Report on Dynamic Programming Project

Algorithm course(fall 2020/2021)
Dr Samer Arandi

Student name	Student id	Grade
Ashraf Adel Habromman	11821493	

Introduction:

This project was done using divide and conquer technique and recursion function which has inefficient tun time almost reach to be an exponential function. So because of the high running time, we decided to use dynamic programming technique which is an efficient method and reduce the running time of the function to be polynomial.

Tool:

1-Hacker Rank website.
2-Eclipse IDE.
3-Paint.

Programming Language: java 8

Code explanation:

Part A: Divide and conquer

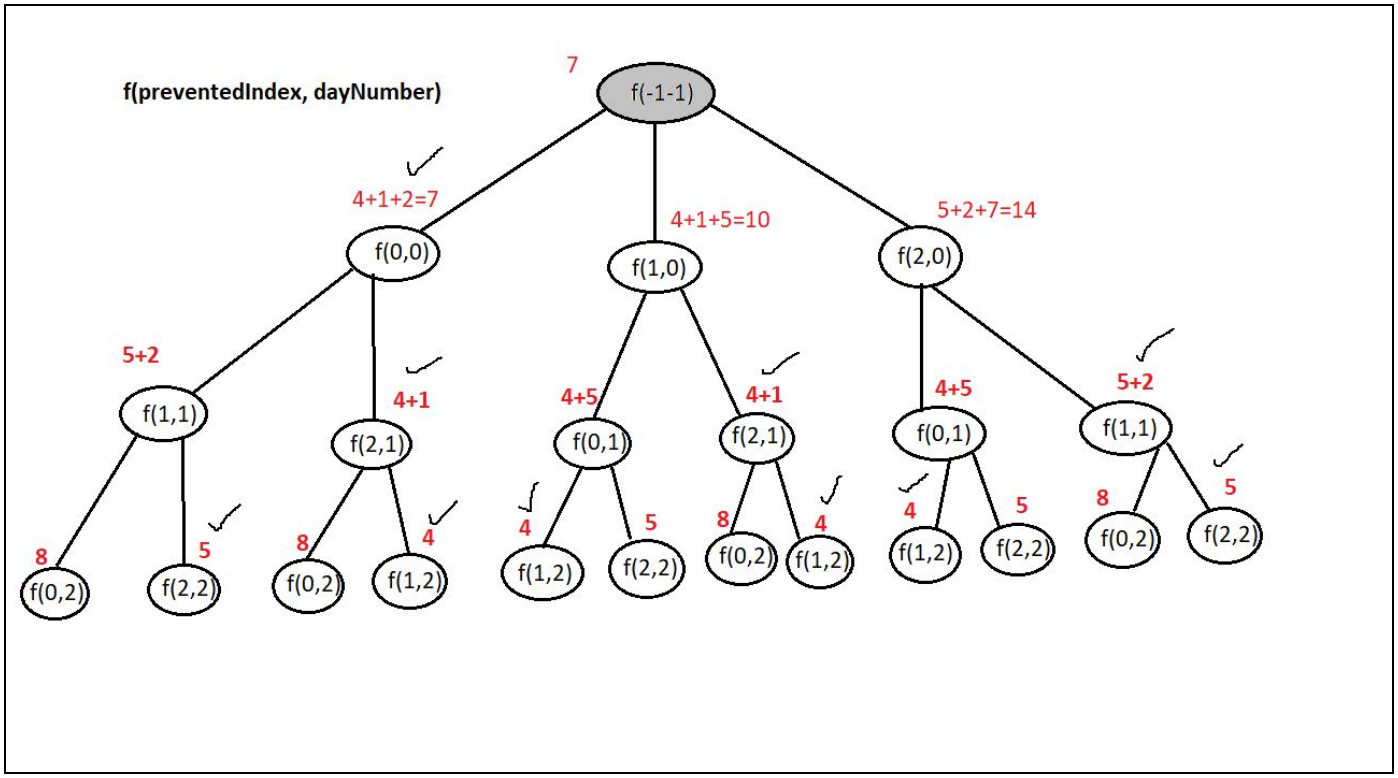
I have implemented a function I called “minimumChakraDAQ” which return an integer number expresses the minimum amount of chakra (energy) which Naruto have to consume in training to be the Hokage of the village.

The function minimumChakraDAQ have parameters, the first parameter is the matrix which includes all of the chakra amounts for different techniques and each day. For the remaining parameters, they should define the size of the problem and make it unique and different from another problem, so I think that if I could define several parameters which express the allowed techniques that Naruto can train on, and parameter to express the day of training called “dayNumber”. But its cloud was not a good idea to define several parameters equal to the number of allowed techniques, what if I have more than 3 different techniques? So I decided to define a parameter which expresses the prevented technique in that call (that day) because Naruto was training on this technique yesterday, the parameter name is “preventedIndex“.

Function prototype:

int minimumChakra(int [][]matrix, int preventedTechnique, int dayNamber);

Visual Recursion tree:



Part B: Dynamic Programming

Chakra table:
(test case)

2	5	8
5	2	4
7	1	5

Dynamic programming overview:

Technique \ day	0	1	2
0	2	5 + 5	
1	5		
2	7		

minChakra[0][1]=min(minChakra[1][0] ,minChakra[2][0])+chakraTable[0][1];

Technique \ day	0	1	2
0	2	10	
1	5	2 + 2	
2	7		

minChakra[1][1]=min(minChakra[0][0] ,minChakra[2][0])+chakraTable[1][1];

Technique \ day	0	1	2
0	2	10	
1	5	4	
2	7	2 + 1	

minChakra[2][1]=min(minChakra[0][0] ,minChakra[1][0])+chakraTable[2][1];

Technique \ day	0	1	2
0	2	10	3 + 8
1	5	4	
2	7	3	

minChakra[0][2]=min(minChakra[1][1] ,minChakra[2][1])+chakraTable[0][2];

Technique \ day	0	1	2
0	2	10	11
1	5	4	3 + 4
2	7	3	

minChakra[1][2]=min(minChakra[0][1] ,minChakra[2][1])+chakraTable[1][2];

Technique \ day	0	1	2
0	2	10	11
1	5	4	7
2	7	3	4 + 5

minChakra[2][2]=min(minChakra[0][1] ,minChakra[1][1])+chakraTable[2][2];

Technique \ day	0	1	2
0	2	10	11
1	5	4	7
2	7	3	9

About the dynamic table, I initialize the first column from the chakra table, then pass on every element in the table column by column to calculate its value. For every element, I have to return to the previous column (previous day) and get the minimum value from, provided that I can't take the value which has the same index in the previous day (can not take same technique in two successive days).
Then get the minimum value from the last column in the dynamic table.
In the same time when values of table calculated, I have implemented a matrix which has the same size of chakra table. And add the indices of minimum chakra in its suitable position in the matrix.
So by this matrix, I can print the path which Naruto can follow to consume the minimum chakra(energy) and be the Hokage in his village.

Conclusion:

I have learned how to deal with a real problem and solve it using different techniques, and for every technique, there is a cost to pay, but this cost may be high or low depends on the complexity of technique, and for sure we have to pay the lower cost.