Decode ways

I/p : str = "1106" → decode

O/p

1 1 10 6 = AAJF
A A J F

11 10 6 = KJF
K J F

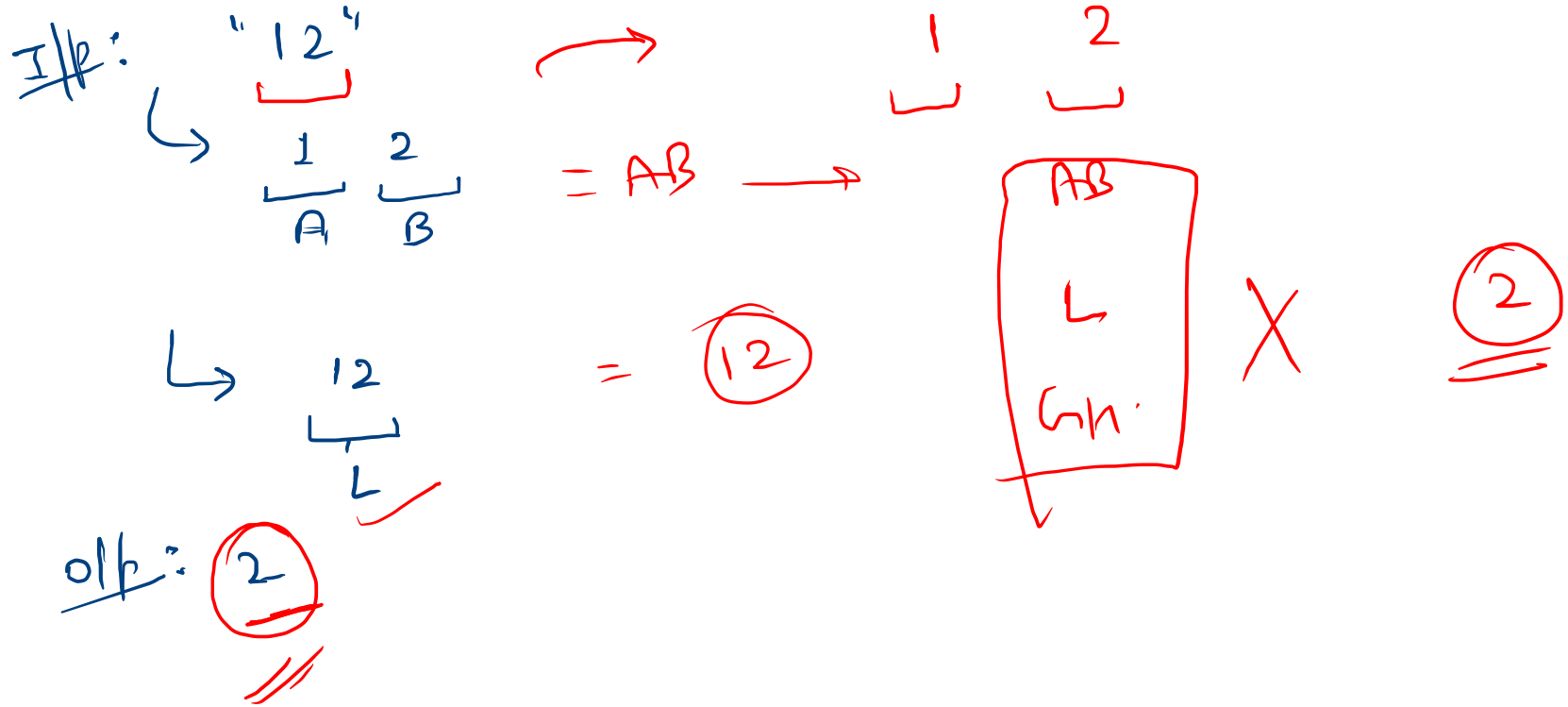
⇒ 2

Code

"A" → "1" ✓
"B" → "2" ✓
"C" → "3"
"Z" → "26"

06 X
6

X (1 11 06 → 06 X, 6 X
0 → 03 X → 3 X


I/p:
"12"
1
2
A
B
= AB
1
2
AB
L
Gn.
X
2
12
= 12
L
o/p:
2

Recursive Solution

"12306" → I/p

"12306"

pos (position)

n-1

I/p

(String s, int pos)

Magical function

o/p

In the string, s, determine no. of ways to decode substring from pos to end of string

"12306"

pos

"306"

way

o/p

No. of ways to decode that substring from pos to end of string

pos to n-1

pos

"12306"

decode(s, pos)

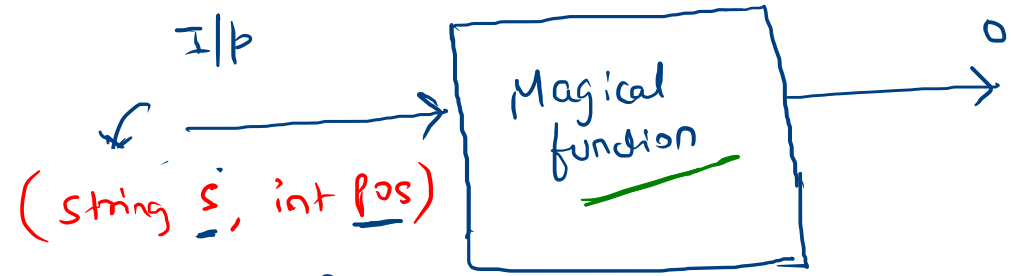
decode (s, 0)

"12306"

pos = 0

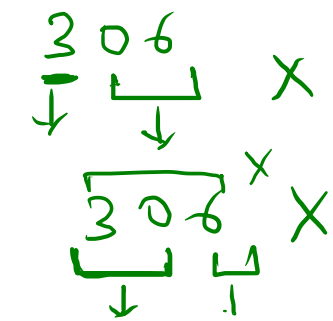
Recursive Solution

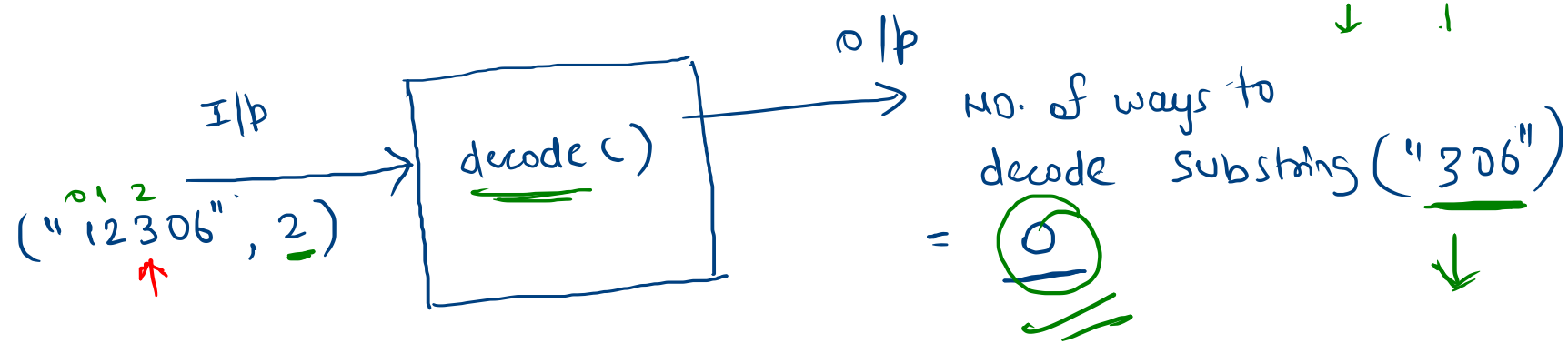
12306
↑
pos
(position)



In the string, S, determine no. of ways to decode substring from pos to end of string

A → 1
1
2 → 26

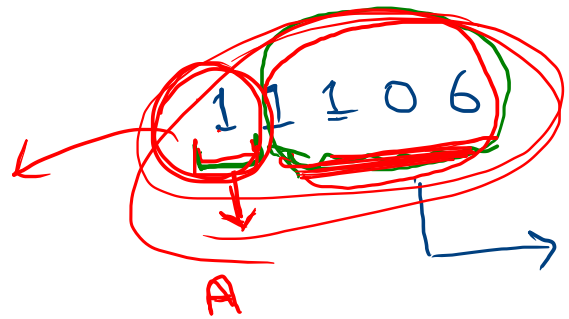


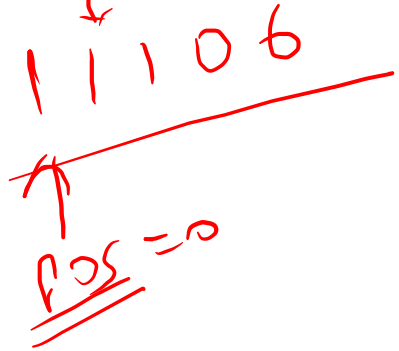


Logical Part

I/p: str = "11106"

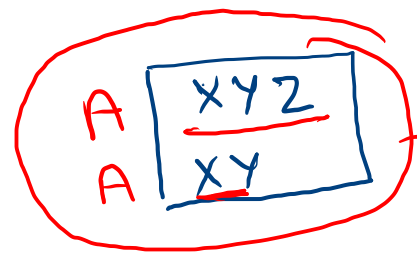
O/p:





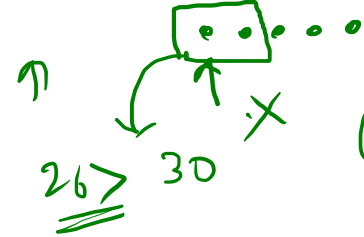
Single Character Consider

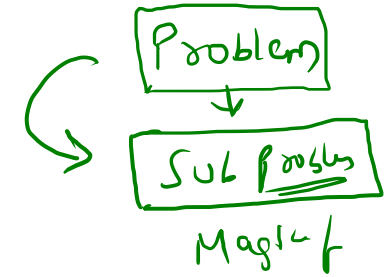
Let's suppose this substring can be decoded in '2' ways



→ 2 ways

int count = decode (s, pos+1);





Logical Part

I/p: str = "11106"

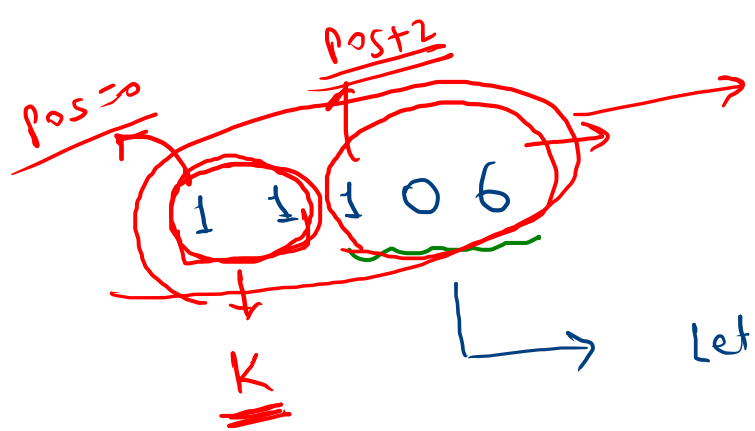
O/p:

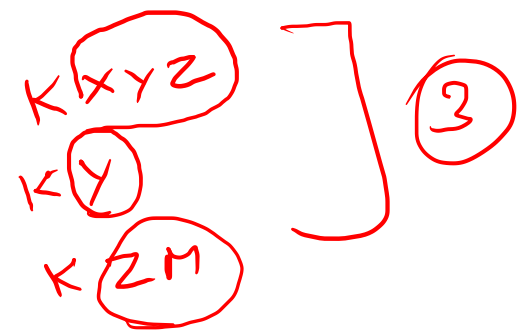
return count_1 + count_2 ;

1 1 1 0 6

A

int count = decode (s, pos+1);





Let's suppose this substring can be decoded in '3' ways

int count_2 = decode (s, pos+2)

XYZ
XY
ZM

Extended version

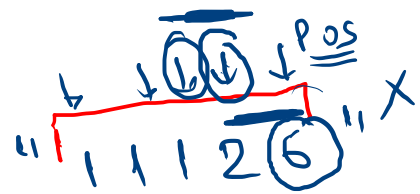
26

Before Proceeding we have to check this valid Number or not ?

Let's suppose this substring can be decoded in '3' ways

XYZ
XY
ZM

```
int count_2 = decode(s, Pos+2)
```



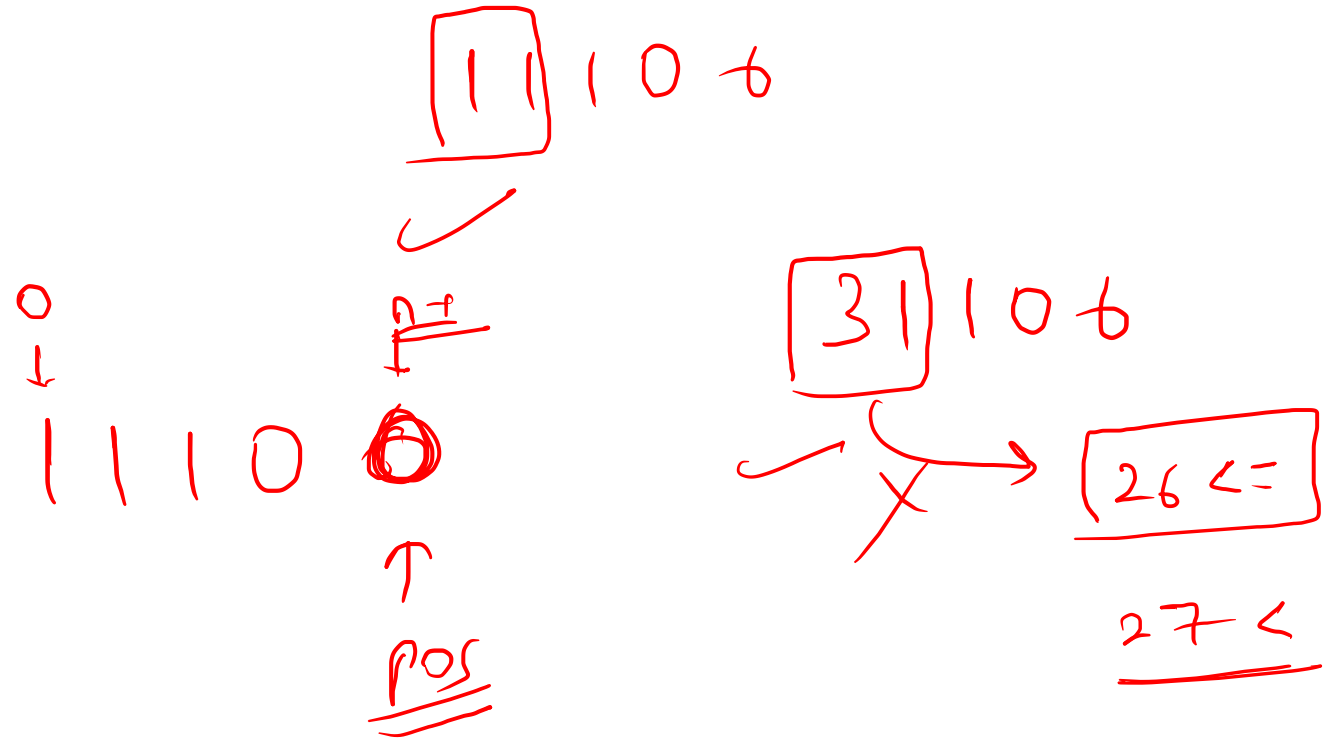
str =

if 'n' = str.length()

(Pos < n-1) **

And, we have to check Number is Smaller than "26"

Beoz, max^m Coded Value is "26"


11106
0
1110
pos
31106
26 <=
27 <

Logical Part

I/p: str = "11106"

O/p:

11106

A

```
int count = decode(s, pos+1);
```

pos < n-1

Chance < 27

11106

K

```
int count_2 = decode(s, pos+2)
```

```
return count + count_2 ;
```

Recursive code

```
int decode ( string s, int pos)
{
    // Base Condition

    int count = decode ( s, pos+1);
    if ( pos < n-1 && s.substr(pos, 2) < "27")
    {
        count += decode ( s, pos+2);
    }

    return count;
}
```

s = 1 1 1 0 6

pos

"10" < "27"

"30" < "27"

Base condition

✓ Smallest valid Input → check output

int decode (string s , int pos)

{

⇒ n = s.size()

1, 2, 3, 4 .....

if (n == 1) return 1 ;

pos

pos = 0

pos = 1

⇒ "2" → ↓

pos+1

0 X

06 X

"0634"

return 0;

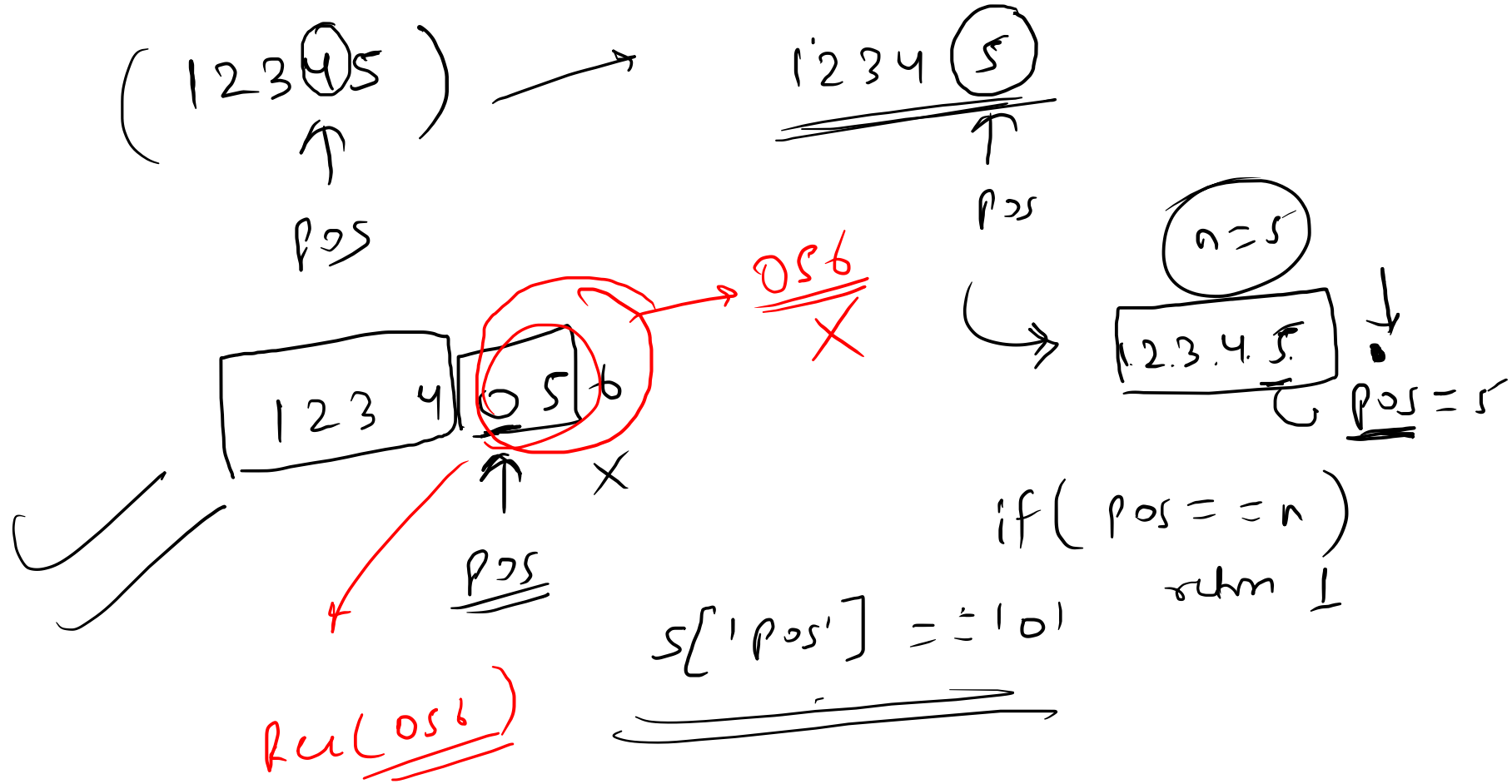
05 X

n = 1

if (n == pos)

return 1.

s[pos] == '0'


(1234 5)
pos
1234 5
pos
n=5
1.2.3.4.5.
pos=5
1234 5 6
X
pos
Del(pos)
if(pos==n)
return 1
s['pos'] = '0'

Recursive code

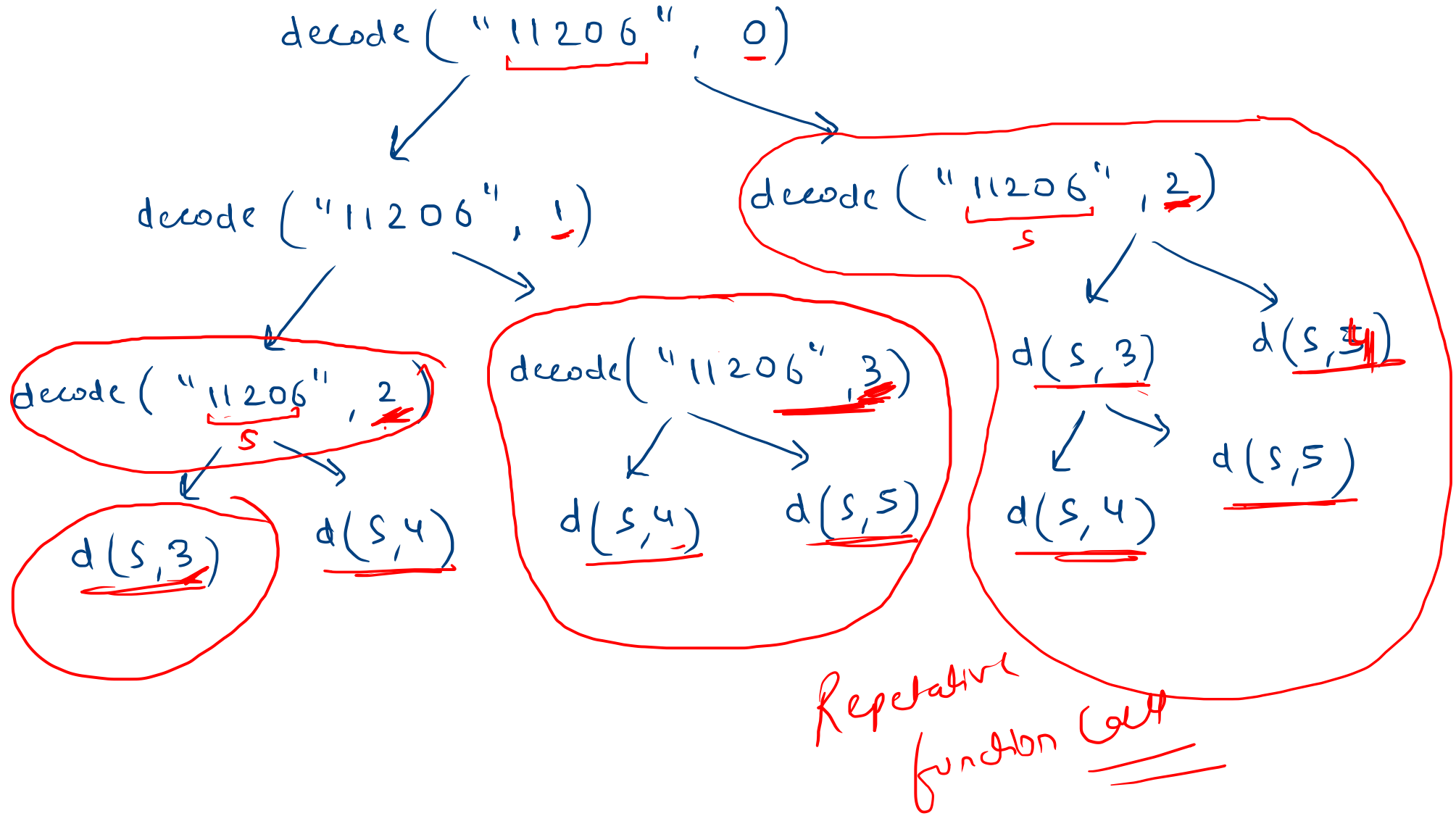
```
int decode ( string s , int pos )
{
    if ( pos == n ) return 1;
    if ( s[pos] == '0' ) return 0;

    int count = decode ( s, pos+1 );
    if ( pos < n-1 && s.substr(pos, 2) < "27" )
    {
        count += decode ( s, pos+2 );
    }

    return count;
}
```

Base Condition

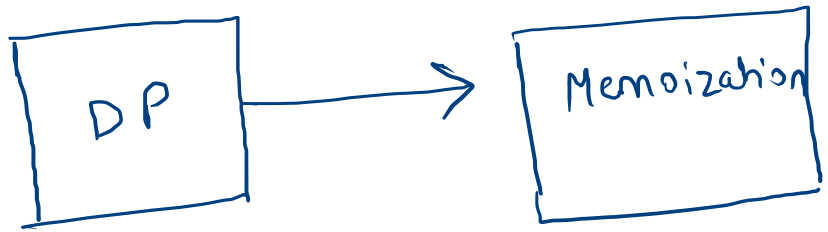
Logical part

Recursive Tree

$O(2^N)$



Memoization Code



vector <int> dp (101, -1) ;

Recursive code

```
int decode ( string s, int pos)
{
    if ( pos == n) return 1;
    if ( s[pos] == '0' ) return 0;
**  if ( dp[pos] != -1) return dp[pos];
    int count = decode ( s, pos+1);
    if ( pos < n-1 && s.substr(pos, 2) < "27")
    {
        count += decode ( s, pos+2);
    }
    return dp[pos] = count;
}
```

Base Condition

Logical part

dp