CME 2204 Assignment 2

Dynamic Programming Project

29.05.2024 23:55 (Sharp deadline – No extension!)

Rules:

- The submissions will be checked for code similarity. Plagiarism will be graded as zero.
- Include a screenshot of your results with the given input file set.
- You are required to upload files with the naming format below :

  (STUDENT1_ID_Name_Surname)_Hw2.zip

  E.g. 1800510045_Ali_Veli_HW2.zip

Problem Definition:

In this assignment, you are expected to design a Dynamic Programming (DP) approach that makes a city tour planning for tourists.

You are working in a travel agency to make city tour plans for your customers. The city is made up of several landmarks connected by streets, forming a landmark graph. Each landmark has an adjusted attractiveness score, which indicates how appealing this landmark is to tourists. Additionally, the base attractiveness score is adjusted by the visitor load of a landmark, i.e., the attractiveness score decreases as the number of visitors at the landmark increases, reflecting the idea that a more crowded attraction might be less enjoyable. Furthermore, each tourist rates his/her interest in each type of landmark on a scale, in which 0 means no interest and 1 means maximum interest. These ratings should be used to further adjust the initial (base) attractiveness score of each landmark. Each landmark connection has an adjusted attractiveness score based on the following formula:

Adjusted Attractiveness Score(Landmark) = Base Attractiveness Score*Personal Interest Score*(max(1-Visitor Load*0.03*Time_To_Landmark,0.1))

You aim to create a tour route that starts and ends at the Hotel and maximizes the total attractiveness score for each tourist. No landmark (except the start landmark - Hotel) can be visited more than once in the tour.

The city's map is a graph, where vertices represent the landmarks, and edges represent the streets connecting them. Each vertex has, besides the connections, the name of the landmark and its attractiveness score. Each edge has a weight that shows the time to reach one landmark to another one.

You need to plan the city tour route so that:

- The tour starts and ends at the Hotel (a designated vertex in the graph).
- The total sum of the attractiveness scores is maximized. You should optimize the route based on dynamically adjusted attractiveness scores.
- Each landmark is visited at most once, except for the hotel, which is visited exactly twice (at the start and end).

Operations:

The following data is provided as text files, and you need to read and save them in suitable data structures.

- Initial attractive score and travel time of each landmark (landmark_map_data.txt)
- Visitor load of each landmark (visitor_load.txt)
- Personal interest in each landmark (personal_interest.txt)

You are expected to implement a DP approach to maximize the total attractiveness score. Your algorithm should not reach the O(n!) runtime in the worst case.

Your algorithm should return the **total attractiveness score, total travel time**, and **the names of landmarks** in their visiting order.

In the actual control time of your code, there will be new input files that contain different landmarks' names (only "Hotel" is the fixed name), different attractiveness scores, and different travel times.

You need to output the time it took for your code to run, you can use any library or method for this purpose.

A sample output of the code:

Assignment 2 for #Student Number, #Student Name

(E.g. Assignment 2 for 1800510045, Ali VELİ)

Please enter the total number of landmarks (including Hotel): 4

Three input files are read.

The tour planning is now processing…

The visited landmarks:
1-Hotel
2-Park
3-Museum
4-Tower
5-Hotel

Total attractiveness score: 20.99

Total travel time: 90 min.

Program took 4 minutes and 10 seconds to run.

// The end of the sample screen.

Note 1: These results are not exact ones, not obtained by the actual input files.

Note 2: If your code is not running properly, write this in submission text. Failure to do so will result in a lower grade than those students who informed about the problem in their code.

Grading Policy

Task	**Percentage**
Naming Format	5
Input processing	10
Total running time	10
Dynamic programming approach	40
Finding exact solutions	35

END OF ASSIGNMENT 2 DOCUMENT