CSC364S (MWF2 section) Problem Set 1 Spring, 2004

Due: Wednesday, Jan 28, beginning of lecture

NOTE: Each problem set only counts 5% of your mark, but it is important to do your own work. These assignments will be followed by term tests, each worth 15% of your final grade. You may consult with others concerning the general approach for solving problems on assignments, but you must write up all solutions entirely on your own. As an experiment you may chose to work in pairs and then submit one assignment. Anything else is *plagiarism*, and is subject to the University's Code of Behavior. You will receive 1/5 points for any question/subquestion for which you say "I do not know how to answer this question". You will receive .5/5 points if you just leave the question blank.

1. Consider the general knapsack problem where an input consists of N items $(g_1, w_1), \ldots (g_N, w_N)$ and a capacity bound C. All input paramteres are non negative integers. Think of g_i as the value/gain of the i^{th} item and w_i as its weight. As in the simple knapsack problem, $S \subseteq \{1, \ldots, N\}$ is a feasible subset if $\sum_{i \in S} w_i \leq C$. The goal now is to try to find a feasible S so as to maximize $P(S) = \sum_{i \in S} g_i$.

   We consider the following natural greedy algorithm:

   ```
   Sort items so that $(g_1/w_1) \geq (g_2/w_2) \ldots \geq (g_N/w_N)$.
   $P = 0,\ W = 0,\ S = \emptyset$;
   for i: 1 .. N

     if $W + w_i \leq C$, then
       $P = P + g_i,\ W = W + w_i,\ S = S \cup \{i\}$;
     endif
   endfor
   ```

   (a) Show that this greedy algroithm is not a c-approximation algorithm for any constant c.

   (b) Now consider the *fractional* general knapsack problem where we can take a fractional amount a_i of any item with $0 \leq a_i \leq 1$. That is, a feasible solution is a tuple $A = (a_1, \ldots, a_N)$ with $\sum_{1 \leq i \leq N} a_i \cdot w_i \leq C$. Now the goal is to maximize $F(A) = \sum_{1 \leq i \leq N} a_i g_i$ for a feasible A.
   Modify the greedy algorithm above to derive an optimal algorithm. Prove that the algorithm always computes an optimal solution.

2. This problem concerns Kruskals greedy algorithm for the MST problem. Let $G = (V, E)$ be a graph and $c : E \rightarrow \Re$ a cost function on the edges.

   (a) Suppose that all edges had unique costs; that is, $c(e_1) \neq c(e_2)$ for $e_1 \neq e_2$. Using the analysis of Kruskals algorithm to show that there is a unique minimum spanning tree.

(b) Let $e \in E$ be a specified edge. Show how to modify Kruskal's algorithm (and its analysis) to compute a minimum spanning tree T subject to the condition that $e \in T$.

3. Consider the following variant of the general knapsack problem. We are given items $(g_1, w_1), \ldots (g_N, w_N)$, a capacity bound C and a cardinality bound M. A feasible subset $S \subseteq \{1, \ldots, n\}$ now satisfies the constraint that $|S| \leq M$ in addition to the usual knapsack constraint that $\sum i \in S w_i \leq C$. Suppose that all input parameters are positive integers with $C \leq N^2$. The objective is (as in the general knapsack problem) to maximize the profit $g(S)$ of a feasible subset S. We want to solve this problem in polynomial (in N) time using dynamic programming.

   (a) Define an appropriate semantic array A and say how to derive the value of an optimal solution from A. Hint: try a three dimensional array.

   (b) Define a corresponding computational array $\tilde{A}$.

   (c) Briefly justify that $A = \tilde{A}$.

   (d) Estimate the time complexity of your algorithm.

   (e) Indicate how you would extend your algorithm to obtain an optimal feasible set S.