Backtracking 2

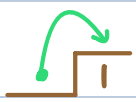
Agenda

1. Print paths in staircase problem
2. Print all paths from source to destination
3. Shortest path in a matrix with huddles

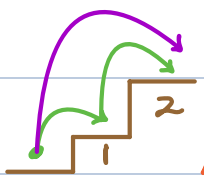
Q → climb N stairs s.t. in each step only 1 or 2 stairs can be climbed. Find # distinct ways to do the task & return all possibilities in lexicographical order.

N = 1

Ans = 1

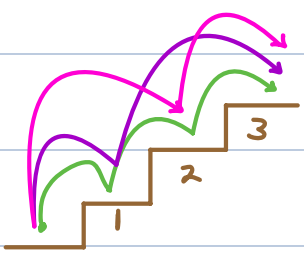


N = 2



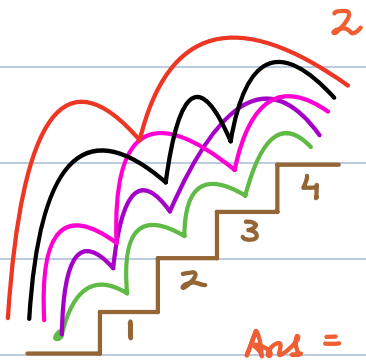
Ans = 1, 1

N = 3

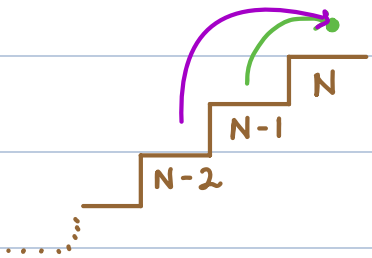


Ans = 1, 1, 1
1, 2
2, 1

N = 4



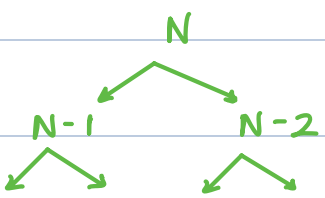
Ans = 1, 1, 1, 1
1, 1, 2
1, 2, 1
2, 1, 1
2, 2



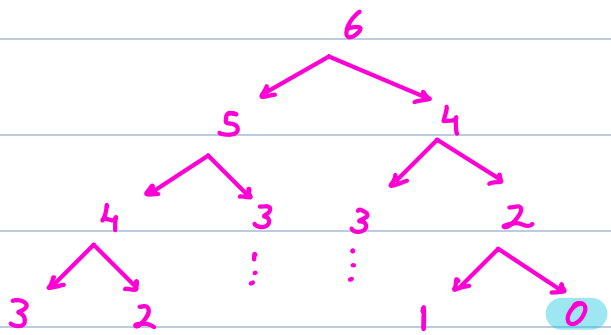
empty list

```
void paths (N, cur[]) {
    if (N == 0) {          // Base case
        print (cur)
        return
    }
    if (N >= 1) {
        cur.add (1)        // Do
        paths (N-1, cur)   // Recursion
        cur.removelast()   // Undo
    }
    if (N >=2) {
        cur.add (2)        // Do
        paths (N-2, cur)   // Recursion
        cur.removelast()   // Undo
    }
}
```

```
void paths (N, cur[]) {
    if (N < 0) return
    if (N == 0) {          // Base case
        print (cur)
        return
    }
    cur.add (1)            // Do
    paths (N-1, cur)       // Recursion
    cur.removelast()       // Undo
    cur.add (2)            // Do
    paths (N-2, cur)       // Recursion
    cur.removelast()       // Undo
}
```



TC < $O(2^N)$ SC = $O(N)$



Q → Given a N * M board.

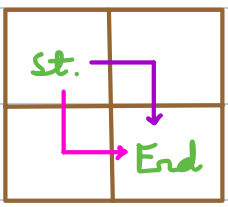
Print all the possible paths from top left to bottom right corner.

Move → Down ('D')

Right ('R')

Print paths in lexicographical order.

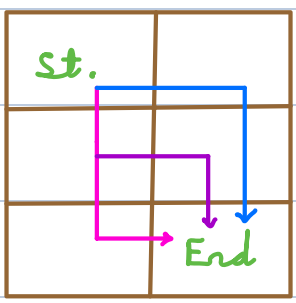
'D' < 'R'



2 * 2

Ans = D R

R D



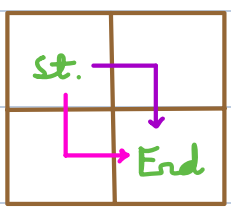
3 * 2

Ans = D D R

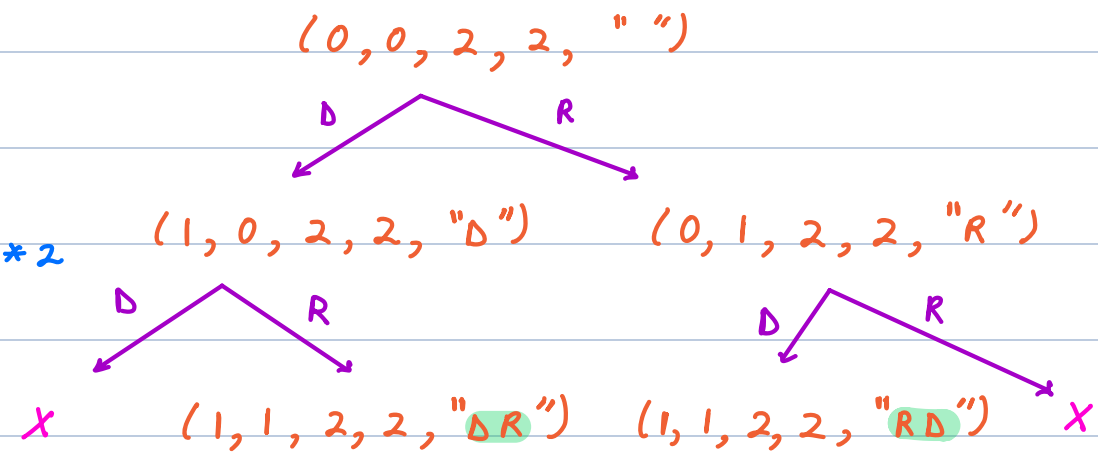
D R D

R D D

```
void maze(r, c, N, M, cur) {
    if (r >= N || c >= M) return
    if (r == N-1 && c == M-1) {
        print(cur)
        return
    }
    maze(r+1, c, N, M, cur + 'D')
    maze(r, c+1, N, M, cur + 'R')
}
```



2*2



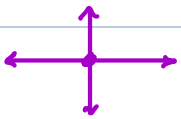
TC < $O(2^{(N+M)})$

SC = $O(N+M)$

Q → Given a N * M board.

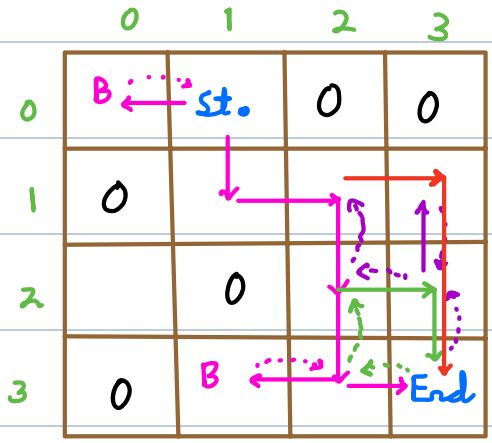
Find the length of **shortest path** from source to destination. If not possible ans = -1.

Move →

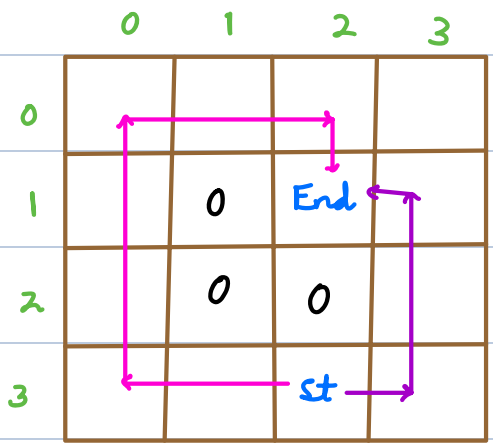


A[i][j] → 0 Blocked cell

→ 1 Empty cell



Ans = 5



Ans = 4

(r-1, c)

(r, c-1) ← (r, c) → (r, c+1)

(r+1, c)

↓ ↑ ← →

dx = [1 -1 0 0]

dy = [0 0 -1 1]

// N, M

ans = ∞

i/p — source — mark vst before calling — 0 — ∀i,j = false

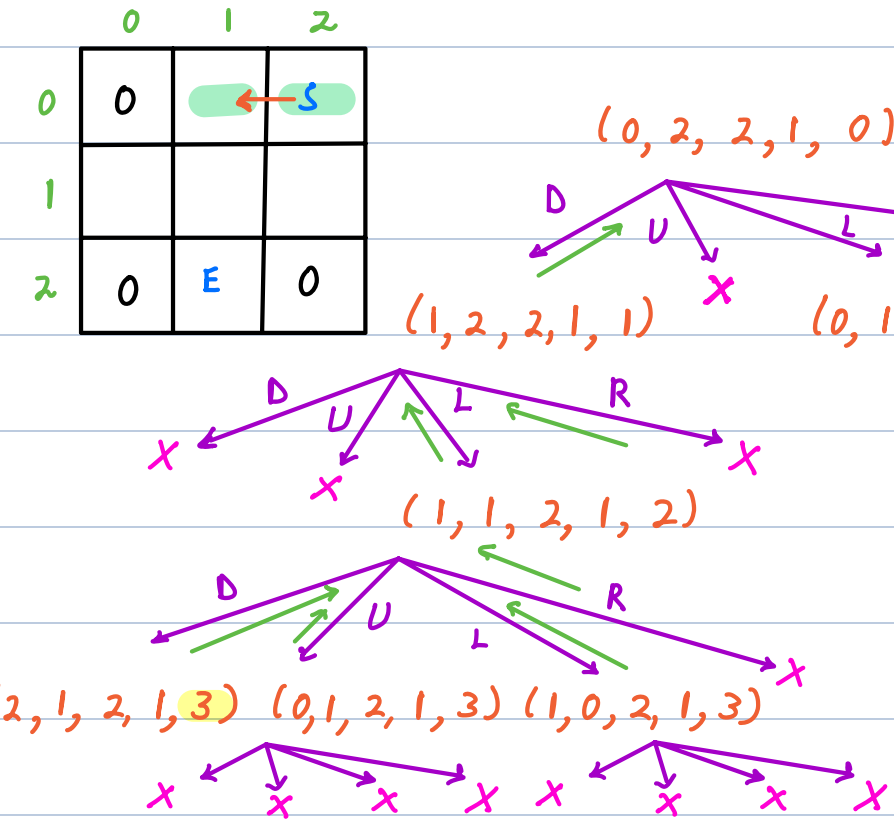
```
void distance ( A[][], r, c, dr, dc, dist, vst[][]) {

    if ( r == dr && c == dc ) {
        ans = min (ans, dist)
        return
    }

    for i → 0 to 3 {
        nr = r + dx[i]
        nc = c + dy[i]
        if ( nr < 0 || nr >= N ||
           nc < 0 || nc >= M || A[nr][nc] == 0 ||
           vst[nr][nc]) continue
        vst[nr][nc] = true
        distance ( A, nr, nc, dr, dc, dist+1, vst)
        vst[nr][nc] = false
    }

}
```

TC < $O(4^{(N*M)})$ SC < $O(N*M)$


0 1 2
0
1
2
0
S
0
E
0
(0, 2, 2, 1, 0)
D
U
L
R
X
(1, 2, 2, 1, 1)
(0, 1, 2, 1, 1)
(1, 1, 2, 1, 2)
(2, 1, 2, 1, 3)
(0, 1, 2, 1, 3)
(1, 0, 2, 1, 3)