CO322: Data Structure and Algorithms

Lab2: Dynamic Programming

Aim: Aim of this laboratory is to get an understanding about the applications of dynamic programming and how it can be achieved by using the primitives provided by a common programming language.

Objective:

- Find a recursive solution to a problem with overlapping sub-problems.
- Use memorization to improve the runtime.
- Evaluate the runtime of the implementations.

Problem:

There are N stations on a train route numbered from 0 to N-1. The ticket cost between two stations (say i, j) is given in a global, two dimensional matrix called cost. The ticket cost between two stations can be represented by *cost[i][j]*. Further, the train travels only in one direction from station 0 to N-1.

Your task is to implement a function called *int minCost(int fromStation, int toStation)* using Java which returns the minimum possible cost to travel between given two stations. Note that depending on the station to station ticket price it may even be attractive to break the journey (get down from some intermediate station and restart the journey) while traveling from i to j.

You may assume that there are less than 100 stations. (might be useful when deriving a key for hashing).

Use the provided skeleton code.

Task: You are required to do the following;

1. Using recursion (or otherwise) implement the *minCost* function.
2. What is the run-time complexity of your implementation.
3. Argue that dynamic programming can be used to improve the run-time.
4. Use dynamic programming to improve the run-time.
5. Calculate the run-time of your implementation in part 4 above. Assume, hashing is O(1).

Submission: Submit the answers to part 1 and 4 in two different java files named *1.java* and *4.java*. Answers to other questions should be submitted in a *written.pdf* file. All files should be in a single tar ball. Deadline for submission 26th July 2019.