Algoritmos de Aproximação
Segundo Semestre de 2012
Primeira Prova – 10 de setembro

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 prova consta de 4 questões. Verifique antes de começar a prova se o seu caderno de questões está completo.
5. Não é permitido o uso de folhas avulsas para rascunho.
6. Não é necessário apagar rascunhos no caderno de questão mas especifique qual é a resposta e qual é o rascunho.
7. A prova é sem consulta.
8. Você pode usar, sem ter que escrever, algoritmos vistos em aula. Neste caso, antes de usá-lo, deixe claro qual é o protótipo do algoritmo, o que ele devolve/faz, e quanto tempo ele consome em função de sua entrada.

Não escrever nesta parte da folha

Questão	Nota	Observação
1		
2		
3		
4		
Total		

Boa prova!

1. [2,0 pontos]
   No **problema da árvore de Steiner com peso nos vértices**, são dados um grafo $G = (V, E)$, pesos $w_i \geq 0$ para cada vértice i em V, custos $c_e \geq 0$ para cada aresta e em E, e um conjunto $T \subseteq V$. O custo de uma árvore é a soma do peso dos vértices na árvore mais a soma do custo das arestas na árvore. O objetivo do problema é encontrar uma árvore de peso mínimo que contenha todos os vértices do conjunto T.

   Mostre que existe uma constante $c > 0$ tal que não existe $(c \ln |T|)$-aproximação para o problema da árvore de Steiner com peso nos vértices a menos que P = NP.

   Lembre-se que o problema da cobertura por conjuntos consiste em, dado um conjunto finito E, uma coleção $\mathcal{S}$ de subconjuntos de E que cobre E, e um custo w_S para cada S em $\mathcal{S}$, encontrar uma subcoleção $\mathcal{S}' \subseteq \mathcal{S}$ que cobre E e tem custo mínimo. Na versão *sem custos* do problema, o custo $w_S = 1$ para todo S em $\mathcal{S}$.

   Você pode usar, na sua demonstração, o seguinte teorema visto em aula:

   Teorema: Existe uma constante $c > 0$ tal que, se existir uma $(c \ln |E|)$-aproximação para o problema da cobertura por conjuntos sem custos, então P = NP.

2. [2,0 pontos]
Lembre-se que o problema da mochila consiste no seguinte: dados um número n, um valor $v_i \geq 0$ e um tamanho $s_i > 0$ para os itens $i = 1, \ldots, n$, e um número $B > 0$ que indica a capacidade de uma mochila, encontrar um subconjunto S de $\{1, \ldots, n\}$ tal que $\sum_{i \in S} s_i \leq B$ e $\sum_{i \in S} v_i$ é máximo. Assumimos que $s_i \leq B$ para todo i.

Considere o seguinte algoritmo guloso para o problema da mochila. Inicialmente ordenamos todos os itens em ordem não-crescente da razão do valor pelo tamanho, ou seja, $v_1/s_1 \geq v_2/s_2 \geq \cdots \geq v_n/s_n$. Seja i^* o índice de um item de valor máximo, ou seja, $v_{i^*} = \max_i v_i$. O algoritmo guloso coloca os itens pela ordem dos índices até o próximo item não mais caber; isto é, ele encontra k tal que $\sum_{i=1}^{k} s_i \leq B$ mas $k = n$ ou $\sum_{i=1}^{k+1} s_i > B$. O algoritmo devolve ou $\{1, \ldots, k\}$ ou $\{i^*\}$, aquele que tiver maior valor. Prove que esse algoritmo é uma 1/2-aproximação para o problema da mochila.

3. [3,0 pontos]
Considere a seguinte relaxação linear (P) do problema da cobertura por vértices: dado um grafo $G = (V, E)$ e um peso $w_i \geq 0$ para cada vértice i em V, encontrar um vetor x indexado por V que

$$\begin{array}{rll} \text{minimize} & \sum_{i \in V} w_i \, x_i & \\ \text{sujeito a} & x_i + x_j \geq 1 & \text{para todo } ij \text{ em } E \\ & x_i \geq 0 & \text{para todo } i \text{ em } V. \end{array}$$

Uma solução x do programa linear (P) tal que $x_i \in \{0, 1/2, 1\}$ é chamada de solução *meio-inteira*. É possível provar que sempre existe uma solução ótima meio-inteira para (P). Mais do que isso, existe um algoritmo polinomial que encontra uma solução ótima meio-inteira para (P).

Uma *k-coloração* de um grafo $G = (V, E)$ é uma função $\gamma : V \to \{1, \ldots, k\}$ tal que $\gamma(i) \neq \gamma(j)$ sempre que $ij \in E$. Sabe-se que, para todo grafo planar, existe uma 4-coloração. (Um grafo é *planar* se pode ser desenhado no plano sem que suas arestas se cruzem, mas você não precisa saber disso para responder a essa questão.) Ademais, existe um algoritmo polinomial que recebe um grafo G planar e devolve uma 4-coloração de G.

(a) Escreva uma 3/2-aproximação para o problema da cobertura por vértices para grafos planares. Você pode usar, no seu algoritmo, os dois algoritmos polinomiais mencionados acima.

(b) Mostre que seu algoritmo de fato devolve uma cobertura por vértices do grafo (planar) dado.

(c) Mostre que o seu algoritmo tem uma razão de aproximação de 3/2.

4. [3,0 pontos]
Neste problema, consideramos uma variante do problema de escalonamento em máquinas idênticas visto em aula. Nesta variante, as máquinas não são idênticas. Lembre-se que d_j é a duração da tarefa j. A máquina i tem uma velocidade $s_i > 0$, e leva d_j/s_i unidades de tempo para processar a tarefa j. Assuma que as máquinas estão ordenadas de modo que $s_1 \geq \cdots \geq s_m$. Chamamos esta variante do problema de *escalonamento em máquinas relacionadas*.

Um *procedimento de decisão ρ-relaxado* para um problema de escalonamento é um algoritmo que, dado uma instância do problema de escalonamento em questão, e um prazo $D > 0$, ou produz um escalonamento com tempo de conclusão no máximo $\rho\, D$, ou estabelece corretamente que não existe escalonamento com tempo de conclusão no máximo D para a instância dada.

Considere a seguinte variante do algoritmo de Graham para escalonamento em máquinas relacionadas, que chamaremos de GRAHAM2. O algoritmo recebe: o número n de tarefas, o número m de máquinas, a duração das tarefas, dadas no vetor d, a velocidade das máquinas, dadas no vetor s, e um prazo $D > 0$.

O algoritmo GRAHAM2 tem duas fases. Na primeira fase, o algoritmo rotula cada tarefa j com a mais lerda máquina i que permite que a tarefa j complete até o instante D, ou seja, o maior i tal que $d_j/s_i \leq D$. Se não existe um tal i para alguma tarefa j, então claro que não existe escalonamento com tempo de conclusão no máximo D para esta instância e GRAHAM2 termina com essa resposta. Se existe um tal i para cada tarefa j, então GRAHAM2 entra na segunda fase, descrita a seguir.

Inicialmente todas as máquinas estão funcionando e estão vagas. A cada momento em que uma máquina i em funcionamento torna-se vaga antes do instante D, a máquina i começa a executar a próxima tarefa com rótulo i que não foi processada. Se não há tarefa com rótulo i por processar, a máquina i começa a executar a próxima tarefa com rótulo $i + 1$ que não foi processada. Se não há tarefa com rótulo $i + 1$ por processar, a máquina i começa a executar a próxima tarefa com rótulo $i + 2$ que não foi processada, e assim por diante. Se não há mais tarefas a serem processadas pela máquina i, ela para. Se uma máquina torna-se vaga no instante D ou depois deste, a máquina também para. O algoritmo GRAHAM2 termina a segunda fase quando todas as máquinas pararam.

Se nem todas as tarefas foram executadas ao final da segunda fase, o algoritmo GRAHAM2 estabelece que não há escalonamento com tempo de conclusão no máximo D.

Prove que o algoritmo GRAHAM2 é um procedimento de decisão 2-relaxado. Ou seja, prove que

(a) quando o algoritmo estabelece que não há escalonamento com tempo de conclusão no máximo D, ele está certo; e

(b) quando o algoritmo devolve um escalonamento, o seu tempo de conclusão é no máximo $2D$.