CS 477 HW-6

Chris Howe

November 2023

1 Problem 1

The goal of problem one is to determine the best possible combination of warehouse deliver cites to maximize profits. Moving between warehouses has a fuel cost. Staying at the same warehouse has a discount from the city. At each warehouse, the owner must pay the warehouse staff a salary that varies each week.

1.1 Part A

The best way to solve this problem is with dynamic programming. Dynamic programming is useful when a divide and conquer method is applicable but the same sub problems come up frequently. Dynamic programming supplements this by saving results to problems that are already solved. In order to create a dynamic programming algorithm for a problem, the first step is usually to define a recursive formula to solve the function. This recursive function used to solve this problem is shown below. Essentially, the recursive function stores the optimal values determined in previous problems. It starts from the end of the problem(The last week) and works backwards to the beginning (The first week). At each time step, it determines where to deliver packages based on the optimal solutions for the previous week. It then determines these previous optimal solutions and so on.

The variable being optimized for is the total cost for the week. The algorithm returns the lowest possible cost that can be obtained given a set of parameters. The solution can be obtained by solving each sub problem since an optimal solution would have to pass through any optimal solutions for sub problems. This is true because the problem adheres to the optimal substructure property. Lets assume that the fastest way through A3 is through B2. The approach demonstrated would have either already taken this path through B2 if B2 is the optimal solution. If B2 was not optimal, the algorithm would have not chosen B2. (See the diagram below to see the nodes referenced here).

In the recursive formula pictured, A_i and B_i. "Low" represents the table of optimal values determined for the sub problems. "i" refers to the week being optimized. D_A and D_B refer to the discount amount each city gives the owner for staying in the city.

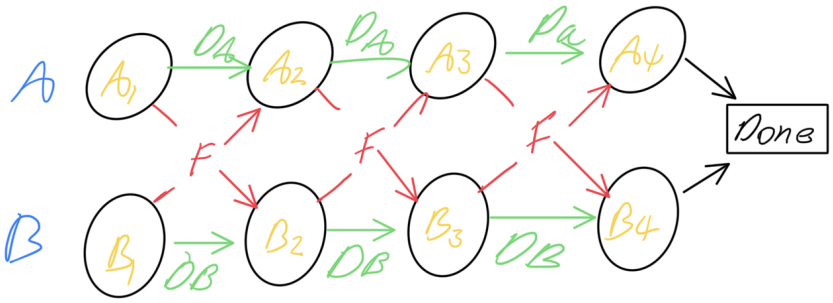

Figure 1: Problem Design

$$low_A[i] = \begin{cases} A_1 & i = 1 \\ min(low_A[i-1] - D_A + A_i, low_B[i-1] + F + A_i) & i \geq 2 \end{cases}$$

$$low_B[i] = \begin{cases} B_1 & i = 1 \\ min(low_B[i-1] - D_B + B_i, low_A[i-1] + F + B_i) & i \geq 2 \end{cases}$$

1.2 Parts B - D

Parts B - D are already implemented in code.