Design and Analysis of Algorithms April 25, 2012
Massachusetts Institute of Technology 6.046J/18.410J
Profs. Dana Moshkovitz and Bruce Tidor Handout 20

Problem Set 8

This problem set is due **at 9:00pm** on **Wednesday, May 2, 2012**.

Both exercises and problems should be solved, but *only the problems* should be turned in. Exercises are intended to help you master the course material. Even though you should not turn in the exercise solutions, you are responsible for material covered by the exercises.

Mark the top of the first page of your solution with your name, the course number, the problem number, your recitation section, the date, and the names of any students with whom you collaborated. The homework template (LaTeX) is available on the course website.

You will often be called upon to "give an algorithm" to solve a certain problem. Your write-up should take the form of a short essay. A topic paragraph should summarize the problem you are solving and what your results are. The body of the essay should provide the following:

1. A description of the algorithm in English and, if helpful, pseudo-code.
2. At least one worked example or diagram to show more precisely how your algorithm works.
3. A proof (or indication) of the correctness of the algorithm.
4. An analysis of the running time of the algorithm.

Remember, your goal is to communicate. Full credit will be given only to correct solutions *that are described clearly*. Convoluted and opaque descriptions will receive lower marks.

Exercise 8-1. Do Exercise 21.2-1 on page 567 of CLRS.

Exercise 8-2. Do Exercise 21.2-5 on page 568 of CLRS.

Exercise 8-3. Do Exercise 21.3-5 on page 572 of CLRS.

Exercise 8-4. Do Exercise 21.4-2 on page 581 of CLRS.

Exercise 8-5. Do Exercise 34.2-5 on page 1066 of CLRS.

Exercise 8-6. Do Exercise 34.3-2 on page 1077 of CLRS.

Exercise 8-7. Do Exercise 34.5-1 on page 1100 of CLRS.

Problem 8-1. Improving Union-by-Rank

Recall the Forest of Trees solution to the Union-Find data structure presented in class. Consider the version with the Union-by-Rank but not the Path Compression improvement. Let n be the total number of elements, and let m be the total number of operations. We gave run time bounds for the following operations:

- MAKE-SET(u): $\Theta(1)$
- WEAK-UNION(u, v): $\Theta(1)$ (where u, v are the roots of two trees)
- FIND-SET(u): $\Theta(\lg n)$
- UNION(u, v) = WEAK-UNION(FIND-SET(u), FIND-SET(v)): $\Theta(1) + 2\Theta(\lg n) = \Theta(\lg n)$

Thus, using this data structure, a series of m operations will take $O(m \lg n)$ time.

Notice that currently, WEAK-UNION is fast ($\Theta(1)$), while FIND-SET is slow ($\Theta(\lg n)$). If we can somehow make FIND-SET slightly faster, at the expense of making WEAK-UNION slightly slower, then the run time of UNION will be slightly faster, so we can do m operations in $o(m \lg n)$ time. This improvement can be achieved without using Path Compression at all. However, regular Union-by-Rank with Path Compression is still the best.

(a) We will develop a data structure over n elements that supports MAKE-SET in $\Theta(1)$, WEAK-UNION in $O(k)$, and FIND-SET in time $O(\log_k n)$, for any integer k.

To lead you off, you will still need a forest of trees; however, the trees in this forest should satisfy certain special properties:

1. Unlike trees in regular Union-by-Rank, the trees in our forest only store elements on its leaves. The root and all the internal nodes (nodes that are neither the root nor a leaf) don't represent elements in the data structure. The root stores the name of the representative element.
2. Let $height(x)$ of a node x be the length of the longest path from x to any leaf below x (ex. height of a leaf is 0). We require that all roots in the forest have height at least 1.
3. We also require that if a root has height $h > 1$, then it must have at least two children. Further, we will ensure that each child of the root must have height $h - 1$.
4. Finally, each internal node with height h must have at least k children whose heights are exactly $h - 1$. It may have any number of children with height less than $h - 1$. Consequently, each internal node will have at least k children, but possibly more.

Develop a data structure containing special trees satisfying the above constraints that supports the aforementioned operations, perhaps augmented with additional data on the nodes. Analyze how each of the operations is performed to ensure the properties of the trees are always satisfied, and show that they have the desired run time.

(b) Explain how you can use the data structure in (a) to achieve $o(m \lg n)$ run time on m total operations.

Problem 8-2. Polynomial or NP-Hard?

Suppose that there are n_1 classes you are thinking of taking. You have a list of which of your n_2 friends are taking which classes. To figure out your schedule for the semester, you've decided to learn as much as you can about all of the classes. You can learn something about a class in two ways: (1) attend the class yourself, or (2) ask a friend who is taking the class what it's like. Attending a class takes one hour. Speaking to one friend also takes one hour, but during that hour you can quiz your friend about *all* of the classes they are taking.

(a) For each available class, you would like to either attend the class in question, or speak to *all* of your friends in the class to get their personal opinion. You only have k hours to devote to this process. Either devise a polynomial-time algorithm for picking out k friends and classes that satisfy these constraints, or show that the problem is NP-hard.

(b) To make your life easier, you've decided that you can learn enough about a class by either attending the class, or talking to *at least one* of your friends in the class. Again, you only have k hours to devote to this process. Either devise a polynomial-time algorithm for picking out k friends and classes that satisfy these constraints, or show that the problem is NP-hard.

6.046J / 18.410J Design and Analysis of Algorithms
Spring 2012