Problem Set 2 for CS 170

Note

When asked for an algorithm you must give (1) a brief informal description of the algorithm, (2) a precise description using pseudo-code, (3) an informal argument for termination and correctness of the algorithm, and (4) an analysis of the running time of the algorithm. Be clear about what the input to the algorithm is, how you measure the size of the input, and what constitutes a "step" in your running-time analysis.

Problem 0. [Any questions?] (5 points)

What's the one thing you'd most like to see explained better in lecture or discussion sections? A one-line answer would be appreciated.

(Sometimes we botch the description of some concept, leaving people confused. Sometimes we omit things people would like to hear about. Sometimes the book is very confusing on some point. Here's your chance to tell us what those things were.)

Problem 1. [Graph Property] (10 points)

Is it possible to construct an undirected graph that has an odd number of vertices and where each vertex has an odd degree? If so, show an example. If not, show the reason. (Graphs are not allowed to have self-loops or multiple edges between the same pair of vertices.)

Problem 2. [Cycle Detection] (15 points)

Design linear time algorithms for the following problems:

(a) Determine whether a directed graph $G = (V, E)$ has a cycle. Your algorithm should run in $O(|V| + |E|)$ time.

(b) Determine whether an undirected graph $G = (V, E)$ has a cycle. Your algorithm should run in $O(|V|)$ time, independent of $|E|$.

Problem 3. [Graph Traversal] (15 points)

Let G be a strongly connected graph with the property that (u, v) is an edge iff (v, u) is also an edge. Does there always exist a path that traverses each edge of G exactly once? If so, design an algorithm for finding such a path; otherwise, show why not.

Problem 4. [Tree Node Labeling] (15 points)

Let T be a tree. Each vertex v of T has a non-negative integer label $l(v)$ that is no greater than the depth of v. For each vertex v, let $p(v)$ be the parent of v (the parent of the root vertex is itself). Define the k-th ancestor $p^k(v)$ of v to be $p^{k-1}(p(v))$ for $k > 1$ and define $p^1(v) = p(v)$, $p^0(v) = v$. Design a linear time algorithm for computing $x(v) = l(p^{l(v)}(v))$ for each vertex v of T.

Problem 5. [Integer Range] (20 points)

There are k integers in the range from 1 to n. Design a data structure and two algorithms: the first algorithm preprocesses the k integers in $O(n+k)$ time and stores them into the data structure to facilitate the second algorithm; the second algorithm uses the data structure to answer questions like "How many of the k integers fall in the range $[a, \ldots, b]$?" in $O(1)$ time.

Problem 6. [Peace for Grudgeville] (20 points)

In Grudgeville, people just can't seem to forgive. Every so often two people get into an argument, and then they hate each other forever after. Hating is a symmetric relation: If Alice hates Bob, then Bob hates Alice, too. People who hate each other can't stand to be in each other's presence. This has put a crimp on the social scene in Grudgeville, and Sheriff Brown is tired of stopping the fights that periodically break out.

Then Sheriff Brown has a flash of insight. There's an island offshore. If he could figure out a way to divide the n townsgoers into two groups, so no one hates anyone else in their group, then he could ship one group out to the island, leave the other behind on the mainland, and everyone would be happy. Can you help keep the peace?

Your job is to build an efficient algorithm to check whether such a division is possible. You're given a list of p pairs of people who hate each other. Your algorithm should check whether such a division into two groups is possible, and if so, it should output a roster stating for each person where they will live (on the mainland or on the island).

List your algorithm, explain informally why it is correct, and state its running time.