[i, j - 1]$ 
16                  $b[i, j] \leftarrow \text{“}\leftarrow\text{”}$ 
17  return c and b
    
```

PRINT-LCS( $b, X, i, j$ )

```

1  if  $i = 0$  or  $j = 0$ 
2      then return
3  if  $b[i, j] = \text{“}\searrow\text{”}$ 
4      then PRINT-LCS( $b, X, i - 1, j - 1$ )
5           print  $x_i$ 
6  elseif  $b[i, j] = \text{“}\uparrow\text{”}$ 
7      then PRINT-LCS( $b, X, i - 1, j$ )
8  else PRINT-LCS( $b, X, i, j - 1$ )
    
```

	j	0	1	2	3	4	5
i		$y_j$	0	1	0	1	1
0	$x_i$						
1	1						
2	0						
3	0						
4	1						

**QUESTÃO 5** [2 pontos] Aplique o algoritmo *backtracking* para resolver o problema das  $n$  rainhas considerando  $n=5$ . O problema das  $n$  rainhas consiste em colocar  $n$  rainhas em um tabuleiro de xadrez  $n$  por  $n$  de modo que duas delas não estejam na mesma linha, coluna ou diagonal.

- Explique como o algoritmo *backtracking* pode ser utilizado para resolver este problema [0.5pontos] ?
- Construa a árvore estado-espço e indique por números sobre os nós da árvore a ordem em que os nós são gerados. Construa a árvore até encontrar uma solução válida para o problema [1.5pontos].

Obs: Os pontos relativos a esta questão correspondem à construção correta da árvore estado-espço.

Estado inicial  
(0)
