

Lab04- Greedy Strategy

Exercises for Algorithms by Xiaofeng Gao, 2016 Spring Semester

Name:_____ Student ID:_____ Email: _____

1. Consider the problem of making change for n cents using the fewest number of coins. Assume that each coin's value is an integer.
 - (a) Describe a greedy algorithm to make change consisting of quarters, dimes, nickels, and pennies. Prove that your algorithm yields an optimal solution.
 - (b) Suppose that the available coins are in the denominations that are powers of c , i.e., the denominations are c^0, c^1, \dots, c^k for some integers $c > 1$ and $k \geq 1$. Show that the greedy algorithm always yields an optimal solution.
 - (c) Give an $O(nk)$ -time algorithm that makes change for any set of k different coin denominations, assuming that one of the coins is a penny.

2. Describe an efficient algorithm that, given a set $\{x_1, x_2, \dots, x_n\}$ of points on the real line, determines the smallest set of unit-length closed intervals that contains all of the given points. Argue that your algorithm is correct.

3. Alice wants to throw a party and is deciding whom to call. She has n people to choose from, and she has made up a list of which pairs of these people know each other. She wants to pick as many people as possible, subject to two constraints: at the party, each person should have at least five other people whom they know and five other people whom they don't know.

Give an efficient algorithm that takes as input the list of n people and the list of pairs who know each other and outputs the best choice of party invitees. Give the running time in terms of n .

4. Consider the following variation on the Interval Scheduling Problem. You have a processor that can operate 24 hours a day, every day. People submit requests to run daily jobs on the processor. Each such job comes with a start time and an end time; if the job is accepted to run on the processor, it must run continuously, every day, for the period between its start and end times. (Note that certain jobs can begin before midnight and end after midnight; this makes for a type of situation different from what we saw in the Interval Scheduling Problem.) Given a list of n such 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 point in time. Provide an algorithm to do this with a running time that is polynomial in n . You may assume for simplicity that no two jobs have the same start or end times.

Example. Consider the following four jobs, specified by (start-time, end-time) pairs.

(6:00 P.M., 6:00 A.M.), (9:00 P.M., 4:00 A.M.),
(3:00 A.M., 2:00 P.M.), (1:00 P.M., 7:00 P.M.).

The optimal solution would be to pick the two jobs (9:00 P.M., 4:00 A.M.) and (1:00 P.M., 7:00 P.M.), which can be scheduled without overlapping.