

Homework Assignment #3

CS 410/584 Algorithm Design & Analysis: Spring 2011

This assignment is due Thursday, 21 April, at the beginning of class. You should work alone on this assignment. However, you are free to discuss the problems on the class mailing list. (Or you can send me email questions directly; please put “CS 584” at the beginning of the subject line.) Please put “410” or “584” on your paper, depending on which section you are registered in.

Reading: 17.1-17.3, 21 (all), 22.1-22.3, 22.5, 23 (all).

Note: On any homework exercise where you are asked to give an algorithm, you must also provide an English description of how it works and at least one example execution.

3A (10 points) Recall the example from class of implementing a queue Q with two stacks SI and SO . SI represents the back of Q , and SO the front of Q . An *enqueue* operation on Q is implemented as a *push* on SI , and a *dequeue* operation as a *pop* on SO , moving the contents of SI to SO if necessary.

a. Suppose we also support a “*requeue*(x)” operation on Q that adds element x to the front, by performing *push*(x) on SO . Explain why a sequence of n *enqueue*, *dequeue* and *requeue* operations can still be done with $O(n)$ stack operations.

b. Now suppose we also support an “*unqueue*” operation on Q that removes an element from the back, by performing a *pop* on SI , with a move of the contents of SO to SI if SI is empty. Explain why a series of n *enqueue*, *dequeue*, *requeue* and *unqueue* operations might now take more than $O(n)$ time.

3B (15 points) Propose a data structure to manage a set S of distinct integers that supports two operations:

Insert(x, S): add integer x to set S

Delete-Top-Half(S): remove the largest $\lceil S/2 \rceil$ elements from S

Give algorithms to perform these two operations on the data structure such that m operations can be performed in $O(m)$ time. (Note: It must also be possible to list the elements in S at any time, but there is no constraint on the cost of that operation.)

3C (10 points) Show the UNION-FIND forest that results from the following sequence of operations on elements 1, 2, ..., 12. **Important:** In the case of merging equal-size trees, make the tree with the smaller number at the root the root of the result. Interpret UNION(i, j) as a request to union the sets containing elements i and j .

Do this exercise two ways: without and with path compression.

UNION(1, 5)
UNION(2, 8)
UNION(5, 8)
UNION(3, 4)
UNION(9, 10)
UNION(10, 3)
FIND(8)
UNION(4, 5)
UNION(11, 12)
UNION(8, 12)
FIND(10)
FIND(12)

3D (15 points) An *Euler Circuit* in a directed graph $G = (N, E)$ is a directed cycle that uses each edge in E exactly once (but it can repeat nodes in N). It turns out that G will have an Euler Circuit if and only if G is connected and, for every node n in N , the number of incoming edges to n equals the number of outgoing edges from n . Give an algorithm to construct an Euler Circuit for G (or discover that such a circuit is impossible). Assume the graph is presented as an adjacency list. Analyze the time complexity of your algorithm.

3E (10 points, 584 only) Suppose we have a collection of n comic-book characters with superpowers. We also have a list of r pairs (p, q) where character p and character q are enemies. We would like to label every character either as a “Superhero” or a “Supervillain”, such that every enemy-pair involves one Superhero and one Supervillain. Describe an $O(n + r)$ algorithm to provide such a labeling (or report that none exists). Show two examples, one where there is a solution and one where there is not.