Largest Square Sub-matrix With All 1's

5 6

0 1 0 1 0 1

1 0 1 0 1 0

0 1 1 1 1 0

0 0 1 1 1 0

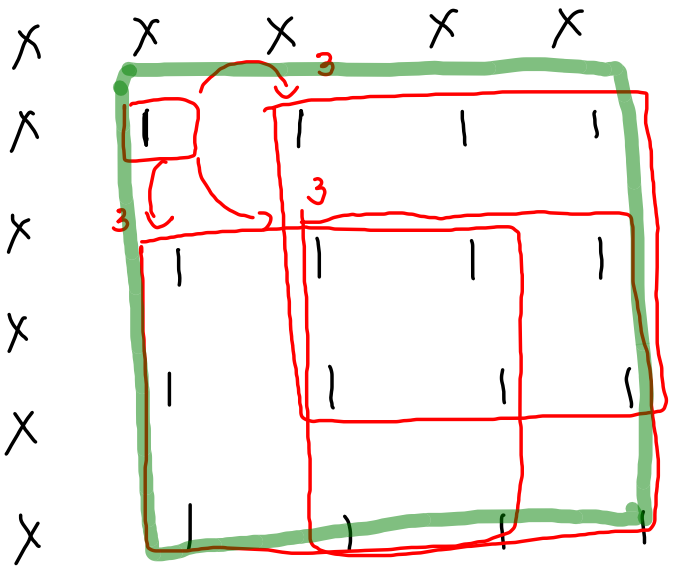
1 1 1 1 1 1

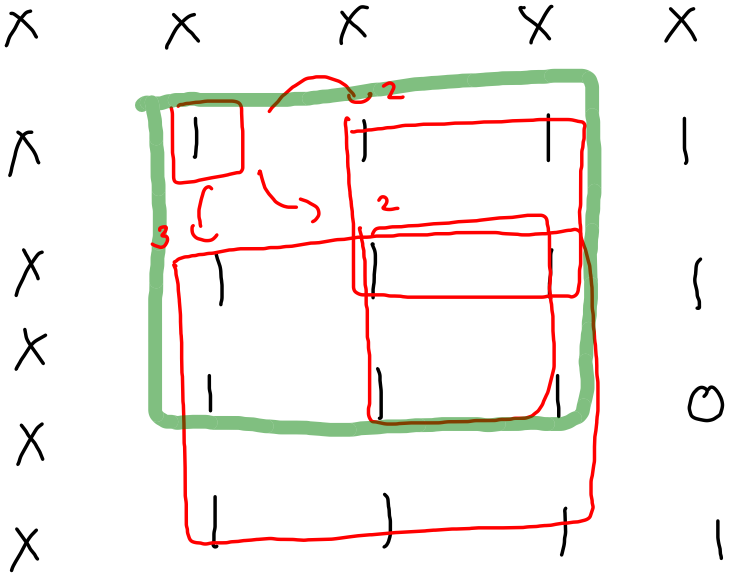
	0	1	2	3	4	5
0	0	1	0	1	0	1
1	1	0	1	0	1	0
2	0	1	3	2	1	0
3	0	0	2	2	1	0
4	1	1	1	1	1	1

$dp[i][j] = \min(dp[i][j+1], dp[i+1][j], dp[i+1][j+1]) + 1;$

$dp[i][j]$ → longest square created if i,j is considered as top-left corner of square

$$dp[i][j] = \min(dp[i][j+1], dp[i+1][j], dp[i+1][j+1]) + 1;$$





Fractional Knapsack - Official

```
5
15 14 10 45 30
2 5 1 3 4
7
```

	0	1	2	3	4
values	15	14	10	45	30
wt	2	5	1	3	4
v/w	7.5	2.8	10	15	7.5
	(1)	(0)	(3)	(4)	(2)

sort (v/w)

(i) item is breakable

(ii) only one instance of each item is available.

	0	1	2 ✓c	3 ✓c	4 ✓p
values	15	14	10	45	30
wt	2	5	1	3	4
v/w	7.5	2.8	10	15	7.5
	(1)	(0)	(3)	(4)	(2)

cap = 7

3-45

1-10

(4-30) × 3/4

rc = ~~7 4 3~~

0

profit = 45 + 10

+ 22.5

= 77.5

	v_c		v_c	v_c	v_p
	0	1	2	3	4
values	15	14	10	45	30
wt	2	5	1	3	4
v/w	7.5	2.8	10	15	7.5
	(1)	(0)	(3)	(4)	(2)

rc

$= \cancel{7}\ \cancel{4}\ \cancel{3}\ 0$

profit $= 45 + 10 + 22.5$

$= 77.5$

3-45

1-10

$(4-30) \times \frac{3}{4}$

	v_c		v_c	v_c	v_p
	0	1	2	3	4
values	15	14	10	45	30
wt	2	5	1	3	4
v/w	7.5	2.8	10	15	7.5
	(2)	(0)	(3)	(4)	(1)

rc $= \cancel{7}$

$\cancel{4}\ \cancel{3}$

$\cancel{2}\ 0$

profit $= 45 + 10 + 15 + 7.5 = 77.5$

3-45

1-10

2-15

$(4-30) \times \frac{1}{4}$

	0	1	2	3	4
values	15	14	10	45	30
wt	2	5	1	3	4

2, 15, 7.5	5, 14, 2.8	1, 10, 10	3, 45, 15	4, 30, 7.5

↓ Sort (ratio)

i

5, 14, 2.8	2, 15, 7.5	4, 30, 7.5	1, 10, 10	3, 45, 15

pair : wt val valwt

```
int rc = W;
double profit = 0.0;

for(int i=n-1; i >= 0;i--) {
    Pair p = a[i];

    if(p.wt <= rc) {
        //use this item completely
        rc -= p.wt;
        profit += p.val;
    }
    else {
        //use this item partially
        profit += (p.r * rc);
        rc = 0;
        break;
    }
}
```

rc = ~~7~~ ~~4~~ ~~3~~ 0

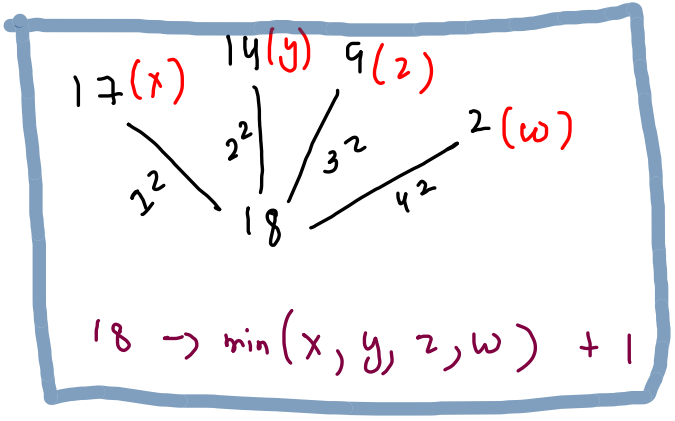
profit = 45 + 10 + 7.5 * 3

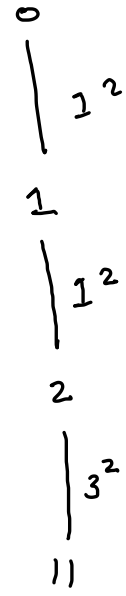
= 77.5

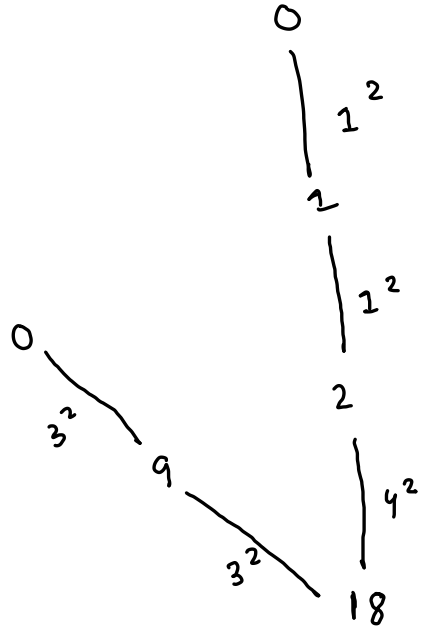
Perfect Squares

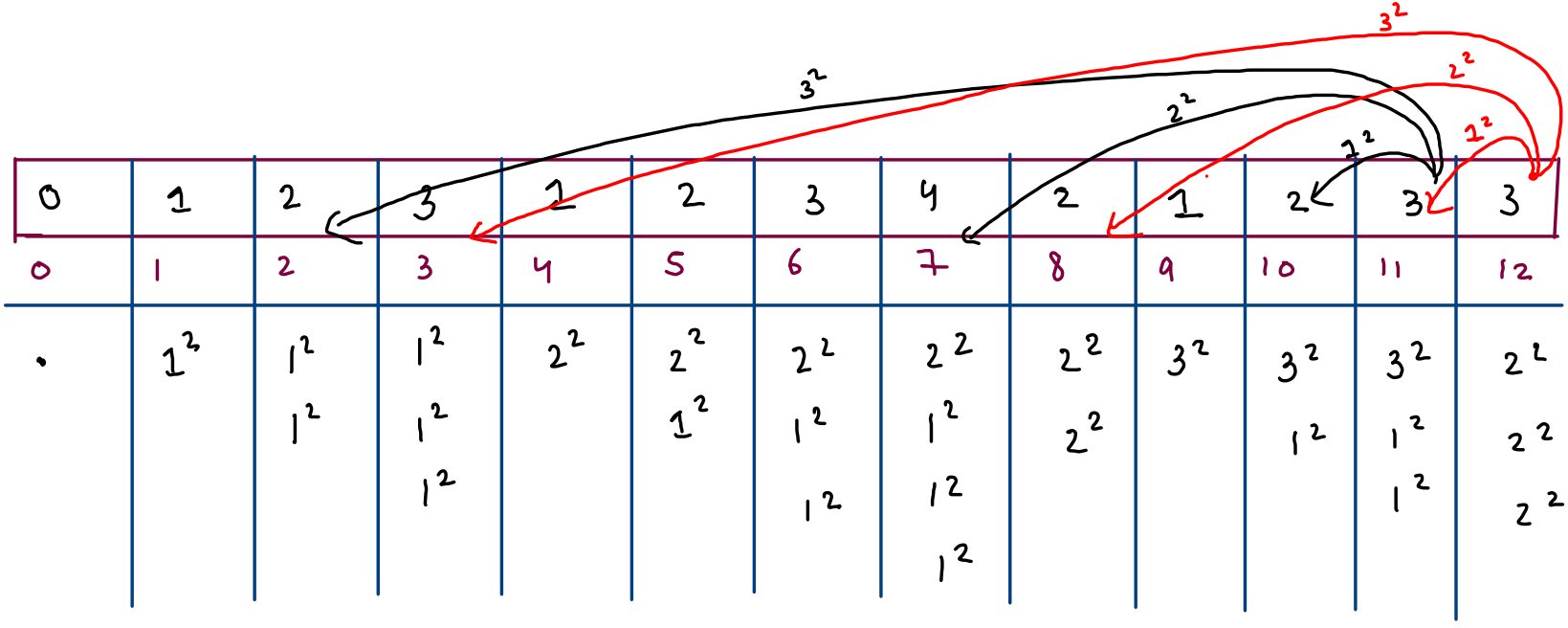
$11 \rightarrow 3^2 + 1^2 + 1^2$

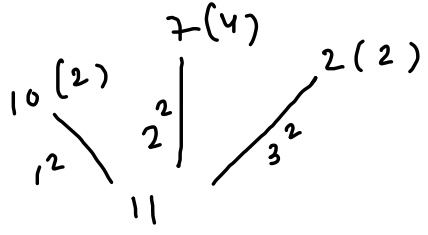
1. You are given integer N
2. You need to find least number of perfect squares that amount to N
3. Input and output is handled for you
4. It is a functional problem ,please do not modify main()

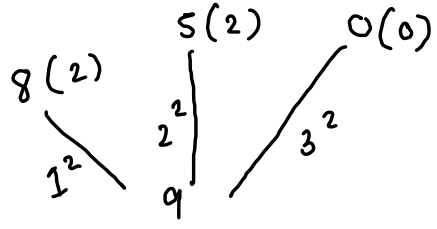










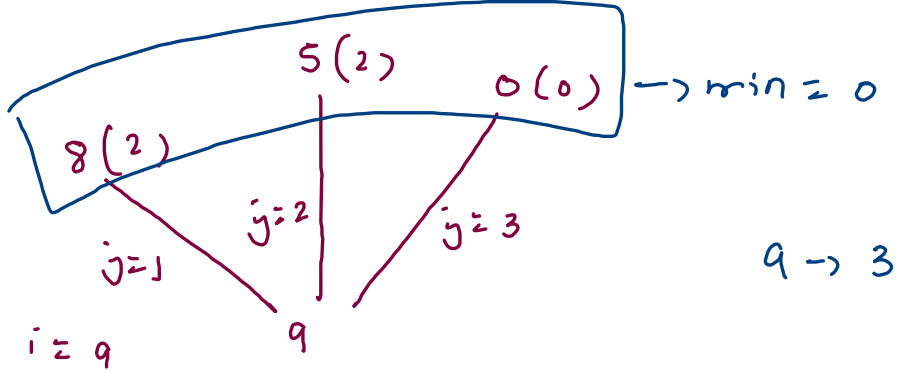


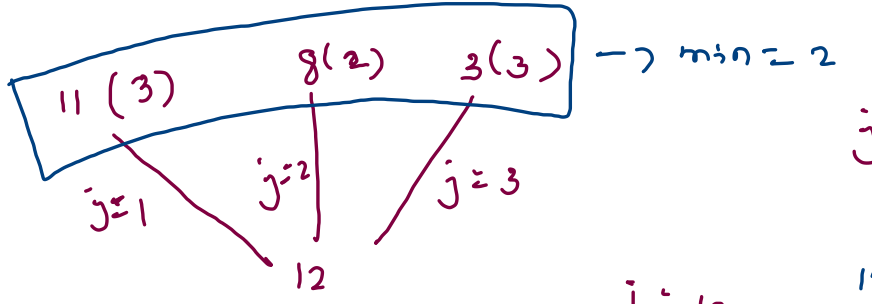
dp[i] -> min steps required to represent i

```java
private static int count(int n) {
    //Write your code here
    int[]dp = new int[n+1];
    dp[0] = 0;

    for(int i=1; i <= n;i++) {
        int min = Integer.MAX_VALUE;
        for(int j=1; j*j <= i ;j++) {
            min = Math.min(dp[i- j*j],min);
        }
        dp[i] = min+1;
    }

    return dp[n];
}
```





Catalan Number

C0 -> 1
C1 -> 1
C2 -> 2
C3 -> 5
..
Cn -> C0.Cn-1 + C1.Cn-2 + .. + Cn-2.C1 + Cn-1.C0

$C_0 = 1$

$C_1 = 1$

$C_2 = C_0 C_1 + C_1 C_0 = 2$

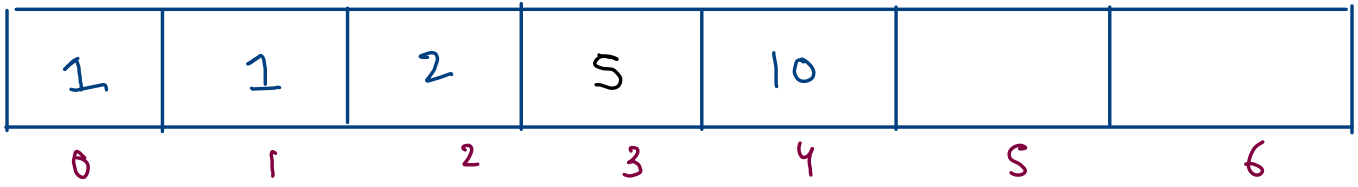
$C_3 = C_0 C_2 + C_1 C_1 + C_2 C_0$

$\rightarrow 1 \times 2 + 1 \times 1 + 2 \times 1 = 5$

$C_4 = C_0 C_3 + C_1 C_2 + C_2 C_1 + C_3 C_0$

$= 1 \times 5 + 1 \times 2 + 2 \times 1 + 5 \times 1 = 14$

$n = 6$



dp[i] → catalan of i

$$C_i = \sum_{l=0}^{i-1} C_l * C_r$$

$l + r = i - 1$

$r = i - 1 - l$

$C_4 = C_0 * C_3 + C_1 * C_2 + C_2 * C_1 + C_3 * C_0$

5 + 2 + 2 + 5

$(i = 4)$

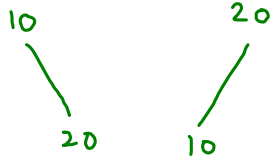
$l = 0$	$l = 1$	$l = 2$	$l = 3$
$r = 3$	$r = 2$	$r = 1$	$r = 0$

T : O(n2)

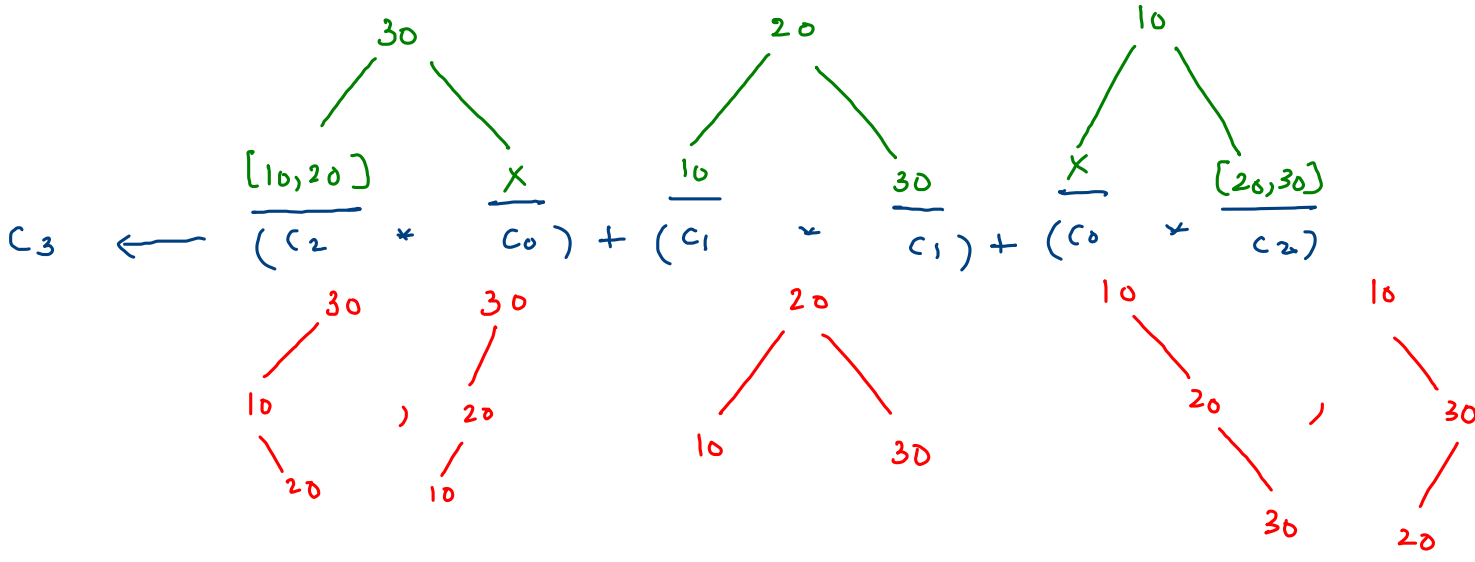
S : O(n)

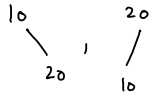
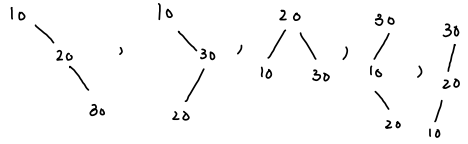
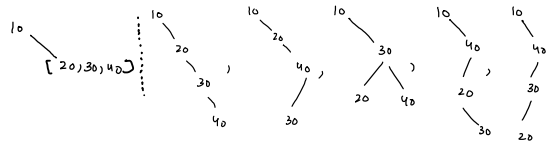
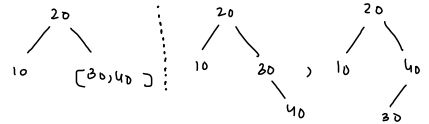
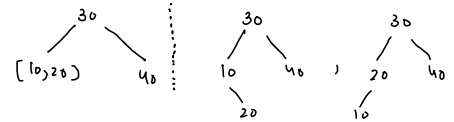
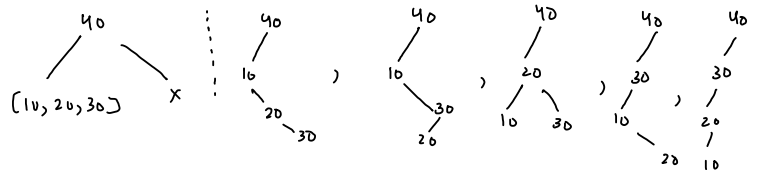
Number Of Bsts

n = 2 (10, 20)



n = 3 (10, 20, 30)



i	catalan	ways
C_0	1	.
C_1	1	10
C_2	2	
C_3	5	
C_4	C_0C_3	
	+	
	C_1C_2	
	+	
	C_2C_1	
	+	
	C_3C_0	

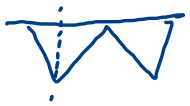
Count Of Valleys And Mountains

n = 2

(n pairs of upstroke, downstroke)

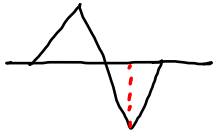
note: at any moment

count of upstrokes ≥ count of downstrokes

i.e  (invalid)

↳ $c_u = 0$

$c_d = 1$

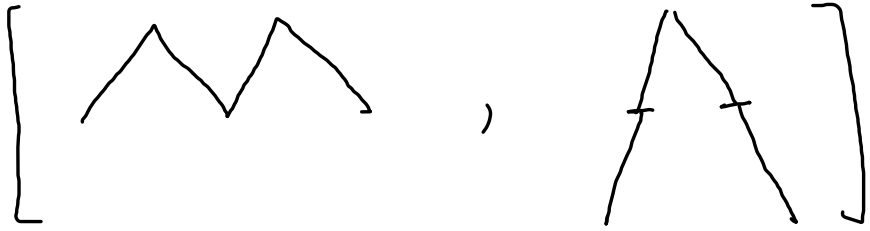


↳ $c_u = 1$

$c_d = 2$

/ → upstroke

\ → downstroke



→ n = 2 valid ways

C_0 1

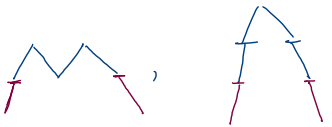
C_1 1

C_2 2

C_3 C_0C_2 →

C_1C_1 →

C_2C_0 →



C_4 C_4C_x ↳ outside with me (above)

C_0C_3 outside: 3, wm: 0

C_1C_2 outside: 2, wm: 1

C_2C_1 outside: 1, wm: 2

C_3C_0 outside: 0, wm: 3