Josh Bell

CS – 325

May 8th, 2021

Homework 3

GitHub Repo: https://github.com/JoshBell302/CS325-AoA-Assignment-3

Problem 1: Road Trip

a) **Design an efficient greedy algorithm to determine which hotels you should stay in if you want to minimize the number of days it takes you to get to your destination. Provide a description and pseudocode.**
This idea can be broken into steps . . .
1. Every day, we should loop through the hotels that are after from the hotel that was stayed at on the night previous
2. When looping, if a hotel's distance is found to be farther than the distance for the day then the previous hotel from the found one is chosen.
3. Repeat Steps 1 and 2 until the final hotel is reached.

b) **What is the theoretical running time of your algorithm?**
The theoretical running time is easy to calculate, the first thing to look at is the worst case which is if the distance continues to be in multiples. This would cause the calculation to calculate the distance of each hotels twice. With the worse time calculated twice the in would be O(2n) => O(n), this is a linear running time.

Problem 2: CLRS 16-1-2 Activity Selection Last-to-Start Greedy Criteria

To solve the activity-selection problem (16.1 CLRS), suppose that instead of always selecting the first activity to finish, we select the last activity to start that is compatible will all previously selected activities. Describe how this approach is a greedy algorithm and prove that it yields an optimal solution.

It is an optimal solution, even though it is last-to-start we can imagine it as the original problem just going in reverse which is going to produce the same results as the original. It is greedy due to choosing the best-looking option in every step.

Problem 3: Activity Selection Last-to-Start Implementation

You may use any language you choose to implement the activity selection last-to-start algorithm described in problem 2. Name your program activity . where ** = ccp, c or py. If you use a sorting function in your program you will need to write that yourself. You can use any of the sorting algorithms from previous homework assignments or implement**

another sorting algorithm. If you sort the activities include the time to sort in your theoretical running time. The program should read input from a file named "act.txt". The file contains lists of activity sets with number of activities in the set in the first line followed by lines containing the activity number, start time & finish time.

In the write up submitted in Canvas include a description of your algorithm, pseudocode, and analysis of the theoretical running time. You do not need to collect experimental running times.

The code is in the GitHub Repo found above. Output from terminal with the act.txt . . .

```
PS C:\Users\joshb\Documents\GitHub\
Set 1
Maximum number of activities = 4
2 4 8 11

Set 2
Maximum number of activities = 2
2 1

Set 3
Maximum number of activities = 3
2 4 6
```

Description: The code first reads all of the data from "act.txt" and stores into a 3d array. Once all the data is stored, we run through each set by looping to find the minimum amount of total time for all the activities in that set. Once it is found we then loop multiple times, using a while loop, to add activities to the max number of activities. If no more are found with the minimum value and with a starting time equal to or greater than the current end time, then we increase the minimum time required. Once we do this a few times and find that no more are being added we exit the loop and print the found data to the terminal.

Pseudocode: This is broken into steps …

1. Collect all data from act.txt
2. Loop each set
3. Find the set's minimum time by subtracting the end time from the start time
4. Once found loop through the activity to find activities which have total time equal to the current minimum value and have their start time greater than or less than to the current end time, if found add it to an array of activities.
5. If a loop is found to not increase the list of max activities, then increase the minimum time in an attempt to find more activities, if this fails when increasing the minimum three times then exit the loop.
6. Print current sets information.
7. Repeat steps 3 – 6 for each set until complete for all sets.

Theoretical Running Time: Looking at the actual sorting process and not when reading the file. Reading from a file depends on the length of the file, which is out of my control, it would be

difficult to place a value for that particular portion. So, for the sorting portion, we find that we loop once finding the minimum value for the set and another three times when adding values to the max activities list. Looking at this we can then say that my time is O(4n) => O(n).