## CSCI406 Dynamic Programming Group Project

Contributors:

2. To develop an optimal solution to this problem, a good approach is to realize that at any given step, the optimal cost is equal to the cost of doing operations in the present city plus the cost of relocating to this city from any other city. With this information in mind, we can solve it through the following recursive relation:

|  |  |
| --- | --- |
|  | The number of months up to where the current optimal path is being calculated, including , a month previous to the first month in any given dataset |
|  | The number of cities in the dataset, 0 indexed |
|  | The opt\_path function, which takes an arbitrary city and a month |
|  | The cost of operating at city during the month of |
|  | The cost of relocating from city to city |

Basis Step:

Recursive Step:

ii. Dynamic Programming Algorithm:  
 function **opt\_path\_dp**( = cities, = number of months):  
 // initialize a ( x ) matrix with 0s  
 M  [0 for i in | |][0 for j in (| |) ]

// initialize first column of matrix with starting costs from each city  
 M[0][0] = [0][0], M[1][0] = [1][0]… M[][0] = [][0]

// populate cells column-major  
 for j in | |:  
 for i in | |:  
 if i == 0: // first row in the matrix (first city)  
 M[i][j] = min(M[i][j-1], // cost of staying in same city  
 M[i+1][j-1] + fixed cost to city i + 1,  
 M[i+2][j-1] + fixed cost to city i + 2, …   
 M[i+][j-1] + fixed cost to city i + )  
 + cost at city i in present month (j)

else if i == 1: // second city  
 M[i][j] = min(M[i][j-1],   
 M[i-1][j-1] + fixed cost to city i - 1,  
 M[i+1][j-1] + fixed cost to city i + 1, …   
 M[i+][j-1] + fixed cost to city i + )  
 + cost at city i in present month (j)  
 else if i == 2: // third city

M[i][j] = min(M[i][j-1],   
 M[i-1][j-1] + fixed cost to city i - 1,  
 M[i-2][j-1] + fixed cost to city i - 2, …   
 M[i+][j-1] + fixed cost to city i + )  
 + cost at city i in present month (j)  
 else if i == – 1 // th city:  
 M[i][j] = (the min value of the (cell in the previous column (j -1) + its transportation costs to city ) + cost at city in present month (loop over all cities from previous month to find the lowest cost at the th city to finalize matrix).

iii. Traceback Algorithm:

b) Complexity of Algorithm:

Using dynamic programming, the complexity of our algorithm is . This is because of a double for-loop which first iterates over every month, and as a substep of each month iteration iterates over every city. If we had instead gone with the typical recursive approach, our complexity would have been .

c) Implementation: <https://github.com/mcbeej/dppython>

d) Optimal Solution:

145

NY NY DEN DEN DEN DEN DEN LA LA LA LA LA