MC548: Projeto e Análise de Algoritmos II

Prof. Cid C. de Souza – 3ª Prova – (02/12/2010)

Nome:

RA: **Turma:**

Questão	frac	Peso	Nota
1			
2			
3			
4			
Total		10,0	

Observação: o peso das questões será decidido pelo docente da seguinte forma: as duas questões que você responder melhor terão peso 3 e as demais terão peso 2. Portanto, uma mesma questão pode ter peso 2 para um aluno e peso 3 para um outro aluno.

Instruções:

1. A duração da prova é de 110 minutos.
2. Coloque o seu nome, RA e turma em **no alto desta página e em todas as folhas de resposta**.
3. Não é permitido usar qualquer material de consulta.
4. **Questões mal justificadas serão consideradas erradas !**
5. Use as folhas de papel almaço entregues pelo docente para responder às questões da prova.
6. A prova pode ser feita a lápis porém, nesse caso, você fica impedido de solicitar revisão de nota.
7. O uso de calculadoras ou quaisquer outros equipamentos eletrônicos, inclusive celulares, está proibido durante a prova.
8. Não desgrampeie o caderno de questões.

1. Seja um grafo não-orientado $G = (V, E)$ com um custo w_{ij} (arbitrário!) associado a cada aresta $(i, j) \in E$. Dado um subconjunto U de vértices de V, o *corte* definido por U em G, denotado por $\delta(U)$, é o conjunto de arestas (i, j) onde $i \in U$ e $j \notin U$. Um corte $\delta(U)$ é dito ser *equilibrado* se $|U| = \lfloor \frac{|V|}{2} \rfloor$.
O *problema do corte equilibrado máximo* em G é $\mathcal{NP}$-difícil. Nele, o objetivo é encontrar o subconjunto U de vértices tal que $\delta(U)$ seja equilibrado e a soma dos custos das arestas em $\delta(U)$ é maximizado. Formule este problema usando Programação Linear Inteira. Explique o significado de cada variável e de cada restrição do seu modelo, especificando suas quantidades em termos de $n = |V|$ e $m = |E|$.

2. Responda aos itens a seguir:

(a) Defina o que é uma *k-aproximação absoluta* para um problema de otimização, sendo k uma constante.

(b) Considere o seguinte "algoritmo" para o problema da clique máxima em um grafo não-orientado $G = (V, E)$.

Passo 1: $S = \{\}; \ell = 0;$ /* S: clique retornada pela heurística (inicialmente vazia) e $\ell = |S|$ */

Passo 2: Encontre o vértice u de maior grau em G e que esteja conectado a todos os vértices de S através de uma aresta de E. Se tal vértice não exisitr, retorne ℓ.

Passo 3: Faça $S = S \cup \{u\}$ e $\ell = \ell + 1$. Volte ao Passo 1.

Mostre que não existe uma constante k tal que o "algoritmo" acima seja uma k-aproximação absoluta. Em outras palavras, mostre que o "algoritmo" é, na verdade, uma heurística e que ela é *"arbitrariamente ruim"* (no sentido definido em aula).

(c) Considere um grafo não-orientado $G = (V, E)$. Diz-se que um subconjunto $\mathcal{C}$ de ciclos em G disjuntos nas arestas é uma *cobertura* de G, se todo vértice de V está contido em pelo menos um dos ciclos em $\mathcal{C}$. O *tamanho* da cobertura é definido como sendo o número de ciclos que a compõem.

(c.1) Mostre que o problema de encontrar a cobertura de um grafo por ciclos disjuntos nas arestas que tenha tamanho mínimo é $\mathcal{NP}$-difícil.

(c.2) Mostre que não existe aproximação absoluta para o problema do item anterior a menos que $\mathcal{P} = \mathcal{NP}$ (Dica: inspire-se na única prova desse tipo que foi feita em sala de aula).

3. João deve despachar pelos Correios n livros cujos pesos (em `kg`) são dados por $\{p_1, p_2, \ldots, p_n\}$. Ele dispõe de caixas para empacotar os livros que suportam um peso máximo de P `kg`. Note que as caixas podem ter dimensões variadas. A única restrição dos Correios é mesmo o peso máximo que elas podem ter. Ou seja você pode supor que, se não fosse pelo peso, poder-se-ia até colocar todos livros numa mesma caixa.
Agora, para cada caixa despachada, os Correios cobram um valor fixo C, independentemente do seu peso real. João quer saber como organizar os livros nas caixas de modo a minimizar o custo de envio dos livros. Para tal, ele elaborou o **algoritmo mostrado ao final dessa página**. Responda aos itens abaixo:

(a) Como seria a alocação dos livros em caixas proposta pelo algoritmo para a instância dada por $n = 9$, $p = \{6, 4, 5, 5, 4, 4, 2, 3, 7\}$ e $P = 10$? Essa solução é ótima ? Justifique.

(b) Qual a complexidade deste algoritmo em função de n ? Justifique.

(c) Mostre que o algoritmo acima é uma 2-aproximação para o problema que João quer resolver (ou seja, prove que o algoritmo é uma 1-aproximação relativa).

```
Livros(p, n, P);
    Ordenar os livros em ordem decrescente de peso;
    (* supor a partir daqui que p_1 ≥ p_2 ≥ ... ≥ p_n *)
    i ← 1;    (* livro mais leve nao empacotado ainda *)
    j ← n;    (* livro mais pesado nao empacotado ainda *)
    k ← 0;    (* numero de caixas usadas *)
    Enquanto i ≤ j faça
        k++;    imprimir("pegue uma nova caixa");
        P' ← P;    (* capacidade residual da caixa *)
        (* coloca primeiro os livros mais pesados na caixa *)
        Enquanto (i ≤ j) e (P' ≥ p_i) faça
            imprimir("coloque livro i na caixa k")
            P' ← P' − p_i;    i++;
        fim-enquanto
        (* coloca agora os livros mais leves na caixa *)
        Enquanto (i ≤ j) e (P' ≥ p_j) faça
            imprime("coloque livro j na caixa k")
            P' ← P' − p_j;    j−−;
        fim-enquanto
    fim-enquanto
    imprimir("custo: kC reais")
fim
```

4. Considere o problema de determinar se um grafo $G = (V, E)$, onde $|V| = n$ e $|E| = m$, possui **uma** cobertura de vértices (`CV`) de tamanho $\leq p$. Complete o procedimento recursivo abaixo de modo que ele implemente uma estratégia de *backtracking* que resolva o problema. As tuplas que compõem o espaço de estados são dadas por um vetor binário x de tamanho n (fixo) onde, para todo $i = 1 \ldots n$, $x_i = 1$ se e somente se o vértice i faz parte da cobertura. O grafo é dado pela matriz de adjacências A.

`Encontra_cobertura`(G,p);
    Para $i = 1$ **até** n **faça** $x_i \leftarrow 0$;
    `CV`(1,0,0,p);
fim.

`CV`(k, s, c, p); (* procedimento recursivo auxiliar *)
(* == *)
(* k: índice do próximo elemento da tupla a ser decidido *)
(* s: o número de vértices escolhidos nas $k-1$ primeiras posições da tupla *)
(* c: o número de arestas cobertas pelos vértices escolhidos nas $k-1$ $1^{\underline{as}}$ posições da tupla *)
(* == *)
    $x_k \leftarrow 1$;
    $\delta \leftarrow 0$; (* número de novas arestas cobertas *)
    Para $i = 1$ **até** n **faça**
        Se $(x_i = 0)$ **e** $(A[i, k] = 1)$
        então $\boxed{}$;
    Se $(s + 1 \leq p)$ **e** $(c + \delta = m)$
    então $\boxed{}$;
    se não
        Se $(s + 1 < p)$ **e** $\boxed{}$
        então $\boxed{}$;

    $x_k \leftarrow 0$;
    Se $\boxed{}$ **então** $\boxed{}$;
fim.

Desenhe a subárvore de espaço de estados que é percorrida ao se executar o algoritmo `Encontra_cobertura` para o grafo mostrado na figura abaixo, sendo $p = 3$. Nessa subárvore, indique para cada aresta a decisão correspondente e para cada nó os valores de k, s e c nas chamadas correspondentes do procedimento `CV`.