Fractional knapsack :-

Cap → 9

due to fractional knapsack

maximise value ? in knapsack

object										
value :→	33	14	50	9	8	11	6	40	2	15
wt :→	7	2	5	9	3	2	1	10	3	3
value per wt →	4.71	7	10	1	2.66	5.5	6	4	0.66	5
Rank.(Max. value per w t) →	6	2	1	9	8	4	3	7	10	5

ob

double.

order of pickup for knapsack

Cap.

~~9~~ ~~4~~

~~2~~

~~1~~

0

Value → 10*5 = 50

7*2 = 14

6*1 = 6

5.5*1 = 5.5

Sorting

Profit/Total val = 75.5

comparison between double ?

return this.val_by_wt - o.val_by_wt;

7.5 — 7.3

Result = 0.2

In integer = 0

'0' mean, this & o. is equal x

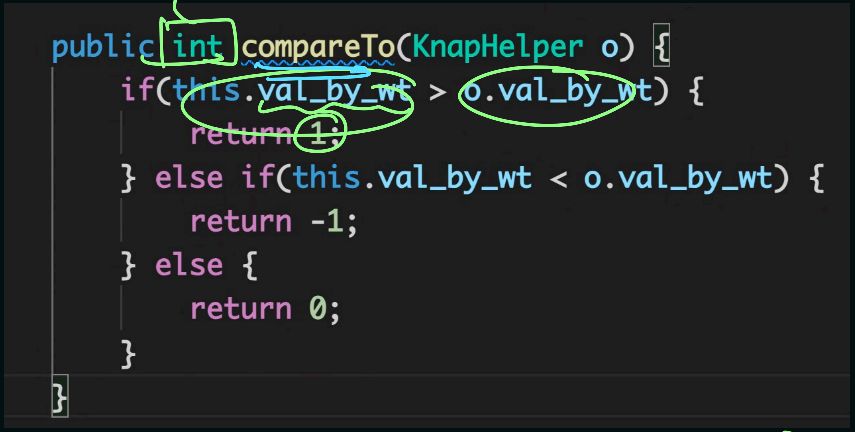
Standard

↳ return this.check - other.check

+ve → 'this' is greater

-ve — this is smaller

0 → both are equal



[illegible]

```
public int compareTo(KnapHelper o) {
    if(this.val_by_wt > o.val_by_wt) {
        return 1;
    } else if(this.val_by_wt < o.val_by_wt) {
        return -1;
    } else {
        return 0;
    }
}
```

```
if ([illegible] — > 0) {
    → -1;
} else if( this < 0) {
    — +1;
} else {
    → 0;
}
```

Perfect square: Given an integer n, return the least number of perfect square numbers that sum to n

$\underbrace{\text{least number}}$ → Min nos.

n= 7 → $\underbrace{1^2+1^2+1^2}_{④}+2^2 = 7$

n= 9 → $\underbrace{3^2}_{①} = 9$

n= 5 → $1^2+2^2=5$ ②

n= 3 → $\underbrace{1^2+1^2+1^2}_{③} = 3$

n= 4 → 2^2 → ①

n=11

0	1	2	3	4	5	6	7	8	9	10	11
0	1	2	3	1	2	3	4	2	1	2	3
-	1^2	1^2 1^2	1^2 1^2 1^2	2^2	1^2 2^2	1^2 1^2 2^2	1^2 1^2 1^2 2^2	2^2 2^2	3^2	1^2 3^2	1^2 3^2 1^2

Print all Longest Increasing subseq :-

length of Longest incre. subseq Ending at 50

max

10	22	9	33	21	50	41	60	80	1
1	2	1	3	2	* 4	4	5	6.	1
10	10 20	9	10 20 33	10 21 OR 9 21	10 20 33 50	10 20 33 41	10 20 33 50 60 OR 10 20 33 41 60	10 20 33 50 60 80 OR 10 20 33 41 60 80	1

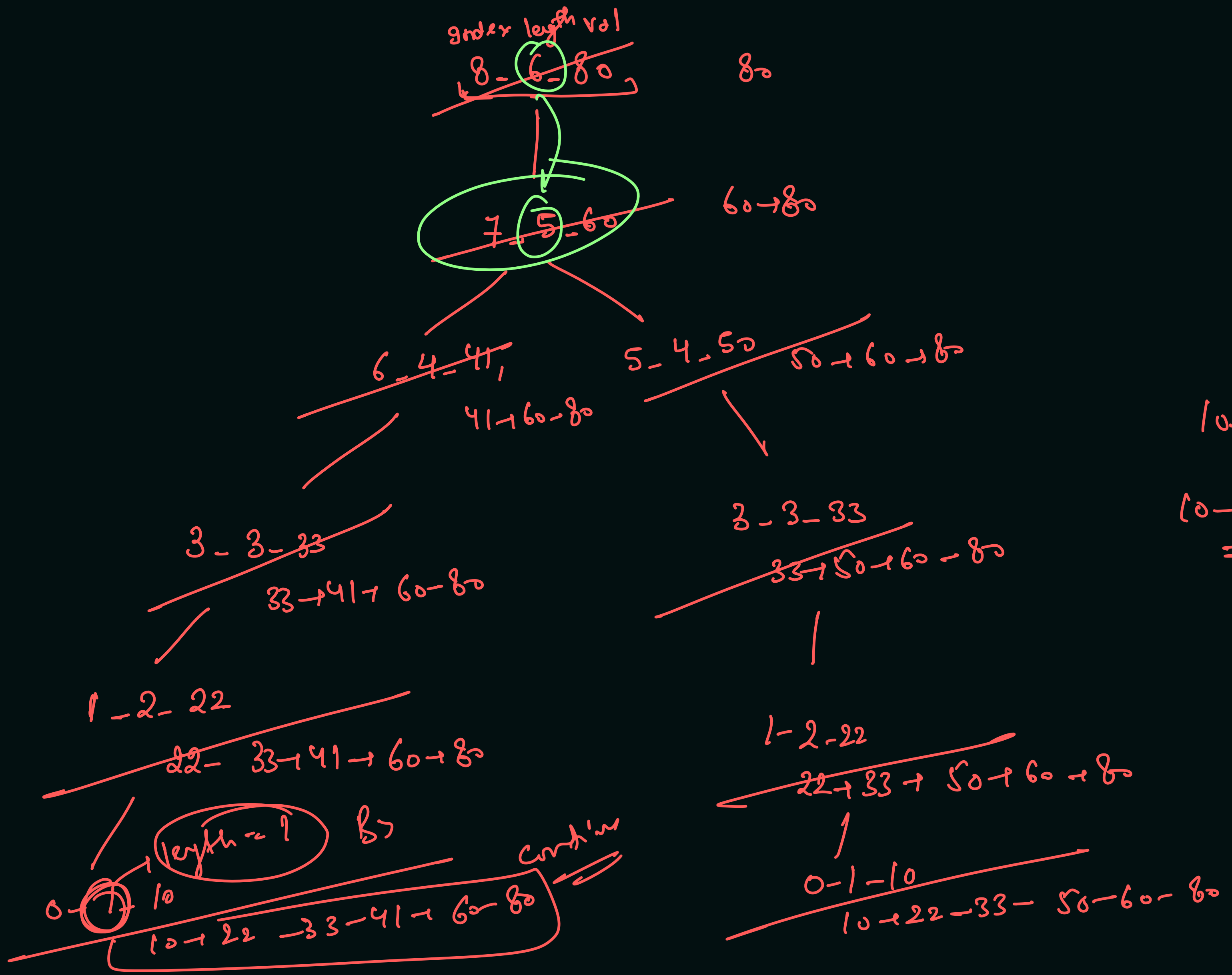
→ All LIS

10 → 20 → 33 → 50 → 60 → 80

10 → 20 → 33 → 41 → 60 → 80

max Index = 8

10	22	9	33	21	50	41	60	80	1
1	2	1	3	2	*4	4	5	6.	1
0	1	2	3	4	5	6	7	8	9



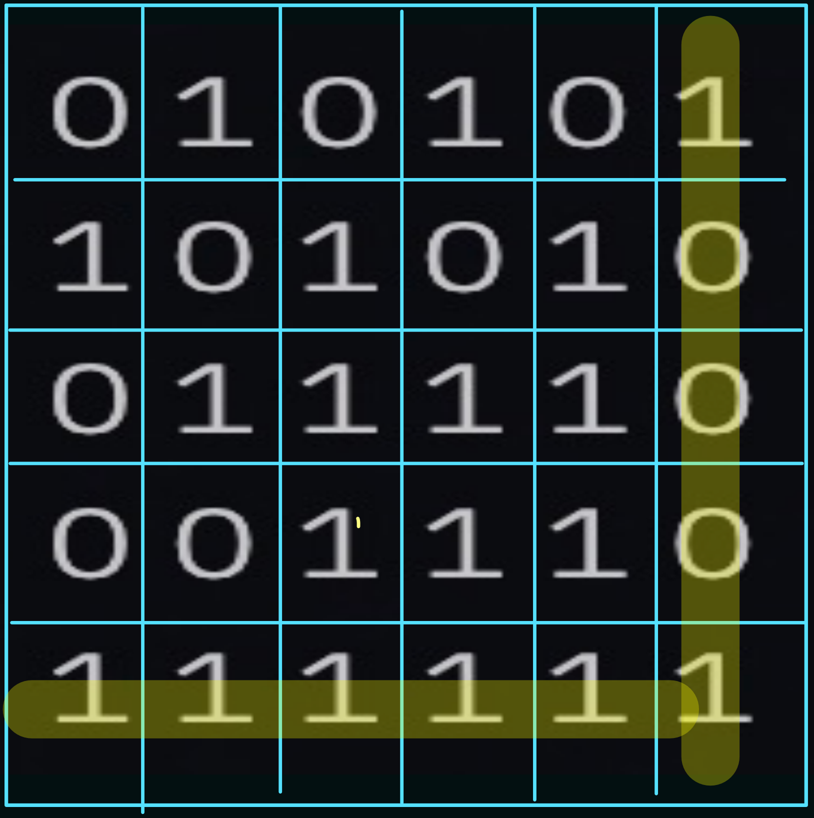
10→22→33→41→60-80

10→22-33-50→60-80

Largest Square Submatrix with all 1s :-

Portal → length of side = l

Leetcode → Area of square = l*l

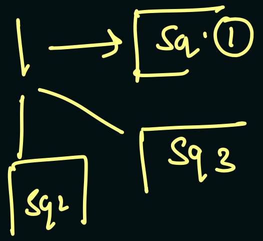


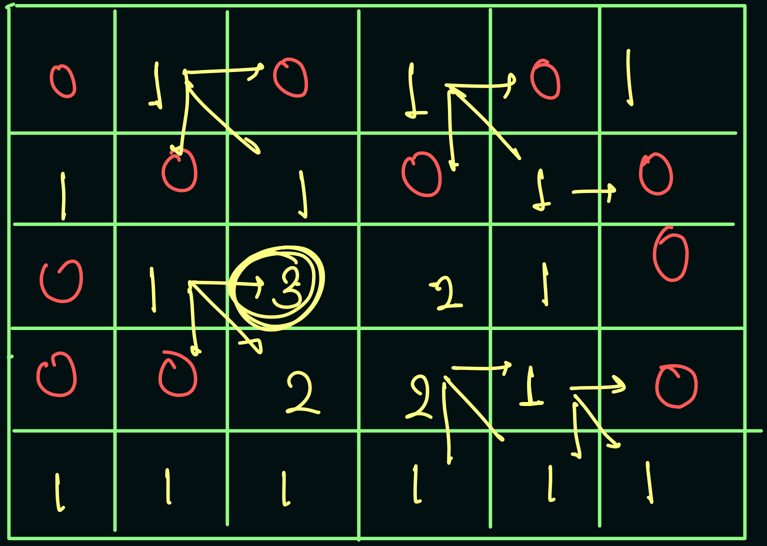
0	1	0	1	0	1
1	0	1	0	1	0
0	1	1	1	1	0
0	0	1	1	1	0
1	1	1	1	1	1

Small problem → $\boxed{1}$

↪ 1 size

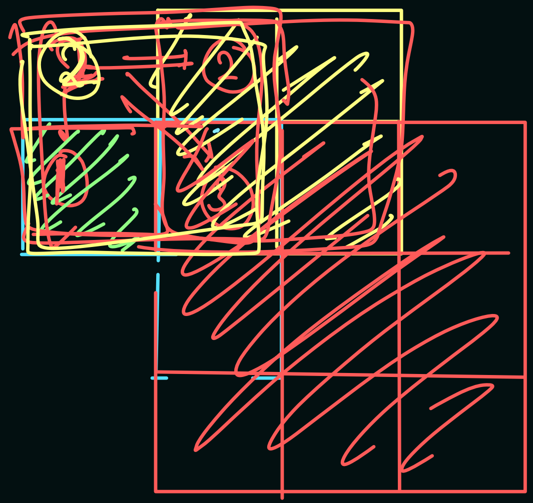
$\boxed{0}$ → 0 size

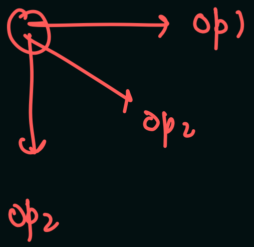




0	1	0	1	0	1
1	0	1	0	1	0
0	1	3	2	1	0
0	0	2	2	1	0
1	1	1	1	1	1

max : ③





It depends on min. size from op1, op2 & op3] +1

NOTE: for maximum size of Rectangle

→ use largst Area Histogram concept of Stack.