Problem Set 7

This problem set is due **in lecture** on **Wednesday, November 27.**

Reading: §30.1–2; Chapter 17; §22.1–2; pp. 580-587; §24.3

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 each sheet with your name, the course number, the problem number, your recitation section, the date, and the names of any students with whom you collaborated.

Each problem should be done on a separate sheet (or sheets) of three-hole punched paper.

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 your essay should provide the following:

1. A description of the algorithm in English and, if helpful, pseudocode.
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. Graders will be instructed to take off points for convoluted and obtuse descriptions.

Exercise 7-1. Do Exercise 30.2-5 on page 838 of CLRS.

Exercise 7-2. Do Exercise 17.1-3 on page 409 of CLRS.

Exercise 7-3. Do Exercise 17.3-3 on page 416 of CLRS.

Exercise 7-4. Do Exercise 22.1-3 on page 530 of CLRS.

Exercise 7-5. Do Exercise 22.2-6 on page 539 of CLRS.

Exercise 7-6. Do Exercise 24.3-2 on page 600 of CLRS.

Problem 7-1. A ***maximum spanning tree*** of an undirected graph is a spanning tree whose edges have the maximum total weight (taken over all spanning trees). Give an algorithm that, on an undirected graph $G = (V, E)$, finds a maximum spanning tree of G.

Problem 7-2. A ***Toeplitz matrix*** is an $n \times n$ matrix $A = (a_{ij})$ such that $a_{ij} = a_{i-1,j-1}$ for $i = 2, 3, \ldots, n$ and $j = 2, 3, \ldots, n$. The following is an example of a Toeplitz matrix:

$$\begin{pmatrix} 5 & 7 & 9 & 11 \\ 8 & 5 & 7 & 9 \\ 4 & 8 & 5 & 7 \\ 6 & 4 & 8 & 5 \end{pmatrix}$$

(a) Is the sum of two Toeplitz matrices necessarily Toeplitz? What about the product? Prove your answer.

(b) Describe how to represent a Toeplitz matrix so that two $n \times n$ Toeplitz matrices can be added in $O(n)$ time.

(c) Give an $O(n \lg n)$-time algorithm for multiplying an $n \times n$ Toeplitz matrix by a vector of length n. Use your representation from part **(b)**. (*Hint:* Use convolution.)

Problem 7-3. Suppose we had code lying around that implemented a stack, and we now wanted to implement a queue. One way to do this is to use two stacks S_1 and S_2. To insert into our queue, we push into stack S_1. To remove from our queue we first check if S_2 is empty, and if so, we "dump" S_1 into S_2 (that is, we pop each element from S_1 and push it immediately onto S_2). Then we pop from S_2.

For instance, if we first insert a and then insert b, S_1 will look like $[b\ a]$ (where the top of stack is to the left). Then to remove from the queue, we dump S_1 into S_2 so that S_1 is empty and S_2 looks like $[a\ b]$; then we pop the a from S_2.

Suppose each push and pop costs 1 unit of work, so that performing a dump when S_1 has n elements costs $2n$ units (since we do n pushes and n pops).

(a) Suppose that (starting from an empty queue) we do 3 insertions, then 2 removals, then 3 more insertions, and then 2 more removals. What is the total cost of these 10 operations, and how many elements are in each stack at the end?

(b) If a total of n insertions and n removals are done in some order, how large might the running time of one of the operations be (give an exact, non-asymptotic answer)? Give a sequence of operations that induces this behavior, and indicate which operation has the running time you specified.

(c) Suppose we perform an arbitrary sequence of insertions and removals, starting from an empty queue. What is the amortized cost of each operation? Give as tight (i.e., non-asymptotic) of an upper bound as you can. Prove your answer using any amortization technique you like.

Problem 7-4. In this problem, we consider special cases of directed graphs with nonnegative edge weights satisfying the triangle inequality, for which the single-source shortest-paths problem can be solved faster.

(a) Suppose that all edge weights are integers between 0 and C for some constant C. Give an $O(V+E)$-time algorithm to solve the single-source shortest-paths problem in such a graph.

(b) Suppose that all edge weights are integers between 0 and $u-1$, where $u > |V|$. Give an $O((V+E)\lg\lg u)$-time algorithm to solve the single-source shortest-paths problem in such a graph.

An interesting property of Dijkstra's algorithm is that it does not require a general priority-queue data structure; instead, it merely requires what is called a ***monotone priority queue***. A monotone priority queue supports the following operations on a dynamic set S:

1. DELETE-MIN: Delete and return the current minimum value in S.
2. INSERT(x): Insert an element x. However, for the operation to be allowed, x must be larger than or equal to the last value returned by DELETE-MIN.

Thus, in a monotone priority queue, the values returned by DELETE-MIN operations are monotonically increasing over any valid sequence of calls to INSERT and DELETE-MIN.

(c) Design a monotone priority queue that maintains a dynamic set of elements from the universe $\{0,\ldots,u-1\}$. INSERT should run in $O(1)$ time. The total time spent by DELETE-MIN over a sequence of k operations should be $O(u+k)$.

(d) Consider a graph in which all shortest paths have integer weights between 0 and $u-1$. Describe how to use your monotone priority queue in Dijkstra's algorithm, and analyze the running time of the resulting algorithm for single-source shortest paths. State your bound in terms of $|V|$, $|E|$, and u.