LIS : longest inc. subseq.

```
[10,9,2,5,3,7,101,18]
```

10	9	2	5	3	7	101	6
0	1	2	3	4	5	6	7

len = $\underline{4}$

1	1	1	2	2	3	4	3
0	1	2	3	4	5	6	7
10.	9.	2.	2 5.	2 3.	2 5 7.	2 5 7 101.	2 5 6.

T: $O(n^2)$

S: $O(n)$

Lis : nlogn

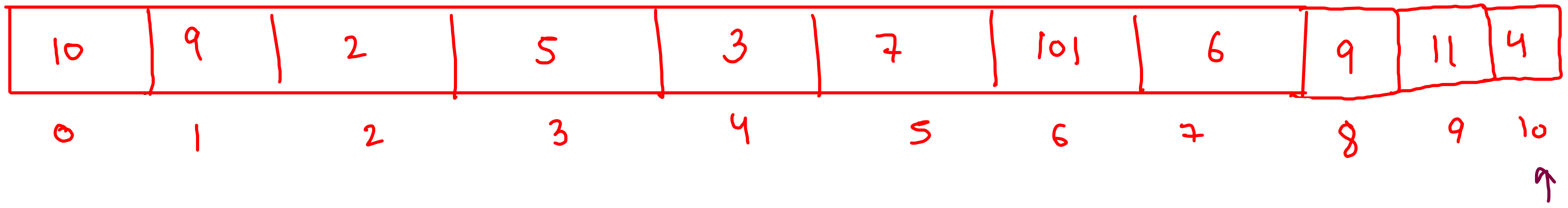
10	9	2	5	3	7	101	6	9	11
0	1	2	3	4	5	6	7	8	9

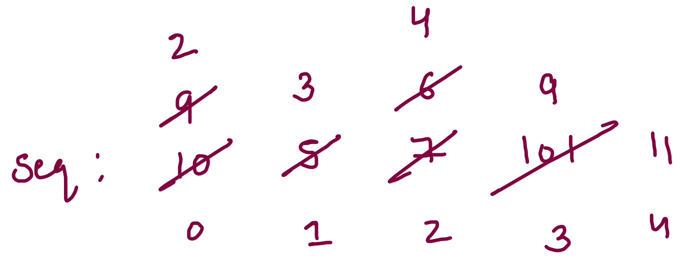
	2				
	~~9~~	3	6	9	
Seq :	~~10~~	~~5~~	~~7~~	~~101~~	11
	0	1	2	3	4

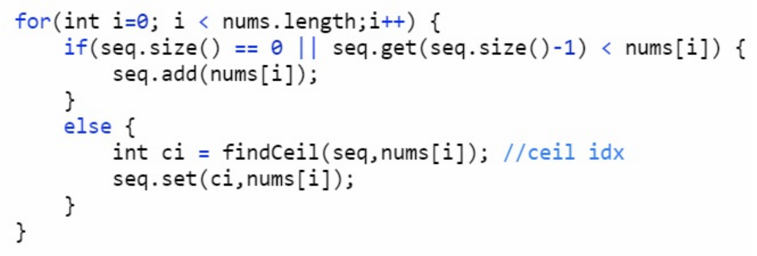
Lis.

(i) seq is not an lis.

(ii) seq. is sorted.







```java
for(int i=0; i < nums.length;i++) {
    if(seq.size() == 0 || seq.get(seq.size()-1) < nums[i]) {
        seq.add(nums[i]);
    }
    else {
        int ci = findCeil(seq,nums[i]); //ceil idx
        seq.set(ci,nums[i]);
    }
}
```

len : 5

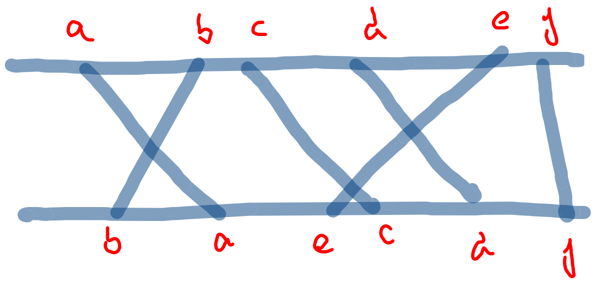
Maximum Non-overlapping Bridges

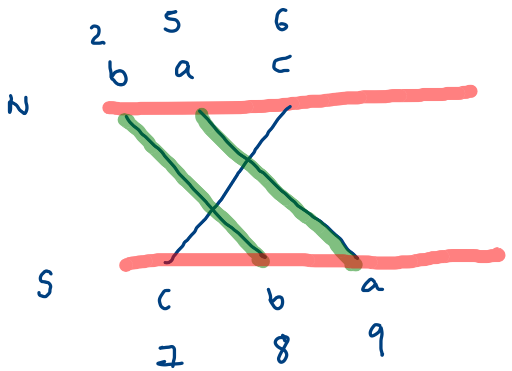
(i) sort on the basis north
(ii) Lis on south

	0	1	2	3	4	5	6	7	8	9
N	10	2	8	17	21	50	41	60	80	1
S	20	7	15	3	40	4	57	80	90	30

$a_n > b_n$	$a_n < b_n$
$a_s > b_s$	$a_s < b_s$

a → 5, 9
b → 2, 8
c → 6, 7





1. You are given a number n, representing the number of bridges on a river.
2. You are given n pair of numbers, representing the north bank and south bank co-ordinates of each bridge.
3. You are required to print the count of maximum number of non-overlapping bridges.

	0	1	2	3	4	5	6	7	8	9
N	10 ✓	2 ✓	8 ✓	17 ✓	21 ✓	50 ✓	41 ✓	60 ✓	80 ✓	1 ✓
S	20	7	15	3	40	4	57	80	90	30

① sort on basis of north

	0	1	2	3	4	5	6	7	8	9
N	1	2	8	10	17	21	41	50	60	80
S	30	7	15	20	3	40	57	4	80	90

② perform LIS based on south.

Russian Doll Envelopes

① . sort on the basis width.

② . perform Lis on the basis height.

```
[[5,4],[6,4],[6,7],[2,3]]
```

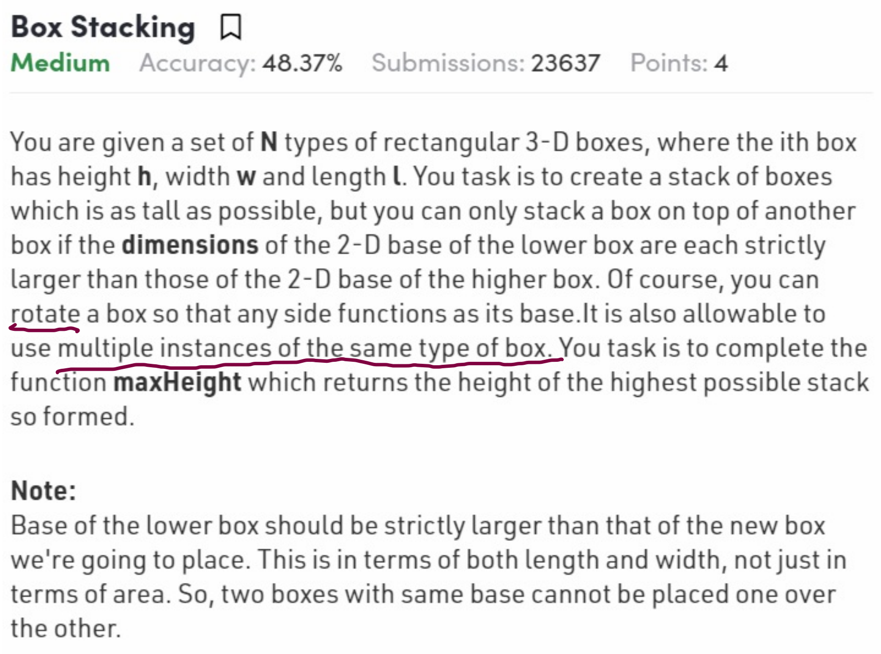
a b c d

	0	1	2	3
w	5	6	6	2
h	4	4	7	3

$$\boxed{\begin{array}{c} x_w < y_w \\ x_h < y_h \end{array}}$$

	0	1	2	3
w	5	6	6	2
h	4	4	7	3

	0	1	2	3
w	2	5	6	6
h	3	4	4	7



Box Stacking

Medium   Accuracy: 48.37%   Submissions: 23637   Points: 4

You are given a set of **N** types of rectangular 3-D boxes, where the ith box has height **h**, width **w** and length **l**. You task is to create a stack of boxes which is as tall as possible, but you can only stack a box on top of another box if the **dimensions** of the 2-D base of the lower box are each strictly larger than those of the 2-D base of the higher box. Of course, you can rotate a box so that any side functions as its base.It is also allowable to use multiple instances of the same type of box. You task is to complete the function **maxHeight** which returns the height of the highest possible stack so formed.

Note:
Base of the lower box should be strictly larger than that of the new box we're going to place. This is in terms of both length and width, not just in terms of area. So, two boxes with same base cannot be placed one over the other.

n = 2 (type)

(1, 2, 4)   (5, 6, 2)

(i) unlimited boxes of each type.

(ii) rotation is allowed.

(1, 2, 4)

$\downarrow$

rotation is allowed

		ba
	1, 2, 4	2
	1, 4, 2	4
✓	2, 1, 4	2
	2, 4, 1	8
✓	4, 1, 2	4
✓	4, 2, 1	8

(5, 6, 2)

$\downarrow$

		ba
	5, 6, 2	30
✓	5, 2, 6	10
✓	6, 2, 5	12
✓	6, 5, 2	30
	2, 5, 6	10
	2, 6, 5	12

d, w, h

$d > w$

why 3n ?

$d > w$

d, w, h we

w, d, h waste

l w h

(1, 2, 4) (5, 6, 2)

	ba		ba
2, 1, 4	2	5, 2, 6	10
4, 1, 2	4	6, 2, 5	12
4, 2, 1	8	6, 5, 2	30

consider all rotations where $l > w$

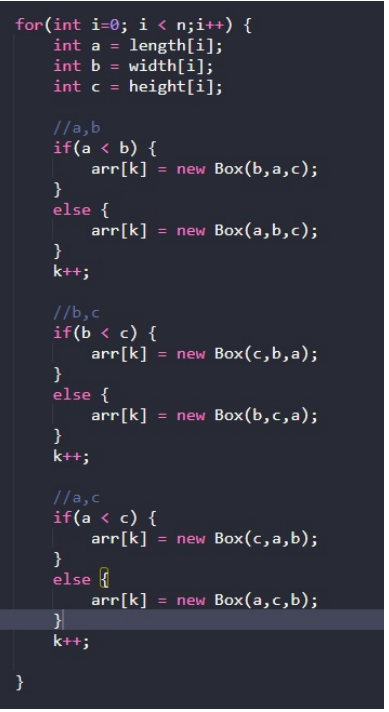
1. sort on the basis of base area

dis -> l, w

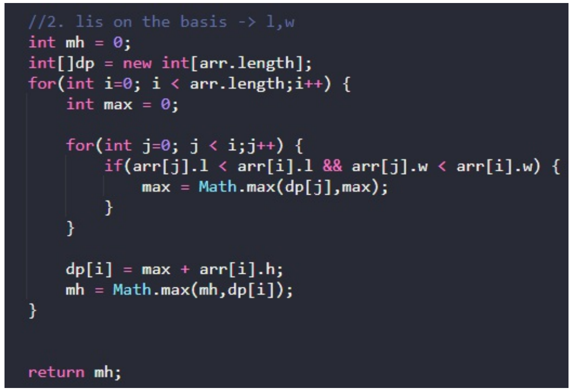
	0	1	2	3	4	5
l	2	4	4	5	6	6
w	1	1	2	2	2	5
h	4	2	1	6	5	2
Area.	2	4	8	10	12	30

	0	1	2	3	4	5
l	2	4	4	5	6	6
w	1	1	2	2	2	5
h	4	2	1	6	5	2
Area.	2	4	8	10	12	30

$$\begin{tikzcd}
\text{ans} & 4 \arrow[rrr, bend right, dash] & 2 & 5 & 10 \arrow[rr, bend right, dash] & 9 & 12
\end{tikzcd}$$



```java
for(int i=0; i < n;i++) {
    int a = length[i];
    int b = width[i];
    int c = height[i];

    //a,b
    if(a < b) {
        arr[k] = new Box(b,a,c);
    }
    else {
        arr[k] = new Box(a,b,c);
    }
    k++;

    //b,c
    if(b < c) {
        arr[k] = new Box(c,b,a);
    }
    else {
        arr[k] = new Box(b,c,a);
    }
    k++;

    //a,c
    if(a < c) {
        arr[k] = new Box(c,a,b);
    }
    else {
        arr[k] = new Box(a,c,b);
    }
    k++;

}
```

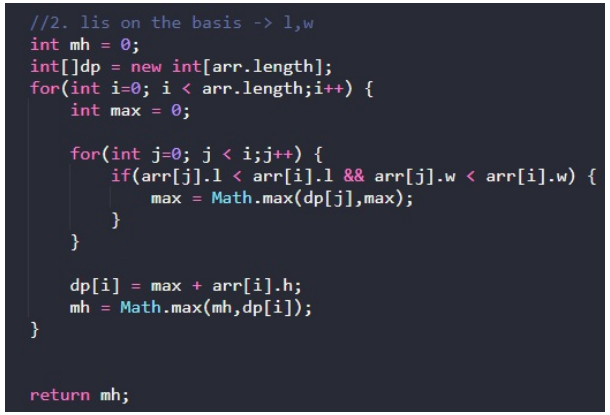


```java
//2. lis on the basis -> l,w
int mh = 0;
int[]dp = new int[arr.length];
for(int i=0; i < arr.length;i++) {
    int max = 0;

    for(int j=0; j < i;j++) {
        if(arr[j].l < arr[i].l && arr[j].w < arr[i].w) {
            max = Math.max(dp[j],max);
        }
    }

    dp[i] = max + arr[i].h;
    mh = Math.max(mh,dp[i]);
}


return mh;
```

1	4	2
3	5	2
4	4	2
a	b	c

	4,1,2	4,2,1	2,1,4	3,2,5	5,2,3	5,3,2	4,4,2	4,2,4	4,2,4
	0	1	2	3	4	5	6	7	8
Ar	4	8	2	6	10	15	16	8	8

	2,1,4	4,1,2	3,2,5	4,2,1	4,2,4	4,2,4	5,2,3	5,3,2	4,4,2
	0	1	2	3	4	5	6	7	8
Ao.	2	4	6	8	8	8	10	15	16
dp[i]	4	2	9	5	8	8	7	11	11



```
//2. lis on the basis -> l,w
int mh = 0;
int[]dp = new int[arr.length];
for(int i=0; i < arr.length;i++) {
    int max = 0;

    for(int j=0; j < i;j++) {
        if(arr[j].l < arr[i].l && arr[j].w < arr[i].w) {
            max = Math.max(dp[j],max);
        }
    }

    dp[i] = max + arr[i].h;
    mh = Math.max(mh,dp[i]);
}


return mh;
```