MCGILL UNIVERSITY – COMP 360

TEST 2

April 4, 2011

Student Number: _____

Family Name(s): _____

Given Name(s): _____

- There are 7 pages in total (including this page).
- You have 60 minutes for this test.
- This test is worth 10% of your final mark.
- Answer each question directly on the test paper, in the space provided. Use the reverse side of the pages for rough work. If you need more space for one of your solutions, use the reverse side of a page and indicate clearly the part of your work that should be marked.

Question 1 (out of 12)	
Question 2 (out of 10)	
Question 3 (out of 8)	
Total (out of 30)	

Question 1 [12pt]

(a) [**3pt**] Recall that Knapsack problem is the following problem:

- **Input**: a set of items, where the i-th item has weight w_i and value v_i which are positive integers; and a positive integer W.
- **Output**: a subset of items of maximum total value, whose total weight is at most W.

This problem can be solved using the dynamic programming technique by considering the following subproblems. For each i and V, the subproblem specified by i and V is to compute the minimum total weight of a subset of the first i items whose total value is exactly V.

More precisely, let $v^\star = \max_{i:1\leq i\leq n} v_i$. Then for $1 \leq i \leq n$ and $0 \leq V \leq nv^\star$, define $B(i, V)$ to be the minimum weight of a subset J of the set of items $\{1, 2, \ldots, i\}$ with total value exactly V. In other words,

$$B(i, V) = \min_J \sum_{i\in J} w_i$$

where the min operation is taken over all subsets J of $\{1, 2, \ldots, i\}$ with the property that $\sum_{i\in J} v_i = V$.

(i) Give the value of $B(1, V)$ for all $0 \leq V \leq nv^\star$:

(ii) Give the recursive formula for computing $B(i, V)$ for $2 \leq i$ and $1 \leq V \leq nv^\star$ (note that $B(i, 0) = 0$):

(b) [**1pt**] Give the precise name of the problem that the Floyd-Warshall algorithm solves.

(c) [**2pt**] Describe the array in step 2 of the dynamic programming algorithm given in lecture for solving the Negative Cycle Detection problem.

Question 1 continues here

(d)[2pt] Give precise definition of a flow network.

(e)[1pt] What is the upper bound on the running time of the Ford-Fulkerson algorithm as given in lecture? State your answer using the $\mathcal{O}$ notation. Specify the meaning of all variables in the expression.

(f) [3pt] Give pseudo-code for the Scaling Max-Flow algorithm. (You don't need to give code for finding an st path in a graph, nor the augment(f, P) subroutine.)

Question 2 [10pt] Consider the following scheduling problem. There are n jobs, the i-th job has deadline t_i and requires a continuous period of d_i units of time. (Here d_i and t_i are positive integers, for all i: $1 \leq i \leq n$.) There is a single machine that can process one job at a time. The problem is to schedule as many jobs as possible on the machine, so that all scheduled jobs meet their deadline, and no two jobs overlap. (Time starts at 0.)

For example, if $n = 3$ and the deadlines and durations of the jobs are:

$$d_1 = 10, t_1 = 2, \quad d_2 = 5, t_2 = 4, \quad d_3 = 8, t_3 = 5$$

then at most 2 jobs can be scheduled. (An example of a schedule with two jobs is: job 1 starts at time 0, job 3 starts at time 2.)

You are asked to give a dynamic programming algorithm that solves this problem. Formally, the input to your algorithm is a set

$$\{(d_1, t_1), (d_2, t_2), \ldots, (d_n, t_n)\}$$

where all d_i, t_i are positive integers, for $1 \leq i \leq n$. The output of your algorithm is a set, which must be of maximum possible size, of pairs of (job,starting time) in the schedule. That is, your output must be a set of the form

$$\{(j_1, s_1), (j_2, s_2), \ldots, (j_k, s_k)\}$$

where: k is the number of jobs in your schedule; $j_1, j_2, \ldots, j_k$ is the list of jobs in the schedule to be processed in that order; and for each $1 \leq i \leq k$, s_i is the starting time for job j_i. In particular

- $s_1 \leq s_2 \leq \ldots \leq s_k$
- $s_i + t_i \leq d_i$ for $1 \leq i \leq k$,
- and $s_i + t_i \leq s_{i+1}$ for $1 \leq i < k$.

(Thus the schedule given in the example above is $\{(1, 0), (3, 2)\}$.)

(a) [2pt] Specify the array that you use to solve the above problem. Clearly describe the size and the entries of the array.

(b) [3pt] Give the initialization and recursive formula for computing the array in part (a).

Question 2 continues here

(c) [**2pt**] Give pseudo-code for a program that computes the array in part (a).

(d) [**2pt**] Give pseudo-code for a program that computes an optimal schedule using the array in part (a).

(e) [**1pt**] What are the running time and space of your algorithm?

Question 3 [8pt] Consider a set of mobile computing clients in a certain town who each need to be connected to one of several possible base stations. We'll suppose there are n clients, with the position of each client specified by its (x, y) coordinates in the plane. There are also k base stations; the position of each of these is specified by (x, y) coordinates as well.

For each client, we wish to connect it to exactly one of the base stations. Our choice of connections is constrained in the following ways. There is a *range* parameter r: a client can only be connected to a base station that is within distance r. There is also a load parameter L: no more than L clients can be connected to any single base station.

Your task is to design an algorithm, using network flow, for the following problem. The input consists of:

- the positions of a set of clients: $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$;
- the positions of a set of base stations: $(a_1, b_1), (a_2, b_2), \ldots, (a_k, b_k)$; and
- the range and load parameters: r and L.

(Here all x_i, y_i, a_j, b_j and r, L are positive integers.) The problem is to decide whether every client can be connected simultaneously to a base station, subject to the range and load conditions in the previous graph.

(a)[3pt] Clearly describe the flow network that you use by specifying its vertices, edges, and the capacity of each edge.

(b)[1pt] What is the running time for the construction of the flow network? (Assume that mathematical operations such as addition, multiplication on the inputs x_i, y_i, a_j, b_j and r, L takes constant time.)

Question 3 continues here

(c)[**1pt**] Show how to use a maximum flow in your flow network to solve the problem.

(d)[**3pt**] Prove that your argument in (c) is correct.