Computer Science 364 — Summer, 2003
St. George Campus — University of Toronto

Homework Assignment #1
Due: June 4, 2003, by 6pm
(in the lecture)

On the cover page of your assignment, you must write ***and sign*** *the following statement: "I have read and understood the homework collaboration policy described on the course webpage." Without such a signed statement your homework will not be marked.*

Question 1. (10 marks)
a. (5 marks) Suppose the runtime of an algorithm is $T(n)$. Which of the expressions below for $T(n)$ would make the algorithm polytime?

(a) $20n^{18} + n^2 \log n$

(b) $6^{3log_2 n}$

(c) $n^{3log_2 n}$

(d) $n^{1/n}$

(e) $3^{(log_2 n)^2}$

b. (5 marks) True or False?

(a) $2^{n^{0.01}} \in O(n^3)$

(b) $2^{3n} \in O(6^n)$

(c) $4^{\log_2 n} \in \Theta(n^2)$

(d) $(\log_2 n)^{\log_2 n} \in O(n^{10})$

(e) $n^{\sqrt{n}} \in \Theta((\sqrt{n})^n)$

Question 2. (20 marks) **Descending Kruskal's algorithm.**
Consider the following Minimum Spanning Tree Algorithm: You are given as an input a connected graph G with m edges and a weight $w(e)$ on each edge e.

Sort the edges $e_1, \ldots, e_m$ in *non-increasing* order of weights
Set $H = G$
for $i = 1$ **to** m **do**
  if $H - e_i$ is connected **then**
    $H = H - e_i$
  end if
end for
Return H

a. (5 marks) Prove that the algorithm always returns a spanning tree of G.
b. (15 marks) Prove that the algorithm always returns a MST of G. The steps of your proof will be similar to that of Kruskal's algorithm.

Question 3. (15 marks) **Monitoring a network.** A typical *communication network* consists of some sites, and some communication links; each communication link joins two sites. It is important to set up monitors on the sites which can alert management if a link breaks down. A monitor on a site can cover all the links involving that site. Since monitors are expensive, we want to try to use as few of them as possible. Here is a greedy algorithm which, when given an arbitrary communication network, will set up some monitors on some sites to cover *all* the links.

> List the links in an arbitrary order $l_1, \ldots, l_n$
> **for** $i = 1$ **to** n **do**
>   **if** l_i is not covered by a monitor **then**
>   Set up *two* monitors, one at each of the two sites joined by l_i
>   **end if**
> **end for**

Setting up two monitors might seem foolish, but in fact, it is quite clever. As you will prove below, this will allow you to find a relatively good solution.

a. (2 marks) Roughly, how long would it take to solve this problem using exhaustive search, checking every possible assignment of monitors to sites, checking each assignment to see if it covers all the links and then choosing the best assignment? What is the running time of above algorithm?

b. (4 marks) Give an example of a network where this algorithm does not use the fewest possible number of monitors and justify your claim.

c. (9 marks) Prove that this algorithm never uses more than twice as many monitors as the optimal solution does. That is, if the algorithm uses $2K$ monitors then any optimal solution requires at least K monitors.

Question 4. (20 marks) **Moving on a checkerboard.**

Suppose that you are given an $n \times n$ checkerboard, called C, and a checker. The square in row i and column j of C is denoted by c_{ij}, with the bottom-left square being c_{11} and the top-right square being c_{nn}. You must move the checker from the bottom edge of the board to the top edge of the board according to the following rule. At each step you may move the checker to one of three squares:

1. the square immediately above,
2. the square that is one up and one to the left (but only if the checker is not already in the leftmost column),
3. the square that is on up and one to the right (but only if the checker is not already in the rightmost column).

Each time you move from square x to square y, you receive $p(x, y)$ dollars. You are given $p(x, y)$ for all pairs (x, y), $x, y \in \{c_{ij} | 1 \leq i, j \leq n\}$, for which a move from x to y is legal. Do *not* assume that $p(x, y)$ is positive. You are to design a dynamic programming algorithm for this problem.

First we want to find the value of the maximum possible gain by moving this checker from the bottom edge to the top edge of the checkerboard.

a. (4 marks) Define a suitable array A to store partial solutions to the problem and state how to find from this table the value of maximum possible gain.

b. (10 marks) Give a recurrence, including the initial values, to compute elements of A. Justify your answer.

c. (3 marks) Give an example of a checkerboard of size $n = 3$ and the function $p(x, y)$. and run your algorithm against this example.

d. (3 marks) Explain how you would extract the sequence of moves in an optimal solution from table A. What is the overall running time of your algorithm?

Question 5. (15 marks) **Descending Partitions.**
A *descending partition* of a positive integer N is a sequence of positive integers $A_1 > A_2 > \cdots > A_k$ with $\sum_{i=1}^{k} A_i = N$.

Given N, compute the number of all possible descending partitions of N.

(For example, if $N = 6$, the descending partitions are: (6), $(5, 1)$, $(4, 2)$, and $(3, 2, 1)$; so your algorithm on input 6 must output 4.)

a. (5 marks) Define a suitable array A to store partial solutions to the problem and state how to find from this table the total number of partitions of N.

b. (10 marks) Give a recurrence, including the initial values, to compute elements of A. Justify your answer. What is the running time of an algorithm based on this recurrence?