Count binary strings

n = 3

~~000~~

~~001~~

010 ✓

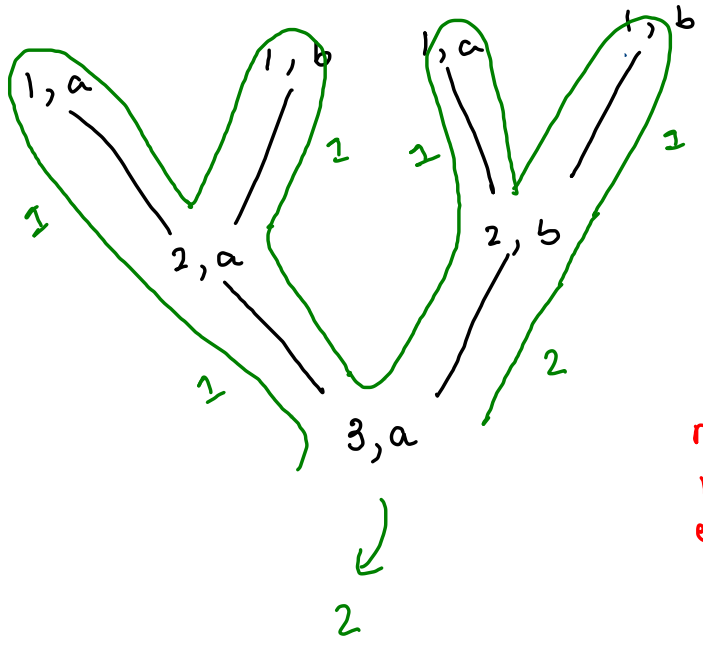
011 ✓

~~100~~

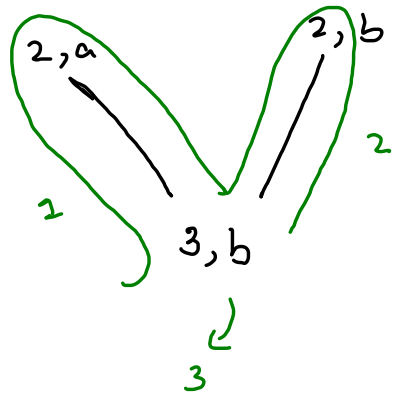
101 ✓

110 ✓

111 ✓



5



n, a → (n-1), b

ending at '0'

n, b → (n-1), a + (n-1), b

ending at '1'

a → count of strings which ends at 0.

b → count of string which ends at 1.

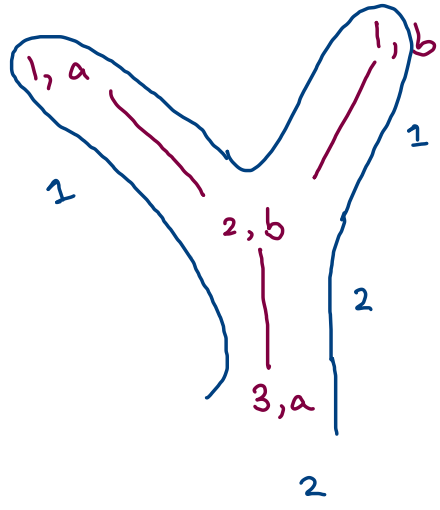
2 + 3 = 5

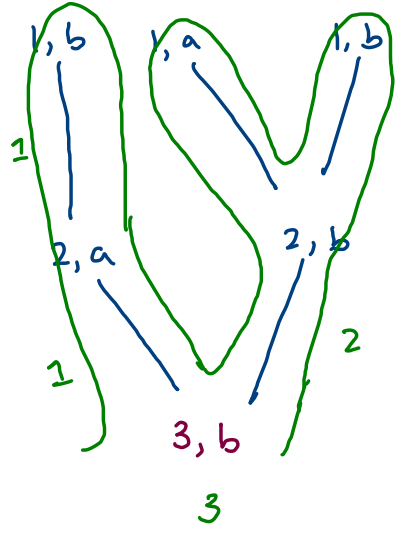
```
int ans = cbs(n,'a') + cbs(n,'b');
```

```
public static int cbs(int n, char type) {
    if(n == 1) {
        return 1;
    }

    int ans = 0;

    if(type == 'a') {
        //n length binary string which ends at 0 and has no consecutive 0's
        ans = cbs(n-1,'b');
    }
    else {
        //n length binary string which ends at 1 and has no consecutive 0's
        ans = cbs(n-1,'a') + cbs(n-1,'b');
    }

    return ans;
}
```

a -> ending at 0, no consecutive 0's

b -> ending at 1, no consecutive 0's

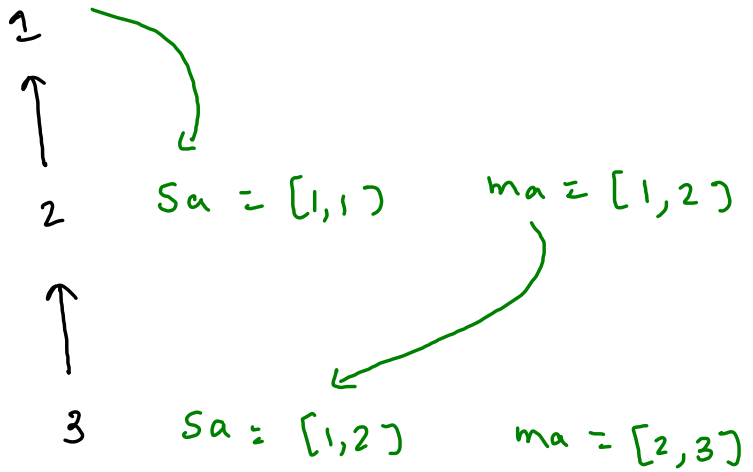




		0	1	2	3	4	5	
ending at 0.	0	–	1 (0)	1 (10)	2 (110, 010)	3	5	(a)
ending at 1.	1	–	1 (2)	2 (11, 01)	3 (101, 111, 011)	5	8	(b)

n = 5

```
public static int[] cbs2(int n) {
    if(n == 1) {
        int[] ba = {1,1};
        return ba;
    }

    int[] sa = cbs2(n-1);
    int[] ma = {sa[1],sa[0] + sa[1]};

    return ma;

}
```



	0	1	2	3	4
ending at 0	0		1	1	2	3 (ocz)
ending at 1	1		1	2	3	5 (oco)

n = 4

```
public static int cbs(int n) {
    int ocz = 1;  //old count zero
    int oco = 1;  //old count one

    for(int i=2; i <= n;i++) {
        int ncz = oco; //new count zero
        int nco = oco + ocz; //new count one

        oco = nco;
        ocz = ncz;
    }

    return ocz + oco;
}
```

8

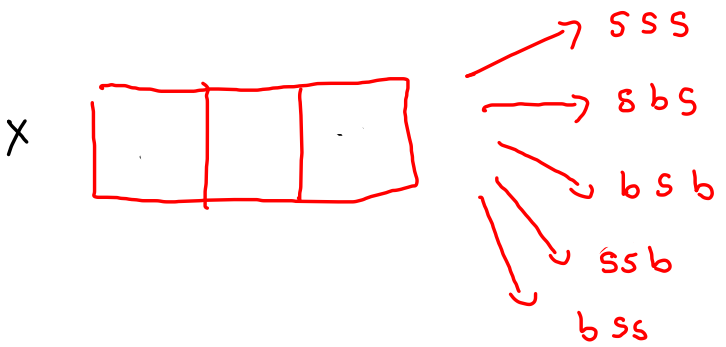
Arrange building

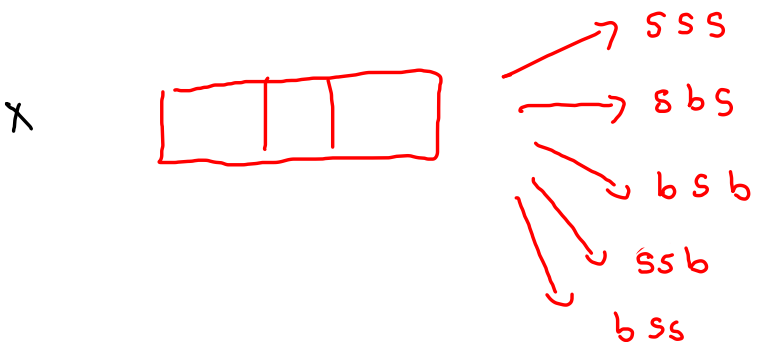
1. You are given a number n, which represents the length of a road. The road has n plots on it's each side.
2. The road is to be so planned that there should not be consecutive buildings on either side of the road.
3. You are required to find and print the number of ways in which the buildings can be built on both side of roads.

b -> building

s -> space

total : x * x





		0	1	2	3	4
ending at s	0	X	1 (s)	2 (ss, bs)	3 (sss, bss, sbs)	5
ending at b	1	X	1 (b)	1 (sb)	2 (ssb, bss)	3

total = 8 * 8 = 64

count encodings

1 1 2 3 →

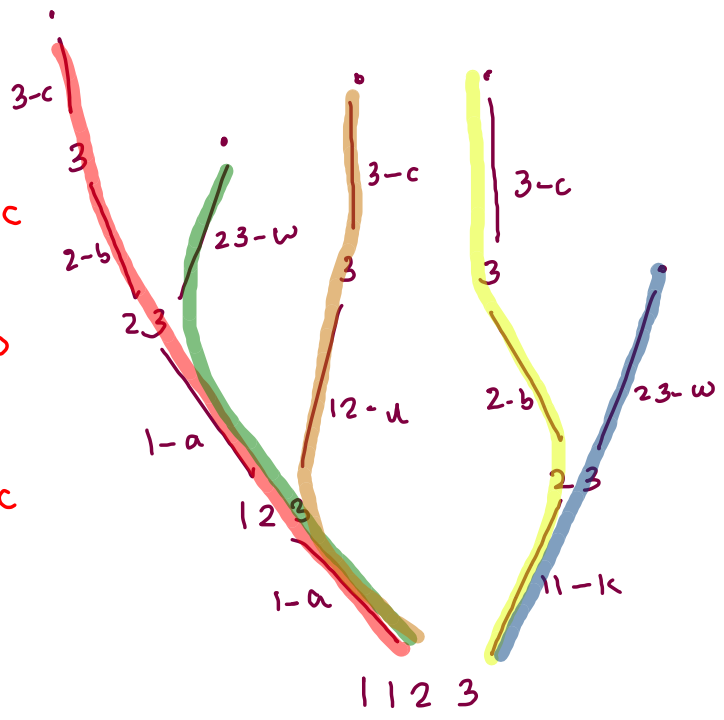
a a b c

a k c

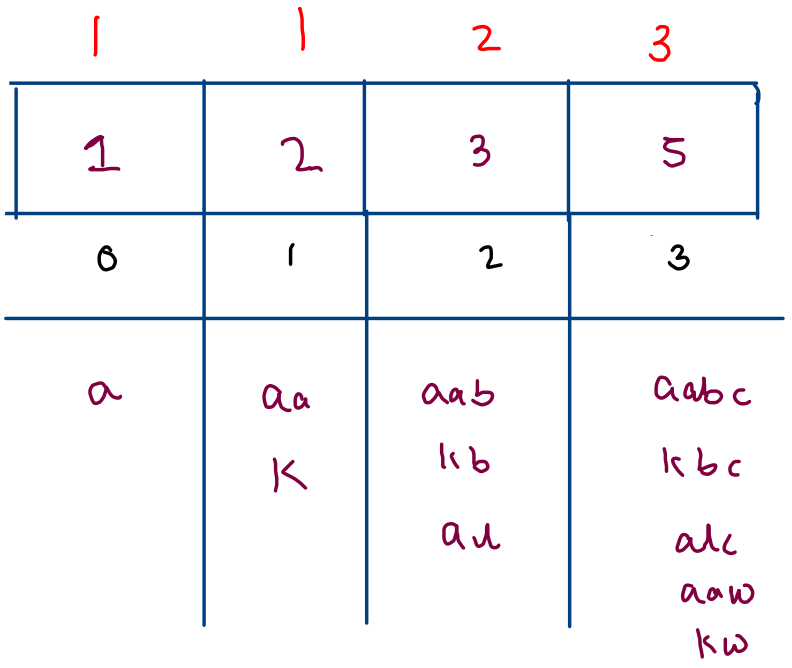
a a w

k w

k b c



1 – a	11 – k	21 – u
2 – b	12 – l	22 – v
3 – c	13 – m	23 – w
4 – d	14 – n	24 – x
5 – e	15 – o	25 – y
6 – f	16 – p	26 – z
7 – g	17 – q	
8 – h	18 – r	
9 – i	19 – s	
10 – j	20 – t	

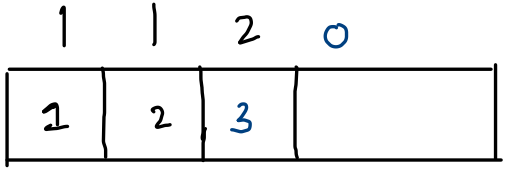


dp[i] -> encoding of

string : [0, i]

dp[i] -> count of encoding of string till ith index.

	1	1	2	6	3	0	4	2	0
1	1	2	3	5	5	0	0	0	0
0	1	2	3	4	5	6	7	8	9
	a	aa k	aab kb au	aabf kbf adf aaz kz	aabfc kbfc adfc aazc kzc				



(i) $i \rightarrow n_2$ $(i-1) \rightarrow n_2$

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

no. created by i and $i-1 \leq 26$

(ii) $i \rightarrow n_2$ $(i-1) \rightarrow 2$

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

(iii) $i \rightarrow 2$ $(i-1) \rightarrow 2$

$dp[i] = 0$

(iv) $i \rightarrow 2$ $(i-1) \rightarrow n_2$

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

↳ $i-1$ char should be '1', '2'

```java
for(int i=2; i < dp.length;i++) {
    char p = str.charAt(i-1);
    char c = str.charAt(i);

    if(c == '0' && p == '0') {
        //c and p both are '0'
        dp[i] = 0;
    }
    else if(c == '0') {
        //only c is '0'
        if(p == '1' || p == '2') {
            dp[i] = dp[i-2];
        }
    }
    else if(p == '0') {
        //only p is '0'
        dp[i] = dp[i-1];
    }
    else {
        //c and p both are non-zero
        dp[i] = dp[i-1];

        int num = Integer.parseInt(p + "" + c);

        if(num <= 26) {
            dp[i] += dp[i-2];
        }
    }
}
```

2 - 4 — 10 — 6

2 4 — 10 — 6

•	2	4	1	0	6
1	1	2	2	2	2
0	1	2	3	4	5
:	b	bd w	bda wa	bdj wj	bdjF wjF