MC658 - Análise de Algoritmos III
Lista de Exercícios
Programação Dinâmica e Algoritmos Pseudopolinomiais

1º semestre de 2019 – Prof. Cid C. de Souza – Instituto de Computação – UNICAMP

1. No Problema Inteiro da Mochila (PIM) são dados n itens e uma mochila com capacidade de peso W, representada por um valor inteiro positivo. Além disso, para cada item i são dados o seu custo c_i e peso w_i, ambos representados por inteiros positivos. Suponha que você dispõe de uma quantidade ilimitada de cada item. Assim, desde que a capacidade de peso não seja violada, você pode colocar na mochila quantas unidades de um item i quiser. O problema que se quer resolver é encontrar um subconjunto de itens cuja soma de pesos não exceda a capacidade da mochila e cujo custo total (dado pela soma dos custos de todos itens no subconjunto) seja máxima.

   (a) Claro que o número máximo de unidades do item i que você pode levar na mochila é $k_i = \lfloor \frac{W}{w_i} \rfloor$. Assim, ao invés de considerar que até k_i unidades do item i estão disponíveis para serem levadas, você poderia pensar em k_i itens distintos de custo c_i e peso w_i cada um que podem ou não serem levados na mochila. Com isto, você reduz o PIM ao Problema Binário da Mochila (PBM) e, consequentemente, pode usar o algoritmo de programação dinâmica com complexidade pseudopolinomial (em W) que foi visto em aula para resolver o PBM para resolver o PIM. Qual seria a complexidade deste algoritmo?

   (b) Baseado no algoritmo de programação dinâmica para o PBM, desenvolva um algoritmo pseudopolinomial para o PIM com complexidade $O(nW^2)$. Explicite claramente qual a fórmula de recorrência em que se baseia o seu algoritmo e quais são os seus casos base.

   (c) Mostre como é possível resolver o PIM em $O(nW)$ fazendo uma pequena modificação no algoritmo de programação dinâmica do item anterior. Ou seja, mostre que você pode resolver o PIM com um algoritmo que tem a mesma complexidade que aquele que resolve o PBM. Explicite claramente qual a fórmula de recorrência em que se baseia o seu algoritmo e quais são os seus casos base.

   (d) Compare a complexidade do algoritmo do item anterior com aquela obtida no item *(a)*.

2. Considere a seguinte versão do problema da mochila, chamado de Problema Mochila Múltipla (PMM para abreviar):

   Entrada: um conjunto S de n itens com custo c_i e peso w_i e k diferentes mochilas com capacidades $W_1, \ldots, W_k$.

   Saída: um subconjunto de S com custo total máximo que pode ser embalado nas k mochilas.

   (a) Forneça um algoritmo de tempo pseudopolinomial que resolva o PMM para $k = 2$ em tempo $O(nW_1W_2)$.

   (b) Estenda seu algoritmo para k geral. Forneça as complexidades assintóticas de tempo de execução e de espaço.

3. Considere o seguinte problema de sequenciamento de tarefas em uma única máquina com prazos (*deadlines*). Temos n tarefas dadas por triplas $(p_1, d_1, t_1), (p_2, d_2, t_2), \ldots, (p_n, d_n, t_n)$, onde:

   - p_i: penalidade incorrida se a tarefa i não for executada até o prazo d_i.

- t_i: tempo de processamento da tarefa i.

Um subconjunto J das tarefas deve ser escolhido para ser processado sem sofrer penalidades. O custo da solução é a soma das penalidades das tarefas **não** pertencentes a J. Deve-se escolher J tal que o custo seja mínimo.

(a) Mostre que o problema é $\mathcal{NP}$-difícil.

(b) Inicialmente, observe que sempre existe uma solução ótima em que não há tempo ocioso entre as tarefas, ou seja, à exceção da primeira tarefa executada, uma tarefa sempre começa a ser executada tão logo a tarefa anterior tenha sido finalizada.

Agora, mostre que sempre existe uma solução ótima dada por uma sequência de execução de tarefas na máquina da forma $i_1, i_2, \ldots, i_s, i_{s+1}, i_{s+2}, \ldots, i_n$ tal que apenas tarefas $i_1, \ldots, i_s$ são terminadas antes dos seus prazos e satisfazem a $d_{i_1} \leq d_{i_2} \leq \ldots \leq d_{i_s}$. Em outras palavras, existe uma solução ótima onde as tarefas não penalizadas são executadas na ordem *não decrescente* dos seus prazos.

(c) Suponha que: *(i)* as tarefas estão indexadas na entrada de modo que se $1 \leq i < j \leq n$, então $d_i < d_j$, e que *(ii)* todos os prazos e tempos de processamento das tarefas sejam dados por inteiros positivos.

Seja $P = \sum_{i=1}^{n} p_i$. Deseja-se projetar um algoritmo de **programação dinâmica** de complexidade $O(nT)$ (**pseudopolinomial**) que resolve o problema. Dê a fórmula de recorrência que governa o funcionamento do seu algoritmo, deixe claro quais são os casos base da recorrência, escreva um pseudocódigo e analise a complexidade do algoritmo de modo a confirmar que ele atende às exigências do enunciado.

Dicas: defina $F_j(t)$ como sendo o custo ótimo do problema considerando apenas as j primeiras tarefas e um tempo máximo t para a execução de todas as tarefas que não sofrerão penalização. Com esta definição, procure responder as seguintes questões: qual o valor de $F_o(t)$ para todo $t \geq 0$? Qual valor de $F_j(t)$ corresponde ao valor ótimo do problema original com n tarefas? Como o valor de $F_j(t)$ se relaciona com o ótimo de outros subproblemas?