Adam Nelson, Lewis Setter, Akshay Swaminathan

CSCI 406 Algorithms, Section A

Prof. Dinesh Mehta

Project 3: Dynamic Programming

Describe the main idea of your approach including how you break the problem into smaller recursive problems

The main recursive problem is that finding the ideal solution for n days does not also give the ideal solution for n+1 days. Each day n has a projected demand of kn, so there are at most kn+1 order size possibilities. This means that day n-1 has kn-1+kn-tank size possibilities, as you can order additional gas for any given day as long it does not exceed the size of tank.

To find the optimal cost for n days, you find every possible cost for n-1 days, and modify the demand for the n-1th day for each possibility stated above. The optimal cost of all those solutions gives the optimal cost and solution for all n days.

The base case is the scenario of having to buy gas for a single day. This cost never changes as it is only the price of placing an order. Since the demand for that single day does not matter, only the overnight storage costs and orders for subsequent days are what change the overall solution costs. This is what allows us to go back and add days when determining the cost of any n number of days.

Write pseudocode for your dynamic programming algorithm

Dynamic Solution Pseudocode:

* Calculate the maximum amount you would want to store overnight based off the storage cost, tank size, and the cost of placing an order
* Create a 2D array with one axis representing the day and the other axis representing how much gas you are storing for the next day, no greater than your maximum storage amount
* Initialize every possibility for the first day using the cost of placing an order and the cost of storing that amount of gas overnight
* For every day from the second to the last day:
  + For every possible amount of gas remaining at the end of the day:
    - Create a new array to store the cost of purchasing every amount of gas ranging from the requirements for that day up to the requirements plus the maximum storage amount
    - Determine your smallest possible purchase based off the current day’s requirements and the tank size
    - For every possible purchase size:
      * Set the cost of the purchase to the cost of acquiring all gas prior to this day
      * Add the cost of today’s delivery
      * Add the cost of storing extra gas tonight
    - Subtract the delivery cost from the scenario of purchasing no gas on this day
    - For every possible purchase size:
      * Compare purchase sizes and find the minimum cost
      * Store how much gas was bought on that day
    - Store the previous day associated with the minimum cost as the parent pointer for the current day
* Go back through the parent pointer chain to find how much gas was ordered each day using the traceback algorithm

Develop a traceback algorithm that returns the days on which to place orders and how much.

Traceback Algorithm:

* Set the starting cell as the the last day in the table with no gas left over
* While the starting cell is not null (meaning this cell is the parent of another cell):
  + If any gas was purchased on this day, increase the total number of days gas was ordered and indicate the day
  + Record how much was purchased
  + Set the new “starting cell” as the parent of the current cell

Theory: Derive the complexity of your algorithm in terms of n.

Insert Text Here