Google Farewell Round 1st Training

CS Chapter IEEE NTUA SB

April 4, 2023

Maximum Coins

Mike has a square matrix with N rows and N columns. Cell (i, j) denotes the cell present at row i and column j. Cell $(1, 1)$ denotes the top left corner of the matrix. Each cell has some amount of coins associated with it and Mike can collect them only if he visits that cell. $C_{i,j}$ represents the number of coins in cell with row i and column j. From a cell (i, j), Mike can decide to go to cell $(i+1, j+1)$ or cell $(i-1, j-1)$, as long as the cell lies within the boundaries of the matrix and has not been visited yet. He can choose to start the journey from any cell and choose to stop at any point. Mike wants to maximize the number of coins he can collect. Please help him determine the maximum number of coins he can collect.

Input: The first line of the input gives the number of test cases, T. T test cases follow. Each test case begins with a line containing the integer N. The next N lines contain N integers each. The j-th integer in the i-th line represents the number of coins $C_{i,j}$ in cell (i, j).

Output: For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is the maximum number of coins Mike can collect.

Limitations	**Sample Input**	**Sample Output**
Time limit: 20 seconds.	2	Case #1: 14
Memory limit: 1 GB.	3	Case #2: 9
$1 \leq T \leq 100$	1 2 5	
$1 \leq C_{i,j} \leq 10^7$	3 6 1	
	12 2 7	
Test Set 1:	5	
$1 \leq N \leq 100$	0 0 0 0 0	
	1 1 1 1 0	
Test Set 2:	2 2 2 8 0	
$1 \leq N \leq 10^3$ in at most 10 cases.	1 1 1 0 0	
$1 \leq N \leq 100$ in all other cases.	0 0 0 0 0	

In Sample Case #1, the maximum number of coins collected can be 14, if Mike follows this path: $(1, 1) \rightarrow (2, 2) \rightarrow (3, 3)$. In Sample Case #2, the maximum number of coins collected can be 9, if Mike follows this path: $(2, 3) \rightarrow (3, 4)$.

Test Set 1

Mike is allowed to go from cell (i, j) to cell $(i + 1, j + 1)$ or to cell $(i - 1, j - 1)$. We can consider each possible starting point and try calculating the value of each possible path Mike can traverse. Initialize the answer as 0. For each starting cell (i, j), Mike can either go diagonally upwards or diagonally downwards. First consider all the paths which start at cell (i, j) and end diagonally above it. We keep on adding the value of coins by traversing upwards diagonally and updating the answer. Similarly, we consider all paths which start at cell (i, j) and end diagonally below it and update the answer. Each path can contain at most

$O(N)$ elements. Thus, it takes $O(N)$ time for each starting position. There can be $O(N^2)$ starting positions. Thus, the overall complexity of the solution is $O(N^3)$.

Test Set 2

An important observation here is that all the numbers are positive. Thus, it is always optimal to collect the coins for each cell present on a particular diagonal instead of choosing some of the cells on the diagonal. This can be done by starting on the top left of each diagonal and traversing down and adding the coins collected. For each diagonal it would take $O(N)$ time to calculate the coins present in that diagonal. There are $O(N)$ diagonals present in the matrix. Thus, the overall time complexity of the solution is $O(N^2)$.

The implementation of the above algorithm on c++ can be found in maximumCoins.cpp file.