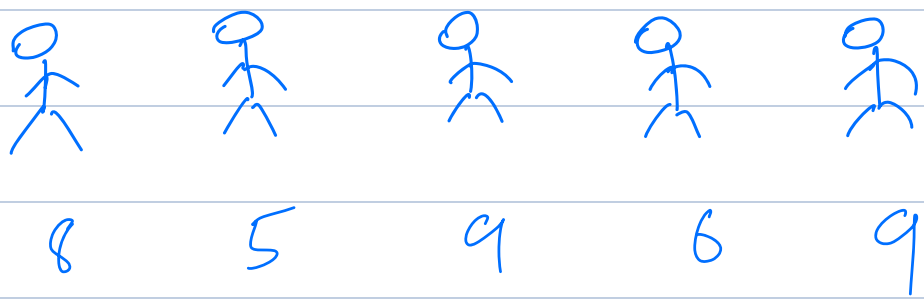
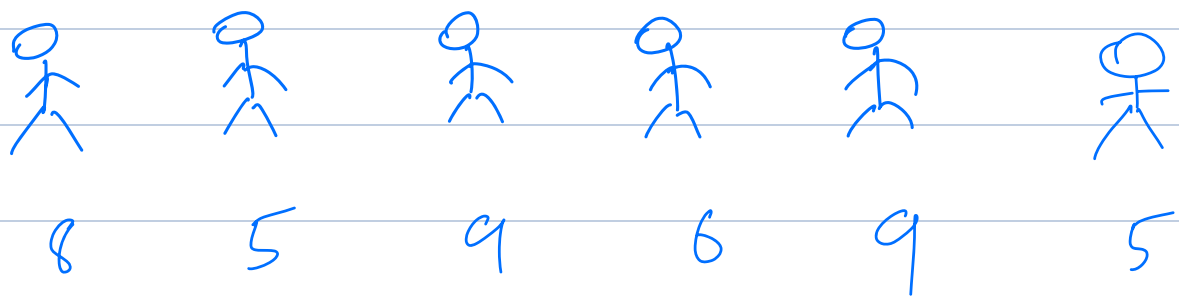
Dynamic Programming.



= 37.



= 42.

If I have a pre-calculated value, I will not calculate it again or rather re-use it.

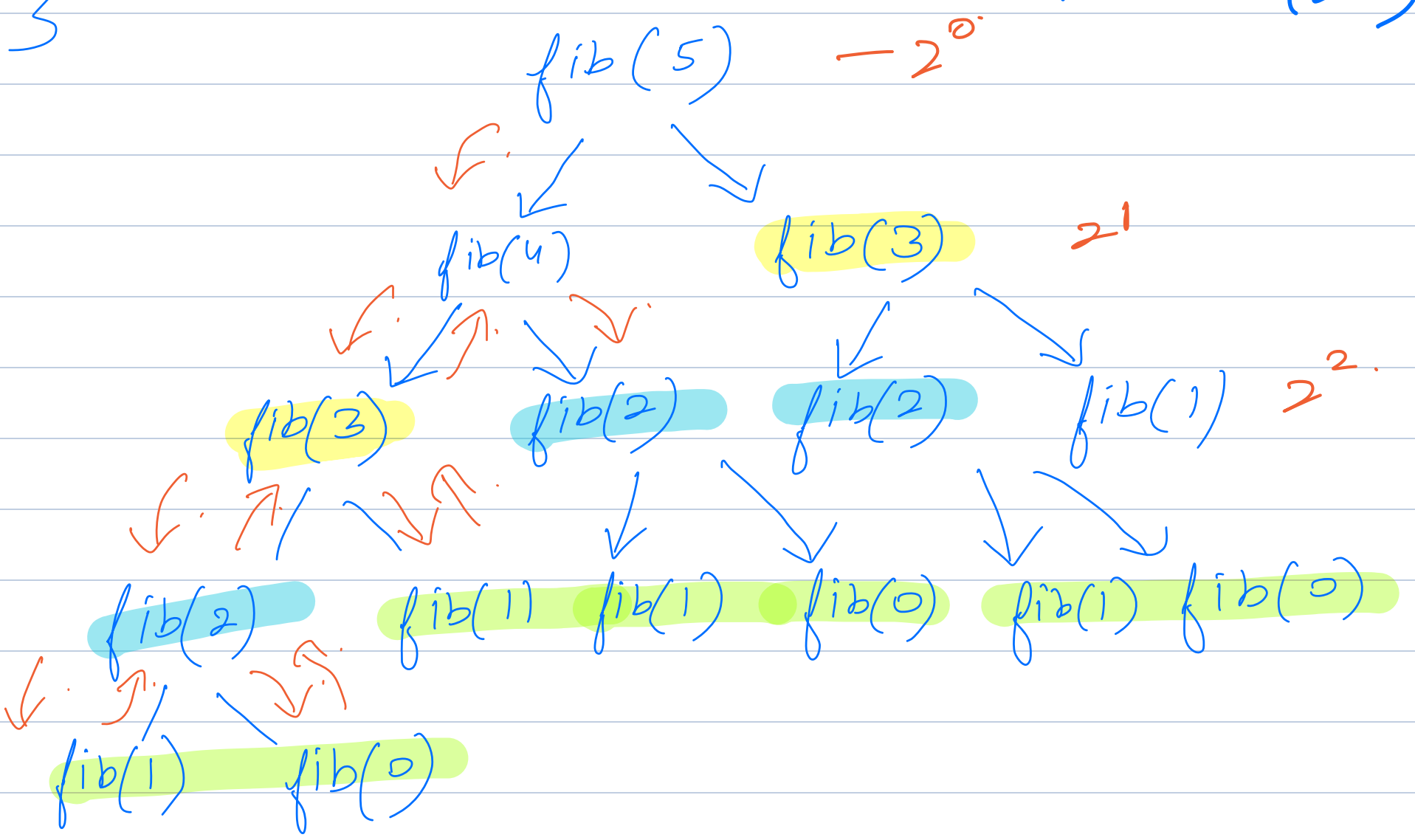
N^{th} fibonnaci Number

N = 10	0	1	1	2	3	5	8	13	21	34	55
		1	2	3	4	5	6	7	8	9	10

$$fib(N) = fib(N-1) + fib(N-2);$$

```
int fib(N) {
  if (N ≤ 1) return N;
  return fib(N-1) + fib(N-2)
}
```

T.C. $O(2^N)$.



① Optimal substructure :- To solve a large problem, use smaller instances.

② Overlapping Subproblems :- Solving same problem again & again.

Sol:- I should store the result

storewhere to avoid re-calculation.

Array of size N+1

dp[]:

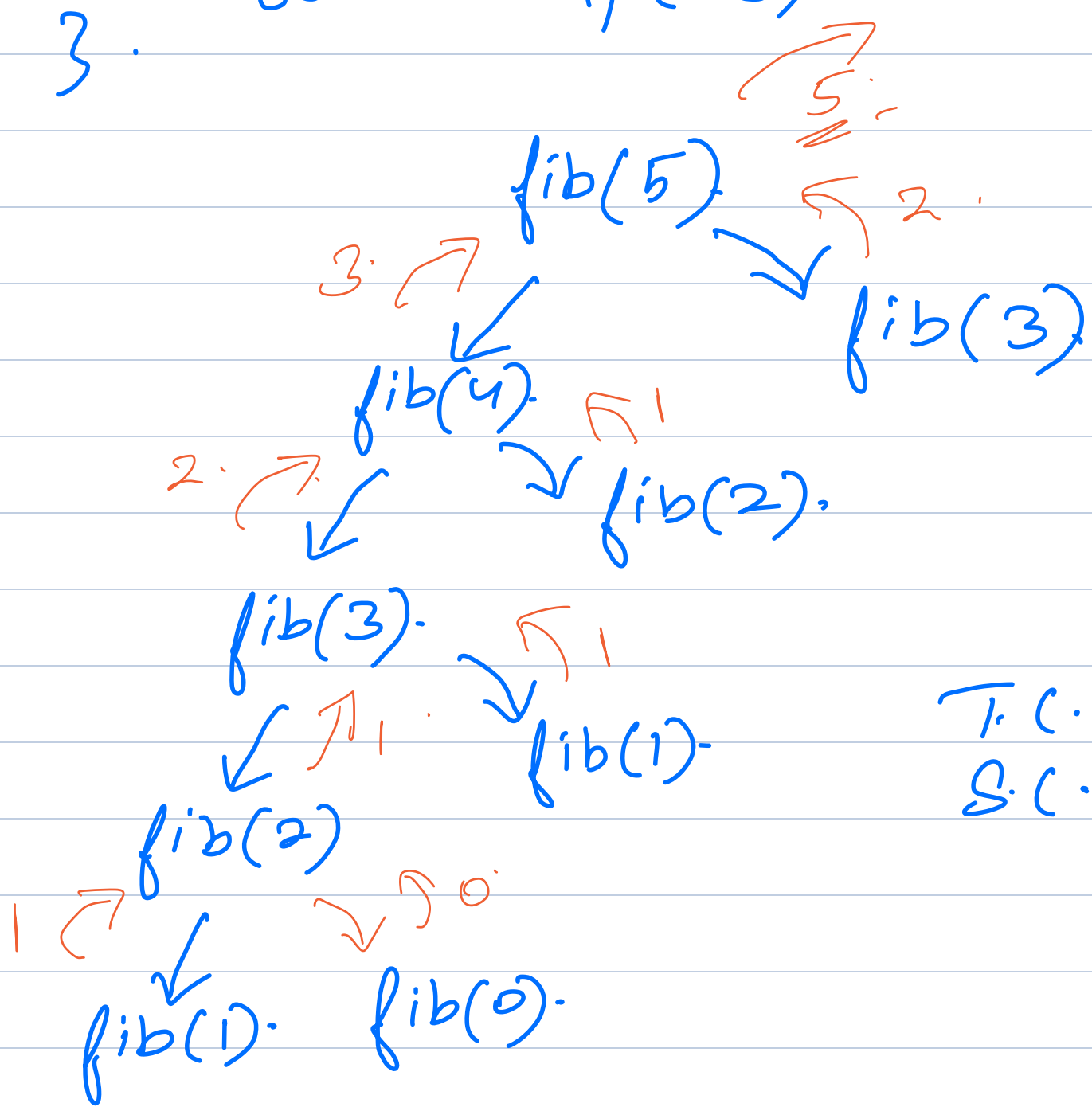
-1	-1	-1	-1	-1	-1
0	1	2	3	4	5

dp[i] → i^{th} Fibonacci No.

Code:

		1	2	3	5
-1	-1	~~-1~~	~~-1~~	~~-1~~	~~-1~~
0	1	2	3	4	5

```
int fib(N) {
    if (N ≤ 1) return N;
    if (dp[N] != -1) {return dp[N];}.

    dp[N] = fib(N-1) + fib(N-2)
    return dp[N];
}.
```



T.C. O(N)
S.C. O(N)

2nd approach.

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

Code.

```
int dp[N+1];
dp[0] = 0;
dp[1] = 1;

for ( i = 2 ; i <= N ; i++) {
    dp[i] = dp[i-1] + dp[i-2];
}
return dp[N];
```

T.C. $O(N)$

S.C. $O(N)$.

$\downarrow$

$SC(O(1))$

M.W.

Top - Down Approach. (Start from the biggest problem)

↳ Recursion.

→ Memoisation.

Bottom - Up Approach. (Start from the smallest problem)

↳ Iterative.

→ Tabulation.

Climbing Stairs

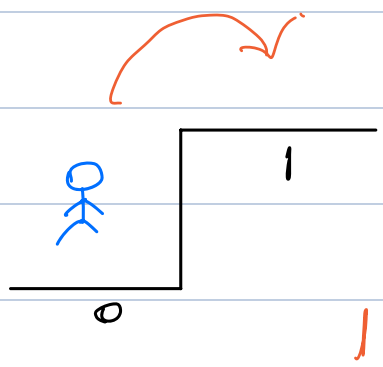
No of ways to reach N^{th} stair?

N stairs

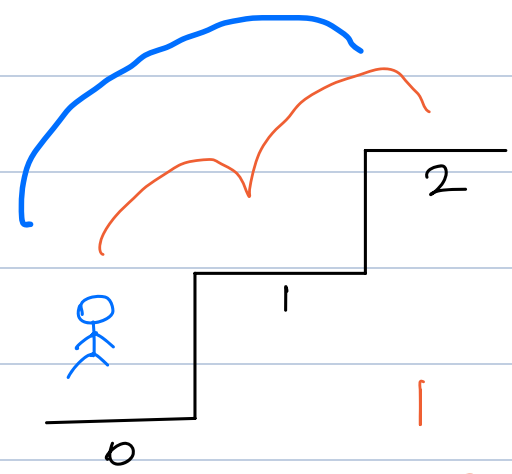
1 jump 2 jump

$N=1$

$N=2$ 2 ways.



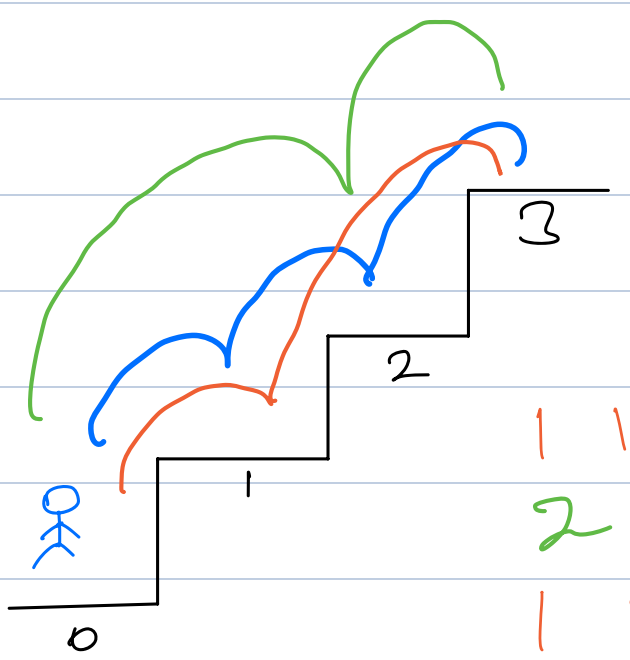
1



1 1

2.

$N=3$

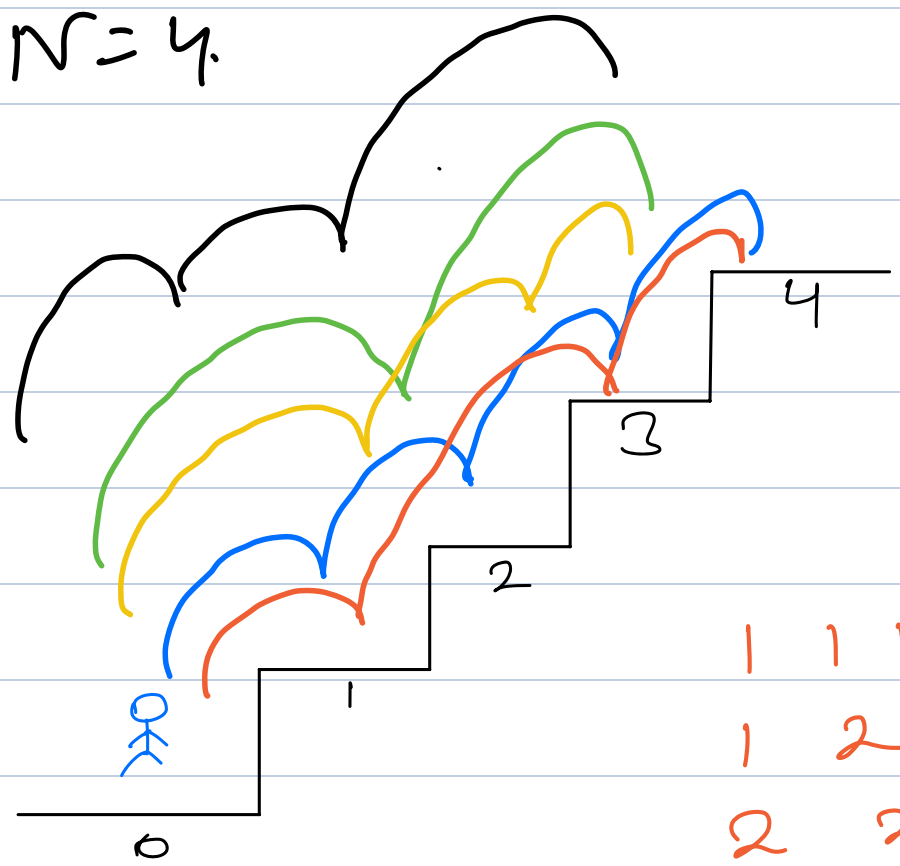


1 1 1

2 1

1 2

$N=4$.



1 1 1 1

1 2 1

2 2.

2 1 1

1 1 2.

ans = 5.

N^{th} stair

$(N-1)^{th}$ stair + $(N-2)^{th}$ stair

ways(N^{th}) = Take 1 jump from $(N-1)^{th}$

stair

OR

Take 2 jumps from $(N-2)^{th}$ stair.

$$= \text{ways}(N-1) * 1$$
$$+$$
$$\text{ways}(N-2) * 1$$

$$\text{ways}(N) = \text{ways}(N-1) + \text{ways}(N-2)$$

ways[1] = 1;
ways[2] = 2;

fib[0] = 0;
ways[0] = 1; → No action.

8:22;

Find the min. no of perfect squares required to get sum = N ?
(No's can repeat?)

N = 6.

$1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2$

$1^2 + 1^2 + 2^2$ ans = 3.

N = 10.

$1^2 + 1^2 + 1^2 \; --- \; 1^2 \rightarrow 10$

$2^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 \rightarrow 7$

$2^2 + 2^2 + 1^2 + 1^2 \rightarrow 4.$

$3^2 + 1^2 \rightarrow 2.$

N = 9.

$3^2 \rightarrow 1.$

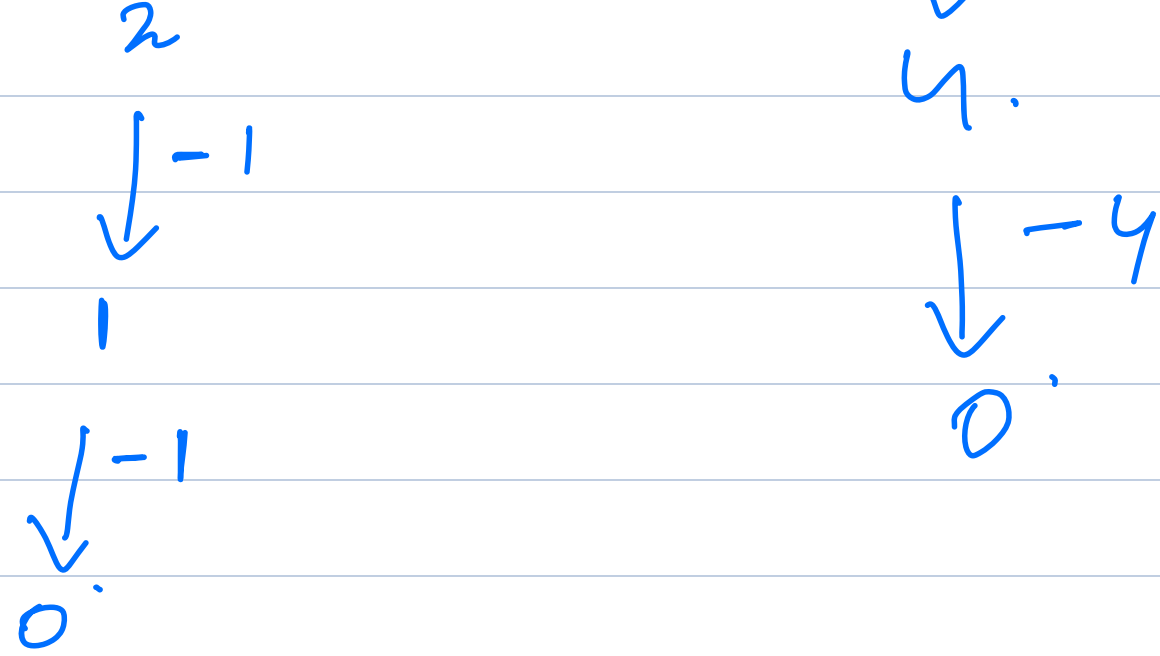
✗ → Not working.

Reduce N to 0 by reducing largest perfect square from N. (Greedy)

N = 12.

$12 \xrightarrow{-9} 3 \xrightarrow{-1}$

$12 \xrightarrow{-4} 8 \xrightarrow{-4}$

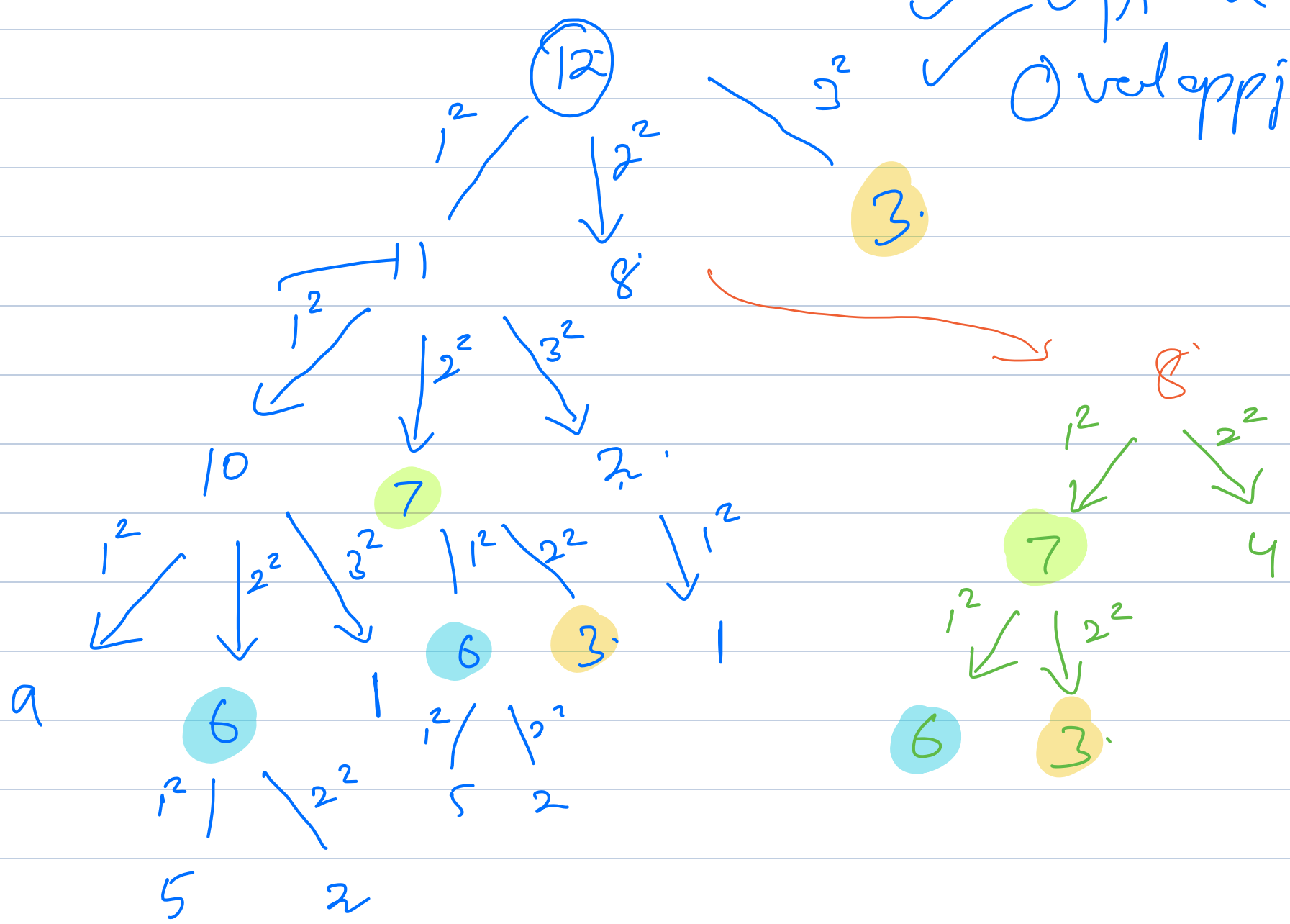


Ans = 3.

B.F. → Try out all the possibilities.

N = 12.

✓ Optimal SS.
✓ Overlapping S.P



$$\min Sq(12) = \min\left(\min Sq(11), \min Sq(8), \min Sq(3)\right) + 1.$$

$$\text{minSq}(N) = \min\left(\text{minSq}(N - x^2) \;\; \forall x^2 \le N\right) + 1.$$

```
int minSquare(N) {
    if (N==0) {return 0;}
    ans = N;
    for (x=1; x*x ≤ N; x++) {
        ans = min(ans, minSquare(N-x²));
    }
    return ans + 1;
}
```

ans = 5

1+1=2

5

1^2 2^2 1

3 4

1^2 2^2 0 0

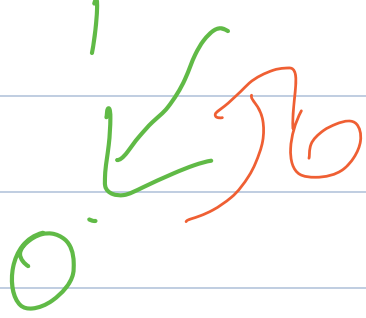
1 1^2 6

0

2 3

0+1 2

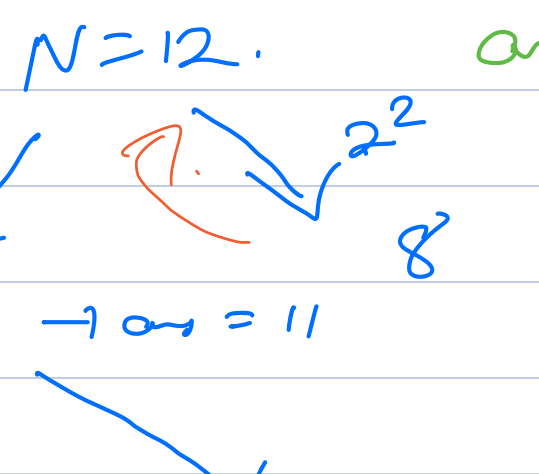
2

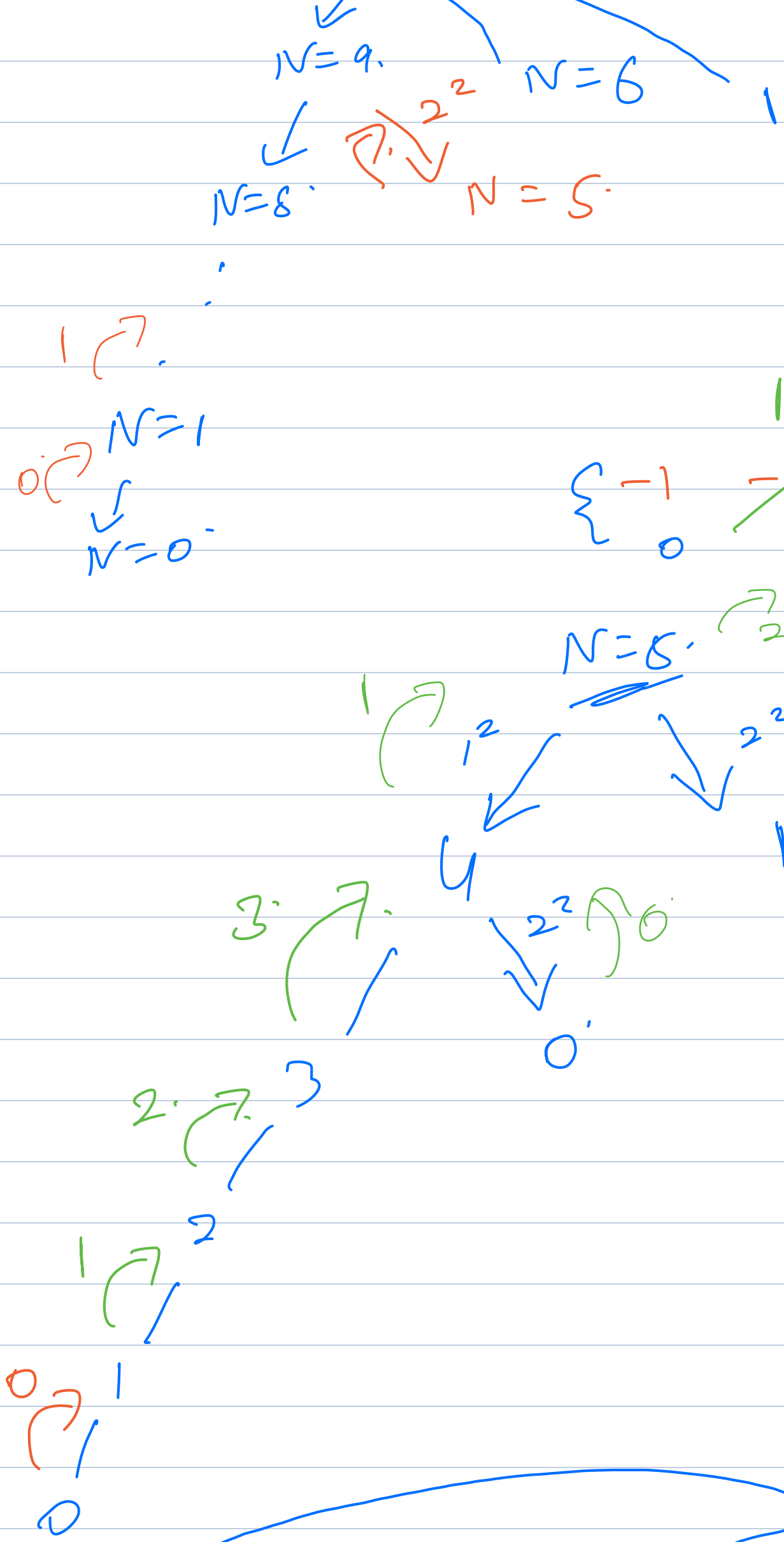


```
int dp[N+1];        ∀i  dp[i] = -1;

int minSquare(N) {
    if (N==0) {return 0;}
    if (dp[N] != -1) {return dp[N];}
        ans = N;
    for (x=1; x*x ≤ N; x++) {
        ans = min(ans, minSquare(N-x²));
    }

    dp[N] = ans + 1;
    return dp[N];
}
```

-1	~~-1~~ 1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1
0	1	2	3	4	5	6	7	8	9	10	11	12

Dry Run as H.W.





$O(N \times \sqrt{N})$

$O(N) + O(N)$

0	1	2	3	1	2	3	4	2	1	2	3	3
0	1	2	3	4	5	6	7	8	9	10	11	12

$dp[2] = 1 + dp[1]$

$dp[3] = 1 + dp[3 - 1^2]$

$1 + \min(dp[9 - 1^2], dp[9 - 2^2])$

$2^2 + 1^2 + 1^2$

$, dp[9 - 3^2]$

$2^2 + 2^2$

Code:

```
int dp[N+1];
dp[0] = 0;

for (i=1; i <= N; i++) {
    ans = INT_MAX;
    for (x = 1; x*x <= i; x++) {
        ans = min(ans, dp[i - x^2]);
    }
    dp[i] = ans + 1;
}

return dp[N];
```

T.C → $O(N * Sqrt(N))$;

S.C. → $O(N)$.

$ways(N) = ways(N-1) + ways(N-2)$

$ways(0) = 0$;