Jahongir Hayitov – CS-01

Problem-Set 5, Theoritical Part

Task:

The National Health Council wants to improve health care in three of the most underdeveloped regions. Currently it has five medical teams available to allocate among such regions to improve their medical care, health education, and training programs. Therefore, the council needs to determine how many teams (if any) to allocate to each of these regions to maximize the total effectiveness of the five teams. The teams must be kept intact, so the number allocated to each region must be an integer. The measure of performance being used is additional person-years of life. (For a particular region, this measure equals the increased life expectancy in years times the region's population.) The following table gives the estimated additional person-years of life (in multiples of 1,000) for each region for each possible allocation of medical teams.

1. Describe a general algorithm for any number M of available medical teams, any number R of regions and table E with estimates:

(a) Summarize the idea for a na¨ıve recursive algorithm.

(b) Identify overlapping subproblems.

(c) Write down pseudocode for the dynamic programming algorithm that solves the problem (top-down or bottom-up). It is enough to compute maximum performance without specifying team allocation

Solution:

a) Naive Recursive Algorithm:

- We consider each possible allocation cases of M medical teams, starting from 0
- Each allocation teams, we calculate the additional person-years of life
- And for each allocation team and region, we recursively compute the maximum total person-years of life, which can be achieved by other teams (M-i) and regions (R-1)
- finally we return the achieved maximum total additional person-years of life

b) The overlapping subproblems may occur. Because subproblems will require to allocate the same number of to multiple regions. Therefore, the same subproblems may be encountered more than two times in different nodes of the recursion tree. For instance: given Region A, B, C and D. If we don't use dynamic programming, we would spend almost the same time when we calculate A+B+C+D and 2*A+C+D. But we already calculated C+D, and did the same operation more than one time.

c) Bottom-Up approach:

We need to initialize a table DP of size (M+1)x(R+1), where DP[i][j] represents the maximum total additional person-years of life which can be achieved with [i] medical teams allocated among j regions:

```
func findMax(M, R, E):
     for i from 0 to M: DP[i][0] := 0;
     for j from 0 to R: DP[0][j] := 0;
     for i from 1 to M:
        for j from 1 to R:
            maxValue := 0;
            for k from 0 to i:
               maxValue := max(DP[i-k][j-1] + E[k][j], maxValue)
            DP[i][j] := maxValue;
     return DP[M][R];
```

2. Provide asymptotic worst-case time complexity with justification:

(a) for the na¨ıve recursive algorithm

(b) for the dynamic programming algorithm

<u>*Solution:*</u>

a) O(R^M) - to prove, we have M medical teams and R regions. If we use recursive algorithm, it searches all cases so there are R^M possible allocations. And for each allocation we need to recursively calculate the maximum total additional-years of life

*b) O(M^2R) – to prove, created a table of size (M+1)x(R+1), as we can see from the pseudocode, iterating through all the elements of this table and inside this loop, added an extra loop `k from 1 to i`, which depends on M. Therefore the total time complexity is O(R*M^2)*

3. Apply the dynamic programming algorithm to an instance of the problem below. You must provide the table with solutions for subproblems that are computed in the algorithm, as well as give the final answer to the problem.

Solution:

This Table **E** which stores the person-years of life in terms of the number of teams and regions

Teams \ Regions	**A**	**B**	**C**
0	0	0	0
1	45	50	20
2	70	70	45
3	90	80	75
4	105	100	110
5	120	130	150

According to the pseudocode, we do the following: for **DP** table

Teams	Region 0	Region A	Region B	Region C
0	0	0	0	0
1	0	45 → A*1	50 → B*1	50 → B*1
2	0	70 → A*1	95 → A*1+ B*1	95 → A*1+ B*1
3	0	90 → A*1	120 → A*2+ B*1	120 → A*2+ B*1
4	0	105 → A*1	140 → A*3+ B*1	140 → A*3+ B*1
5	0	120 → A*1	160 → A*3+ B*2	170 → A*1+ B*1+c*3

Therefore the maximum person-years of life is 170;