CSC 364H L0101 | Test 1 | Summer 2000

Name ______________________ Student No. ______________________

No aids allowed. Answer all questions on test paper. Use backs of sheets for scratch work.

Total Marks: 30

[8] 1. The Simple Knapsack problem *search* (SKS) takes as input: $d, w_1, w_2, \ldots, w_d, C \in \mathbb{N}$ and requires as output a set $S_0 \subseteq \{1, \ldots, d\}$ such that $K(S_0) = M$, where:

$$M = \max_{S \subseteq \{1,\ldots,d\}} \{K(S) | K(S) \leq C\}$$

and $K(S) = \sum_{i \in S} w_i$. Recall that the greedy method for SKS produces an answer $\overline{M}$ such that $\overline{M} \leq C$ but possibly $\overline{M} < M$.

Lemma: Either $\overline{M} = M$, or $\overline{M} > \frac{1}{2}C$.

Your task is to prove this lemma by following the outline below:

Proof of Lemma: Assume that $\overline{M} \neq M$, and let S_0 be the result of greedy, i.e. a subset of $\{1, \ldots, d\}$ such that $K(S_0) = \overline{M}$.

(a) Show that there is at least one weight in S_0:

(b) Show that there is at least one weight not in S_0:

(c) Show now that $\overline{M} > \frac{1}{2}C$ by considering the first weight, call it w_j, which has been rejected after at least one weight has been added:

[9] 2. Recall the Job Scheduling with Deadlines and Profits problem, where each job takes unit time to complete:

Input: A list of jobs $(d_1, g_1), \ldots, (d_n, g_n)$ where $d_i \in \mathbb{N}$ is the deadline of job i and $g_i \in \mathbb{R}^+$ is the profit.

Problem: Find an optimal schedule S.

(a) Define a *schedule* S:

(b) Define a *feasible* schedule S:

(c) Define what it means for a feasible schedule S to be *optimal*:

(d) Define what it means for a schedule S' to *extend* a schedule S:

(e) Define what it means for a schedule S to be *promising*:

(f) Fill in the blanks in the following greedy algorithm for finding an optimal schedule:

Sort the jobs as follows: ______________________________

$d \longleftarrow$ __________

for $t : 1..d$

  $S(t) \longleftarrow$ __________

end for

for $i : 1..n$

  Find the largest t such that _________________________ and let $S(t) \longleftarrow i$

end for

[7] 3. In our dynamic programming solution to the Simple Knapsack Problem (SKS), we defined a Boolean array $R(i,j)$ for $0 \le i \le d$ and $0 \le j \le C$, as follows:

$$R(i,j) = \begin{cases} \mathsf{T} & \text{if } \exists S \subseteq \{1,\ldots,i\} \text{ s.t. } K(S) = j \\ \mathsf{F} & \text{otherwise} \end{cases}$$

(where the convention is that if $i = 0$, then $\{1,\ldots,i\} = \emptyset$).

First, give the recurrence used for computing this array:

(a) Initialization:

(b) For $i > 0$, $R(i,j) = \mathsf{T}$ iff

Now, recall the General Knapsack Problem (GKS):

Input: $d, w_1, \ldots, w_d, v_1, \ldots, v_d, C \in \mathbb{N}$

Problem: Find $\max_{S\subseteq\{1,\ldots,d\}}\{V(S)|K(S) \le C\}$, where $K(S) = \sum_{i\in S} w_i$, and $V(S) = \sum_{i\in S} v_i$.

To solve the general Knapsack problem, we start by computing the same Boolean array $R(i,j)$ that was used to solve SKS. Thus $R(i,j)$ ignores the values v_i, and only depends on the weights w_i. Next we define another array $V(i,j)$ that depends on the values v_i as follows: For $0 \le i \le d$, $0 \le j \le C$

$$V(i,j) = \max\{V(S)|S \subseteq \{1,\ldots,i\} \text{ and } K(S) = j\}$$

Give a recurrence for V:

[6] 4. The Activity Selection with Profits problem is defined as follows:

Input: A list of activities $(s_1, f_1, p_1), \ldots, (s_n, f_n, p_n)$, where each activity consists of a start time s_i and a finish time f_i (such that $0 \leq s_i < f_i$), and a profit $p_i > 0$ obtained if the activity is selected.

Problem: Find a set $S \subseteq \{1, \ldots, n\}$ of selected activities such that no two activities in S are in conflict and $\sum_{i \in S} p_i$ is maximal.

The dynamic programming solution that was given in lectures involved computing an array $A(j)$ whose value is the maximum profit that can be obtained by selecting a set of activities that completes by time u_j (where $u_1 < u_2 < \ldots < u_k$ is the list of all the distinct finish times, sorted in increasing order).

Assuming that the array just described has already been evaluated, show how to use the values in the array to compute the actual set S of selected activities:

Hint: consider what happens if $A(k) = A(k-1)$, and what happens if $A(k) > A(k-1)$.