

CPSC 311: Analysis of Algorithms (Honors)

Exam 2

November 15, 2002

Name: _____

Instructions:

1. This is a closed book exam. Do not use any notes or books, other than your 8.5-by-11 inch review sheet. Do not confer with any other student.
2. Show your work. Partial credit will be given. Grading will be based on correctness, clarity and neatness.
3. I suggest that you read the whole exam before beginning to work any problem. Budget your time wisely—according to the point distribution.
4. There are 4 questions worth a total of 100 points, on 7 pages (including this page).

DO NOT BEGIN THE EXAM UNTIL INSTRUCTED TO DO SO. GOOD LUCK!

1. (25 pts) Given a string $S = s_1s_2 \dots s_k$, which is a sequence of characters, we can modify the string S using the following *edit operations* which produce a new string:

- $\text{insert}(S, i, a)$: insert character a immediately after the i -th character of string S
- $\text{delete}(S, i)$: delete the i -th character of string S
- $\text{replace}(S, i, a)$: replace the i -th character of string S with the character a .

Given two strings $X = x_1x_2 \dots x_n$ and $Y = y_1y_2 \dots y_m$, use dynamic programming to compute the minimum number of edit operations required to transform X into Y .

Let $M[i, j]$ be the minimum number of operations required to transform the prefix $x_1 \dots x_i$ of X into the prefix $y_1 \dots y_j$ of Y .

(a) (4 pts) What are the base cases (the entries in M that are trivial to compute)? *Hint*: Think about what must be done to transform the empty string (no characters) into a non-empty string.

(b) (4 pts) What is the goal (the entry in M that gives the final answer)?

(c) (5 pts) What is the recursive characterization of $M[i, j]$? *Hint*: There are three ways to do the transformation (recursively): (1) delete x_i , then transform $x_1 \dots x_{i-1}$ to $y_1 \dots y_j$; (2) transform $x_1 \dots x_{i-1}$ to $y_1 \dots y_{j-1}$ and then replace x_i with y_j if necessary; (3) transform $x_1 \dots x_i$ to $y_1 \dots y_{j-1}$ and then insert y_j .

(d) (4 pts) What are the dependencies among the subproblems, i.e., which entries in M must already be calculated before calculating $M[i, j]$?

(e) (4 pts) What is a good order in which to calculate the entries of M (to make sure that subproblems are calculated before they have to be used)?

(f) (4 pts) What is the running time of your DP algorithm as a function of n and m (the lengths of the two strings)?

2. (25 pts total) Consider the disjoint sets abstract data type with operations MakeSet, Union and FindSet. Assume the arguments to Union are the representative elements of two sets.

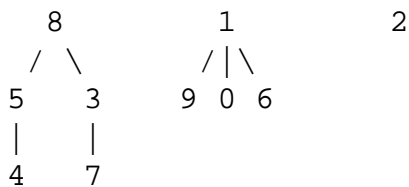
(a) (4 pts) Recall the implementation in which each set is kept as a linked list with a pointer to the representative element. Which operation is faster, Union or FindSet, and why?

(b) (4 pts) Recall the implementation in which each set is kept as a tree with the representative element at the root. Which operation is faster, Union or FindSet, and why?

(c) We will now consider a very efficient (with respect to constant factors) implementation of the tree representation with path compression and union-by-rank.

Suppose the items are non-negative integers $0, 1, 2, \dots$. We'll use an array to record the parent of each item. If an item has no parent, we'll record the height of its tree *if no path compression had been done*. To distinguish the height from a parent reference, we'll record the height h as the negative number $-h - 1$. Think of this as the negative of the number of nodes on the longest path from the root to a leaf. Initially, every item is the root of its own tree, so we set every array element to -1.

(c.1) (3 pts) Draw the array representing this forest, assuming no path compression has been done:



(c.2) (5 pts) Write pseudocode for Union with this representation. Use union by rank — rank is the height that the tree would have if there were no path compression.

(c.3) (5 pts) Write pseudocode for FindSet with this representation. Use path compression.

(d) (4 pts) What is the worst-case running time of a sequence of m disjoint set operations, n of which are MakeSets, using the representation from part (c)?

3. (25 pts total) Let $G = (V, E, W)$ be a connected weighted undirected graph. Let \mathcal{S}_P be the set of all spanning trees of G that contain a particular acyclic set P of k edges of G .

(a) (10 pts) Describe how to modify Kruskal's MST algorithm to find the spanning tree in \mathcal{S}_P with the minimum weight.

(b) (10 pts) Analyze the worst-case running time of your algorithm in terms of $|V|$, $|E|$, and k (the number of edges in P). Assume G is represented by an adjacency list.

(b) (5 pts) Why is your algorithm correct?

4. (25 pts total) Consider the following algorithms for solving the single source shortest path (SSSP) problem on a graph $G = (V, E, w)$, where all the weights are integers between 0 and W :

- (i) Dijkstra's algorithm with the priority queue implemented as a binary heap
- (ii) Dijkstra's algorithm with the priority queue implemented as an array with each array entry a linked list of all nodes with that d -value
- (iii) Bellman-Ford algorithm

(a) (10 pts) Suppose you want to solve the SSSP problem on a graph $G = (V, E)$ where $|E| = \Theta(V \log V)$. Which of the options above is asymptotically fastest and why?

(b) (10 pts) Suppose you want to solve the SSSP problem on a graph $G = (V, E)$ where $|E| = \Theta(V^2)$. Which of the options above is asymptotically fastest and why?

(c) (5 pts) Suppose the graph in part (b) is a complete binary tree. Describe an even more efficient way to solve the problem. What is its running time?