Benjamin Fondell

Homework 4

1.

Pre-condition: Sort the classes by earliest finish time in ascending order.

1. Select the class from the sorted list with earliest finish time.

2. Eliminate classes in sorted list that would not suit this schedule. Add eliminated classes to sub-list.

3. Repeat the selection from step one on remaining subset of possible classes.

4. If the lecture hall becomes filled for a specified time window proceed to filling the next lecture hall in list of lecture halls.

5. Repeat steps one through three on the sub-list of conflicting classes. Placing into further lecture halls as needed.

The important part of this algorithm is the sorting of the classes by earliest finish time. Equally important is the greedy choice of choosing by the class with the earliest possible finish time in order to maximize the available remaining time of a lecture hall. This allows the greatest number of classes to be scheduled in the least number of lecture halls.

The runtime for this algorithm would be bounded above by O(nlogn). This is because the precondition of the algorithm requires the list of classes to be sorted. This sorting can be done in O(nlogn) time. The remaining algorithm is bounded only by the number of classes passed to the program so the remaining runtime can be O(n). Because O(nlogn) dominates the O(n) time the runtime for this greedy scheduling algorithm would be O(nlogn).

2.

Pre-Condition: List of hotels is in ascending sorted order.

Base Case

1. Create variables that hold: a list of hotels stayed in, distance traveled, the current hotel, max daily travel distance, and days traveled.

2. Reset daily travel distance. Start loop from cur hotel. (If first loop current hotel = 0)

Outer loop:

2. Compare the current hotel to the end of the list of sorted hotels, If it is at the end of that list then the algorithm is complete. If not keep passing through the algorithm.

Inner Loop:

3. Consider the first hotel in the list, if the distance traveled + the daily travel maximum is greater than the distance of that hotel, then set that hotel to the current hotel and decrease the daily travel distance available by the distance to that hotel - distance traveled, and proceed to check the next hotel.

4. If the hotel is the last hotel in the list of available hotels or no hotel travel distance fits in the remaining available daily travel distance, break the inner loop:

5. Remove the current hotel from the list of and append to the list of hotels stayed in.

6. Continue the loops until the current hotel is the final hotel in the list. This will reach the destination in the least amount of time by maximizing the travel distance per day of driving.

Assuming that I read the problem correctly and the list of hotels is supposed to be sorted in ascending order then…. Theoretically the runtime of this algorithm would be bounded by the worst case of O(n^2) this is because two loops are used to pass over the same values. The bound is at worst O(n^2) but its actual runtime would likely be faster this is because it often does not need pass through to the end of the second loop. This runtime is true for most algorithms where it is necessary to try all items to find the optimal item then loop through again to try the remaining items.

3.

1. Sort an array of jobs in descending order by their values of penalties.

2. Outer loop passes through sorted array of jobs considering each job in the sorted sequence.

3. Inner loop compares the job considered by the outer loop and finds the latest possible position in a ‘schedule” array by looking at the deadlines of jobs as it loops through the remaining values. It then places the job at the latest possible position within the array which represents the schedule. If it cannot find an unoccupied position in the array of scheduled jobs in a place before the job’s deadline then it will not be scheduled.

The important parts of this algorithm are in the sorting of the list of jobs in descending order and the comparison of jobs in the inner loop to find the latest possible scheduling time in order to maximize jobs scheduled and minimize penalty. The algorithm would be bounded above by O(n^2). In the worst case the algorithm loops through all values once and each value again in an inner loop for this reason we can say that the algorithm is bounded by O(n^2). There is also a need to sort the array of values which will process in O(nlogn) time but the O(n^2) runtime of the nested loop dominates this logarithmic sorting procedure.