Christopher Merrill

CS325

Homework 4

**Problem 1**

To determine which class should use which lecture hall at a given time, the class cj with the earliest finish time should always be chosen. By always choosing the class with the earliest finish time, the lecture halls will be left with the maximum available time. If class c1 is determined to have the earliest finish time, then the only remaining subproblem is finding classes that start after c1 finishes.

Assuming that the activities have already been sorted by ascending finish times, this algorithm would only have to examine each class exactly one time. That means that the runtime would be ϴ(n).

**Problem 2**

A greedy algorithm to solve the “Road Trip” problem would involve always choosing the hotel whose distance from your current location is closest to d. In other words, drive to the furthest possible hotel each day until you reach your destination.

If c is your current location, d is the furthest distance you can drive in a day, and h is the hotel you should choose to stay in on a particular day, you would choose your hotel using the following:

where xi is the distance to each hotel you are evaluating.

Using this strategy, you would only have to evaluate each hotel once. From your starting location each day, you increment through the remaining hotels until the distance surpasses d. You would then add the previous hotel to an array of hotels to stay at.

Since you are only examining each hotel once, the runtime would be ϴ(n).

**Problem 3**

A greedy algorithm for the “Scheduling Jobs with Penalties” problem would begin with ordering the jobs from highest to lowest penalty (they will be added to the schedule in this order). Each job will be added using the following criteria:

If a time interval exists, between 1 and the deadline dj, then schedule job j in the last such available interval (avoiding the penalty). Otherwise the penalty will be paid for that job j regardless, so in this case place the job in the latest available time slot. This will leave the most room for remaining jobs that may be able to avoid their penalty.

Pseudocode:

JOB-SCHEDULING-WITH-PENALTIES (A[1…n], n)

schedule = S[1…n]

FOR all jobs j from 1 to n

timeSlot = LATEST-AVAILABLE (S, 1, dj)

IF timeSlot = 0

timeslot = LATEST-AVAILABLE (S, 1, n)

S[timeSlot] = j

RETURN S

LATEST-AVAILABLE (S[1…n], start, end)

FOR i from end to start

IF S[i] = null

RETURN i

RETURN 0

In JOB-SCHEDULING-WITH-PENALTIES we pass an array A of jobs sorted in descending order by penalty and the number of jobs n. It returns an array S that represents the schedule containing the jobs from A. LATEST-AVAILABLE returns an index for the first empty slot in a schedule, between the start and end indexes. If no slot is available, LATEST-AVAILABLE returns 0.

JOB-SCHEDULING-WITH-PENALTIES looks at each job and first checks if there is a LATEST-AVAILABLE slot between 1 and the deadline dj. If not, it sets the timeSlot to be the LATEST-AVAILABLE in the schedule S.

We examine each job once, and for each job we have to examine each time slot at most one time to determine that jobs placement in the schedule. Thus, the runtime would be ϴ(n2).

**Problem 4**

This is essentially the same as the problem described in Problem 1, only we are building the schedule in reverse. By choosing the activity with the latest start time and placing it first, we are still left with the maximum available time in the lecture halls. If class c1 is determined to have the latest start time, then the only remaining subproblem is finding classes that finish before c1 starts.

Assuming that the activities have already been sorted by descending start times, this algorithm would only have to examine each class exactly one time. That means that the runtime would still be ϴ(n). (Which is equal to the optimal solution found in Problem 1 where we selected activities by the first to finish)

Problem 5