CSE 5311
Spring 2007
Design and Analysis of Algorithms
Exercise Problems for Discussions

1. There is a network of n mobile devices carried by human beings who move around. The moving devices can be defined by a graph at any point as follows: a node represents each of n devices and there is an edge between devices i and j if the locations of i and j are no more than 500 meters apart. Decide whether the following claim can ensure that all devices are connected all the time.

   Claim: Let *G* be a graph on n nodes, where n is an even number. If every node of *G* has degree at least *n*/2, then *G* is connected.

   [Ex. 7, Ch. 3]

2. Suppose that an n-node undirected graph $G = (V,E)$ contains two nodes s and t such that the distance between s and t is strictly greater than $n/2$. Show that there must exist some node v, not equal to either s or t, such that deleting v from G destroys all s-t paths. That is to say that, the graph obtained from G by deleting v contains no path from s to t. Give an algorithm with running time $O(m+n)$ to find such a node v.

   [Ex. 9, Ch. 3]

3. The manager of a large student union on campus comes to you with the following problem. She's in charge of a group of students, each of whom is scheduled to work one shift during the week. There are different jobs associated with these shifts , but we can view each shift as a single contiguous interval of time. There can be multiple shifts going on at once. The manager is trying to find a subset of these students to form a supervising committee that she can meet with once a week. She considers such a committee to be complete if, for every student not on the committee, the student shift overlaps (at least partially) the shift of some student who is on the committee. In this way, each student's performance can be observed by at least one person who is serving on the committee.

   Give an efficient algorithm that takes the schedule of n shifts and produces a complete supervising committee containing as few students as possible.

   [Ex. 15. Ch. 4]

4. You have a processor that can operate 24 hours a day, every day. People submit requests to run daily jobs on the processor. Each job has a *start time* and a *end time*; if a job is accepted to run on the processor it must run continuously, every day, for the period between its *start* and *end* times.

   Given a list of n jobs, your goal is to accept as many jobs as possible (regardless of their length), subject to the constraint that the processor can run at most one job at any given time. Provide an algorithm to do this with a running time that is polynomial in n. Assume that no two jobs have the same *start* and *end* times.

   [Ex. 17, Ch. 4]

5. Suppose you're consulting for a company that manufactures PC equipment and ships it to distribute all over the country. For each of the next n weeks, they have a projected supply s_i of equipment (measured in pounds), which has to be shipped by an air freight carrier.

   Each week's supply can be carried by one of two air freight companies, A or B.

   - Company A charges a fixed rate r per pound (so it costs $r \times s_i$ to ship the week's supply s_i).
   - Company B makes contracts for a fixed amount c per week, independent of the weight. However, contracts with company B must be made in blocks of four consecutive weeks at a time.

   A schedule, for the PC Company, is a choice of air freight company (A or B) for each of the n weeks, with the restriction that company B, whenever chosen, must be chosen for blocks of four contiguous weeks at a time. The cost of the schedule is the total amount paid to company A and B, according to the description above.

   Give a polynomial-time algorithm that takes a sequence of supply values $s_1, s_2, \ldots s_n$ and returns a schedule of minimum cost.
   [Ex. 11, Ch. 6]

6. Suppose we wish to replicate a file over a collection of n servers, labeled $S_1, S_2, \ldots S_n$. To place a copy of the file at server S_i, results in a placement cost of c_i for an integer $c_i > 0$.

   Now, if a user requests the file from a server S_i, and no copy of the file is present at S_i, then the servers $S_{i+1}, S_{i+2}, S_{i+3} \ldots$ are searched in order until a copy of the file is finally found, say at server S_j where, $j > i$. This results in access cost of j-i(note that lower-indexed servers are not consulted in this search.) The access cost is 0 if server S_i holds a copy of the file. We will require that a copy of the file be placed at server S_n, so that all such searches will terminate, at least, at S_n.

   We'd like to place copies of files at servers so as to minimize the sum of placement and access costs. Formally, we say that a configuration is a choice for each server S_i with $i = 1, 2, \ldots n$-1, of whether to place a copy of the file at S_i or not (Recall that a copy is always placed at S_n.) The total cost of the configuration is the sum of all placement costs for servers with a copy of the file, plus the sum of all access costs associated with all n servers.

   Give a polynomial time algorithm to find a configuration of minimum total cost.
   [Ex. 12, Ch. 6]

7. We define the *Escape* problem as follows. We are given a directed graph $G = (V,E)$ (picture a network of roads). A certain collection of nodes $X \subset V$ are designated as populated nodes and a certain other collection $S \subset V$ are designated as safe nodes (Assume that X and S are disjoint). In case of emergency, we want evacuation routes from populated nodes to safe nodes. A set of evacuation routes is defined as a set of paths in G so that (i) each node in X is the tail of one path, (ii) the last node on each path lies in S, and (iii) the paths do not share any edges. Such a set of paths gives a way for occupants of the populated nodes to escape to S, without overly congesting any edge in G.

a. Given *G, X,* and *S*, show how to decide in polynomial time whether such a set of evacuation routes exists.
b. Suppose we change (iii) to "the paths do not share any nodes", show show how to decide in polynomial time whether such a set of evacuation routes exists, given G, X and S.

[Ex. 14, Ch. 7]

8. Consider a set $A = \{a_1, a_2, \ldots a_n\}$ and a collection $B_1, B_2, \ldots B_m$ of subsets of A (i.e., $B_i \subseteq A$ for each i).

   We say that set $H \subseteq A$ is a *hitting set* for the collection $B_1, B_2, \ldots B_m$ if H contains at least one element from each B_i – that is if $H \cap B_i$ is not empty for each i (so H hits all the sets B_i).

   We define the hitting set problem as follows. We are given a set $A = \{a_1, a_2, \ldots a_n\}$, a collection $B_1, B_2, \ldots B_m$ of subsets of A, and a number k. We are asked: Is there a hitting set $H \subseteq A$ for $B_1, B_2, \ldots B_m$ so that the size of H is at most k?

   Prove that the Hitting Set is NP-Complete.

   [Ex. 5, Ch. 8]

9. Suppose you are given a set of positive integers $A = \{a_1, a_2, \ldots a_n\}$ and a positive integer B. A subset $S \subseteq A$ is called a feasible if the sum of the numbers in S does not exceed B: $\sum_{a_i \in S} a_i \leq B$. The sum of the numbers in S will be called the total sum of S. You would like to select a feasible subset S of A whose total sum is as large as possible.

   [Ex. 3, Ch. 11]

10. A dynamic programming solution was given for the 0-1 Knapsack problem. The algorithm runs in $O(nW)$, where n is the number of number of items and W is the size of the knapsack. It was assumed that the weights are integers and that W is small. How do you solve this problem if W is large and if the weights are not integers.

    [Pages. 644-349, Ch. 11]