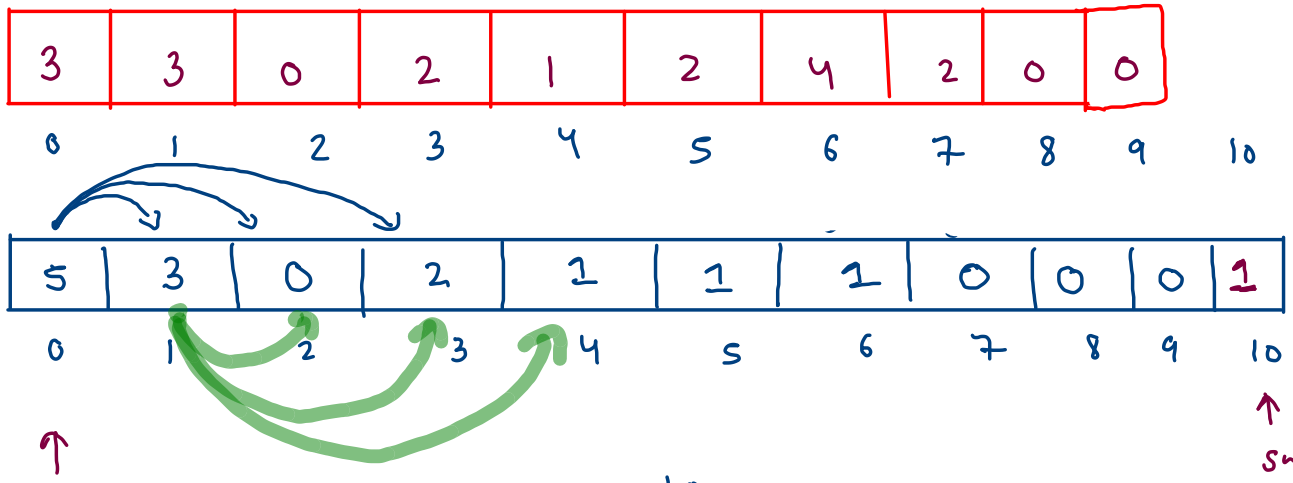
# Climb stair with variable jumps

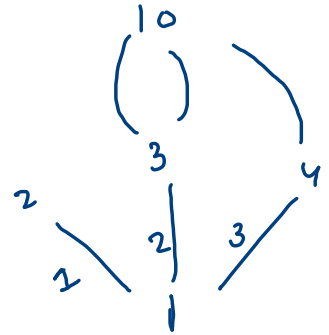



10 3 3 0 2 1 2 4
2 0 0

(i) create string

(ii) assign meaning

(iii) transit & solve




dp[i] -> i to n

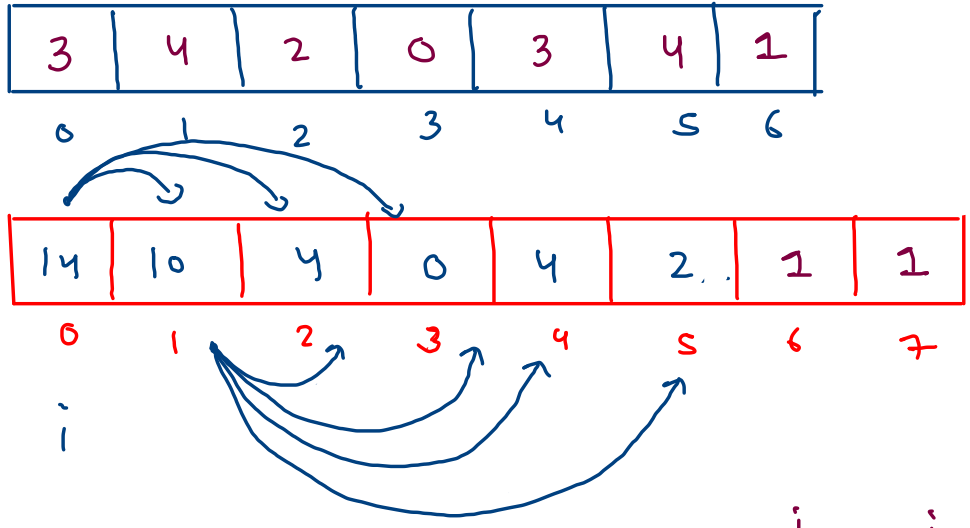
no. of ways

0 + 2 + 1

ans -> dp[0].

```java
public static int csvj_tab(int[]jumps) {
    int n = jumps.length;
    int[]dp = new int[n+1];

    //dp[i] -> i to n number of ways

    dp[n] = 1;

    for(int i=n-1; i >= 0;i--) {
        for(int j = 1; j <= jumps[i] && i+j <= n;j++) {
            dp[i] += dp[i + j];
        }
    }

    return dp[0];
}
```

n = 7



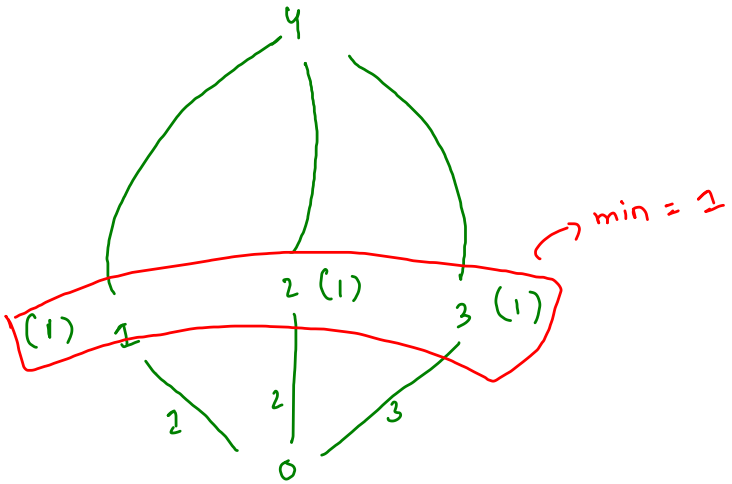

i -> i to n ways

ans -> 14

| 3 | 3 | 0 | 2 | 1 | 2 | 4 | 2 | 0 | 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

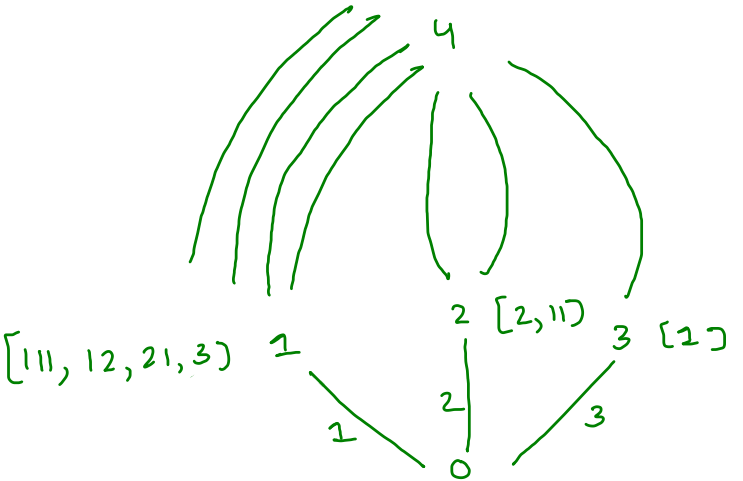
10

min-moves



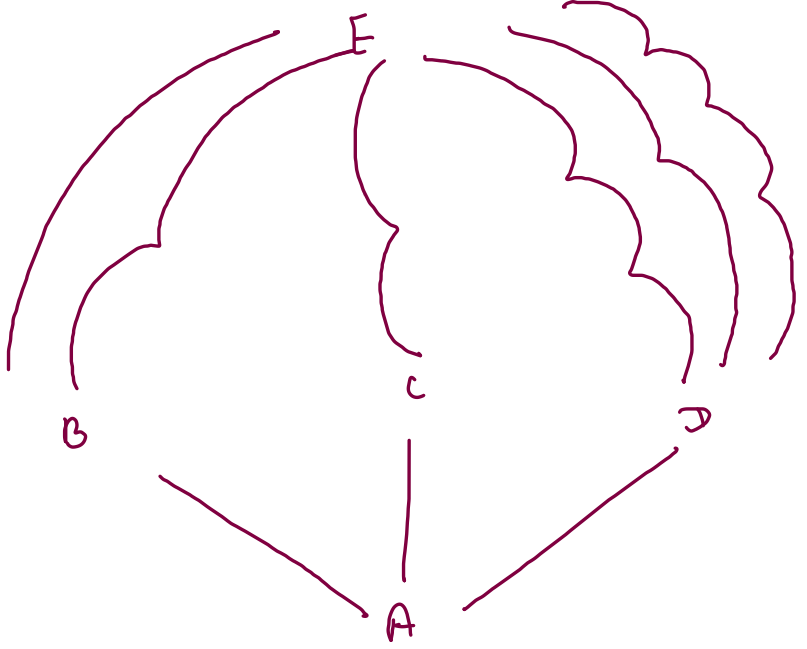

0 to 4 = 1+1 = 2

Climb stairs




0 to 4 = [1111, 121, 211, 31, 22, 112, 13]

[7]

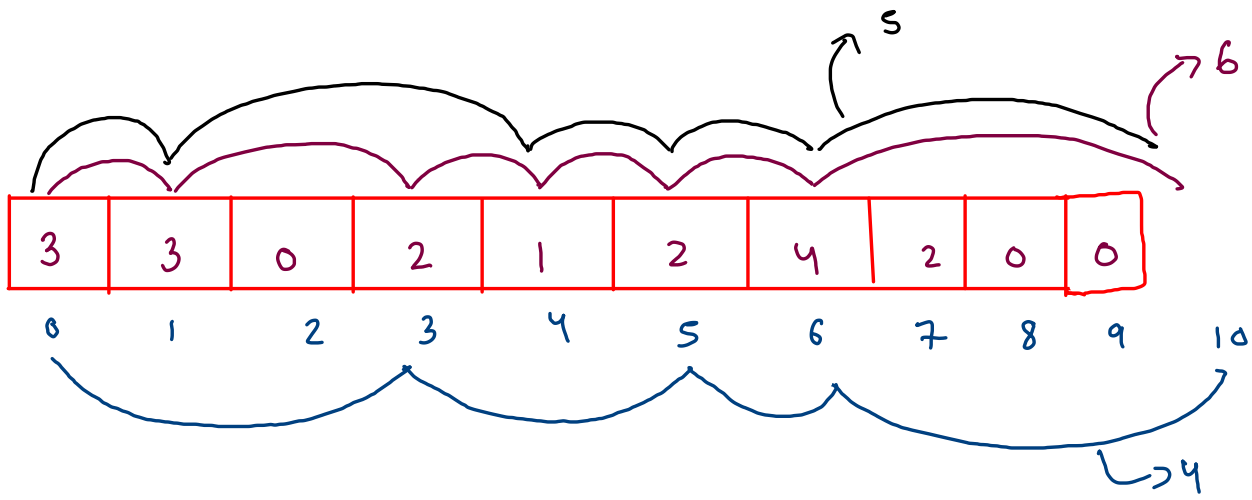



A to E = B to E + C to E + D to E

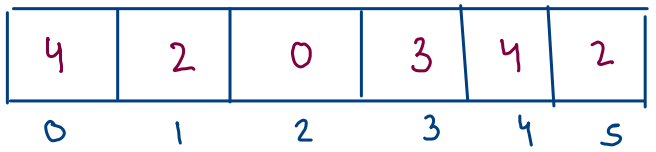
(ways) (ways) (ways) (ways)

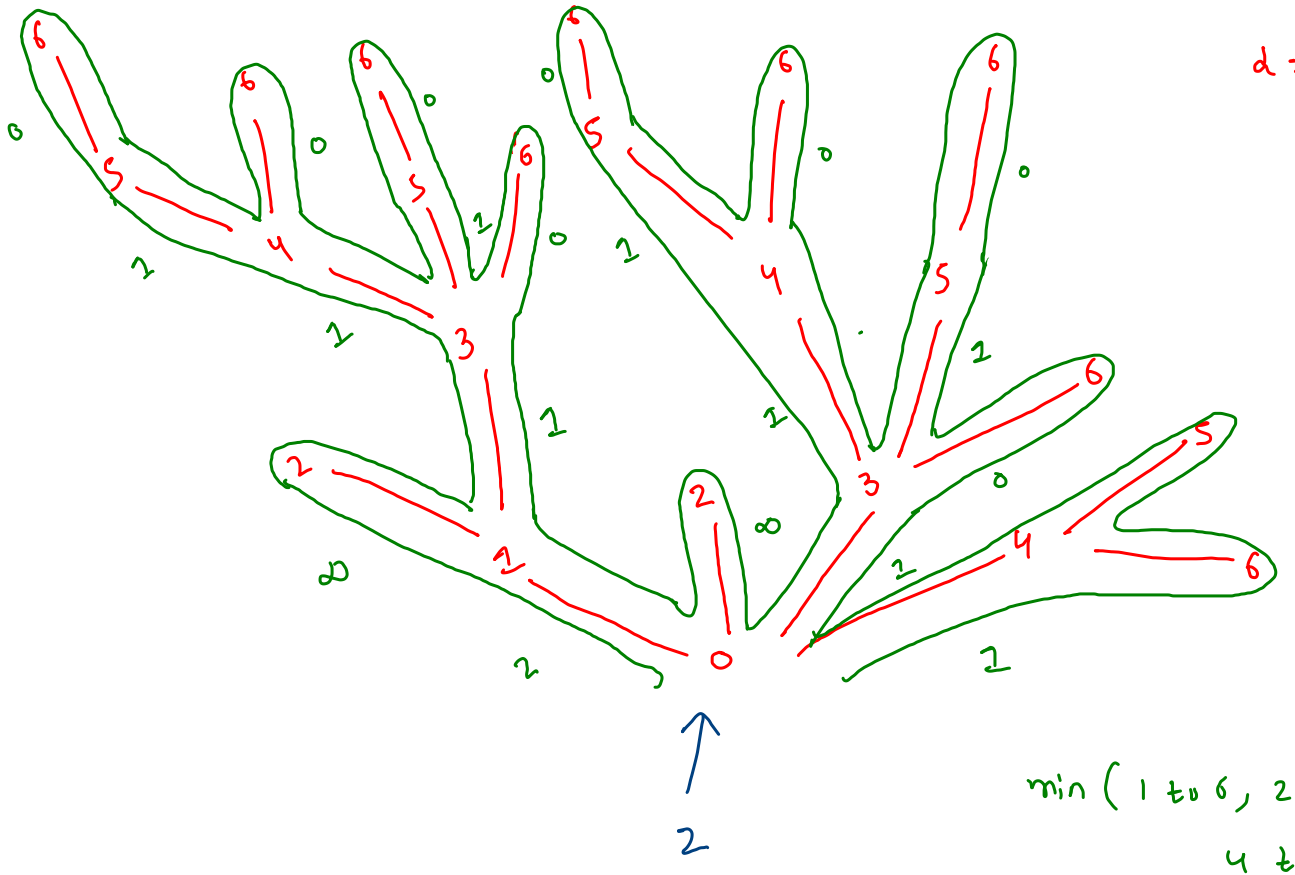
A to E = min(B to E , C to E , D to E ) + 1

(min moves) (min moves) (min moves) (min moves)




4 min moves



d = 6

ways
n to n -> 1

moves
n to n -> 0

min ( 1 to 6, 2 to 6, 3 to 6
4 to 6)

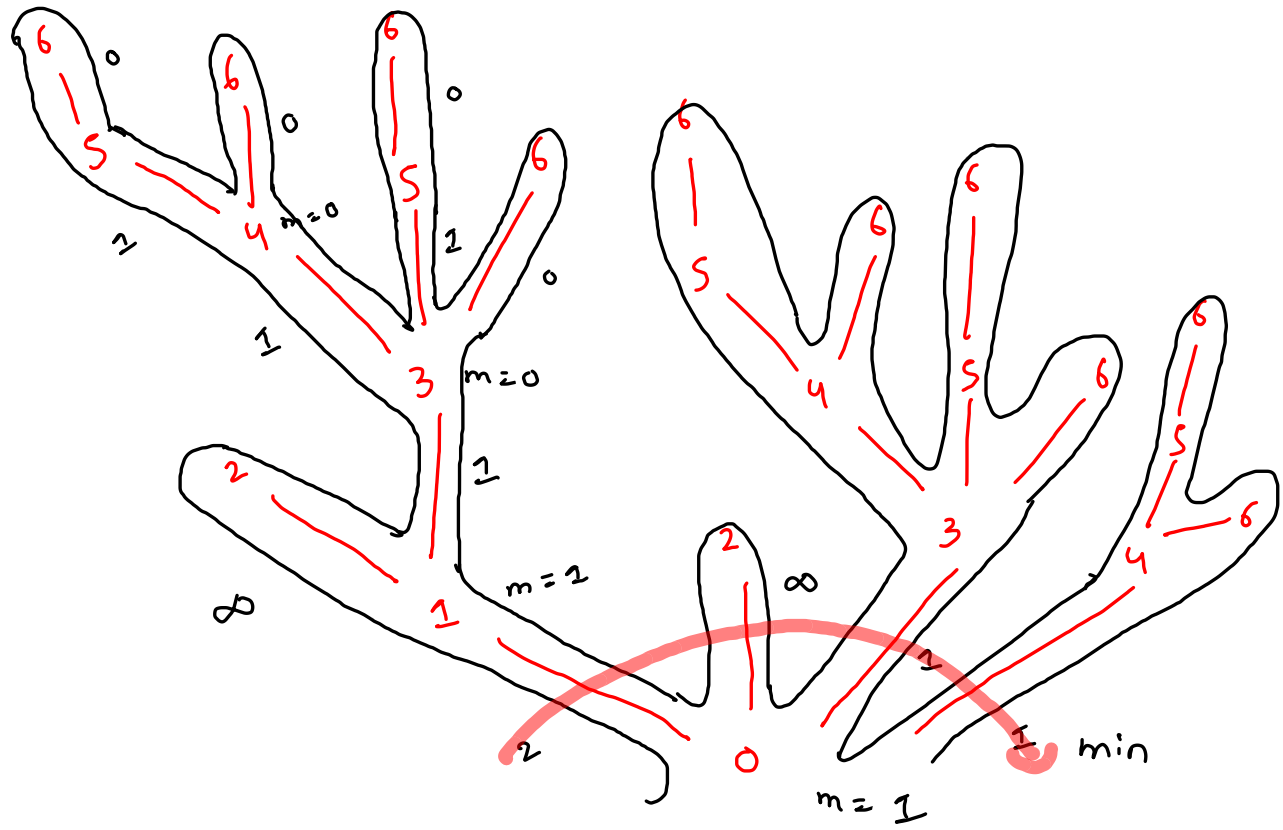
-> 1

jumps

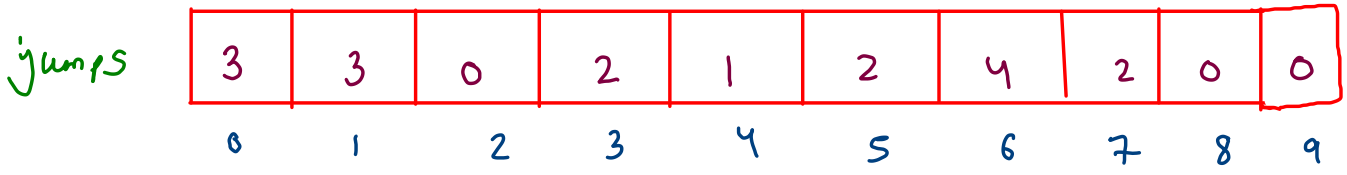
| 4 | 2 | 0 | 3 | 4 | 2 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

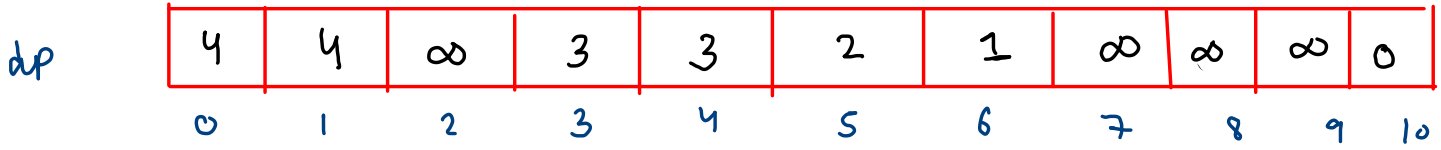
q.6

```java
public static int csvj_mm_rec(int src,int[]jumps) {
    if(src == jumps.length) {
        return 0;
    }

    int min = Integer.MAX_VALUE;

    for(int k = 1; k <= jumps[src] && src + k <= jumps.length;k++) {
        int mmntod = csvj_mm_rec(src + k,jumps); //min moves from nbr to dest

        if(mmntod < min) {
            min = mmntod;
        }
    }

    if(min == Integer.MAX_VALUE) {
        return Integer.MAX_VALUE;
    }
    else {
        return min+1;
    }
}
```



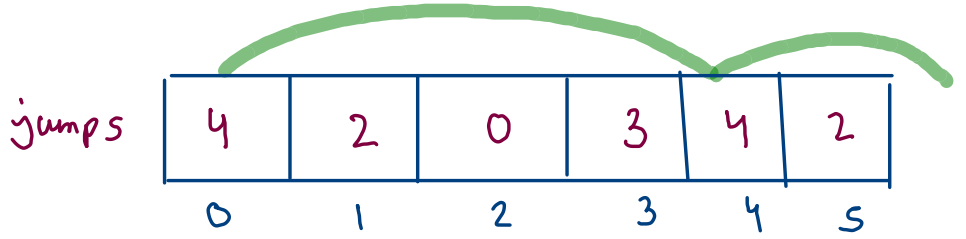

ans = 2

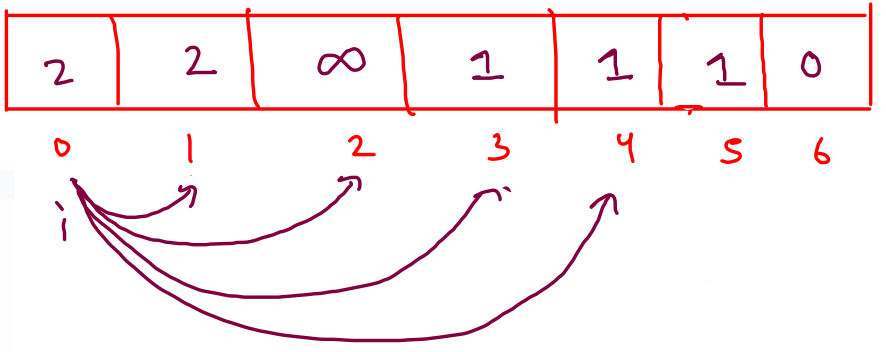



dp[i] -> i to dest

min moves




min = ~~∞~~ 1

k = 1 to 4

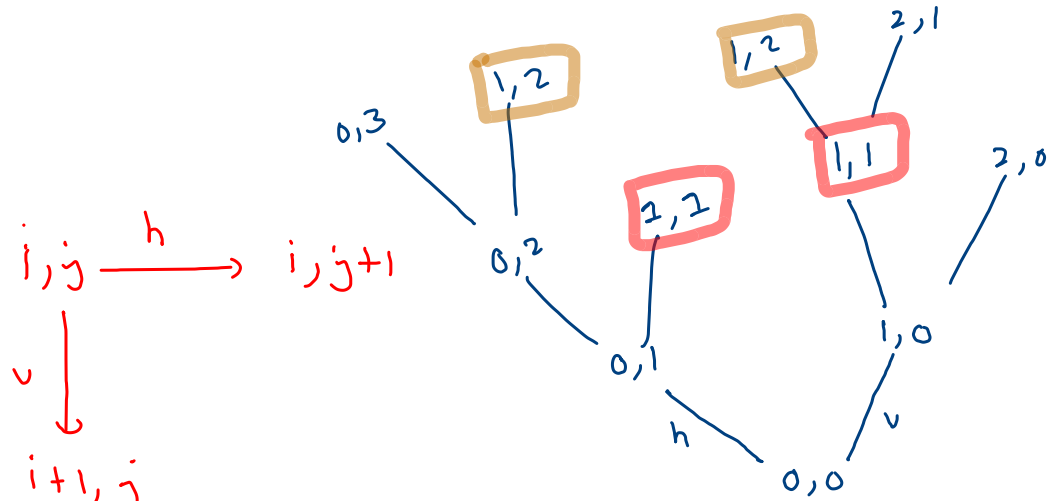
```java
for (int i = n - 1; i >= 0; i--) {
    int min = Integer.MAX_VALUE;

    for (int k = 1; k <= jumps[i] && i + k <= n; k++) {
        int nbr = i + k;
        int mmntod = dp[nbr];

        if (mmntod < min) {
            min = mmntod;
        }
    }

    if (min == Integer.MAX_VALUE) {
        dp[i] = Integer.MAX_VALUE;
    } else {
        dp[i] = min + 1;
    }
}
```

# min cost in maze traversal

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 1 | 2 | 0 | 3 |
| 1 | 4 | 6 | 7 | 1 |
| 2 | 3 | 10 | 12 | 4 |
| 3 | 8 | 9 | 3 | 15 |
| 4 | 4 | 6 | 7 | 10 |

src -> 0,0

dest -> 4,3




i,j to dest -> min [ (i,j+1 to dest) , (i+1,j to dest) ]

min cost — min cost — min cost

+ cost[i][j]

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 1 | 2 | 0 | 3 |
| 1 | 4 | 6 | 7 | 1 |
| 2 | 3 | 10 | 12 | 4 |
| 3 | 8 | 9 | 3 | 15 |
| 4 | 4 | 6 | 7 | 10 |

Cost

| | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 36 | 35 | 33 | 33 |
| 1 | 42 | 43 | 37 | 30 |
| 2 | 38 | 39 | 32 | 29 |
| 3 | 35 | 29 | 20 | 25 |
| 4 | 27 | 23 | 17 | 10 |

DP

$dp[i][j] = \min(dp[i+1][j], dp[i][j+1]) + cost[i][j]$

$dp[i][j]$ -> min cost from $(i,j)$ to dest