moves =

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

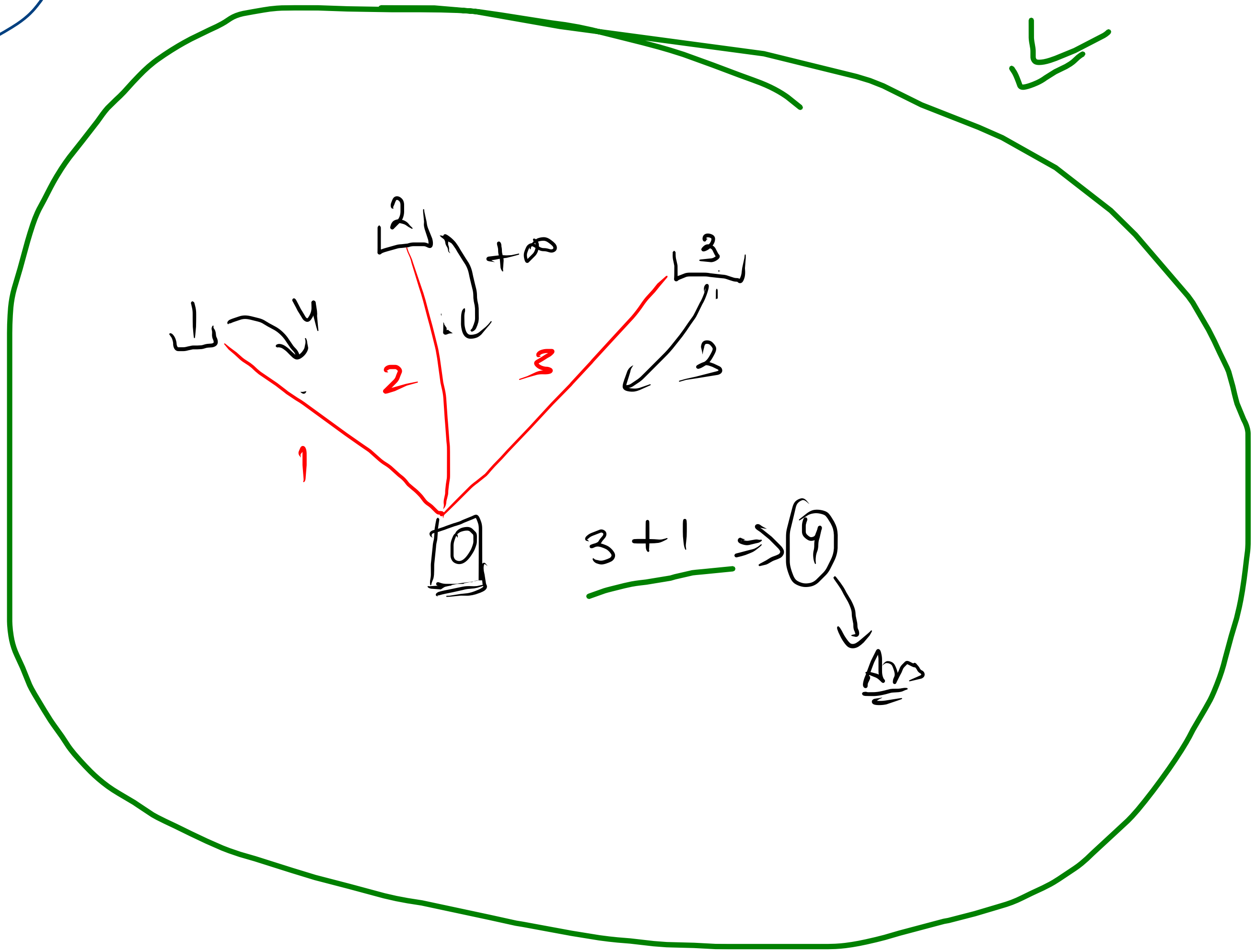
dest

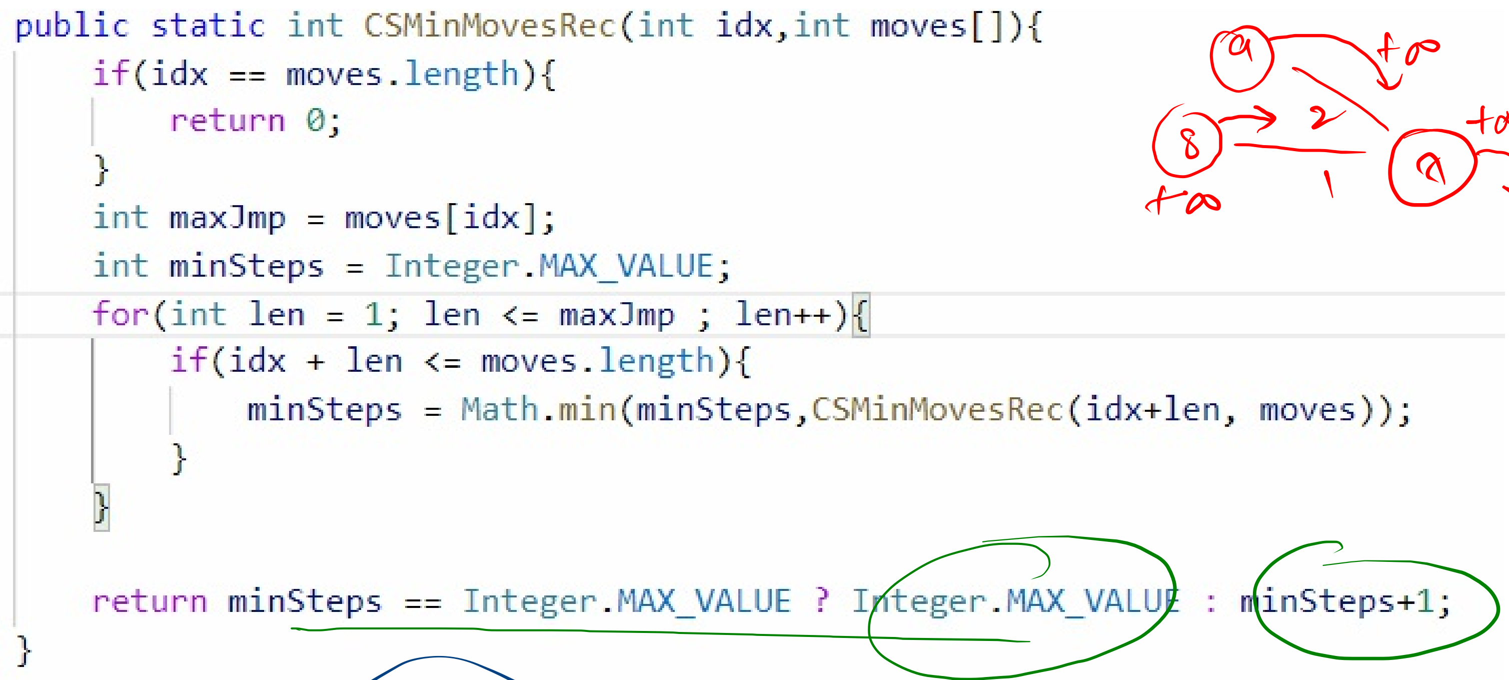
Min moves to reach dest

dest = 10

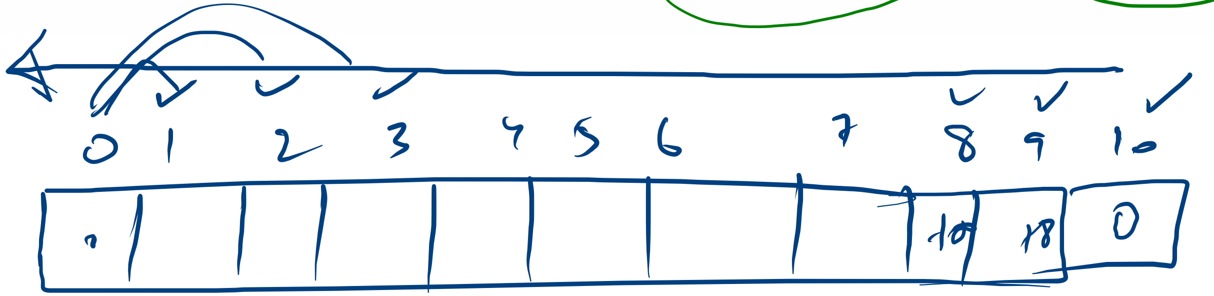
0 → dest (work)

dest = 10



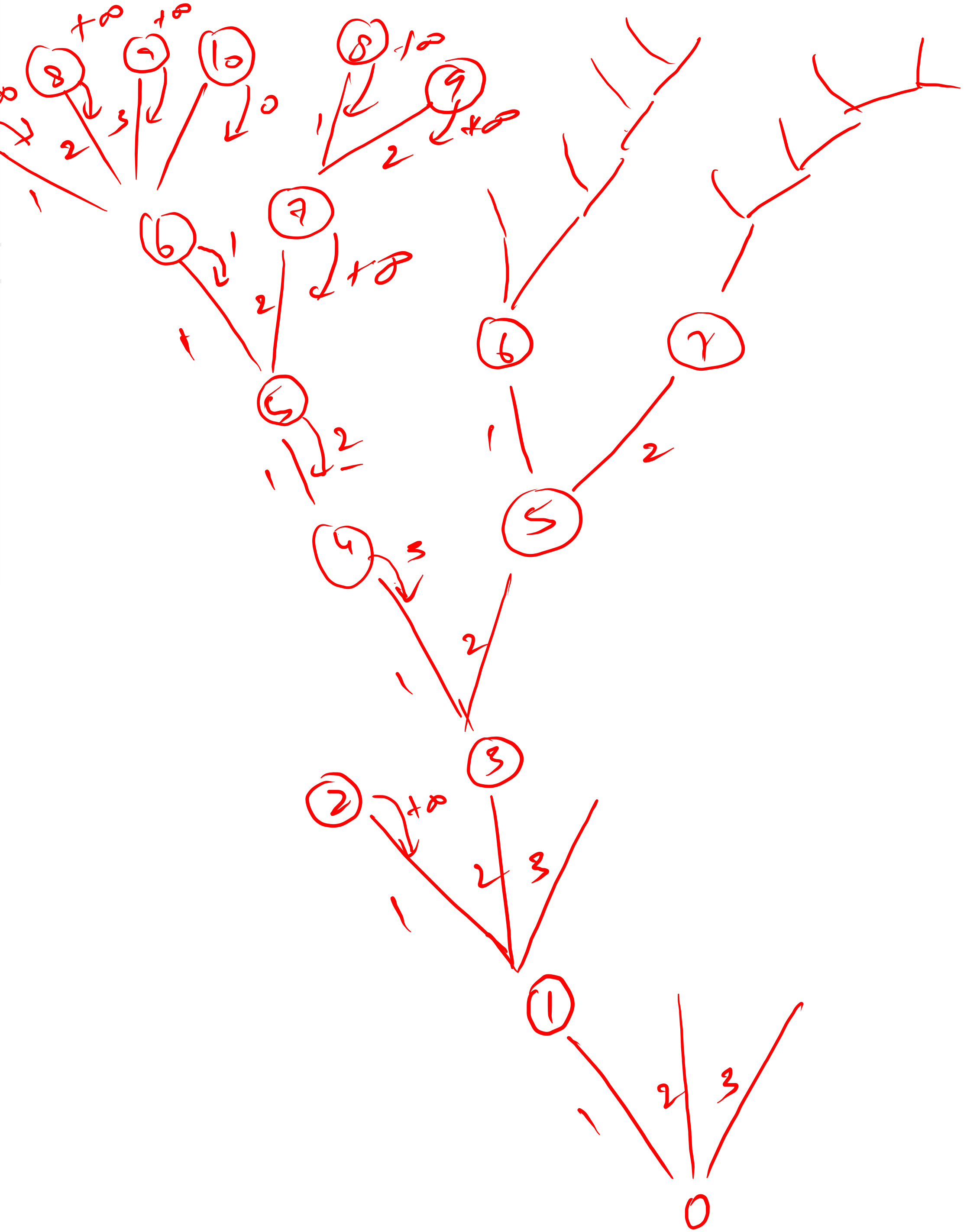


```
public static int CSMinMovesRec(int idx,int moves[]){
    if(idx == moves.length){
        return 0;
    }
    int maxJmp = moves[idx];
    int minSteps = Integer.MAX_VALUE;
    for(int len = 1; len <= maxJmp ; len++){
        if(idx + len <= moves.length){
            minSteps = Math.min(minSteps,CSMinMovesRec(idx+len, moves));
        }
    }

    return minSteps == Integer.MAX_VALUE ? Integer.MAX_VALUE : minSteps+1;
}
```



moves=

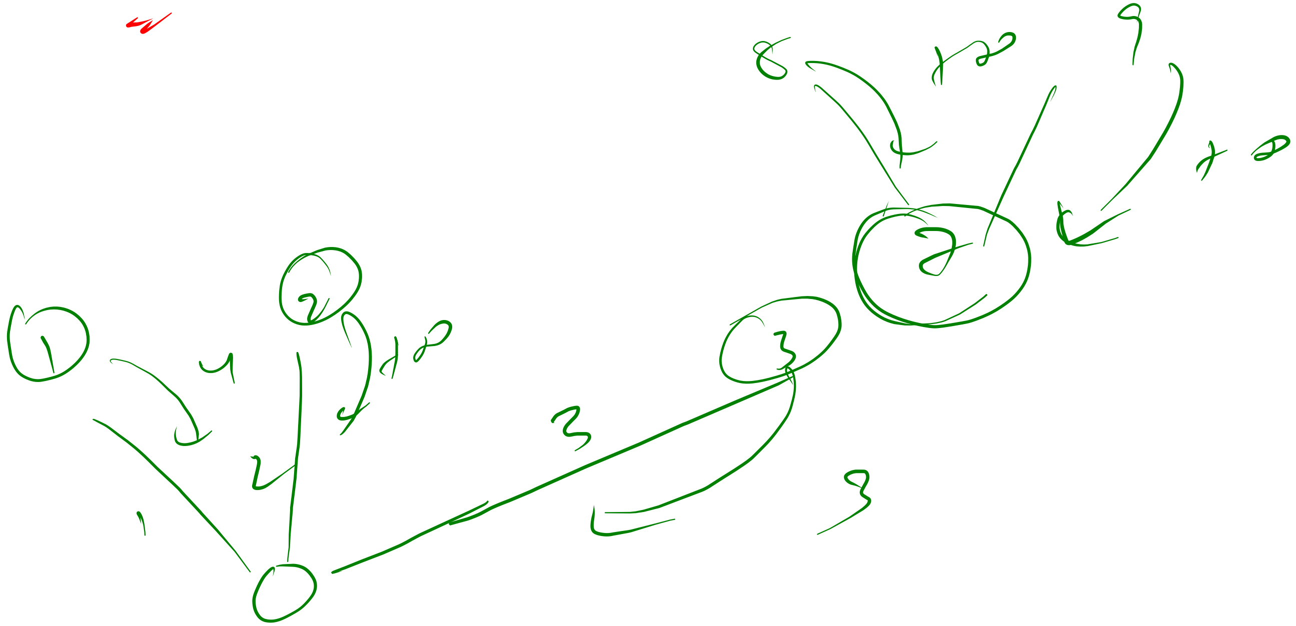
0	1	2	3	4	5	6	7	8	9
3	3	0	2	1	2	4	2	0	0

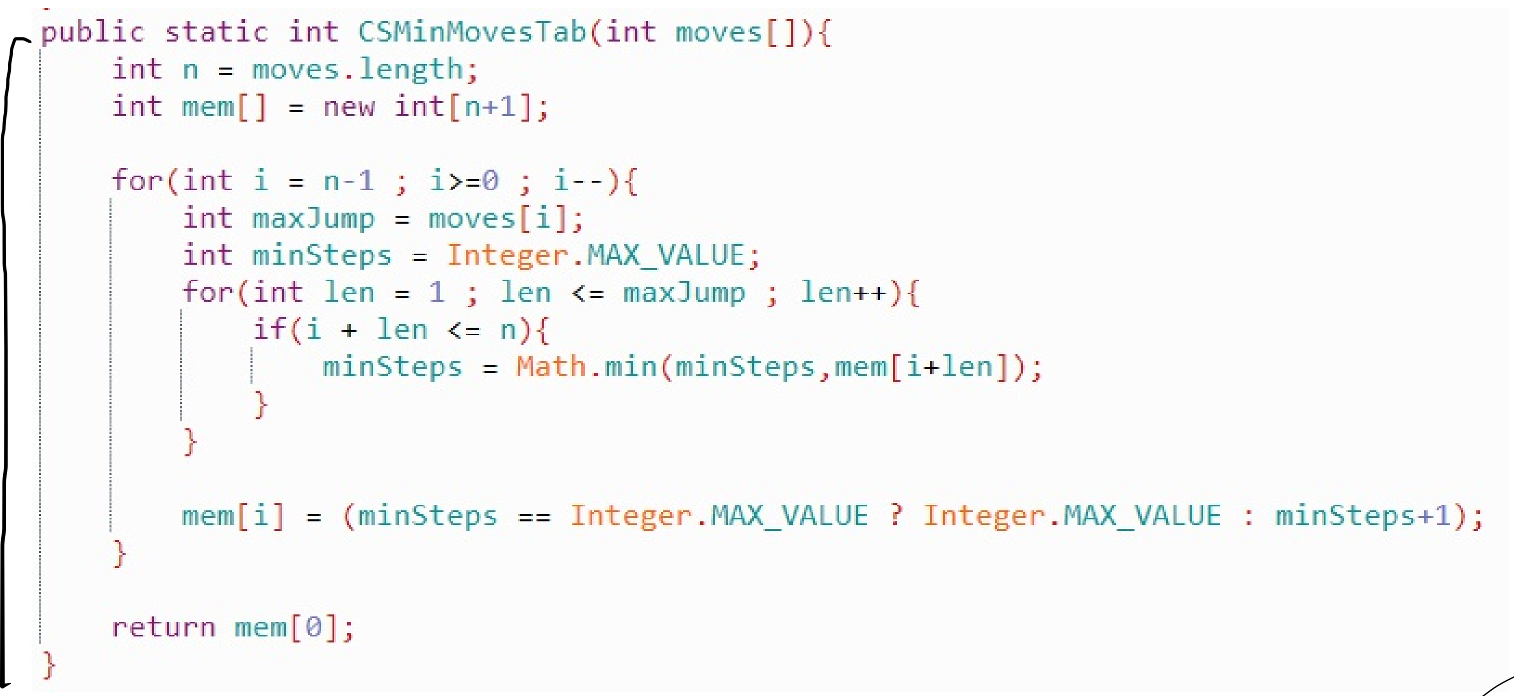


	0	1	2	3	4	5	6	7	8	9
moves =	3	3	0	2	1	2	4	2	0	0

Tabulation

4	4	+∞	3	3	2	1	+∞	+∞	+∞	0
0	1	2	3	4	5	6	7	8	9	10
									n-1	n

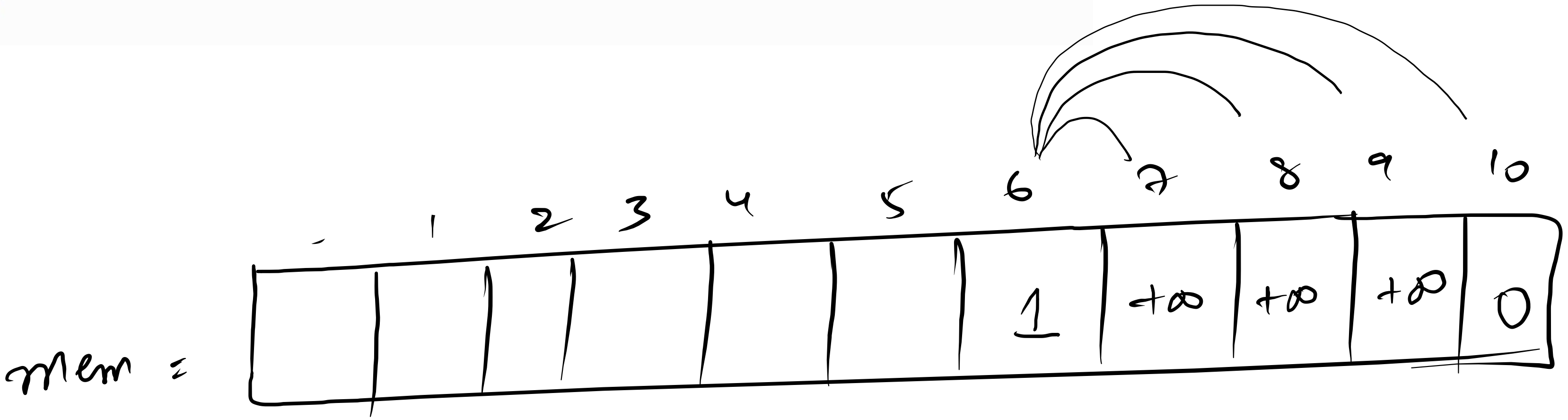




```
public static int CSMinMovesTab(int moves[]){
    int n = moves.length;
    int mem[] = new int[n+1];

    for(int i = n-1 ; i>=0 ; i--){
        int maxJump = moves[i];
        int minSteps = Integer.MAX_VALUE;
        for(int len = 1 ; len <= maxJump ; len++){
            if(i + len <= n){
                minSteps = Math.min(minSteps,mem[i+len]);
            }
        }

        mem[i] = (minSteps == Integer.MAX_VALUE ? Integer.MAX_VALUE : minSteps+1);
    }

    return mem[0];
}
```

	0	1	2	3	4	5	6	7	8	9
moves =	3	3	0	2	1	2	4	2	0	0

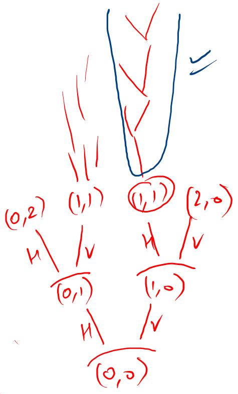
n = 10



mem

	0	1	2	3	4	5
0						
1						
2						
3						
4						
5						1

mem[i][j] → mincost (i,j) → dest



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

nr
nc

Ques → Find Min cost to reach dest from src

Src → (0,0)

dest = (5,5)

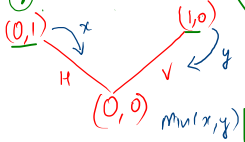
(nr-1, nc-1)

$$\begin{tikzcd}
(r,c) \arrow[r] \arrow[d, "V"] & H\ (r,c+1) \\
(r+1,c) &
\end{tikzcd}$$

Dun

MinCost((0,1) → dest) = x

MinCost((1,0) → dest) = y



mem[r][c] = Math.min(mem[r][c+1],
mem[r+1][c]) + cost[r][c]

Min(x,y) + cost[0][0]

	0	1	2	3	4	5
0	23	23	24	20	21	19
1	24	22	23	18	13	18
2	20	19	17	13	13	13
3	21	19	19	12	9	7
4	23	20	19	16	7	5
5	23	21	14	14	6	1

[H.W.] ✓

Tabulation ✓

✓

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

nr ✓
nc ✓

```
mem[r][c] = Math.min(mem[r][c+1], mem[r+1][c]) + Cost[r][c]
```

Max gold

Src ⇓ Dest

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

rec ✓

mem ✓

Tab

1 mem

2 meaning

3

(r,c) → (r-1,c+1) d1 — x

(r,c) → (r, c+1) d2 — y

(r,c) → (r+1, c+1) d3 — z

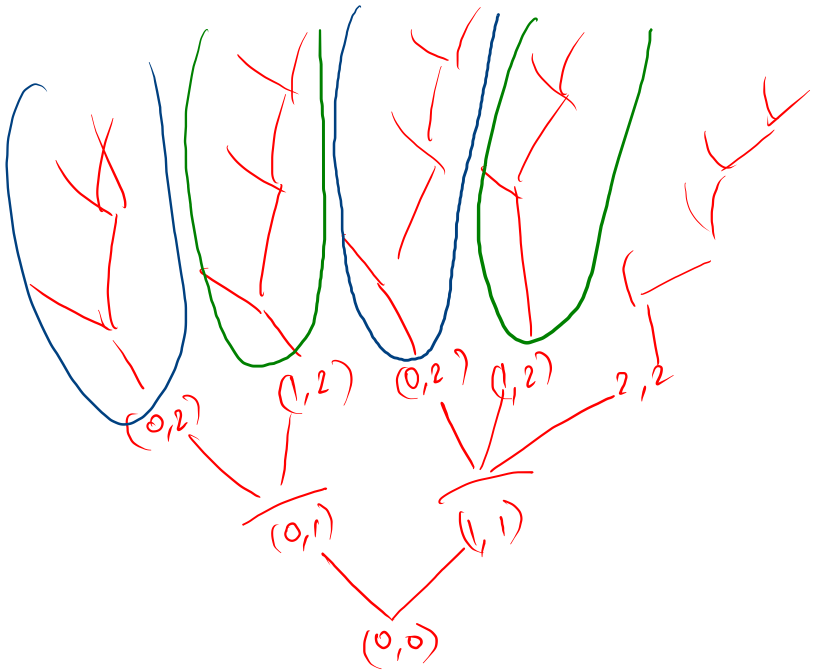
Max(x, y, z) + mine[r][c]

⇓ ⇓

	0	1	2	3	4	5
0	26	24	21	14	12	2
1	31	26	23	17	6	4
2	28	27	21	11	10	6
3	29	25	25	13	8	2
4	33	26	23	18	6	4
5	32	30	18	12	9	1

mem[r][c]

→ max gold collected from (r,c) → dest


(0,2)
(1,2)
(0,2)
(1,2)
2,2
(0,1)
(1,1)
(0,0)