1) $\boxed{\text{dp}[i][j]}$ = Number of ways to reach from $\underline{S}$ → $\underline{D}$? (Always)

Movements: → ↓

2) Dp state: $\text{dp}[i][j]$ = Number of ways to reach from $(0,0)$ to (i,j)

3) Dp expression: $\boxed{\text{dp}[i][j] = \text{dp}[i-1][j] + \text{dp}[i][j-1]}$

4) overlapping: ✓

$\text{dp}[i][j]$

$\text{dp}[i-1][j]$ $\text{dp}[i][j-1]$

$\text{dp}[i-2][j]$ $\text{dp}[i-1][j-1]$ $\text{dp}[i-1][j-1]$ $\text{dp}[i][j-2]$

overlapping

Order & filling

up → down

down → up

5) Dp table: 2D matrix → $\text{dp}[N][M]$

6) Base Conditions: for which are cells we cannot have an upper or left one?

(i,j)

$(i,0)$ → (i,j) will depend on which cells

$(0,j)$

$\text{dp}[i][0]$ and $\text{dp}[0][j]$ are base conditions = 1

$\text{dp}[i][0] = 1$, $\text{dp}[0][j] = 1$

$i = 0$, $j = 0$

cleaned

We need to fill data

$\text{dp}[0][0] = [\rightarrow]$

$\text{dp}[1][0] = [\rightarrow]$

$\text{dp}[2][0] = [\rightarrow]$

for (int i=1; i<N; i++){
  for (int j=1; j<M; j++){
    dp[i][j] = dp[i-1][j] + dp[i][j-1]
  }
}

Space optimization

At any time point we only need 2 rows

Dp states:

$\text{dp}[i][0] = 1$

$\text{dp}[0][j] = 1$

for (int i=1; i<N; i++){
  # Fix $\text{dp}[i][0]$, $\text{dp}[i][0] = 1$
  for (int j=1; j<M; j++){
    $\text{dp}[i][j] = \text{dp}[i-1][j] + \text{dp}[i][j-1]$
  }
}

Final ans: $\text{dp}[N-1][M-1]$

$\text{dp}[\underline{N-1}][\underline{M-1}] \rightarrow \text{dp}[\underline{(N-1)\%2}][\underline{M-1}]$

↳ Fixed down

SC: $O(M)$

Bonus: for my $\underline{\text{SC}} = O$

$N-1, M-1$

$\binom{N+M-2}{N-1}$

N+M-2 → N-1, M-1

$\boxed{\binom{(N+M-2)!}{(N-1)!\,(M-1)!}}$ or $\binom{(N+M-2)}{N-1}$ → Find an this way also

How many ways $\underline{\text{SC}} = 0$

Note: If a cell is blocked → $\text{dp}[i][j]$

1) # $\text{dp}[i][j]$ = Number of ways we can reach from $\underline{S}$ to $\underline{D}$

2) Dp expr: $\left\{ \begin{array}{l} \text{If cell is blocked, } \text{dp}[i][j] = 0 \\ \text{dp}[i-1][j] + \text{dp}[i][j-1] \end{array} \right\}$

3) overlapping ✓

4) Table: $\text{dp}[N][M]$

5) Base conditions:

0th row, 0th col

Along 0th row all values are 1 until we reach a 0

Along 0th col all values are 1 until we reach 0

6) Code

7) TC: $O(NM)$

Space → SC

SC: $O(NM)$

8) Can we do space optimization

At any time point we only need 2 rows

9) Final ans: $\text{dp}[N-1][M-1]$

How many cost 0,0 → Dest

1) $\text{dp}[i][j]$ = min cost to reach from 0,0 to (i,j)

2) Dp expression: $\text{dp}[i][j] = \{ \text{cost}[i][j] + \min(\text{dp}[i-1][j], \text{dp}[i][j-1]) \}$

3) Fill The Accounting Matrix

$\text{cost}[i][j]$

$\text{dp}[i-1][j]$ $\text{cost}[i][j]$

$\text{dp}[i][j-1]$ $\text{cost}[i][j]$

$\text{dp}[i][j]$

Backward

1) $\text{dp}[i][j]$ = Number of ways we can reach (i,j) from $\underline{0,0}$

2) Dp expression: $\text{dp}[i][j] = \{ \text{dp}[i-1][j] + \text{dp}[i][j-1] \}$ *

Dp expression: overlapping ✓

Dp expression table: which are cells can reach

$(4,4)$: $\{(2,4), (3,4), (4,1), (4,2), (4,3)\}$ → 5 ways

$(4,4)$, $(4,3)$, $(3,4)$, $(4,2)$, $(4,1)$, $(2,4)$

$(1,1)$: we need to iterate all state to find $(1,1)$ cell

$\text{dp}[0][0] = 0$

// Iterate on rows

for (int k=0; k<i; k++){
  call $\underline{\text{fn}(k,j)}$ // taken can call $\text{fn}(k,j)$ recursion (k,j)
  if (k == $\text{mat}[k][j]$ == 1){
    $\text{dp}[i][j] = \text{dp}[i][j] + \text{dp}[k][j]$
  }
}

// Iterate on column

for (int k=0; k<j; k++){
  call $\underline{\text{fn}(i,k)}$ // recursion (i,k)
  if (k == $\text{mat}[i][k]$ == 1){
    $\text{dp}[i][j] = \text{dp}[i][j] + \text{dp}[i][k]$
  }
}

Dp expression

Dp expression

2) TC

Note: If duplicate → TC & calculate

$(N^2)^2 = (N^4)$

Bonus TC: $O(N^3 + NM^2)$

↳ cubic

SC: $O(NM)$

$\text{dp}[i][j]$ = # ways $(0,0) \rightarrow (i,j)$ Directions →

Eg: $\text{mat}[1][1] = 2$ → right → $(1,4)$

down

$(2,2)$

Eg: $\text{mat}[2][2] = 0$ → right → $(2,5)$

down

$(3,5)$ → our # nodes

Note: If a cell is 0 we can't reach it, but we consider it from the

Dp:

$(0,0) \rightarrow (0,1)$: $(0,0) \rightarrow (0,1) \rightarrow (1,1)$

$(0,0) \rightarrow (1,1)$: $(0,0) \rightarrow (1,1) \rightarrow (1,2) \rightarrow (1,3) \rightarrow (1,4)$

$(0,0) \rightarrow (2,1)$: $(0,0) \rightarrow (2,1) \rightarrow (2,2) \rightarrow (2,3) \rightarrow (2,4) \rightarrow (2,5)$

1) $\text{dp}[i][j]$ = Number of ways we can reach (i,j) from $\underline{0,0}$

2) Dp expression: $\text{dp}[i][j] = \{ \text{dp}[i-1][j] + \text{dp}[i][j-1] \}$ *

Dp expression: overlapping ✓

$\text{dp}[0][0] = 0$

// Iterate on rows

for (int k=0; k<i; k++){
  call $\underline{\text{fn}(k,j)}$ // taken can call $\text{fn}(k,j)$ recursion (k,j)
  if (k == $\text{mat}[k][j]$ == 1){
    $\text{dp}[i][j] = \text{dp}[i][j] + \text{dp}[k][j]$
    // and ways
    you can reach
    since 0,0 for (i,j)
  }
}

// Iterate on column

for (int k=0; k<j; k++){
  call $\underline{\text{fn}(i,k)}$ // recursion (i,k)
  if (k == $\text{mat}[i][k]$ == 1){
    $\text{dp}[i][j] = \text{dp}[i][j] + \text{dp}[i][k]$
  }
}

2) TC

Note: If duplicate → TC & calculate

$(N^2)^2 = (N^4)$

Bonus TC: $O(N^3 + NM^2)$

↳ cubic

SC: $O(NM)$

$\text{dp}[i][j]$ = # ways $(0,0) \rightarrow$ Dest

$\text{dp}[i][j]$ = # ways to go Dest from (i,j)

call (i,j) from $(2,3)$ how many ways we can reach destination

$(2,3) \rightarrow (2,5) \rightarrow (2,4)$

$(2,3) \rightarrow (3,3) \rightarrow (3,5) \rightarrow (3,4)$

$(2,3) \rightarrow (3,3) \rightarrow (3,5) \rightarrow (4,5)$

// $\text{dp}[i][j] = \text{dp}[i-1][j] + \text{dp}[i][j+1]$

We need to fill $\text{dp}[i][j]$

Find x, $\max(\text{dp}[i][j])$

$\text{dp}[i][j] = \text{dp}[i+x][j] + \text{dp}[i][j+x]$

$\{ 0 \le x < N \}$

$\{ 0 \le x < M \}$

Dp expression:

1) Table

2) Base condition

3) Code:

can go right

can go bottom

order & filling

R → L, Bottom → top

B → T, top → bottom

$\text{dp}[N-1][M-1] = 1$;

for (int i=N-1; i>=0; i--){
  for (int j=M-1; j>=0; j--){
    if (i == N-1 && j == M-1) { continue; } // skip base condition
    $\text{dp}[i][j] = 0$
    for (int x = $\max(i,j)$; x<N; x++){
      if (i+x < N) { $\text{dp}[i][j] += \text{dp}[i+x][j]$ }
      if (j+x < M) { $\text{dp}[i][j] += \text{dp}[i][j+x]$ }
    }
  }
}

Base cases, $\text{dp}[i][j] = 1$? (because from this we can reach dest)

$\text{dp}[N-1][M-1] = 1$

→ (i,j) to can depend on random state

$(i,j+x)$

Final ans: $\text{dp}[1][1]$

TC: $O(N^3 M)$ SC: $O(N^2 M)$

$(1,1)$

Dungeons & Dragons

1) $\text{mat}[i][j]$: Health gained or lost at cell

2) Find min health we need to start at source to think we reach destination

3) Note: At some point health ≤ 0 can die

4) From a cell →

Case 1:

Case 2:

Case 3:

Case 4:

Case 5:

$\text{dp}[i,j] = \{ \min(\text{dp}(i+1,j), \text{dp}(i,j+1)) - \text{mat}[i][j] \}$

Solution we need to think to left right down, bottom right

Solution what he already have at $\text{mat}[i][j]$

$\text{dp}[i,j] = \max(1, \{ \min(\text{dp}(i+1,j), \text{dp}(i,j+1)) - \text{mat}[i][j] \})$

dp expression

$(i,j) \rightarrow (i,j+1)$

$(i+1,j)$

order & filling

R → L

Bottom → up