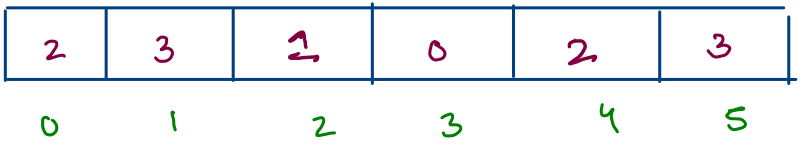
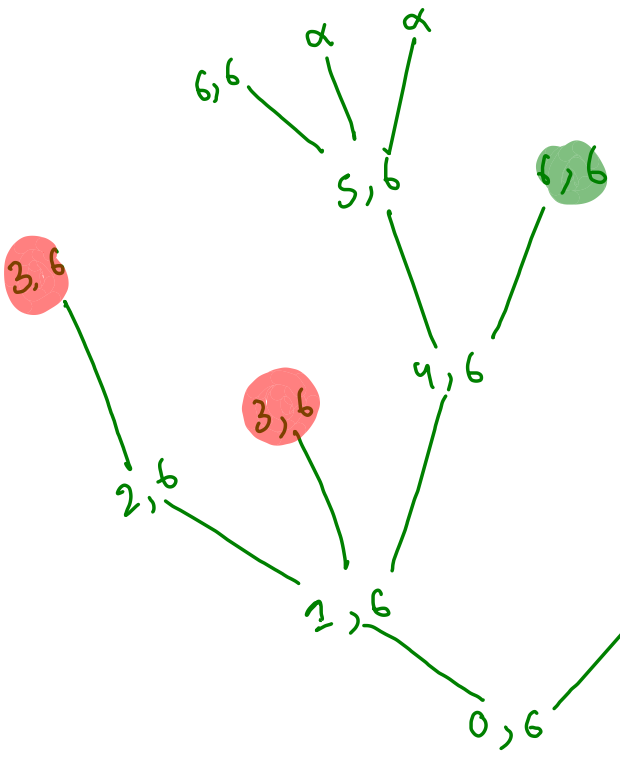
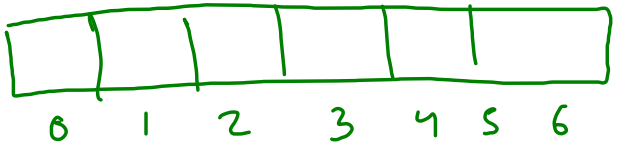



n = 6

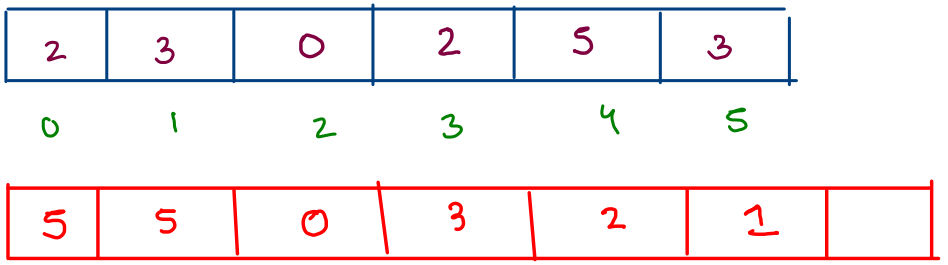


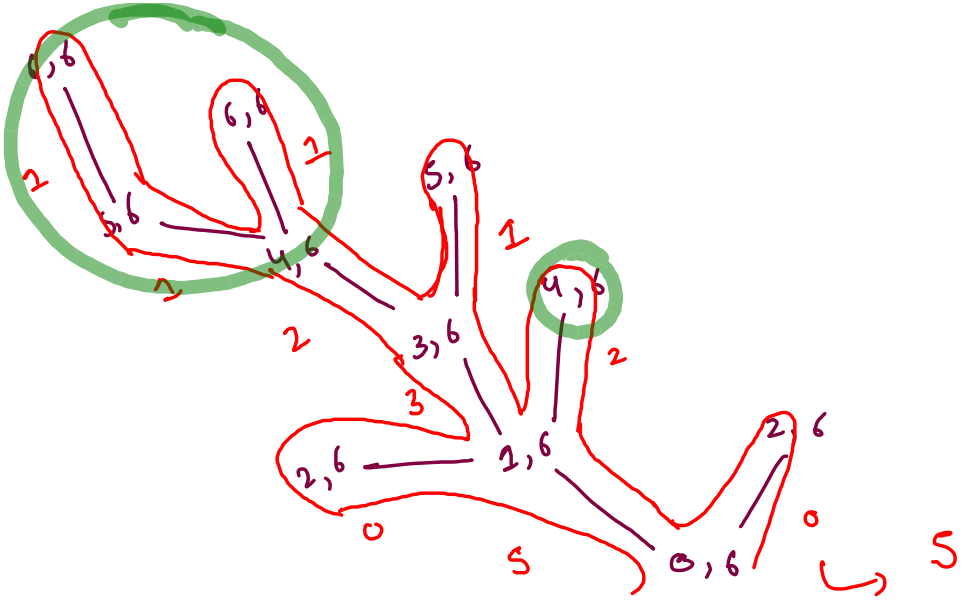



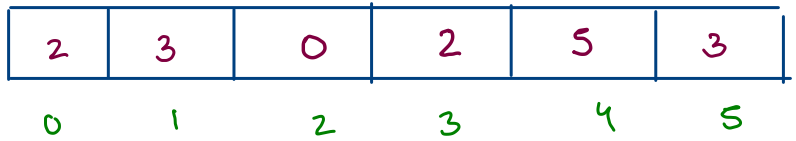

q,b

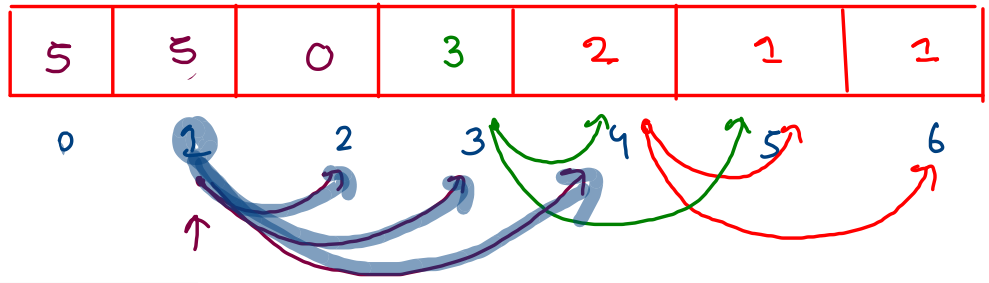
src, dest

```java
public static int CSVJ_mem(int[]jumps,int src,int dest,int[]qb) {
    if(src == dest) {
        return 1;
    }

    if(src > dest) {
        return 0;
    }

    if(qb[src] != -1) {
        return qb[src];
    }

    int stod = 0;
    for(int s = 1; s <= jumps[src];s++) {
        int ntod = CSVJ_mem(jumps,src + s,dest,qb);
        stod += ntod;
    }

    qb[src] = stod;

    return stod;
```





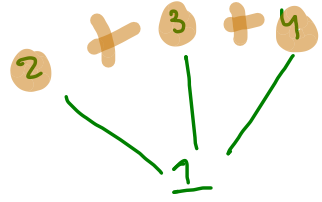

```java
public static int CSVJ_tab(int[]jumps) {
    int n = jumps.length;
    int[]dp = new int[n+1];

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

    dp[n] = 1;

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

    return dp[0];
}
```

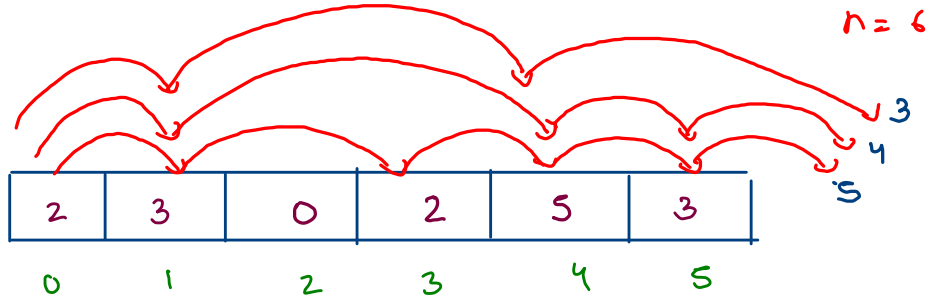
6 dp[i] -> i to n ways

s = 1

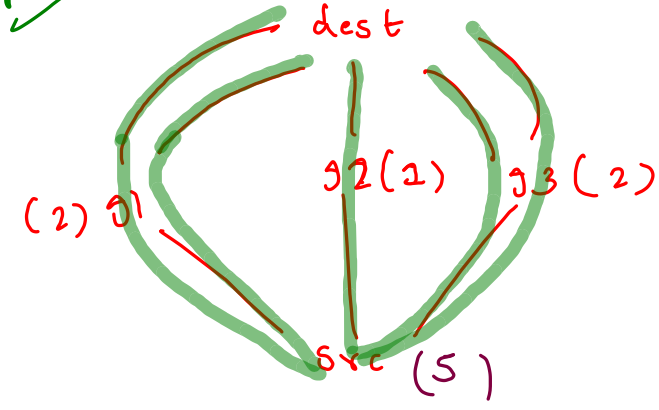
s = 2

s = 3

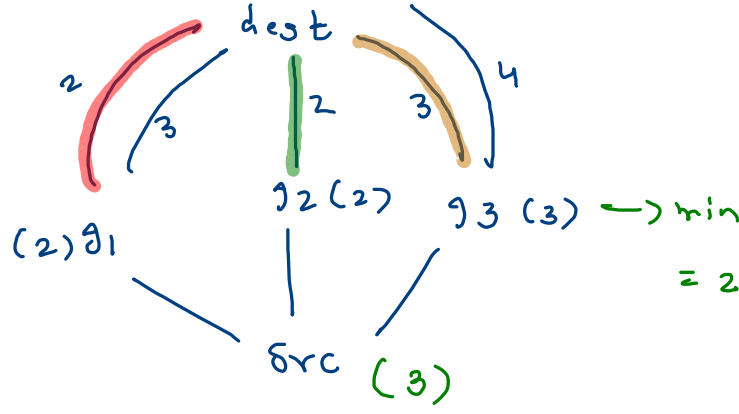


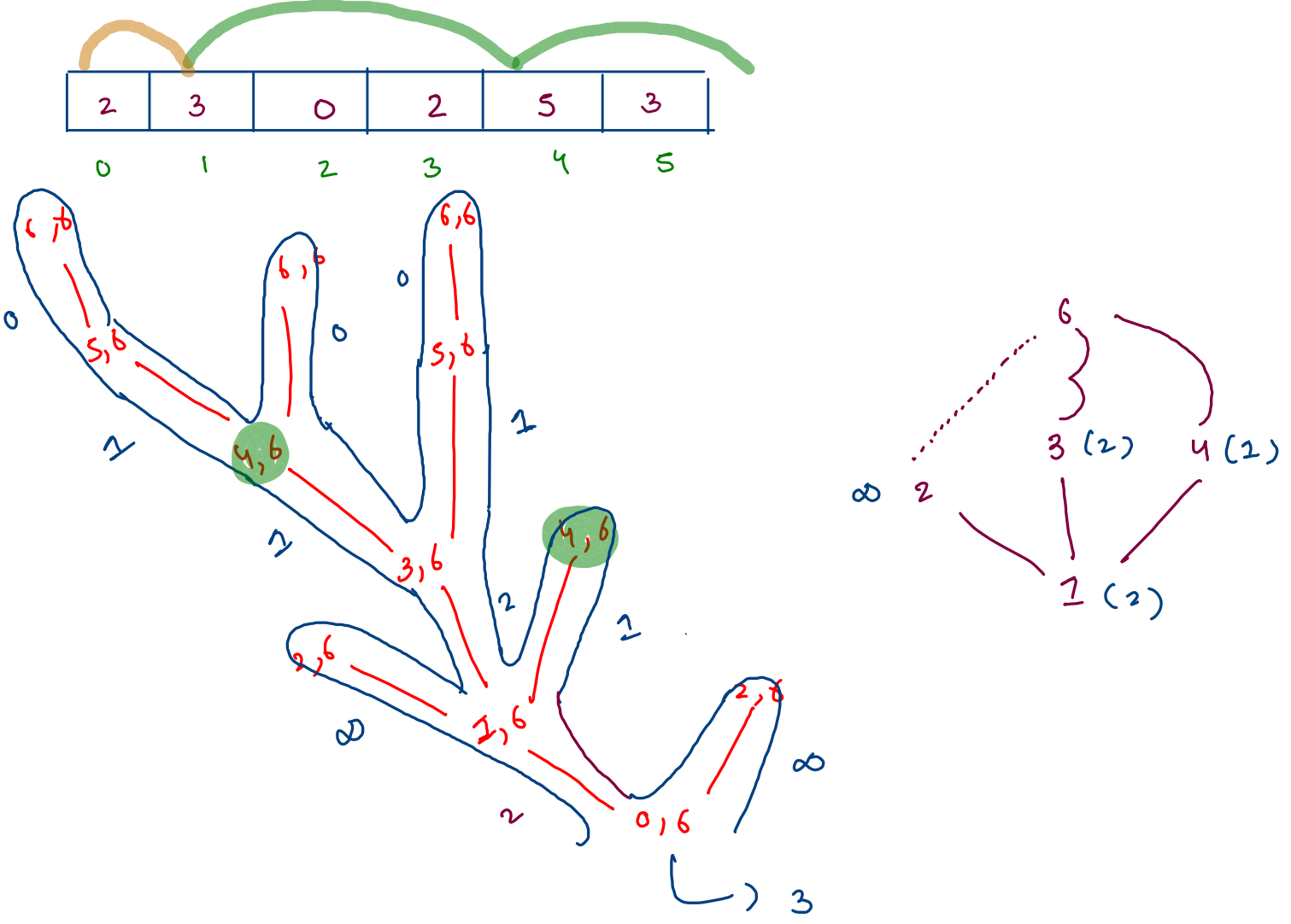



paths

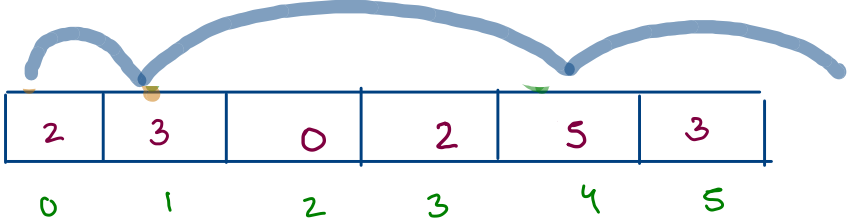


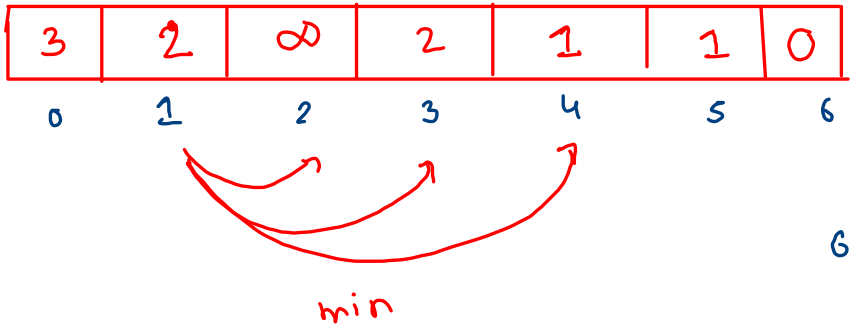

min-jumps




2 3 0 2 5 3
0 1 2 3 4 5
6,6
5,6
4,6
3,6
2,6
1,6
0,6
2,6
0
1
2
∞
3


6
3 (2)
4 (1)
∞
2
1 (2)

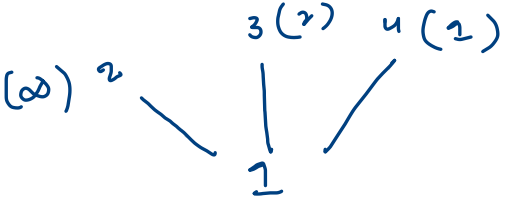


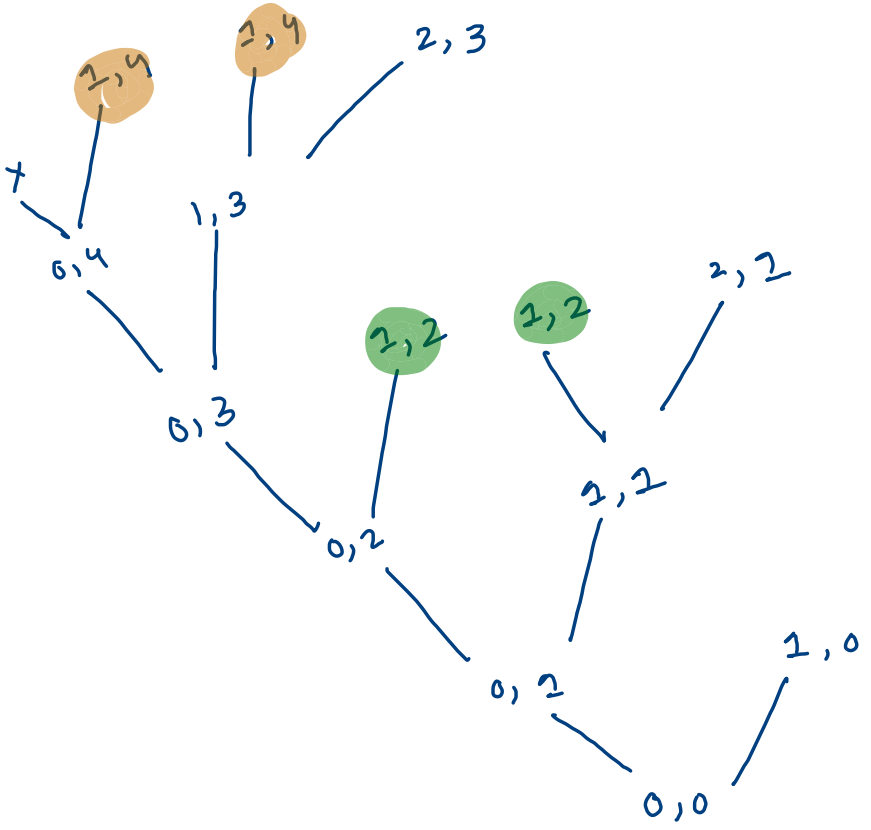

dp[i] -> i to n

6

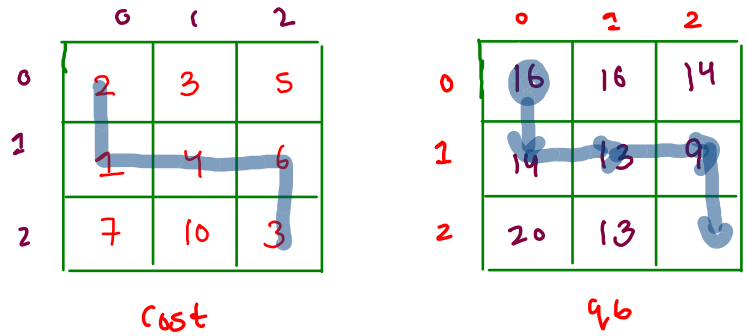
min_moves



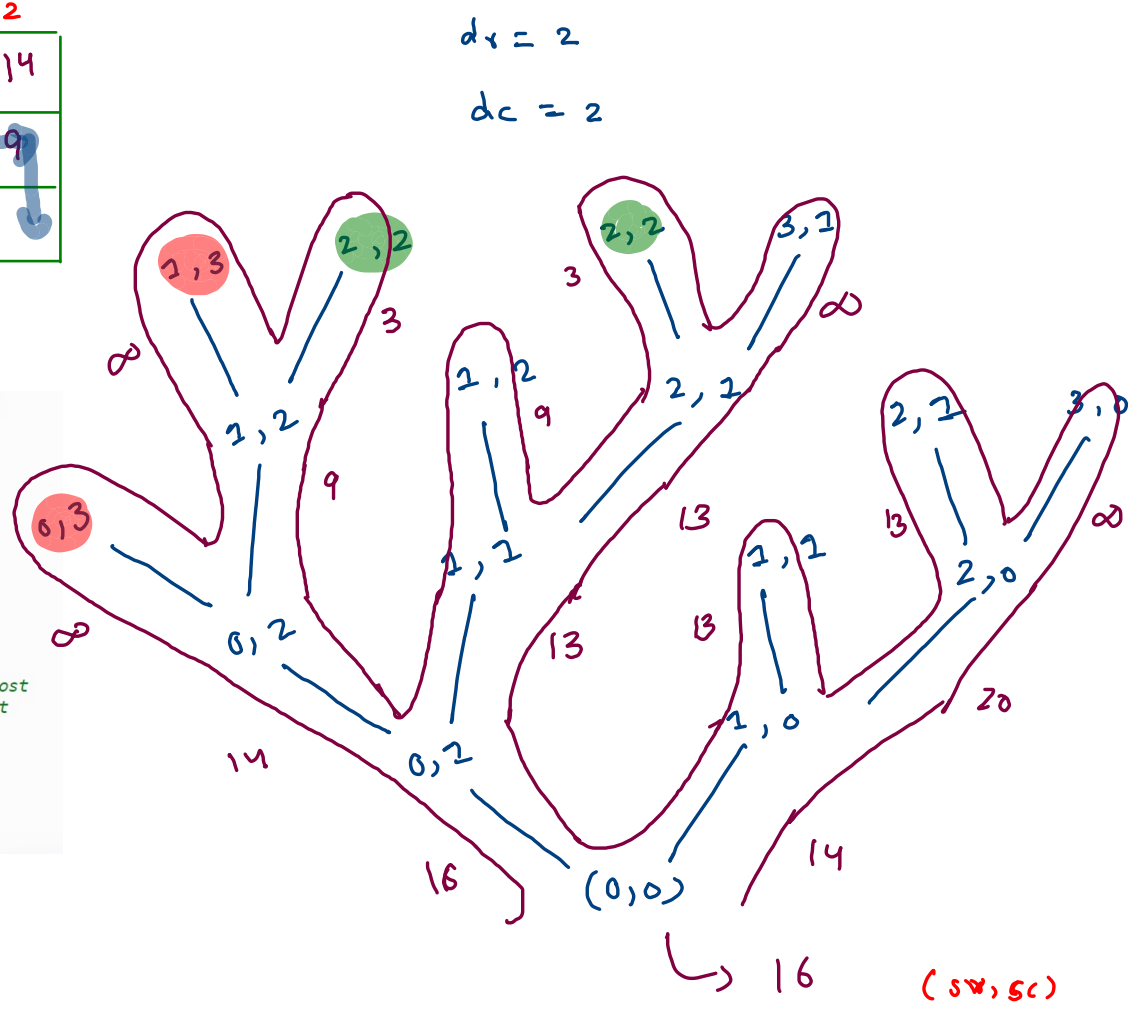
| | | | | |
|---|---|---|---|---|
| 2 | 4 | 3 | 6 | 10 |
| 2 | 9 | 8 | 2 | 9 |
| 11 | 7 | 1 | 6 | 3 |
| 10 | 22 | 4 | 15 | 4 |

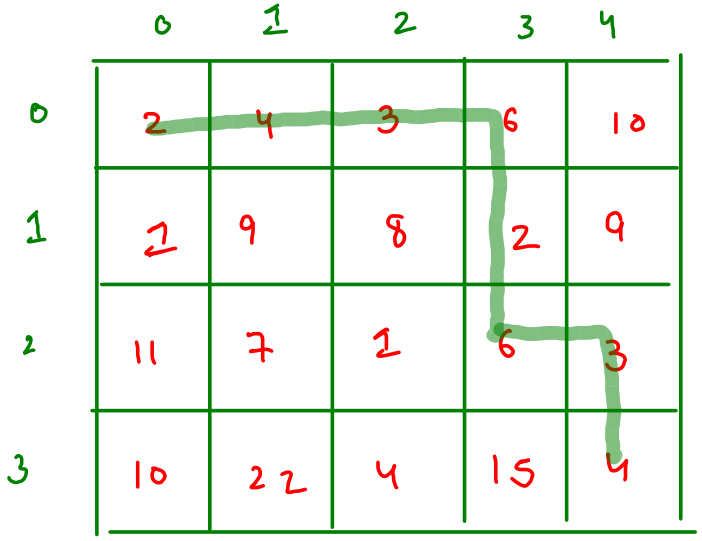



q,b -> 22

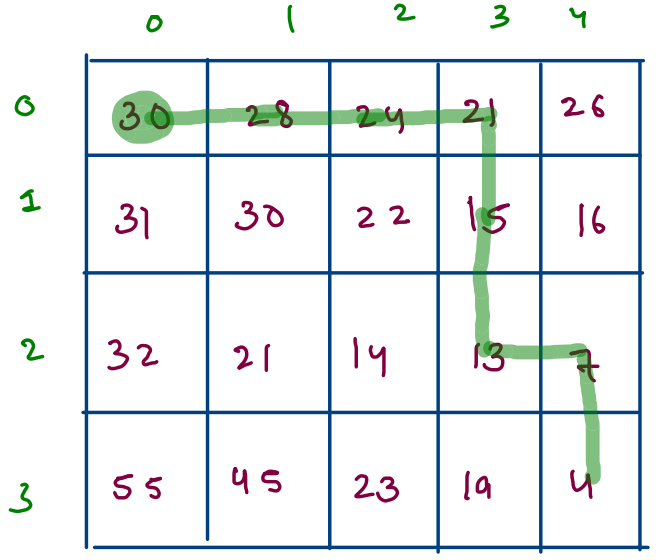



```java
public static int mazeMinCost_mem(int[][]cost,int sr,int sc,int dr,int dc,int[][]qb) {
    if(sr == dr && sc == dc) {
        return cost[sr][sc];
    }

    if(sr > dr || sc > dc) {
        return Integer.MAX_VALUE;
    }

    if(qb[sr][sc] != 0) {
        return qb[sr][sc];
    }

    int hnc = mazeMinCost_mem(cost,sr,sc + 1,dr,dc,qb); //horizontal nbr to dest min Cost
    int vnc = mazeMinCost_mem(cost,sr + 1,sc,dr,dc,qb); //vertican nbr to dest min cost

    int stodmc = Math.min(hnc,vnc) + cost[sr][sc]; //src to dest min cost
    qb[sr][sc] = stodmc;

    return stodmc;
}
```






cost




dp

dp[i][j] → i, j to dest mincost

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