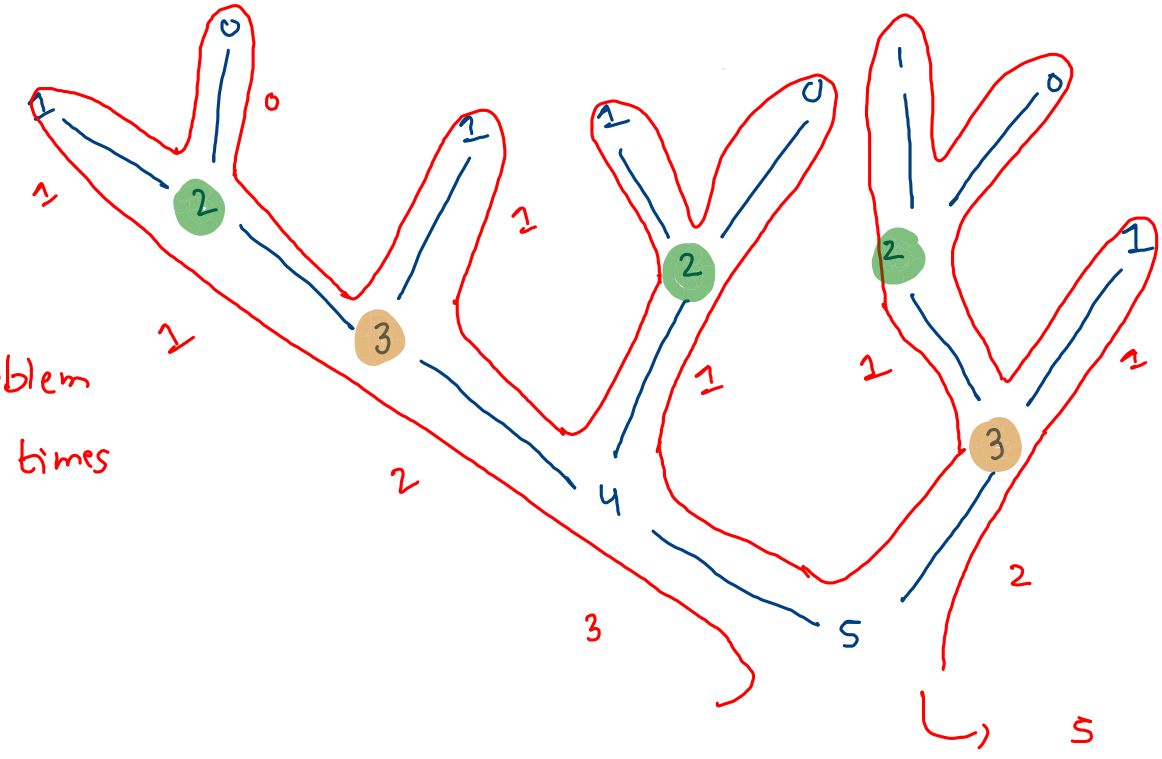
DP

$$fib(n) = fib(n-1) + fib(n-2)$$

```java
public static int fib(int n) {
    if(n == 0 || n == 1) {
        return n;
    }

    int fibn = fib(n-1) + fib(n-2);

    return fibn;
}
```



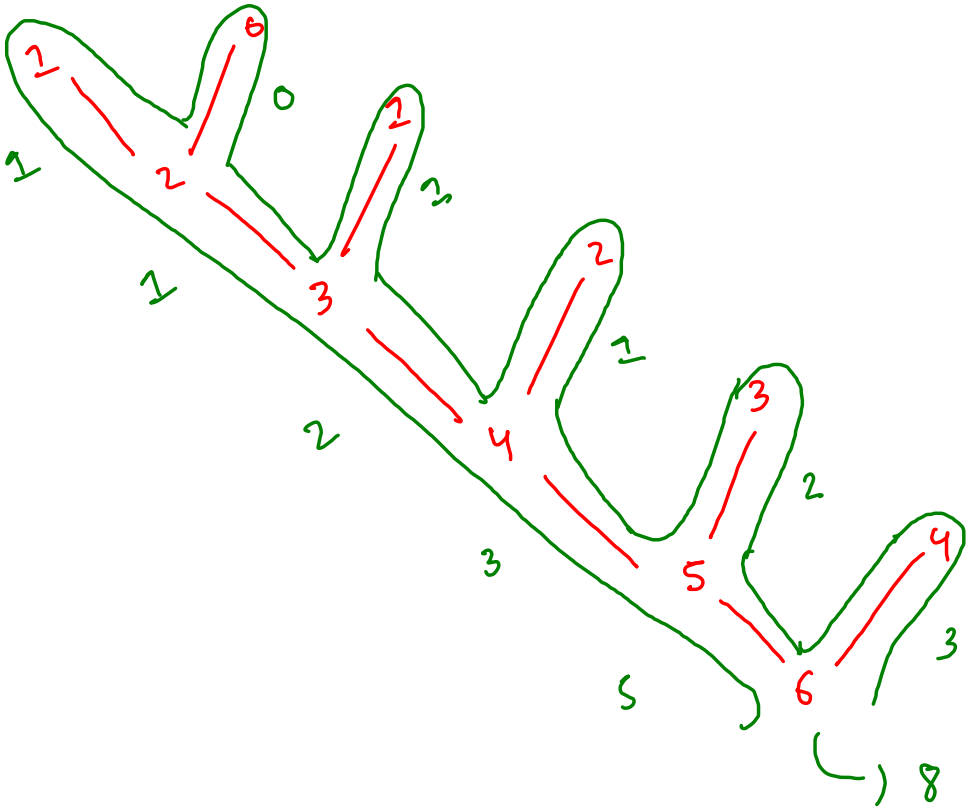
same problem multiple times

dp : memoisation

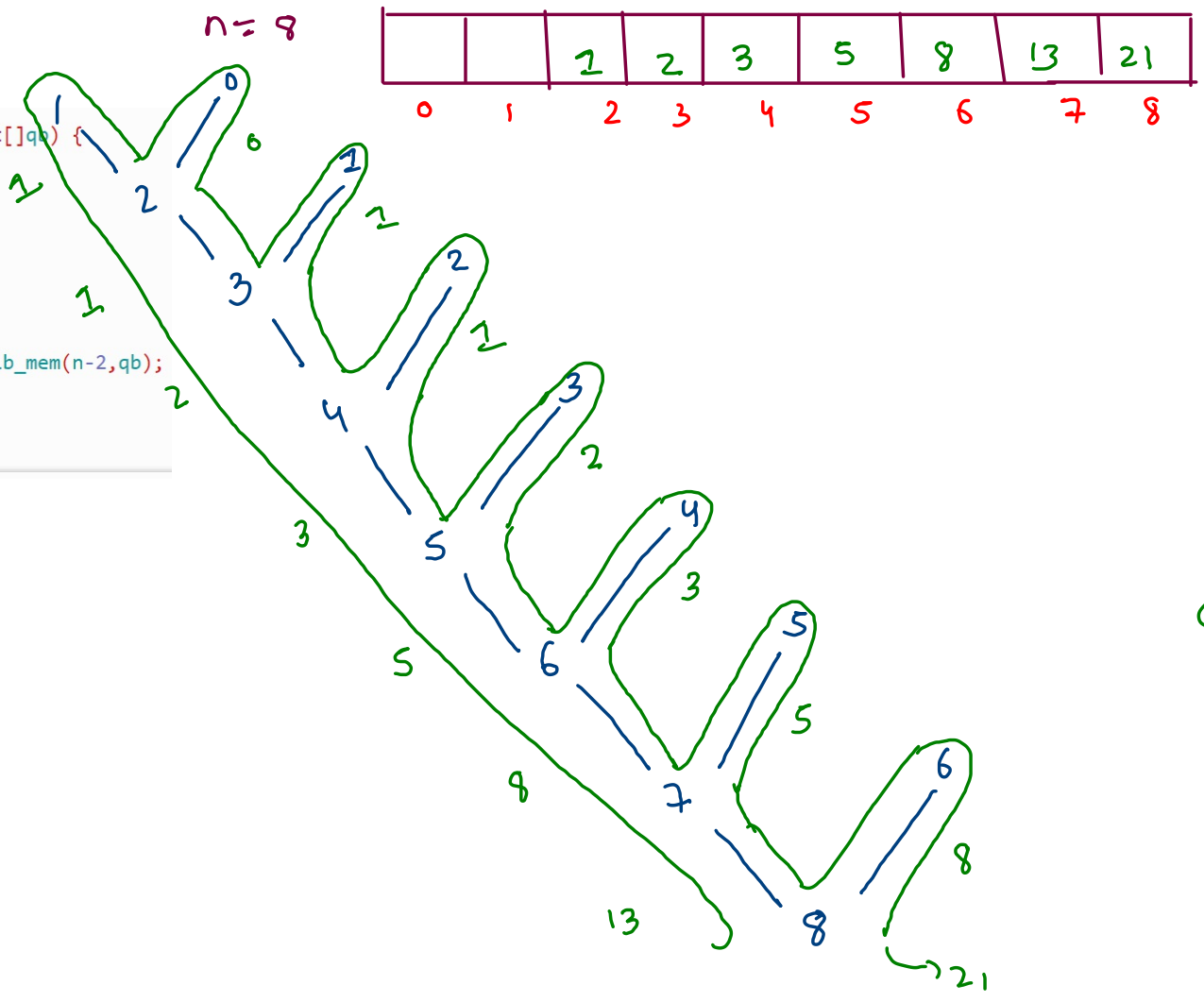
n = 6,

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

memoise



```java
public static int fib_mem(int n,int[]qb) {
    if(n == 0 || n == 1) {
        return n;
    }

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

    int fibn = fib_mem(n-1,qb) + fib_mem(n-2,qb);
    qb[n] = fibn;

    return fibn;
}
```



$n = 8$

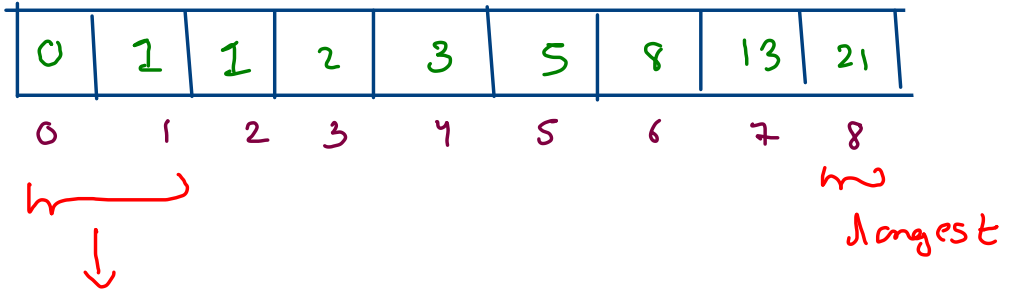
tabulation

(i) create strg

(ii) assign meaning to your strg.

(iii) travel and solve, from smaller problem to larger

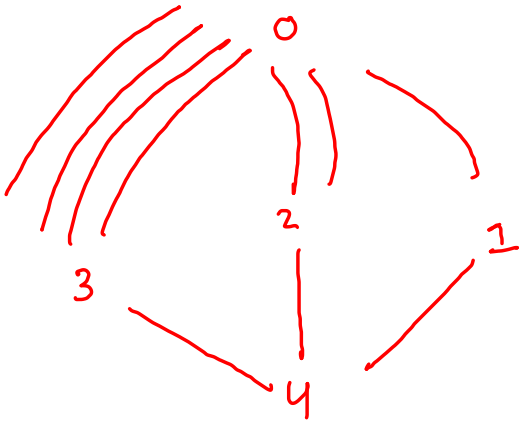
dp



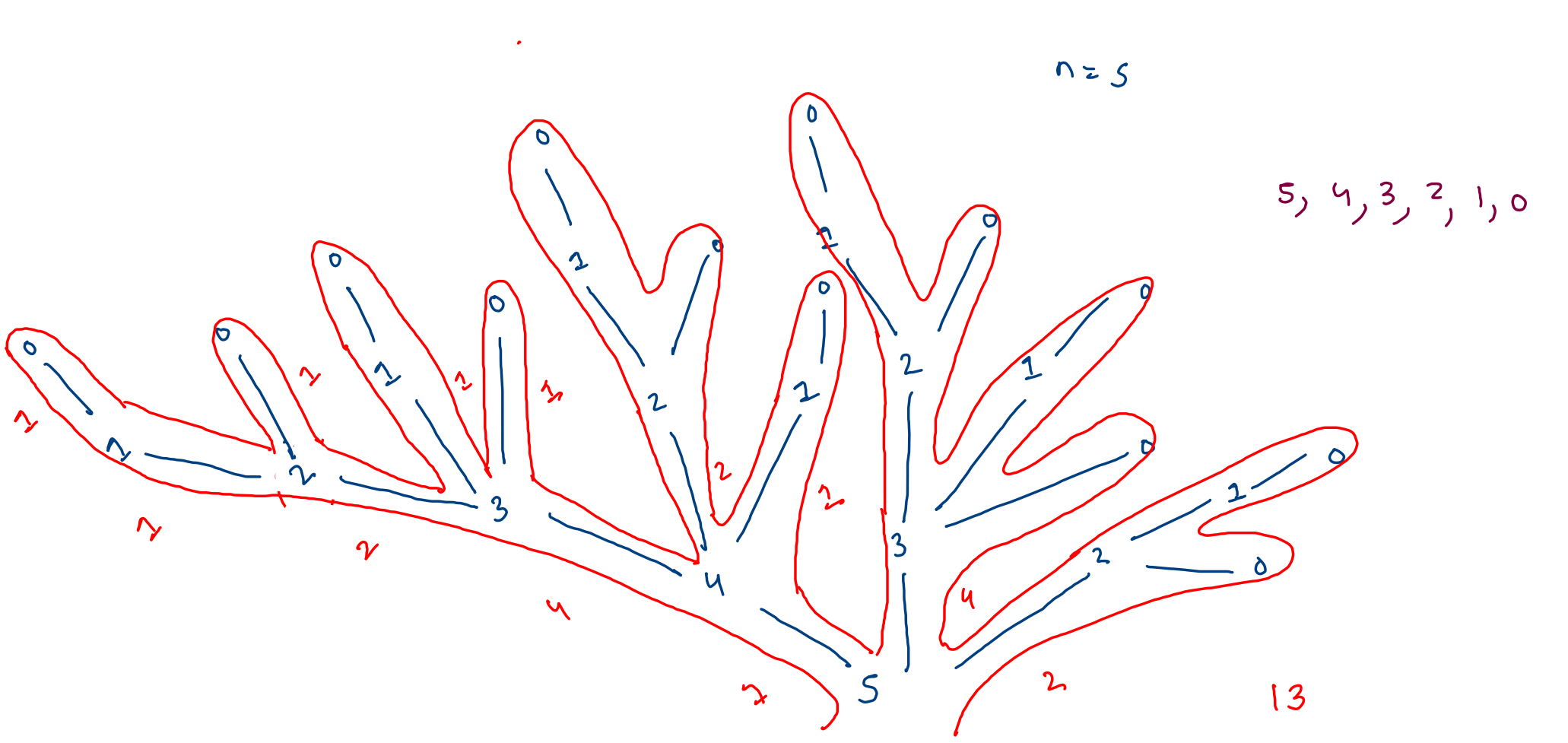
dp[i] -> fib(i)

dp[i] = dp[i-1] + dp[i-2];

Stair paths



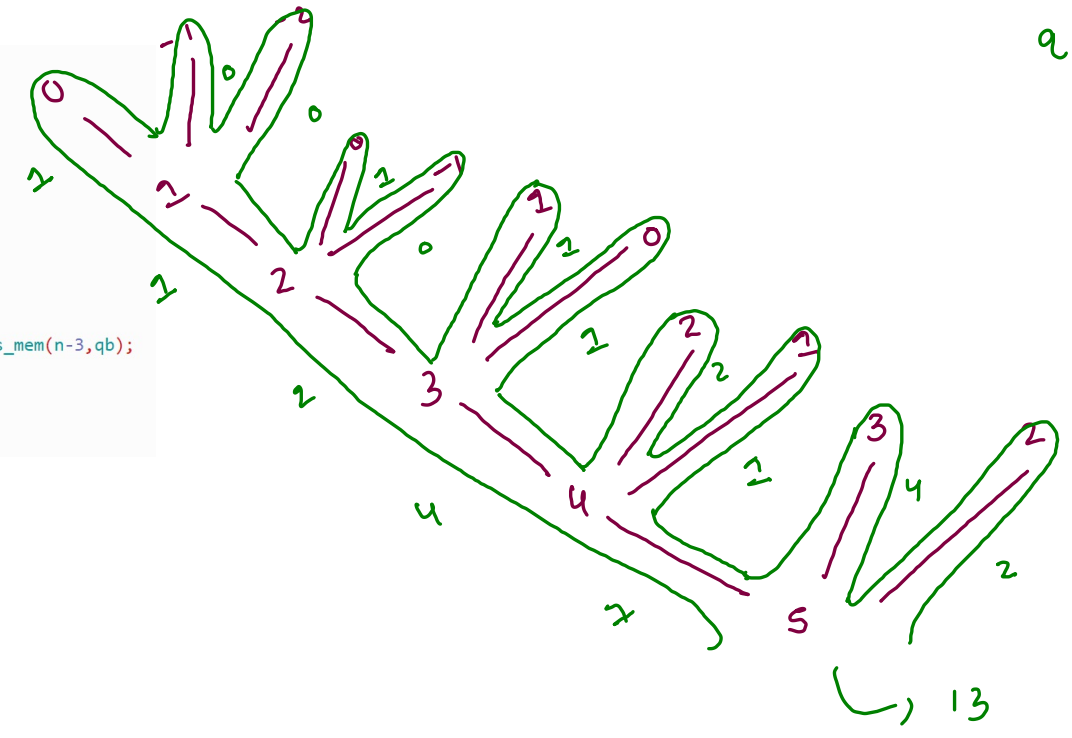
ways(n) = ways(n-1) + ways(n-2) + ways(n-3)


n = 5
5, 4, 3, 2, 1, 0
13

qb

	1	2	4	7	13
6	1	2	3	4	5

```java
public static int cs_mem(int n,int[]qb) {
    if(n == 0) {
        return 1;
    }

    if(n < 0) {
        return 0;
    }

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

    int ways = cs_mem(n-1,qb) + cs_mem(n-2,qb) + cs_mem(n-3,qb);
    qb[n] = ways;

    return ways;
}
```



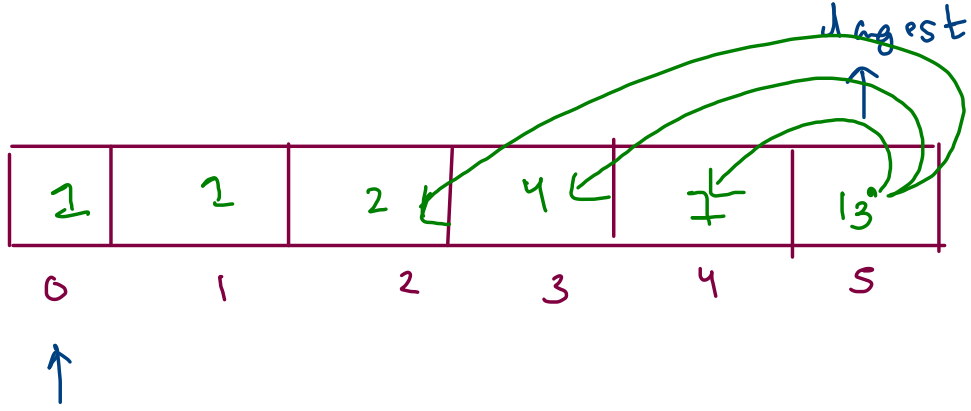
qb[i] -> no. of ways ith floor to ground floor

tabulation

(i) create strg

(ii) assign meaning

(iii) travel and solve

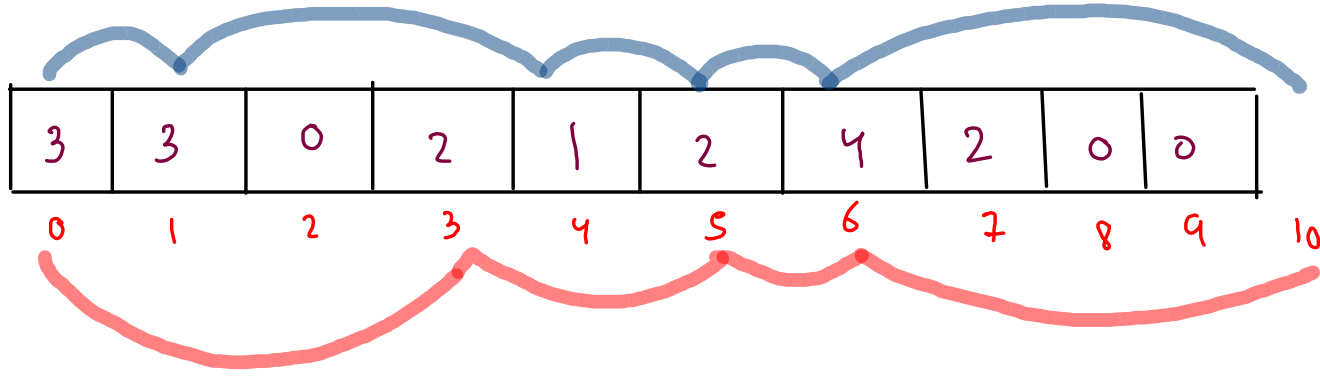


dp[i] -, i to 0 ways

dp[i] = dp[i-1] + dp[i-2] + dp[i-3]

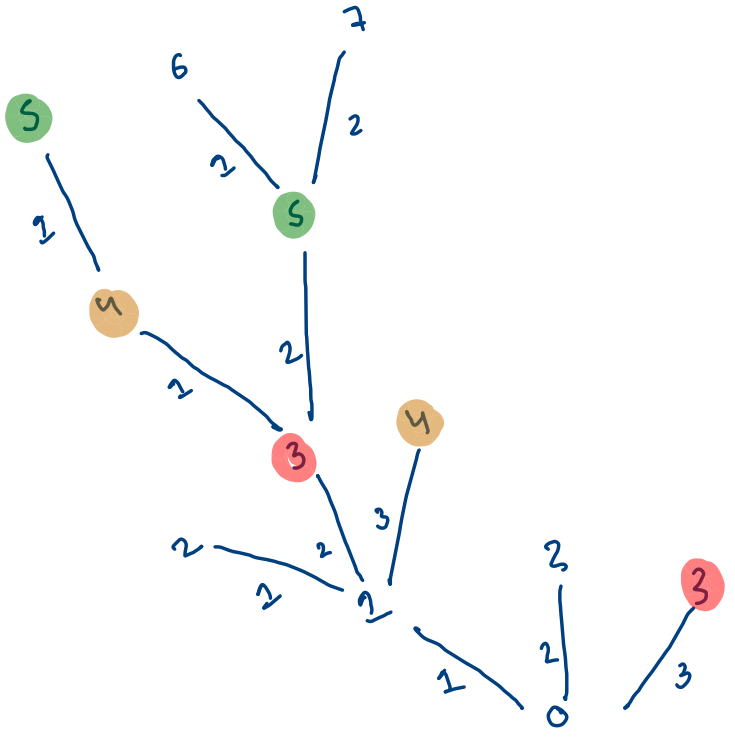
1. You are given a number n, representing the number of stairs in a staircase.
2. You are on the 0th step and are required to climb to the top.
3. You are given n numbers, where ith element's value represents - till how far from the step you could jump to in a single move.
   You can of course jump fewer number of steps in the move.
4. You are required to print the number of different paths via which you can climb to the top.

10 3 3 0 2 1 2 4 2 0 0

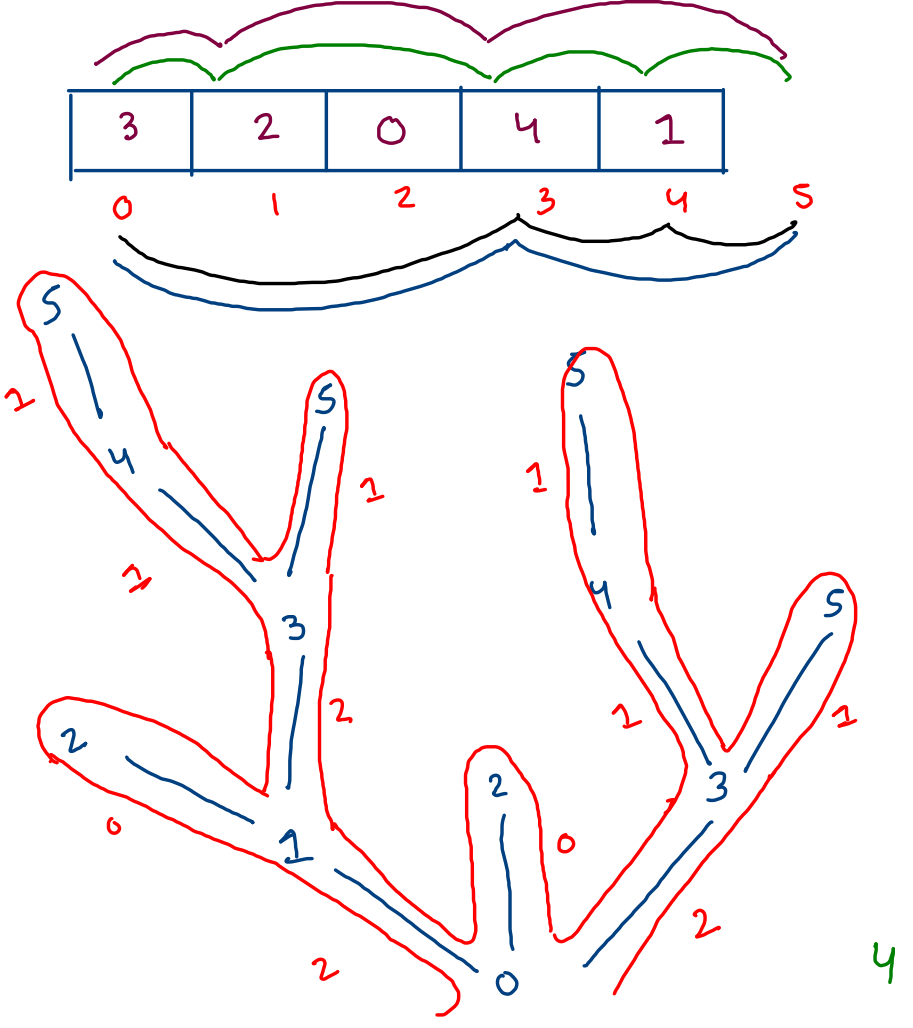


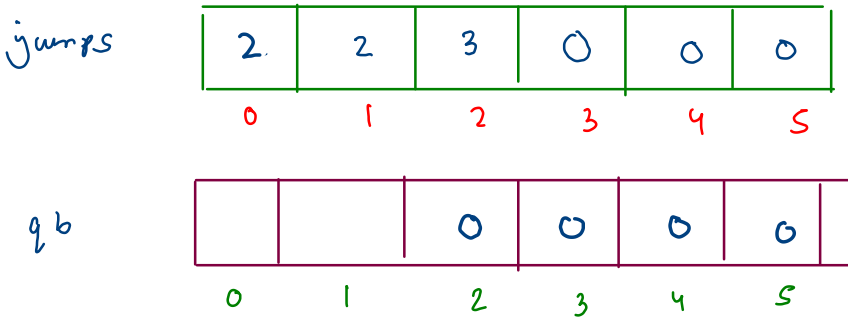
3	3	0	2	1	2	4	2	0	0
0	1	2	3	4	5	6	7	8	9

dest = 10

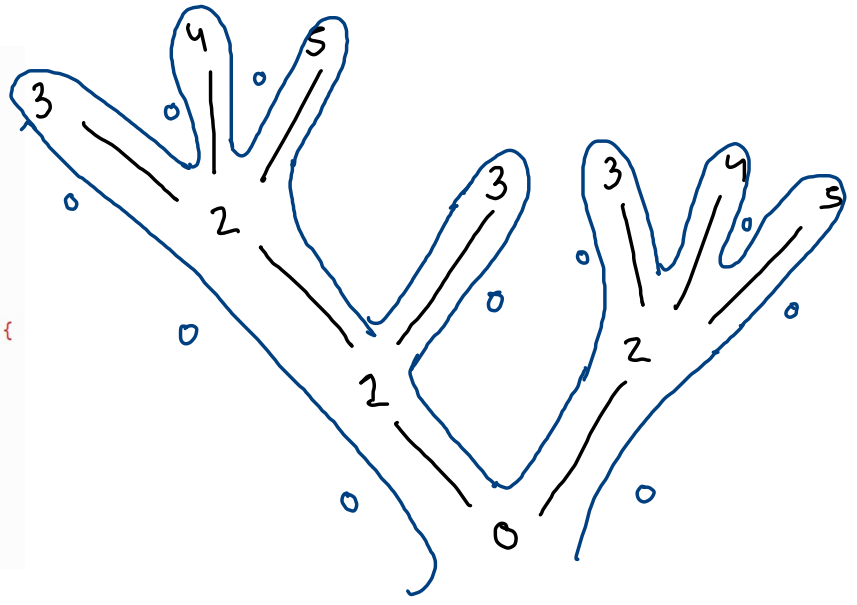


```java
public static int csvj_rec(int src,int[]jumps) {
    if(src == jumps.length) {
        return 1;
    }

    int total = 0;
    for(int k = 1; k <= jumps[src] && src + k <= jumps.length;k++) {

        int ntodw = csvj_rec(src + k,jumps);

        total += ntodw;
    }

    return total;
}
```






```
public static int csvj_mem(int src,int[]jumps,int[]qb) {
    if(src == jumps.length) {
        return 1;
    }

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

    int total = 0;

    for(int k=1; k <= jumps[src] && src + k <= arr.length;k++) {
        int ntodw = csvj_mem(src + k,jumps,qp);
        total += ntodw;
    }

    qb[src] = total;

    return total;
}
```



unsolved -> 0

ans -> 0

↓ improve

unsolved -> -1

ans -> 0