DOKUZ EYLUL UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CME 2204 ALGORITHM ANALYSIS

ASSIGNMENT II

DYNAMIC PROGRAMMING AND GREEDY APPROACH

Lecturers

Asst. Prof. Dr.ZERRİN IŞIK

Res.Asst. ALİ CÜVİTOĞLU

Res.Asst. EZGİ DEMİR

by

ŞEFİKA ÖZLEM PUL

2017510067

CONTENTS

INTRODUCTION

The aim of the assignment is to find the conditions of a car company with the least cost and the most profit. The assignment consists of two parts, part1 and part2. To briefly explanation part1: The car company has a fixed number of cars that it can produce every month. A different number of car production demands come to the car company every month. If the demand is less than we can produce car, there are processes that need to be checked, such as putting it in the garage or if the demand is big, employing intern. There are costs we will pay if we put a car in the garage or if we need the intern, we have to pay them. To briefly explanation part2: According to the monthly offers of different investment companies, we have the money to be calculated according to which company we have the most money. The car company should deposit the payment you receive from your sales. The cost of each car is “B” and the car company takes half the price at the beginning of the month and the rest is taken at the end of the month. “C” offers to the car company from different rates for each month. At the end of each month, the car company can change the investment company by paying a tax of ‘t’ of the money invested, or continue with the same investment company without paying any taxes. For both part1 and part2, we have to calculate both greedy and dynamic programming method.

TASK SUMMARY

COMPLETED TASKS

The required file reading operations were used for the assignment. In the calculation of the cost, the way to follow the dynamic programming method was considered and the coding of the assignment was completed by using two-dimensional array. In the greedy algorithm method, code was written using a one dimensional array. In the profit calculation, the coding of the assignment was completed by using two-dimensional array in dynamic programming method and one-dimensional in greedy method.

INCOMPLETE TASKS

There are no completed tasks.

RUN TIME COMPLEXITY AND SPACE COMPLEXITY

c: number of investment companies, x:number of months r:total number of cars demanded

Dynamic Programming Method for Cost:

Run Time : O(r * r*x)

Space Complexity : O(r*x)

Greedy Method for Cost :

Run Time : O (c *x)

Space Complexity : O (1*x)

Dynamic Programming Method for Profit:

Run Time : O(c * c* x)

Space Complexity : O(c* x)

Greedy Method for Profit :

Run Time : O (c *x)

Space Complexity : O (1*x)

EXPLANATION OF ALGORITHMS

While calculating the cost, two-dimensional array was used while using the dynamic programming method. The dimensions of two-dimensional array are the sum of the demanding month and the number of requests up to that month. The line with (i=0) array[i][j] was filled with garage cost(j) values. If the number of cars demanded in the i. month is more than our production capacity: If the index [i-1][demand-production] of the previous month (i-1) is different from the number of productions (demand-productions) than the zero; If the cost of working the interns as much the difference of the demand number of production is greater than the value of the difference of the demand of the previous month from the number of production of this month, the cost value of the current month is the sum of the difference of the previous month and the value of the previous month. If the cost working the interns as much as the difference of the demanded production number is less than the value of the difference of the demand of the previous month from the production number of this month, the cost value of the current month is the sum of the value the production difference of this month and the value of the previous month. If the number cars demanded in the i. month is less than our production capacity; If the number of cars demanded in the first month is less than our production capacity, I used loop in the array as much as the difference of the number of cars demanded that month.

```
if(m_demand[i]>p)
{
    if(array1[i-1][m_demand[i]-p]!=0)
    {
        if(((m_demand[i]-p)*d)>array1[i-1][m_demand[i]-p])
        {
            array1[i][j]=array1[i-1][m_demand[i]-p]+array1[i-1][j];
        }
        else if(((m_demand[i]-p)*d)<array1[i-1][m_demand[i]-p])
        {
            array1[i][j]=((m_demand[i]-p)*d)+array1[i-1][j];
        }
        else
        {
            array1[i][j]=array1[i-1][m_demand[i]-p]+array1[i-1][j];
        }
        break;
    }
    else if(array1[i-1][m_demand[i]-p]==0)
    {
        array1[i][j]=((m_demand[i]-p)*d)+array1[i-1][0];
        break;
    }
}
else if(p>m_demand[i])
{
    for(int k=0;k<=p-m_demand[i];k++)
    {
        if(k==0)
        {
            array1[i][k]=array1[i-1][k];
        }
        else if(k>0)
        {
            array1[i][k]=garage[k];
        }
    }
```

In the greedy algorithm method when calculating the cost, if the number of cars demanded in the next month is more than the number of cars we can produce and if the number of cars demanded this month is less than the number of cars production, we will pay less if we keep the car in the garage or if we pay to the inters.

```
else if(p>m_demand[i])
{
    if(i+1!=m_demand.length) {
        if(m_demand[i+1]>=p && (p-m_demand[i])>(m_demand[i+1]-p))
        {
            temp=garage[p-m_demand[i]-(m_demand[i+1]-p)];
            arr[i]=arr[i-1]+garage[p-m_demand[i]-(m_demand[i+1]-p)];
        }
        else if(m_demand[i+1]>=p && p-m_demand[i]==(m_demand[i+1]-p))
        {
            temp=garage[p-m_demand[i]];
            arr[i]=arr[i-1]+garage[p-m_demand[i]];
        }
        else if(m_demand[i+1]>=p && (p-m_demand[i])<(m_demand[i+1]-p))
        {
            arr[i]=arr[i-1];
        }
        else if(m_demand[i+1]<p)
        {
            arr[i]=arr[i-1];
        }
```

When making the profit calculation, we take half of the money we have in the dynamic programming method at the beginning of the month and add it to the half of the money we have by calculating the interest rate offered by the investment company that month. We add our money to the percentages given by the investment company offers and compare every possibility. If we change the investment company while comparing, we return within the two-dimensional array considering that there will also be a loss of money in ‘t’ percentage. According to the different combinations of each investment company coming from each month, the best results obtained for each investment company are kept. At the end of the month, it is collected with the highest value among the investment companies and the last half of the last month.

```
for(int k=1;k<=c;k++)
{
    if(j==k)
    {
        if(array[i-1][k-1]!=0 && array[i-1][k-1]<(res+temp)+(((res+temp)*cs.get(k)[i])/100))
        {
            array[i-1][k-1]=(res+temp)+(((res+temp)*cs.get(k)[i])/100);
        }
        else if (array[i-1][k-1]==0)
        {
            array[i-1][k-1]=(res+temp)+(((res+temp)*cs.get(k)[i])/100);
        }
    }
    else if (j!=k)
    {
        temp=temp-(temp*t)/100;
        if(array[i-1][k-1]!=0 && array[i-1][k-1]<(res+temp)+(((res+temp)*cs.get(k)[i])/100))
        {
            array[i-1][k-1]=(res+temp)+(((res+temp)*cs.get(k)[i])/100);
        }
        else if (array[i-1][k-1]==0)
        {
            array[i-1][k-1]=(res+temp)+(((res+temp)*cs.get(k)[i])/100);
        }
        temp=array[i-2][j-1];
    }
}
```

(Dynamic Programming method for part2)

In the method of greedy algorithm in profit calculation, we keep our money at the value of which investment company gives the highest value for each month, we do not need to examine other situation. While trying to find the maximum value, for every month, if the previous investment company are different, the value of ‘t’ is subtracted from our money, multiplied by the proposal of the investment company and added to the existing money we have. If the result is greater than our maximum, that will be our new maximum value.

```
else if(i>1)
{
    if(temp2!=j+1)
    {
        if(max< ((((res+(array2[i-2]-(array2[i-2]*t)/100))*cs.get(j+1)[i]))/100)+(res+(array2[i-2]-(array2[i-2]*t)/100))
        {
            max=((((res+(array2[i-2]-(array2[i-2]*t)/100))*cs.get(j+1)[i]))/100)+(res+(array2[i-2]-(array2[i-2]*t)/100));
            temp1=j+1;
        }
    }
    else if(temp2==j+1)
    {
        if(max< (((res+array2[i-2])*cs.get(j+1)[i]))/100+(res+array2[i-2]))
        {
            max=((res+array2[i-2])*cs.get(j+1)[i])/100+(res+array2[i-2]);
            temp1=j+1;
        }
    }
}
```

PROBLEMS ENCOUNTERED

I had difficulties while installing the algorithm because I could not think very effectively how the greedy method should proceed while making cost calculations. So I could not be sure of the results from the greedy method.

CONCLUSION

The dynamic programming method gives us the best possible result. When we do the dynamic programming method, the run time is less and space complexity less than the brute force algorithm method. The greedy method chooses only the best option at the time it is located, not considering other possibilities. If we compare dynamic programming gives more reliable results. Therefore, it is better to operate with dynamic programming method when calculating cost and profit.