Computer Science 364 | Winter, 2004
St. George Campus | University of Toronto

Homework Assignment #1
Due: Jan 28, 2003, by 6pm
(in the tutorial)

Greedy algorithms

[20] 1. **Powers of 2 subset sum**
The input is a set of natural numbers $(a_1, \ldots, a_n)$ and a number t. You need to output a set $S \subseteq \{a_1, \ldots, a_n\}$ such that $\Sigma_{a_i \in S} a_i = t$. That is, given a set of numbers and a number t you need to output a subset of these numbers whose sum is equal to t, if such a subset exists.

Suppose now that all of $\{a_1, \ldots, a_n\}$ are distinct powers of 2. In this case, the problem can be solved by using greedy strategy.

[5] (a) Give a greedy algorithm that, given as an input set $\{a_1, \ldots, a_n\}$ where all a_i are of the form 2^k, and a number t (not necessarily a power of 2) outputs a set S of numbers that sum up to t, or "No solution" if no solution exists.

[10] (b) Prove correctness of your algorithm. That is, state the definition of "promising" appropriate for that problem, and prove by induction that at each step of the algorithm an intermediate solution is promising.

[5] (c) Prove that there is at most one possible solution for any given input. . That is, now you have to prove *uniqueness* rather than *existence* of the solution.

[20] 2. **Kruskal's algorithm with distinct-cost MST**
Let G be a graph where every edge e has cost $c(e)$. Suppose that there is a minimum spanning tree of G in which no two edges have the same cost (that is, all edges of this minimum spanning tree have distinct costs). A minimum spanning tree all of whose edges have distinct costs will be called *distinct-cost MST*. In this problem you need to prove that Kruskal's algorithm always produces a distinct-cost MST for such a graph G.

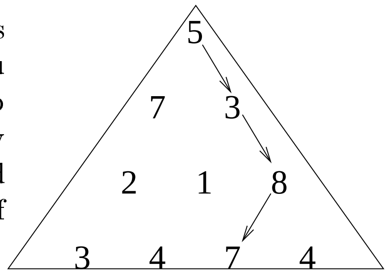
[5] (a) Give an example of a graph where some edges have equal costs, and such that your graph has at least two distinct-cost MSTs. (Draw the graph, and draw two distinct-cost MSTs for your graph.)

[15] (b) Show how to modify the proof of correctness of Kruskal's algorithm to prove that if there exists a distinct-cost MST, then the MST outputted by Kruskal's algorithm is distinct-cost. That is, give an appropriate definition of "promising" and state which parts of the proof of correctness of Kruskal's need to be modified and how.

Dynamic programming

[20] 3. **Moving on a pyramid**

Your input is a list of n rows of numbers, where i^{th} row has i numbers in it. You can visualize them as arranged in a triangular shape. You start from the top number (that is, number 1 in row 1) and in one step you can reach one of the two number in the row below, either directly to the right or to the left of the current number. Your goal is to find the path from the top number to the bottom row such that the sum of numbers in your path is the largest.



Assume that your algorithm gets as an input a $n \times n$ array C, where $C(i,j)$ holds the value of j^{th} number in row i. The values $C(i,j)$ for $j > i$ are undefined.

[5] (a) Define a suitable array A to store partial solutions to the problem and state how to find from it the sum of the numbers on an optimal path.

[12] (b) Give a recurrence to compute elements of A, including initialization. Justify your answer.

[3] (c) Show how your algorithm works (i.e., fill the array) for a small example. Your example should be such that an obvious greedy algorithm does not give the correct answer.

[20] 4. **DNA mapping**

You are given a DNA string y of length m over alphabet $\{A, T, C, G\}$, and "genes" strings $x_1, \ldots, x_n$ of lengths $k_1, \ldots, k_n$ respectively, also over the alphabet $\{A, T, C, G\}$ x_i has an associated probability measure $0 < p_i < 1$. The goal is to find the most probable way of concatenating (with repetitions allowed) some strings from $\{x_1, \ldots, x_n\}$ to obtain y, if one exists. The probability of a sequence of strings is a product of the probabilities of individual strings.

For example, take $y = AGAGTC$, $x_1 = TC$, $x_2 = AG$, $x_3 = AGAG$ with probabilities $p_1 = 0.3, p_2 = 0.7, p_3 = 0.4$. Then the two possible mappings of y are $y = x_2 x_2 x_1$ with probability 0.147 and $y = x_3 x_1$ with probability 0.12. Your algorithm should find the first mapping, since it has higher probability.

[7] (a) Give the definition of an array A that you will use to solve the problem. (Hint: use a one-dimensional array and consider prefixes of y of increasing length.) How can you get the probability of the most probable sequence from this array A? How can you check if there is no solution?

[10] (b) Give a recurrence to compute elements of A, including initialization.

[3] (c) Show the filled array for the example above produced by your algorithm.