Path with max grid

10 June 2022 07:46

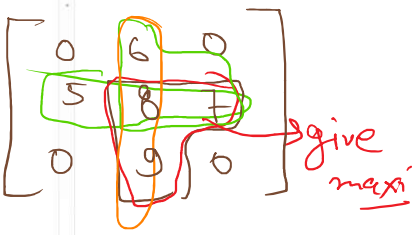
1219. Path with Maximum Gold

Medium 1928 50 Add to List Share

In a gold mine `grid` of size `m x n`, each cell in this mine has an integer representing the amount of gold in that cell, `0` if it is empty.

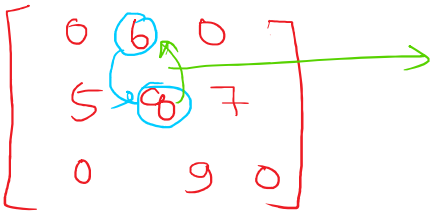
Return the maximum amount of gold you can collect under the conditions:

- Every time you are located in a cell you will collect all the gold in that cell.
- From your position, you can walk one step to the left, right, up, or down.
- You can't visit the same cell more than once.
- Never visit a cell with `0` gold.
- You can start and stop collecting gold from **any** position in the grid that has some gold.

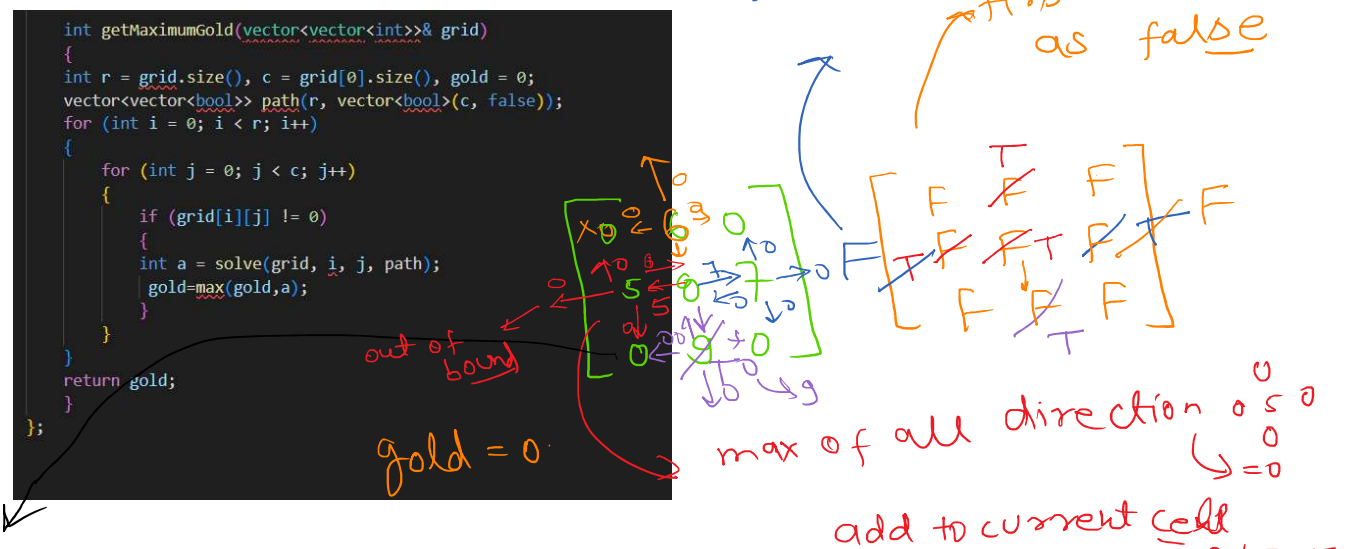


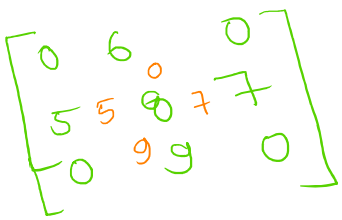
Example 1:

```
Input: grid = [[0,6,0],[5,8,7],[0,9,0]]
Output: 24
```




```
int getMaximumGold(vector<vector<int>>& grid)
{
int r = grid.size(), c = grid[0].size(), gold = 0;
vector<vector<bool>> path(r, vector<bool>(c, false));
for (int i = 0; i < r; i++)
{
    for (int j = 0; j < c; j++)
    {
        if (grid[i][j] != 0)
        {
        int a = solve(grid, i, j, path);
         gold=max(gold,a);
        }
    }
}
return gold;
}
};
```





```
    int solve(vector<vector<int>> &maze, int row, int col, vector<vector<bool>> &path)
{

//All Stopping Conditions; i.e. row,col should never cross the dimesion of given 2D Grid
//Cell having 0 is not to be visited & already visited cant be re-visited
//Path is passed by reference to store the current visited cell to prevent repeation

//Base-Condition
    if (row < 0 || col < 0 || row >= maze.size() || col >= maze[0].size())
```

return 5

```
//Path is passed by reference to store the current visited cell to prevent repeation

//Base-Condition
    if (row < 0 || col < 0 || row >= maze.size() || col >= maze[0].size())
        return 0;
    if (maze[row][col] == 0 || path[row][col] == true)
        return 0;

//marking the current cell as visited
    path[row][col] = true;

//Induction-step
    int sum = 0;
    sum += maze[row][col];

//Hypothesis-step
    //Right
    int right = solve(maze, row, col + 1, path);

    //Left
    int left = solve(maze, row, col - 1, path);

    //Up
    int up = solve(maze, row - 1, col, path);

    //Down
    int down = solve(maze, row + 1, col, path);

//Marking it false (unvisited) to Backtrack
    path[row][col] = false;
```

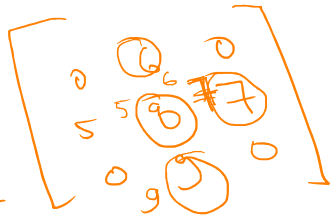
0, 5, 9, 7
max = 9
add to current
9+8 = 17

0 0 17 0
max = 17
6+17 = 23

so current max = 23

now same for 5, 9, 7, 9

max = 17+5 = 23



9+8 = 17

17+7 = 24

15+9 = 24