Task 1 :- Optimize Me

Srishty Gandhi, roll no. 20CS30052

***Abstract*— The task was to reduce the run time of the given program by using various optimization techniques. In the given program recursion was used to find cost of the path in two nXn matrices, which was replaced by an iterative approach to reduce the run time of the algorithm**

I used some other time reduction approaches too reduce the time complexity. Since in the real world every second the situation changes that's why we need to the least time to run the algorithm, for example lets take a drone, if a given algorithm takes less time to compute that means the drone would be able to make decision faster and hence fly with a greater speed. We need reduced space complexity so that it would save our resources, and in a drone that means a bit lesser weight, which results in less power consumed for a given velocity, that's why the task of optimizing a program is very important.

I. INTRODUCTION

The task was to reduce time and space complexity of a given program. Initially the code that was provided to us did not run for big values of sizen. The approach used to solve this task was to replace the recursive function with an iterative function.

II. PROBLEM STATEMENT

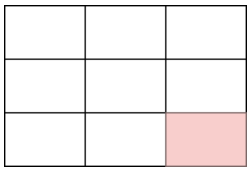
costMatrixA and costMatrixB are two nXn matrices given to us, such that each cell in these matrices is the cost that we need to pay for landing on that cell.

And there are two functions 'Find min cost A' and 'FindMaxCostB'

FindMinCostA finds the minimum cost of going from cell i,j to n-1,n-1

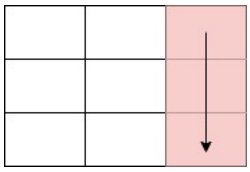
If we take a close look on function, we would find that we can go only towards right and down.

Therefore the cost of last cell will be same as the integer stored in that.

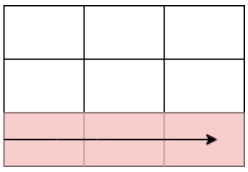


Then we calculate the costs of the cell of the last column of the table. All the cells of the last column have only one place to go, i.e downwards. So the cost of that cell will be equal to the cost of landing on that cell + the path cost of the cell below it.

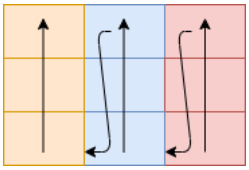
*Write anyone who might have helped you accomplish this eg any senior or someone



Similarly the last the cells in last row have only one path to go, i.e. towards right. So the cost of that cell will be equal to the cost of landing on that cell + the path cost of the cell on the right side of it it



Now if we traverse throughout the table, starting from the last cell to the top most cell. Then all of the cell will either fall into the above three conditions, or it will have two possible paths ahead, out of which we have to choose the one with minimum path cost.



A similar function can also be created for MaxCostB. The only difference will be that instead of minimum we have to choose the maximum cost for the path.

Part 2:-

It is a basic multiplication of both the matrices. It calls the MinCostA function for each cell for CostMatrixA and MaxCostB function for each cell in CostMatrixB.

By calling the MinCost/MaxCost function for each cell , we are finding the cost of some cells more than once. To avoid this we call the function only once, and then store all the cost values in the same array.

Part 3:-

In this part the same operation is being repeated several times. Therefore i have applied parallel computation based optimisation to optimise this code.

At first I used SPMD algorithm strategy to divide the work in different tasks, but later on replaced it with inbuilt function for parallel loop.

III. Results and Observation

Sizen	Given Algo	Optimized algo
4	0	0
8	0.005	0
12	1.903	0
16	613.986	0
100	-	0.003
200	-	0.022
500	-	0.395
700	-	1.238
716	-	1.262

IV. Future Work

There are some bugs still left to fix. I have used different methods to calculate the CostMatrixA and costMatrixB, because if I use the same method for b which i have used for A, the program starts crashing(which ideally should not be the case). There's a scope of vectorising the code, and the parallel computation can be further improved.

CONCLUSION

The problem was to optimize a given code so that, it could do the same operation on a bigger data-set, in less time.While solving it, I used various methods, to reduce the time and complexity of program, and as a result the max value of sizen increased to 700, which previously was around 12.

References

[1] https://www.youtube.com/watch?v=nE-xN4Bf8XIlist=PLLX-Q6B8xqZ8n8bwjGdzBJ25X2utwnoEGindex=1