Q: Count the no. of subsets with sum = K.

Amazon |
Intuit |
Flipkart |
Myntra |
MS....

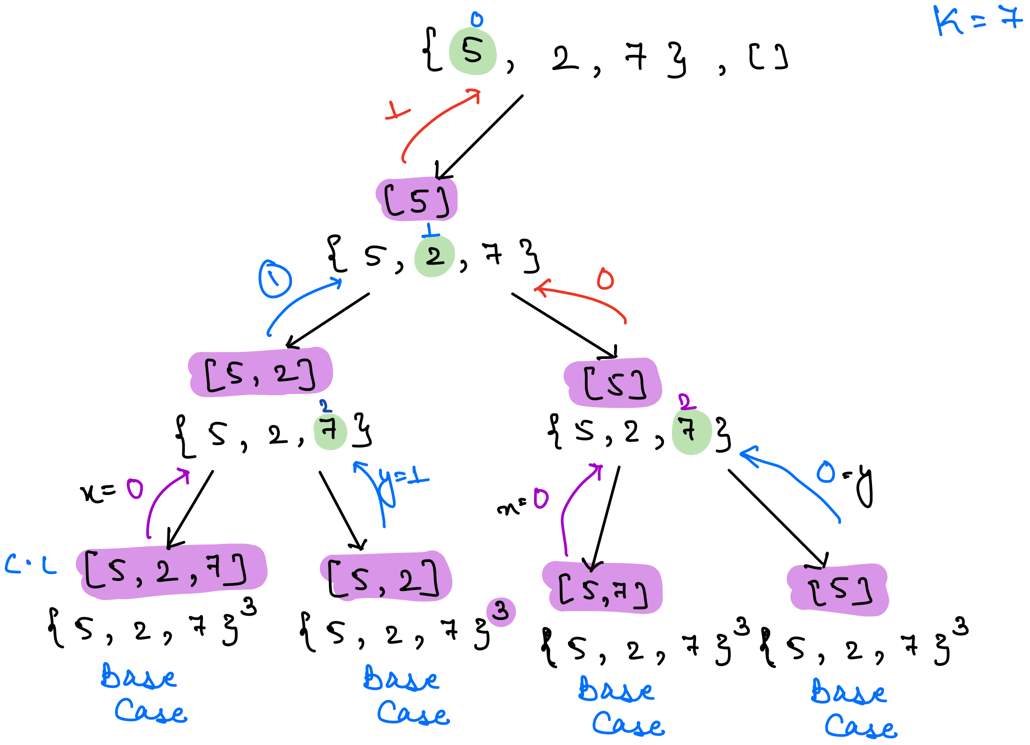
[5, 2, 7], K = 7 → [5, 2]
→ [7]
} ②

Note:- Array only contains distinct elements.

```
generateAllSubsets(currList, index, A[], K) {
        if( index == N) {
                if( Sum of currList == K) return 1;
                else return 0;
        }

        // Include A[index] in subset
        currList.add(A[index]);
         x = generateAllSubsets(currList, index+1, A);
         // Delete the last added element (A[index])
         // from the curr List
         // Exclude A[index] in the Subset.
         currList.pop()
          y = generateAllSubsets(currList, index+1, A);
          return x+y;
}
```

TC: $O(N \cdot 2^N)$



```
generateAllSubsets(index, A[], K, currSum) {
        if(index == N) {
              if(currSum == K) return 1;
              return 0;
        }
        //Include A[index] in subset
        currSum = currSum + A[index]
        x = generateAllSubsets(index+1, A, K, currSum);
        //Delete the last added element (A[index])
        //from the currList
        // Exclude A[index] in the subset.
        currSum = currSum - A[index]
        y = generateAllSubsets(index+1, A, K, currSum);
        return x+y;
}
```

TC : $O(2^N)$
SC : $O(N)$

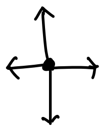
Q. Rat in a Maze.

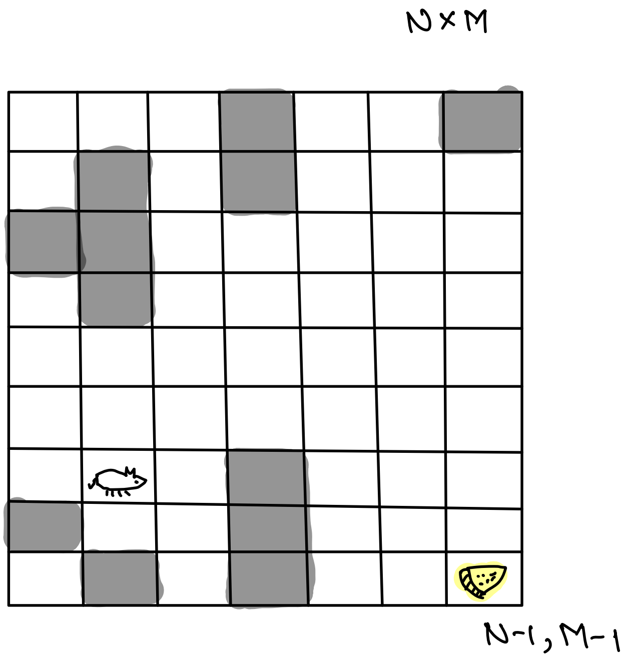
Amazon| MS| Flipkart| GS...

Given a maze (2D matrix) & the initial location of the rat (x,y). Return true if there exist a path from rat's location to cheese location.

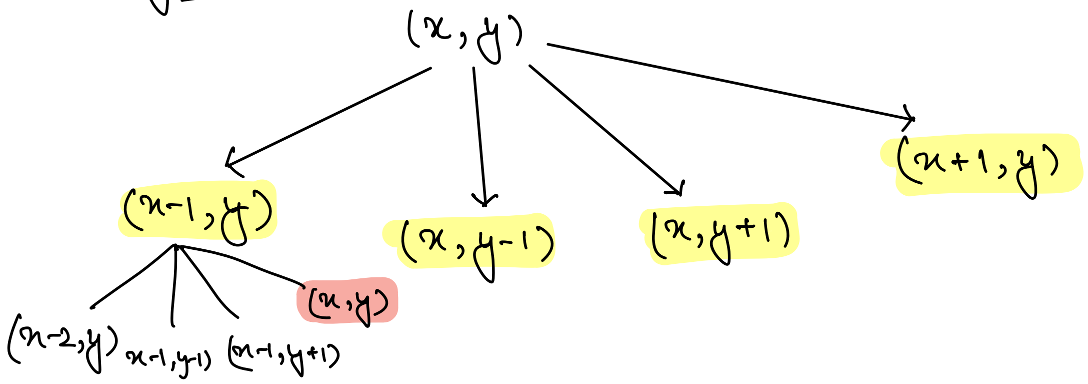
mat[i][j] = 0
→ Non Blocked

mat[i][j] = 1
→ Blocked

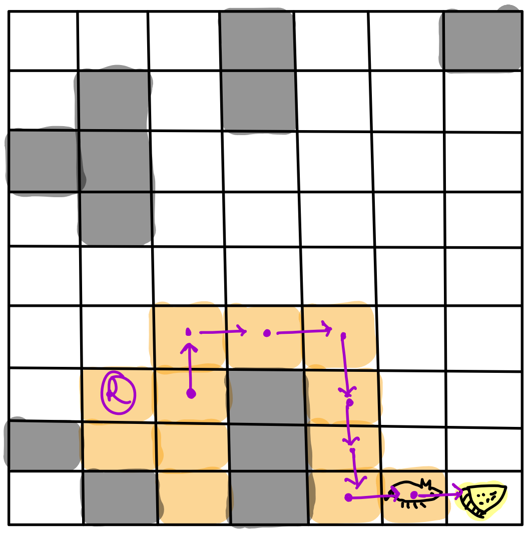


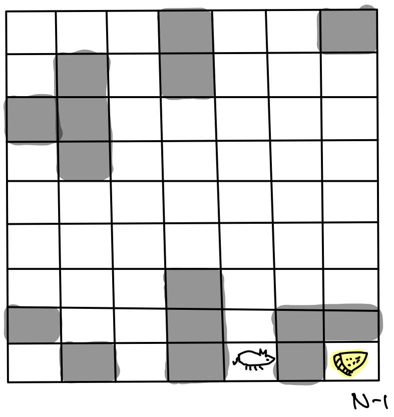


⇒ Rat can only visit a cell, if it is NOT visited yet.



```
boolean   mazeSolver( mat[][], N, M, x, y ) {
        if( x== N-1  && y == M-1)
                return true;
        if( x<0 || x>=N || y<0 || y>=M)
                return false;
        if( mat[x][y] == 1 || mat[x][y] == 2)
                return false;
        mat[x][y] = 2;
        return   mazeSolver(mat, N,M, x+1, y)
                         ||
                 mazeSolver(mat, N,M, x, y+1)
                         ||
                 mazeSolver(mat, N,M, x-1, y)
                         ||
                 mazeSolver(mat, N,M, x, y-1);
}
```





TC: O(N·M)

SC: O(NM)

Q: Rat in a maze.

Google (Hard)

Given

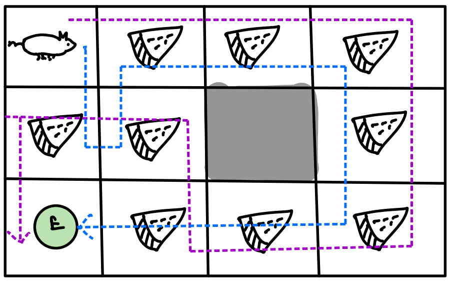
i) Start point of the rat.

ii) End point (Destination)

iii) Blocked points.

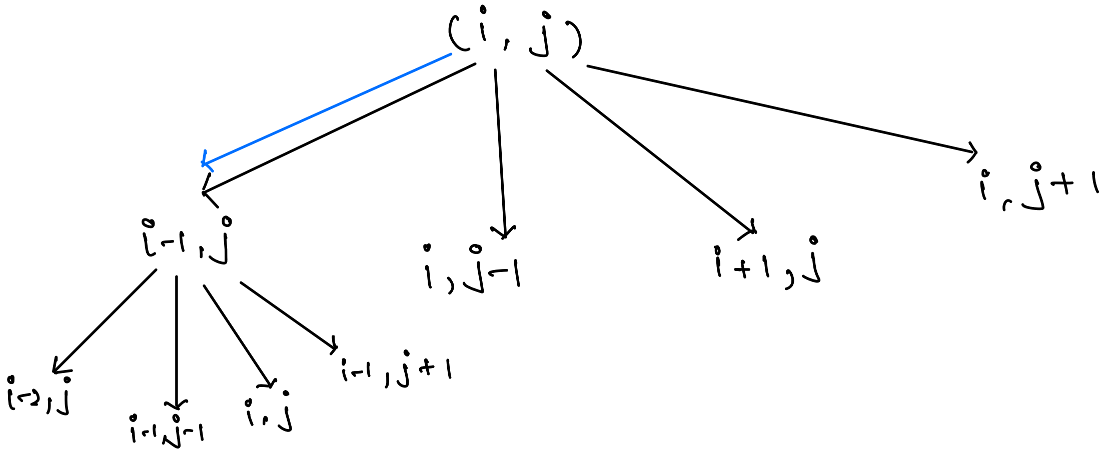
IV) Cells which are filled with cheese.

→ Count the no. of paths from the start to end, s.t rat can eat all the cheese available in the maze without stepping on the same cell more than once in a single path.



⇒ 2

Start	:	S_i, S_j
End	:	E_i, E_j
Cheese	:	0
Blocked	:	1
Empty	:	2

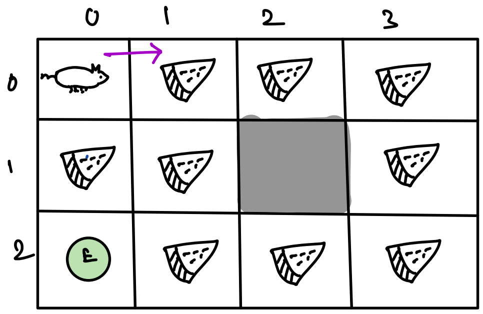


```
int countPaths(mat, N, M, si, sj, ei, ej, cheeseTotal, cheeseCurr){
        if( si < 0 || si >= N || sj < 0 || sj >= M)
                    return 0;
        if(mat[si][sj] == 1 || mat[si][sj] == -1)
                    return 0;
        if( si == ei && sj == ej){
                    if(cheeseCurr == cheeseTotal)
                                return 1;
                    return 0;
        }
        int temp = mat[si][sj]
        mat[si][sj] = -1; // Visited
        int ans = countPaths(mat, N, M, si+1, sj, ei, ej, cheeseTotal, cheeseCurr)
                        +
                  countPaths(mat, N, M, si, sj+1, ei, ej, cheeseTotal, cheeseCurr)
                        +
                  countPaths(mat, N, M, si-1, sj, ei, ej, cheeseTotal, cheeseCurr)
                        +
                  countPaths(mat, N, M, si, sj-1, ei, ej, cheeseTotal, cheeseCurr)

        mat[si][sj] = temp; // Backtracking.
        return ans;
}
```

if (mat[si][sj]==0)
cheeseCurr+1

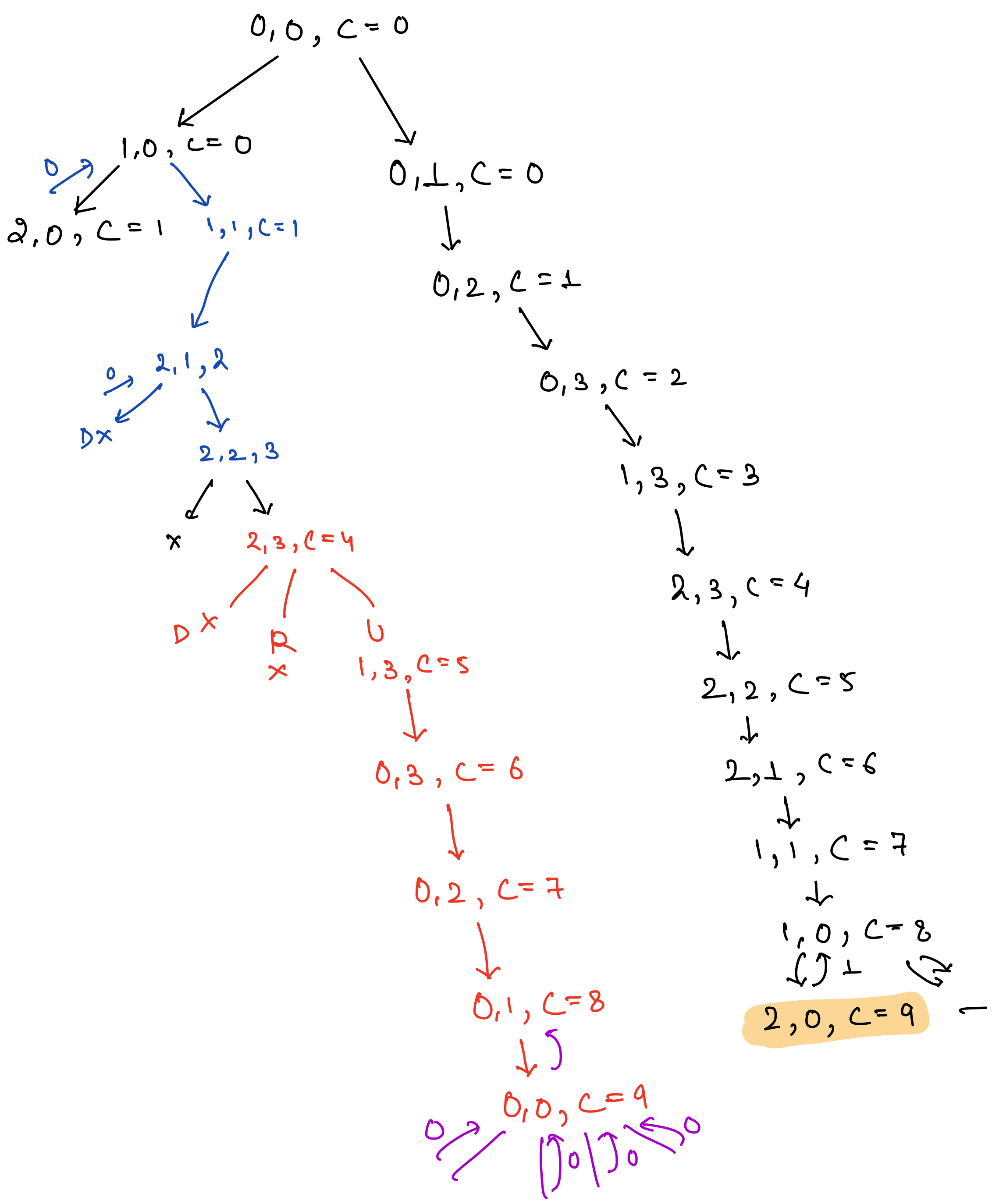
HW :- TC analysis



temp = 2

CheeseTotal = 9

1


0,0, C=0
1,0, C=0
0,1, C=0
0
2,0, C=1
1,1,C=1
0,2, C=1
2,1,2
0
D X
0,3, C=2
2,2,3
1,3, C=3
x
2,3,C=4
2,3, C=4
D X
R
x
U
1,3, C=5
2,2, C=5
2,1, C=6
0,3, C=6
1,1, C=7
0,2, C=7
1,0, C=8
1
2,0, C=9
0,1, C=8
0,0, C=9
0
0
0
0