1/9/2023

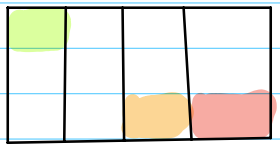
Contest 6 discussion

Q1 Unique paths 3.

Given $n * m$ grid

A[i][j]
- → 0 empty cells
- → 1 start cell (only 1)
- → 2 end cell (only 1)
- → -1 obstacles (can't walk over it)

Return the no of 4 directional walks from starting cell to ending cell, that walk over every non-obstacle cell only once.



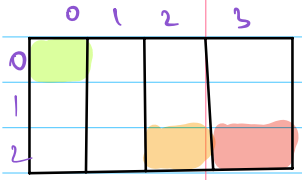
Ans = 2

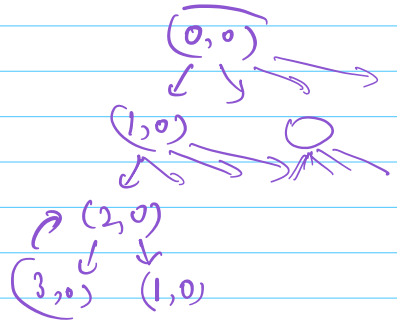
```
int empty = 1
int UP (int[][] grid){
    n = grid.length,  m = grid[0].length.
    for (i→0 to n){
        for (j→0 to m){
            if (grid[i][j] == 0){
                empty++;
            }
            else if (grid[i][j] == 1){
                sx = i;
                sy = j;
            }
        }
    }
```

```
        solve(grid, sx, sy);
    }

void solve(grid[][], x, y) {
    if(x<0 || x>=grid.length || y<0 || y>=grid[0].len ||
       grid[x][y]<0) {
        return;
    }
    if(grid[x][y] == 2) {
        if(empty == 0)
            ans++;
        return;
    }

    grid[x][y] = -2 //Do

    empty--;  //Do

    solve(grid, x+1, y);
    solve(grid, x-1, y);
    solve(grid, x, y+1);
    solve(grid, x, y-1);

    grid[x][y] = 0
    empty++;        } Undo.
}
```

① int[] r = {1, -1, 0, 0}
int[] c = {0, 0, +1, -1}

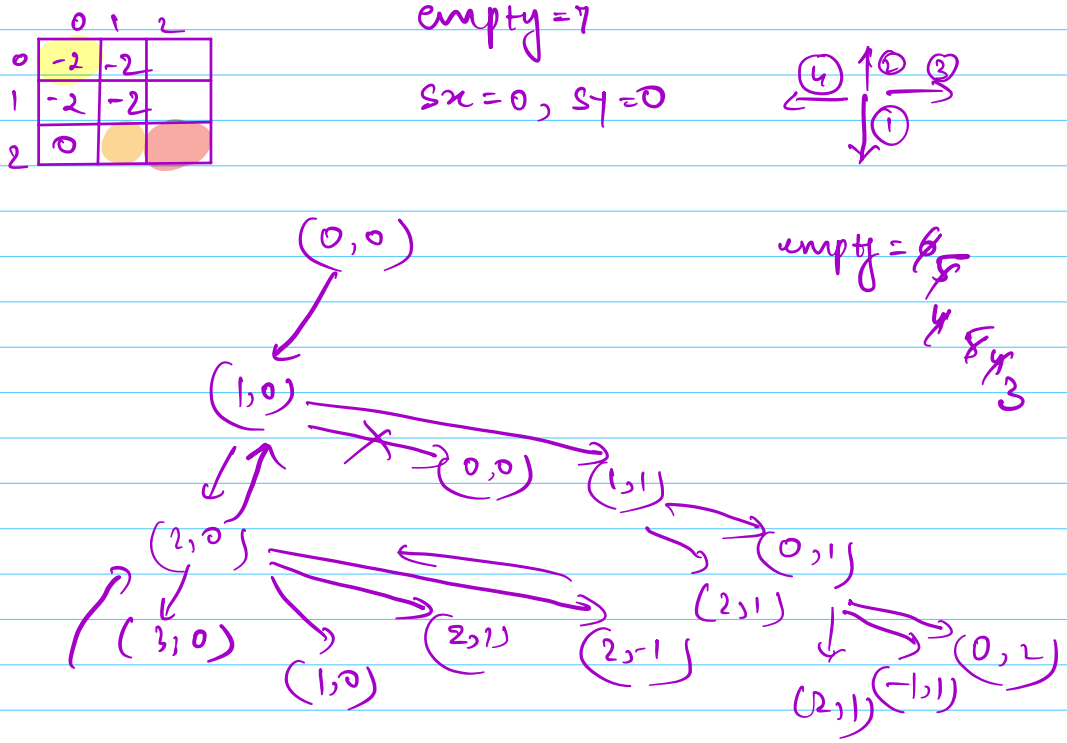
new r = x + r[i]
new c = y + c[i]





empty = 10

TC ≤ O(m*n)


0 1 2
0 -2 -2
1 -2 -2
2 0
empty = 7
sx = 0, sy = 0
4 1 2 3
(0,0)
(1,0)
(0,0)
(1,1)
(2,0)
(0,1)
(2,1)
(3,0)
(2,1)
(2,-1)
(1,0)
(0,2)
(2,1)
(-1,1)
empty = 6
4
3

Q2 Alice & pizza

→ each pizza needs some amount of time to be eaten.
→ find max no. of pizzas we can eat in B minutes.

A = [1, 2, 3], B = 3, ans = 2, $A[i] \simeq 10^9$, $B \simeq 10^9$

A = [10, 2, 6], B = 13, ans = 2

① sort the array
② add & compare.

```
int solve (int[] A, int B) {
    sort (A);
    int ans = 0;
        long
    ~~int~~ sum = 0;
    for i = 0 to N-1 {
        sum += A[i];
        if (sum > B) {
            break;
        }
        ans += 1;
    }
    return ans;
}
```

greedy

we can subtract as well.

TC: $O(N \log N + N) \simeq O(N \log N)$

B = 2×10^9
A = [2×10^9, 2×10^9]

Q3 = Wood cost

Find min cost to join woods into a single piece.

Cost of joining 2 woods = sum of len of the 2 woods.

$A[i] \approx 10^9$

$A = [1, 2, 3, 4, 4, 6]$ → $A = [10^9, 10^9, 10^9, 10^9, 10^9]$

$[3, 3, 4, 4, 6]$ → $[10^9, 10^9, 10^9, 2\times10^9]$

$[4, 4, 6, 6]$ → $[10^9, 2\times10^9, 2\times10^9]$

$[6, 6, 8]$ → $[2\times10^9, 3\times10^9]$

$[8, 12]$ → $[5\times10^9]$

$[20]$

```
Long solve (int[] A) {
    PriorityQueue<Long> pq;
    for (i → 0 to N-1)
        pq.add(A[i]);

    long ans = 0
    while (pq.size > 1) {
        long s = pq.poll() + pq.poll();
        pq.add(s)
        ans += s;
```

```
    }
    return ans;
}
```

```
for (i -> 0 to A.size(); i++) {
    Long x = new Long (A.get (i));
    pq.add (x);
}
```

Autoboxing.

int x — Integer — Object