Problem Set 6

This problem set is due **in recitation** on **Friday, November 14.**

Reading: Chapters §17.1-17.4, §22.1-22.5, §23.1-23.2, §24.1-24.5.

There are **four** problems and **one** exercise. You should only hand in **Problems 6.1 through 6.4**. The first exercise is for review purposes and will not be graded. Each problem is to be done on a separate sheet (or sheets) of three-hole punched paper. 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.

Exercise 6-1. Graphs: Terminology and Definitions
Note: This problem is not to be handed in.

A *graph* $G = (V, E)$ is a set of *vertices*, V, and *edges*, $E \subseteq V \times V$. In this problem, we will review some basic properties of graphs. For each question, answer True or False or fill in the blank where indicated. Justify your answers. (It may be useful to refer to Section B.4 in CLRS.)

(a) An undirected acyclic graph contains $\Theta(n)$ edges.

(b) A directed acyclic graph contains $\Theta(n)$ edges.

(c) A *dense* graph contains ________ edges.

(d) Let $G = (V, E)$ be an undirected graph. If $\max_{v \in V} \delta(v) = O(n)$, then G is dense.

(e) If a graph is *sparse*, then it is a tree.

(f) For every directed graph, there is a unique *topological sort*.

(g) A graph is connected iff it contains a cycle.

(h) Every undirected graph has a minimum spanning tree.

(i) A directed acyclic graph is not *strongly connected*.

(j) A directed acyclic graph contains a vertex v such that $\delta_{IN}(v) = 0$.

(k) If there is a directed path from i to j in a directed graph, then vertices i and j form a *connected component*.

(l) Suppose we have a directed graph $G = (V, E)$ that is strongly connected. For any depth-first search of G, if all the forward edges of G (with respect to the depth-first forest) are removed from G, the resulting graph is still strongly connected.

Problem 6-1. Cycles

(a) Give an $O(E^2 \lg V)$ algorithm to find a minimum weight cycle in a given weighted, undirected, connected graph in which all edge weights are non-negative. Prove your algorithm is correct and analyze its running time. (**Optional Extra Credit Problem:** Can you find a more efficient algorithm for this problem?)

(b) Give an $O(VE)$ algorithm to determine if a given directed, connected graph contains an edge that is in every cycle. Prove your algorithm is correct and analyze its running time. (**Optional Extra Credit Problem:** Can you find a more efficient algorithm for this problem?)

(c) Suppose that a weighted, directed graph $G = (V, E)$ has a negative-weight cycle. Give an efficient algorithm to list the vertices of one such cycle. Prove your algorithm is correct and analyze its running time.

Problem 6-2. Minimum Spanning Trees

(a) Consider the following algorithm, NEW-MST, for computing a minimum spanning tree. The algorithm takes as input an undirected, weighted, connected graph G. It sorts the edges in non-increasing order according to edge weight. It then goes through each edge in G, in the sorted order, and determines if removing that edge disconnects the graph. If not, then the edge is removed permanently, otherwise the edge remains.

Below, we provide pseudocode for the NEW-MST algorithm.

NEW-MST(G)
Sort edges according to edge weight in non-increasing order: $e_1, e_2, \ldots, e_m$
$i \leftarrow 1$
While $i \leq m$:
    If $G \setminus e_i$ is connected
        $G \leftarrow G \setminus e_i$
    $i \leftarrow i + 1$
Output G.

Prove that the NEW-MST algorithm outputs a minimum spanning tree of the input graph G.

(b) Give an efficient algorithm to find a spanning tree for a connected, weighted, undirected graph G such that the weight of the maximum-weight edge in the spanning tree is minimized. Prove your algorithm is correct.

Problem 6-3. Labeled Graphs and Longest Paths

You are given a directed graph $G = (V, E)$, in which each vertex has a unique label $\ell(v) : V \rightarrow Z^+$, a cost function $w : E \rightarrow \mathcal{R}$ assigning a weight to each edge, and a source $s \in V$. Additionally, every edge (i, j) has the property that $\ell(i) < \ell(j)$. Design an algorithm to construct an output array D such that $D[i]$ is the length of the longest path from s to v_i in G.

Problem 6-4. Amortized Weight-Balanced Trees

Consider an ordinary binary search tree augmented by adding to each node x the field $size[x]$ giving the number of keys stored in the subtree rooted at x. Let α be a constant in the range $\frac{1}{2} \leq \alpha < 1$. We say that a given node x is *α-balanced* if

$$size[left[x]] \leq \alpha \cdot size[x]$$

and

$$size[right[x]] \leq \alpha \cdot size[x].$$

The tree as a whole is *α-balanced* if every node in the tree is α-balanced. We will study the following amortized approach to maintaining weight-balanced trees.

(a) Given a node x in an arbitrary binary search tree, show how to rebuild the subtree rooted at x so that it becomes $\frac{1}{2}$-balanced. You algorithm should run in time $\Theta(size[x])$ time and can use $O(size[x])$ auxiliary storage.

(b) Show that performing a search in an n-node α-balanced binary search tree takes $O(\lg n)$ worst-case time.

For the remainder of this problem, assume that the constant α is strictly greater than $\frac{1}{2}$. Suppose that INSERT and DELETE are implemented as usual for an n-node binary search tree, except that after every such operation, if any node in the tree is no longer α-balanced, then the subtree rooted at the highest such node in the tree is "rebuilt" so that it becomes $\frac{1}{2}$-balanced.

We shall analyze this rebuilding scheme using the potential method. For a node x in a binary search tree T, we define

$$\Delta(x) = |size[left[x]] - size[right[x]]|,$$

and we define the potential of T as

$$\Phi(T) = \sum_{x \in T : \Delta(x) \geq 2} c \cdot \Delta(x),$$

where c is a sufficiently large constant that depends on α.

(c) Argue that any binary search tree has nonnegative potential and that a $\frac{1}{2}$-balanced tree has potential 0.

(d) Suppose that m units of potential can pay for rebuilding an m-node subtree. How large must c be in terms of α in order for it to take $O(1)$ amortize time to rebuild a subtree that is not α-balanced.

(e) Show that inserting a node into or deleting a node from an n-node α-balanced tree costs $O(\log n)$ amortized time.