

Name \_\_\_\_\_ Student No. \_\_\_\_\_

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

Total Marks: **100**

*The test consists of 4 questions and 6 pages.*

[25] 1. Define the problem *Dispersed Knapsack* as follows:

**Input:**  $w_1, \dots, w_d, C$ , all positive integers, such that the  $w_i$ 's satisfy the following condition:

$$w_i \geq \sum_{j=i+1}^d w_j, \quad \text{for } i = 1, \dots, d-1$$

**Output:**  $S_{\max} \subseteq \{1, \dots, d\}$  such that  $K(S_{\max}) = \max_{S \subseteq \{1, \dots, d\}} \{K(S) \mid K(S) \leq C\}$ .

Give a greedy algorithm which solves Dispersed Knapsack by filling in the following two blanks:

$S \leftarrow \emptyset$

for  $i : 1..d$

if \_\_\_\_\_ then

\_\_\_\_\_

end if

end for

**Solution:** In the first blank put  $w_i + \sum_{j \in S} w_j \leq C$ , and in the second blank put  $S \leftarrow S \cup \{i\}$ .

[25] 2. Let  $G = (V, E)$  be an undirected (connected) graph where each edge  $e \in E$  has a cost  $c(e)$  associated with it.

(a) Describe Kruskal's algorithm for finding the minimum cost spanning tree  $T$  of  $G$ .

**Solution:** See algorithm 2.1, pg. 31, in textbook.

(b) Suppose that the costs of all the edges are distinct; that is, after ordering the edges by costs we have  $c(e_1) < c(e_2) < \dots < c(e_m)$ . Show that in that case  $G$  has a *unique* spanning tree. (**Hint:** use an appropriate definition of promising.)

(Use next page if necessary.)

**Solution:** Let  $T_o$  be any MCST; we are going to show that  $T_k = T_o$ , where  $T_k$  is the MCST resulting from Kruskal's algorithm.

Let " $T$  is promising" mean  $T \subseteq T_o$ . We are going to show that this is a loop invariant. The BC is trivial. For the induction step, if edge  $e$  is not added, then  $T$  continues being promising.

If edge  $e$  is added, and  $e \in T_o$ , then there is no problem as by IH  $T \cup \{e\} \subseteq T_o$ .

On the other hand, if edge  $e$  is added, but  $e \notin T_o$ , then  $e \in T_k - T_o$ , and so by the *Exchange Lemma*, there exists an  $e' \in T_o - T_k$ , and  $T_{\text{new}} = (T_o \cup \{e\}) - \{e'\}$  is a ST. Since  $c(e) < c(e')$  (if  $c(e') < c(e)$ , then  $e'$  would be considered before  $e$ , and since  $e' \in T_o$ , and  $T \subseteq T_o$ ,  $e'$  would have been placed in  $T$ , and so it would have been in  $T_k$ , which we know not to be the case). But then  $c(T_{\text{new}}) < c(T_o)$ , contradiction. It follows that this last case ( $e \notin T_o$ ) is not possible.

[25] 3. Consider the following variation of the Longest Monotone Subsequence problem:

**Input:**  $d, a_1, a_2, \dots, a_d \in \mathbb{N}$ .

**Output:** What is the length of the longest subsequence of  $a_1, a_2, \dots, a_d$ , where any two consecutive members of the subsequence differ by at most 1?

For example, the longest such subsequence of  $\{7, 6, 1, 4, 7, 8, 20\}$  is  $\{7, 6, 7, 8\}$ , so in this case the answer would be 4.

Give a recurrence for this problem.

**Solution:** First define the array  $R(j)$  to be the longest such sequence ending in  $a_j$ . Second, give the following recurrence:

$$R(j) = \begin{cases} 1 & \text{if } |a_i - a_j| > 1 \text{ for all } 1 \leq i < j \\ 1 + \max_{1 \leq i < j} \{R(i) : |a_i - a_j| \leq 1\} & \text{otherwise} \end{cases}$$

- [25] 4. An *activity*  $i$  has a fixed start time  $s_i$ , finish time  $f_i$ , and profit  $p_i$ . Given a set of activities, we want to select a subset of non-overlapping activities with maximum total profit.

**Input:** A list of activities  $(s_1, f_1, p_1), \dots, (s_n, f_n, p_n)$ . Assume  $p_i > 0$ ,  $s_i < f_i$ , and  $s_i, f_i, p_i \in \mathbb{R}$  where  $1 \leq i \leq n$ .

**Output:** Find a set  $S \subseteq \{1, \dots, n\}$  of selected activities such that no two selected activities overlap, and the profit  $P(S) = \sum_{i \in S} p_i$  is as large as possible.

To solve this problem, we sorted the activities by their finish time:  $f_1 \leq f_2 \leq \dots \leq f_n$ . Then we partition the activities according to their finish times, and denoted these *distinct* finish times by  $u_1 < u_2 < \dots < u_k$ .

Let  $u_0$  be  $\min_{1 \leq i \leq n} s_i$ , i.e., the earliest start time. Thus,  $u_0 < u_1 < u_2 < \dots < u_k$ . We define an array  $A(0..k)$  as follows:

$$A(j) = \max_{S \subseteq \{1, \dots, n\}} \{P(S) \mid S \text{ is feasible and } f_i \leq u_j \text{ for each } i \in S\}.$$

where  $S$  is *feasible* if no two activities in  $S$  overlap.

Note that  $A(k)$  is the maximum possible profit for all feasible schedules  $S$ .

**Your job is to answer the following question:**

Given that  $A$  has been computed, how do you find an actual set of activities  $S$  such that  $P(S) = A(k)$ ?

(Use next page if necessary.)

**Solution:** We show how to find the actual set of activities: Suppose  $k > 0$ . If  $A(k) = A(k-1)$ , then no activity has been scheduled to end at time  $u_k$ , so we proceed recursively to examine  $A(k-1)$ . If, on the other hand,  $A(k) \neq A(k-1)$ , then we know that some activity has been scheduled to end at time  $u_k$ . We have to find out which one it is. We know that in this case

$$A(k) = \max_{1 \leq i \leq n} \{p_i + A(H(i)) \mid f_i = u_k\}$$

so we examine all activities  $i$ ,  $1 \leq i \leq n$ , and output the (first) activity  $i_0$  such that  $A(k) = p_{i_0} + A(H(i_0))$  and  $f_{i_0} \leq u_k$ . Now we repeat the procedure with  $A(H(i_0))$ . We end when  $k = 0$ .

( $H(i)$  is the index of the largest distinct finish time no greater than the start time of activity  $i$ . Formally,  $H(i) = l$  if  $l$  is the largest number such that  $u_l \leq s_i$ .)

**End of Test**