91: Decode Ways

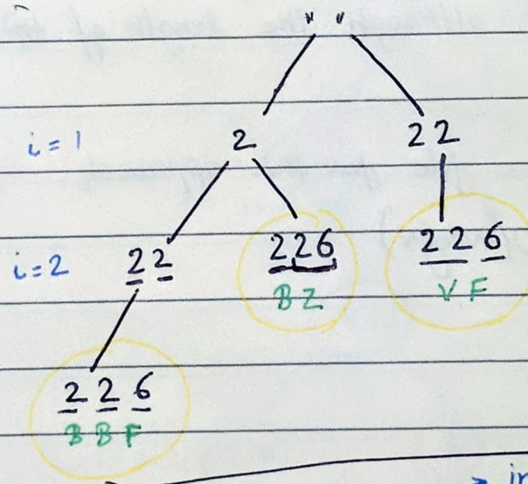
A → 1, B → 2 .... Z → 26

Given the string s = "226" for example, return the total number of ways in which it can be decoded.

226 ⇒ 2 2 6 - BBF
→ 22 6 - VF
2 26 - BZ

Brute force solution

At any position we can choose either one ~~a~~ character or two character (if two characters less than "26").



→ We can see the answer is number of leaves

→ We can recursively sum the number of leaves on left subtree and right subtree to get the answer.

index

```
def func (pos):
    if pos == length(s): return 1
    if s[pos] == '0':
    count = 0
    count += func(pos+1)
    if int(s[i:i+2]) >= 10 & <= 26      ← Pruning
        count += fun(pos+2)
    return count:
```

The below algorithm can be improved with memorization.
space & time → $O(n)$

(2) Iterative approach:-

recurrence relation:

if decoded one way only

$dp[i] = dp[i-1]$ → $\forall\ s[i] \;!= '0'$

if decoded two way only

$dp[i] = dp[i-2]$ → $\forall\ s[i-1:i+1] == 'x0'$
where $x = 1, 2$

if decoded both ways

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

$\forall\ s[i-1:i+1] \geq 10\ \&\ \leq 26$

2 6 1 1 0

dp =

1	1	2	2	4	$\boxed{2}$

↑ answer.

Check-code simple to understand

time & space = $O(n)$

(3) In previous approach we see we are only using last two values of dp to calculate the answer.
∴ instead of keeping the whole array we only keep the last two $dp[i-1]$ and $dp[i-2]$ and keep updating them.

Space complexity - $O(1)$ Time = $O(n)$