Dynamic Programming

Video - 75

Note :- This playlist is only for explanation of dps & solutions.

See my "DP Concepts & dps" playlist for understanding DP from scratch...

Facebook, Instagram → codestorywithMIK

Twitter → cswithMIK

→ codestorywithMIK

Leetcode - 91 Medium

Google Meta Microsoft UBER

Medium 10.8K 4.4K

Companies

A message containing letters from `A-Z` can be **encoded** into numbers using the following mapping:

```
'A' -> "1"
'B' -> "2"
...
'Z' -> "26"
```

To **decode** an encoded message, all the digits must be grouped then mapped back into letters using the reverse of the mapping above (there may be multiple ways). For example, `"11106"` can be mapped into:

- `"AAJF"` with the grouping `(1 1 10 6)`
- `"KJF"` with the grouping `(11 10 6)`

Note that the grouping `(1 11 06)` is invalid because `"06"` cannot be mapped into `'F'` since `"6"` is different from `"06"`.

Given a string `s` containing only digits, return *the **number** of ways to **decode** it*.

The test cases are generated so that the answer fits in a **32-bit** integer.

Example 1:

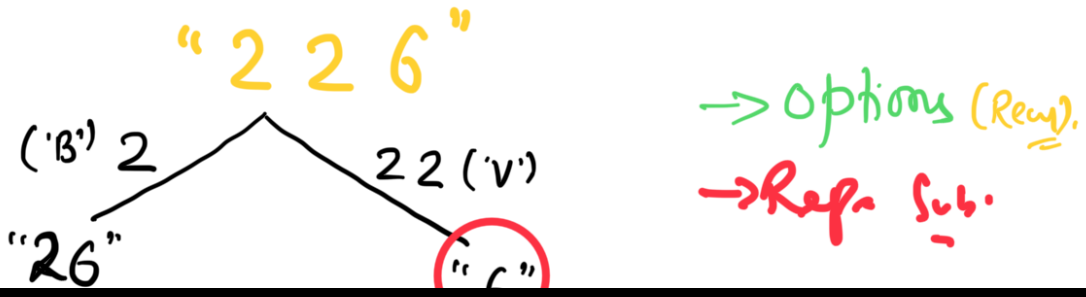
```
Input: s = "12"
Output: 2
Explanation: "12" could be decoded as "AB" (1 2) or "L" (12).
```

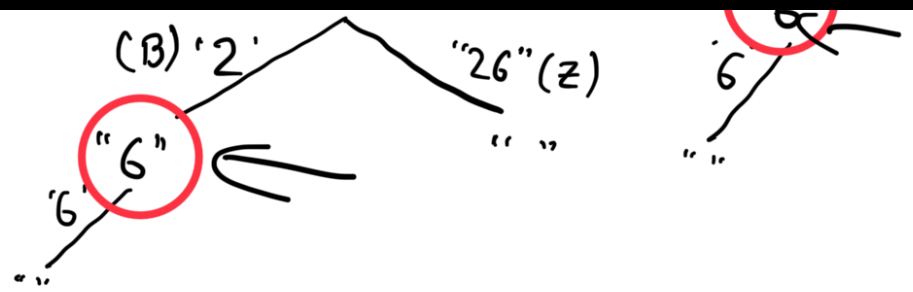
Example 2:

```
Input: s = "226"
Output: 3
Explanation: "226" could be decoded as "BZ" (2 26), "VF" (22 6), or "BBF" (2 2 6).
```

Example 3:

```
Input: s = "06"
Output: 0
Explanation: "06" cannot be mapped to "F" because of the leading zero ("6" is
different from "06").
```

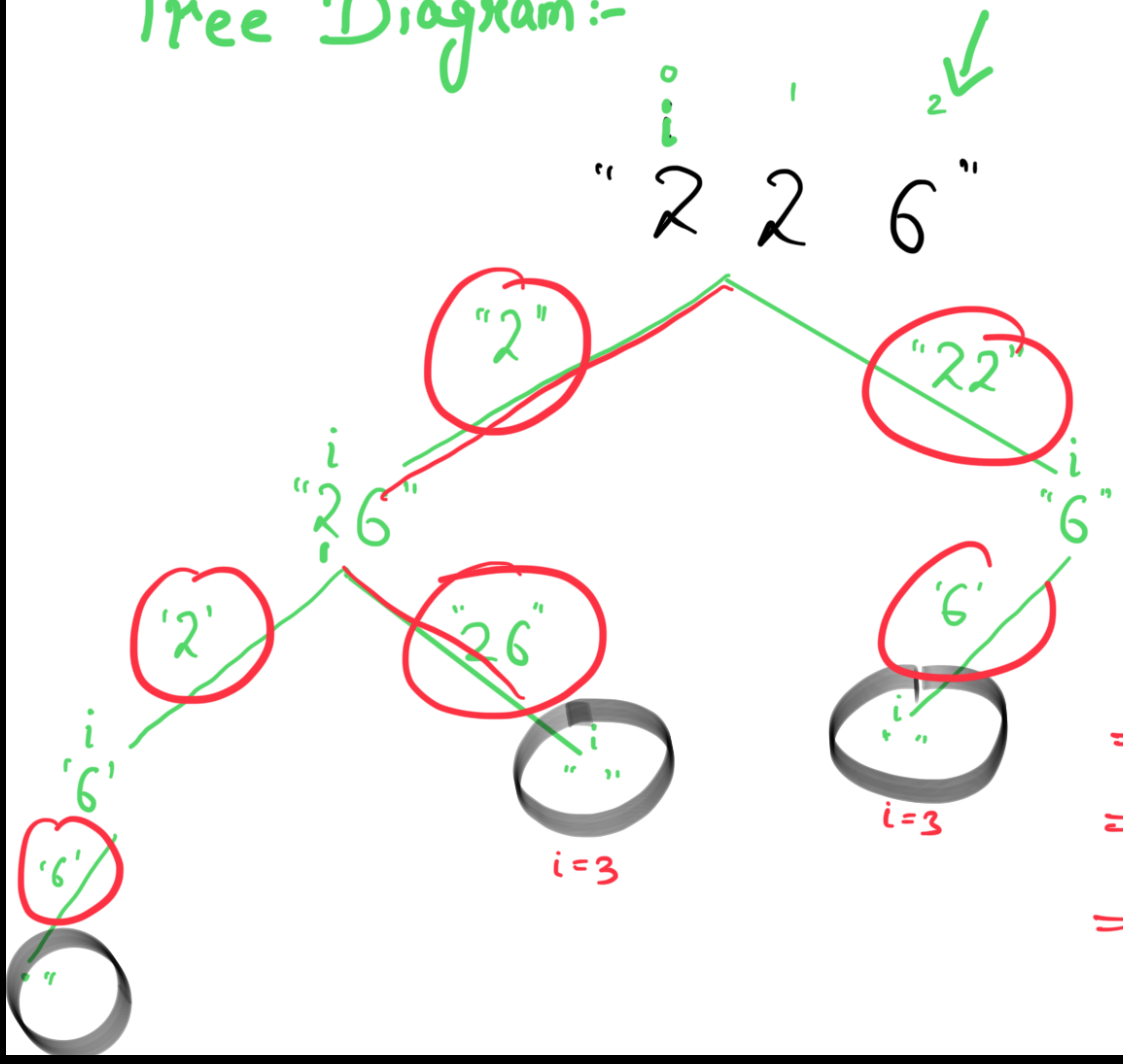




Why DP ??

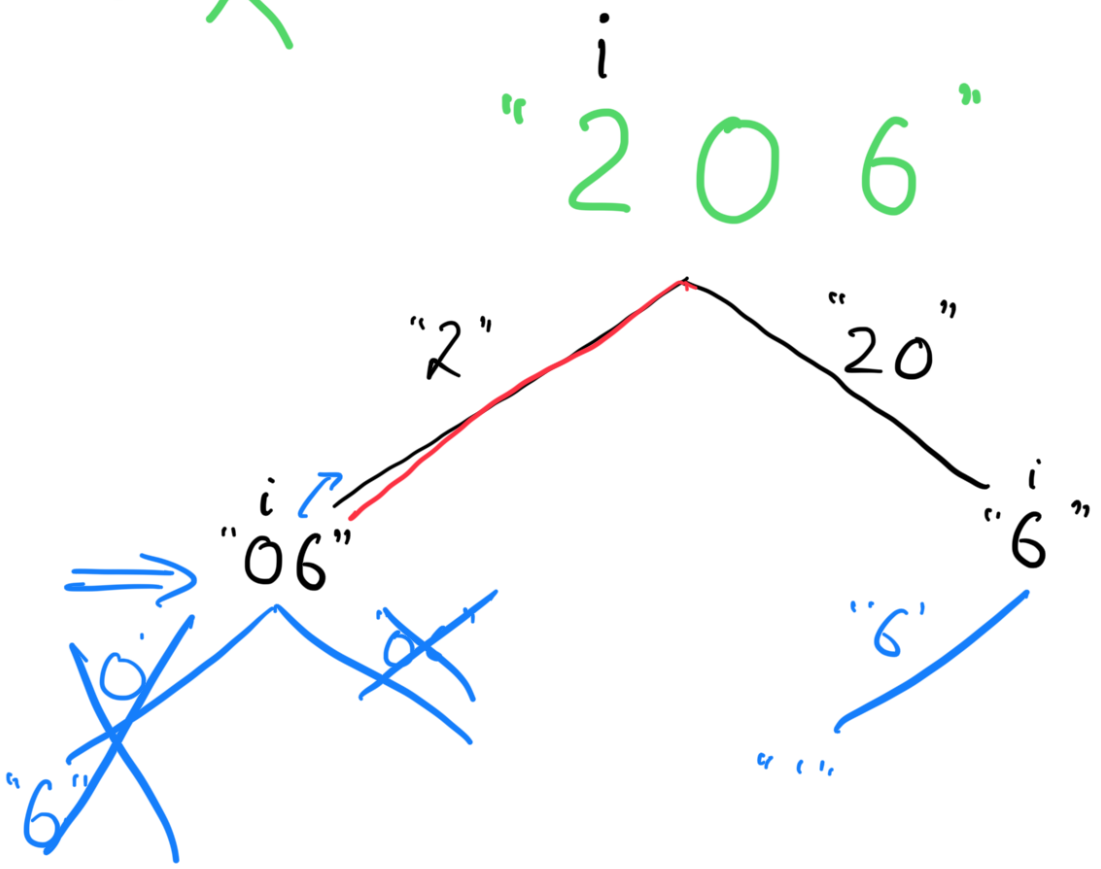
- (•) Options (Recursion) ✓
- (•) Repeating Subproblem (Memoize) ✓

Tree Diagram :-



i = 3

"06" X



20, 6

1	10	20
2	11	21
3	⋮	22
⋮	⋮	⋮
		26
9	19	

10

11

12

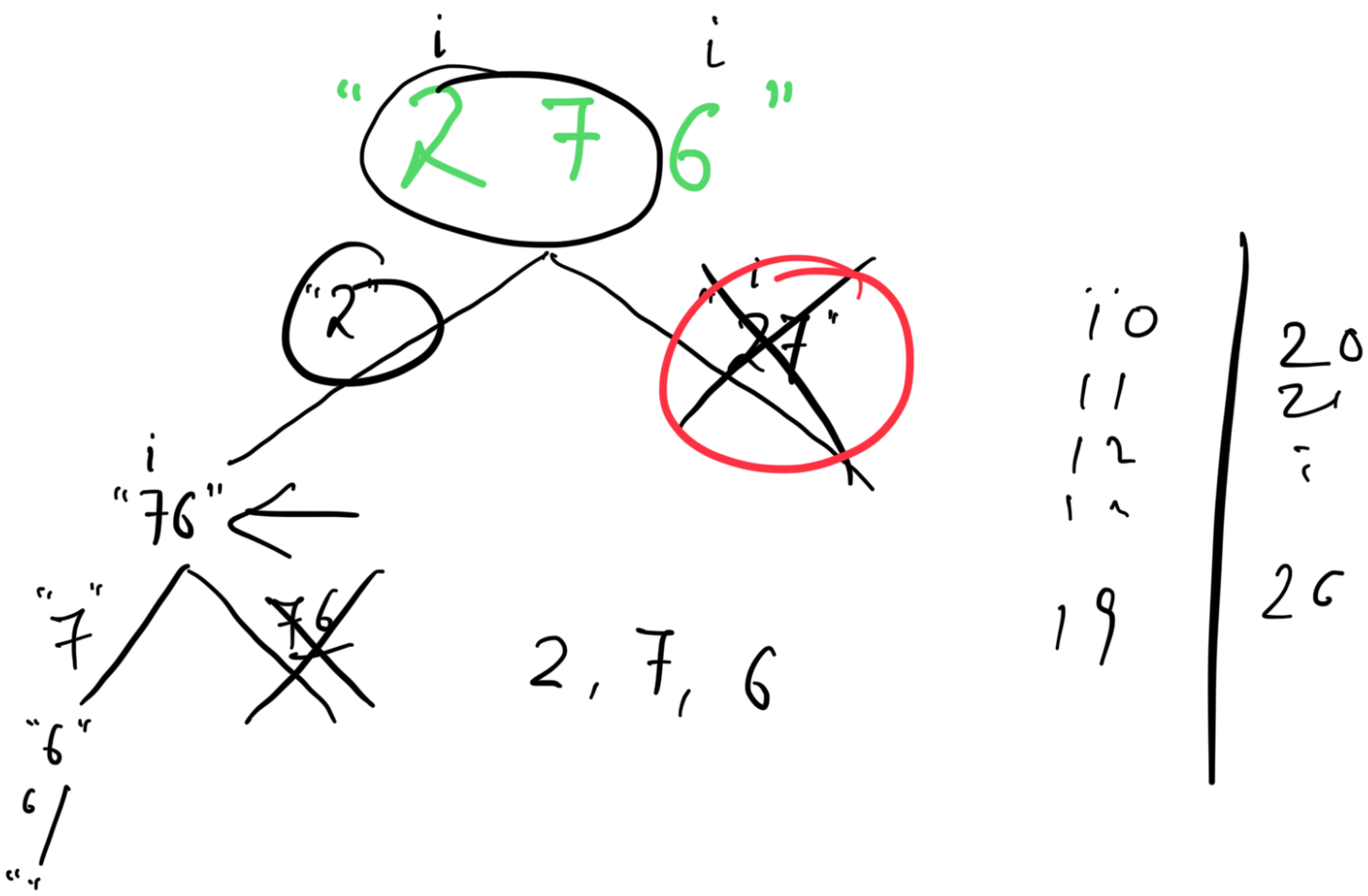
⋮

"2" → <= "6"

26 24 21

25 23 20

22



i

↳ Solve (i+1) ;

↳ Solve (i+2) ;

Bottom UP :-

t[i] = no of decode ways for string s from index i to n

t[i] = t[i+1];

solve(0)

```
int solve(int i, string &s, int &n) {
    if(t[i] != -1) {
        return t[i];
    }

    if(i == n) {
        return t[i] = 1; //one valid split done
    }

    if(s[i] == '0') {
        return t[i] = 0; //not possible to split
    }

    int result     = solve(i+1, s, n);
    if(i+1 < n) {
        if(s[i] == '1' || (s[i] == '2' && s[i+1] <= '6'))
            result += solve(i+2, s, n);
    }

    return t[i] = result;
```

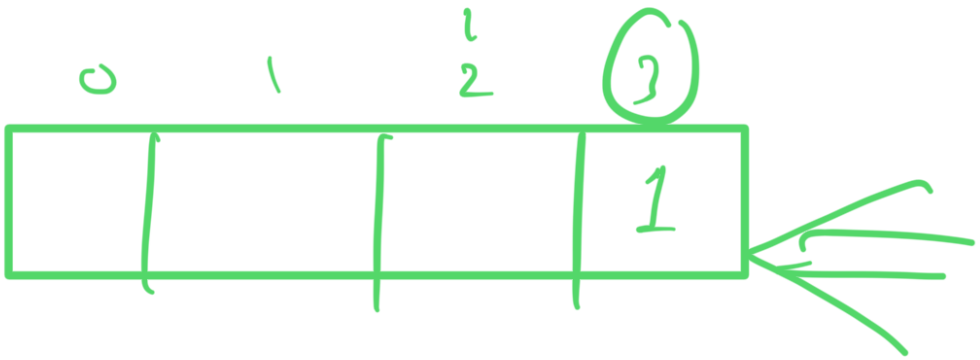
<recursive>t(n+1,0);

t[n] = 1;

for(i=n-1; i>=0; i--){

}

t[i] += t[i+2];



t[n] = 1;

for(i= n-1; i>=0; i++) {

```
if (s[i] == '0') {
    t[i] = 0;
} else {
    t[i] = t[i+1];
    if (i < n-1) {
        if (s[i] == s[i+1])
            t[i] = t[i+2];
    }
}
return t[0];
```