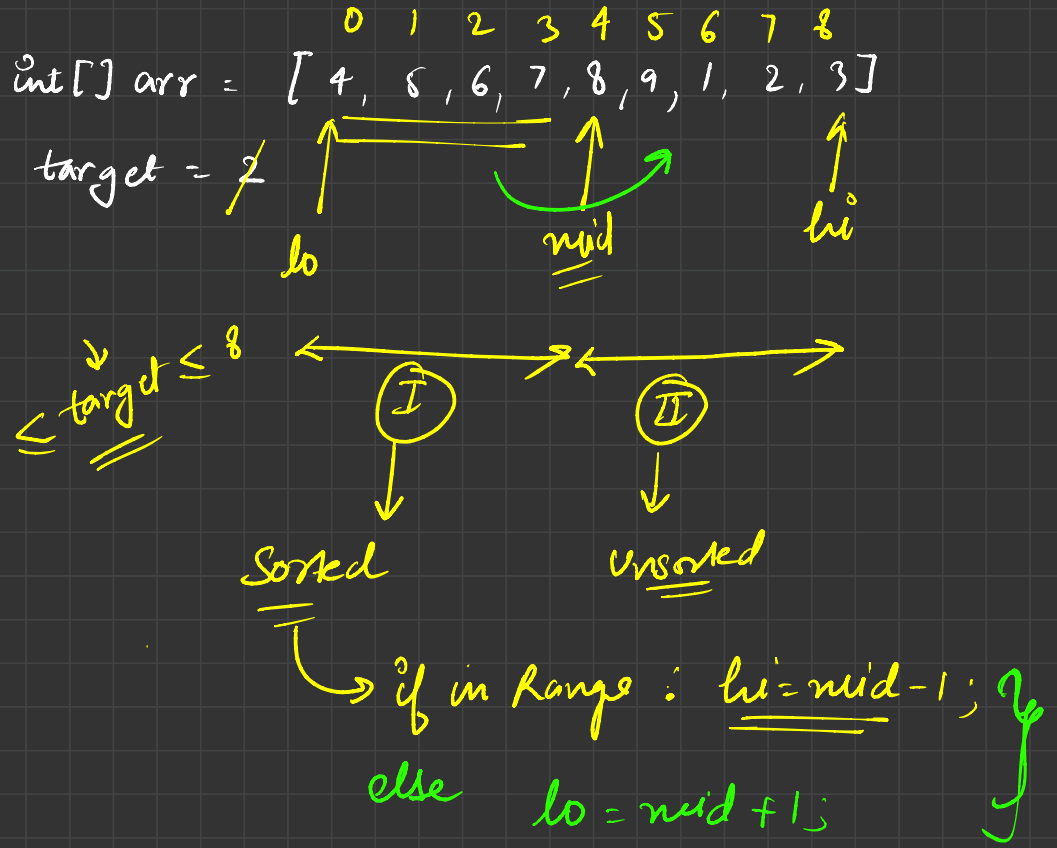
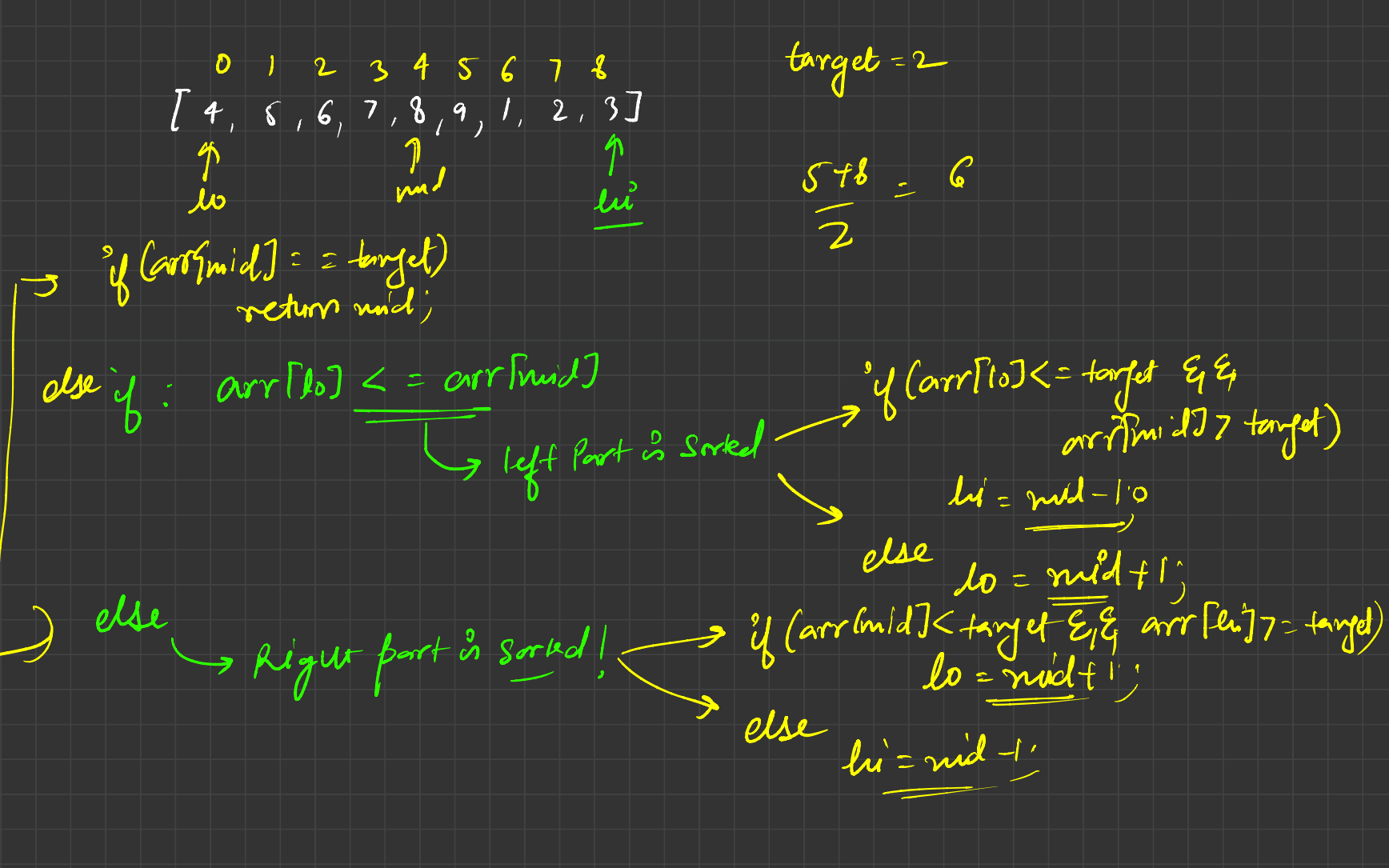
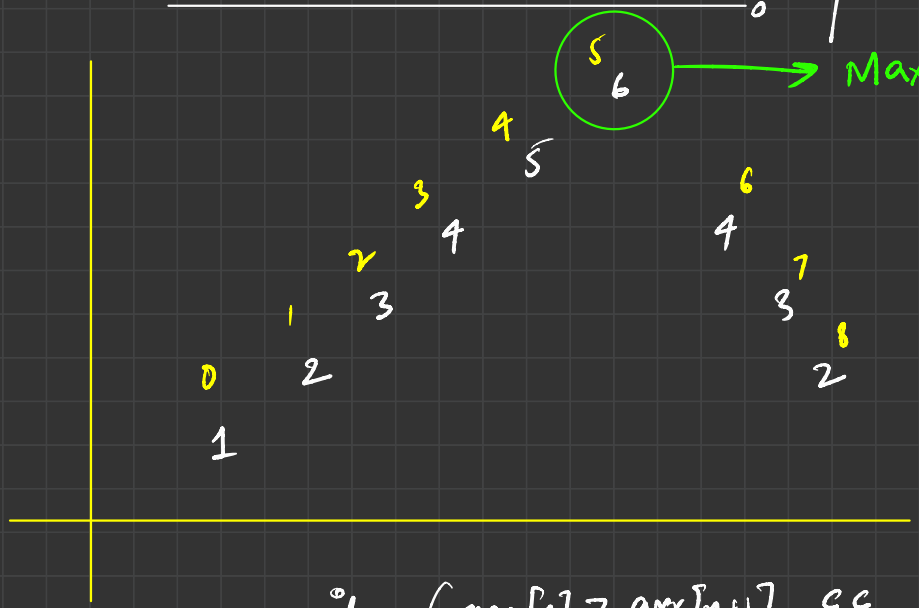
Search In A Rotated Sorted Array




0 1 2 3 4 5 6 7 8
[4, 5, 6, 7, 8, 9, 1, 2, 3]
lo
mid
hi
target = 2
5+8/2 = 6
if (arr[mid] == target)
return mid;
else if: arr[lo] <= arr[mid]
Left Part is Sorted
if (arr[lo] <= target && arr[mid] > target)
hi = mid - 1;
else
lo = mid + 1;
else
Right part is Sorted!
if (arr[mid] < target && arr[hi] >= target)
lo = mid + 1;
else
hi = mid - 1;

Peak Index In A Mountain Array



Maximum of this array

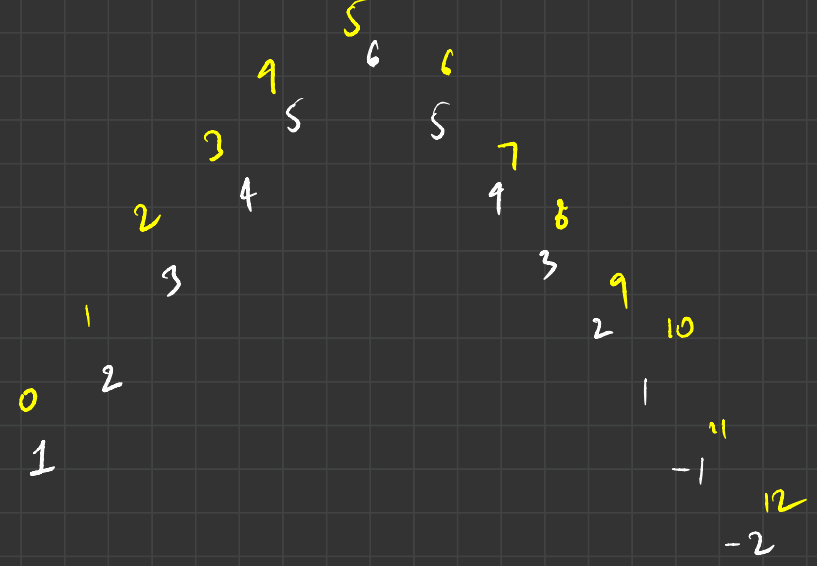
Brute force

{ O(N) }

if (arr[n] > arr[n+1] && arr[n] > arr[n-1]) → peak at n

$\frac{5+6}{2} = 5$

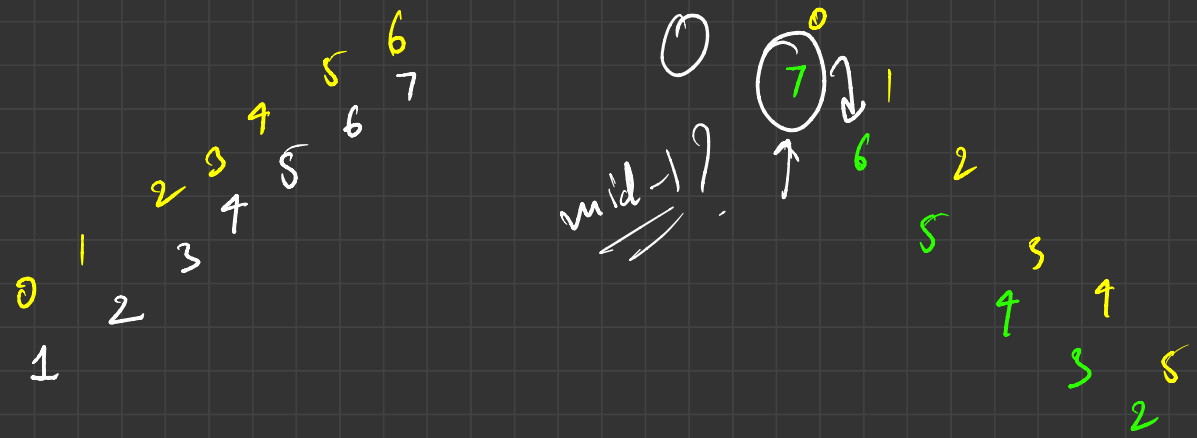
```
if ( Peak(mid) )
    return mid;
if ( arr[mid-1] <= arr[mid] ) // left Part Sorted
    lo = mid+1;
else hi = mid-1;
```



arr[mid-1] ≤ arr[mid]

```
if ( Peak(mid) )
    return mid;                    → (5)

if ( arr[mid-1] <= arr[mid] )      // left Part Sorted
    lo = mid+1;
else  hi = mid-1;
```

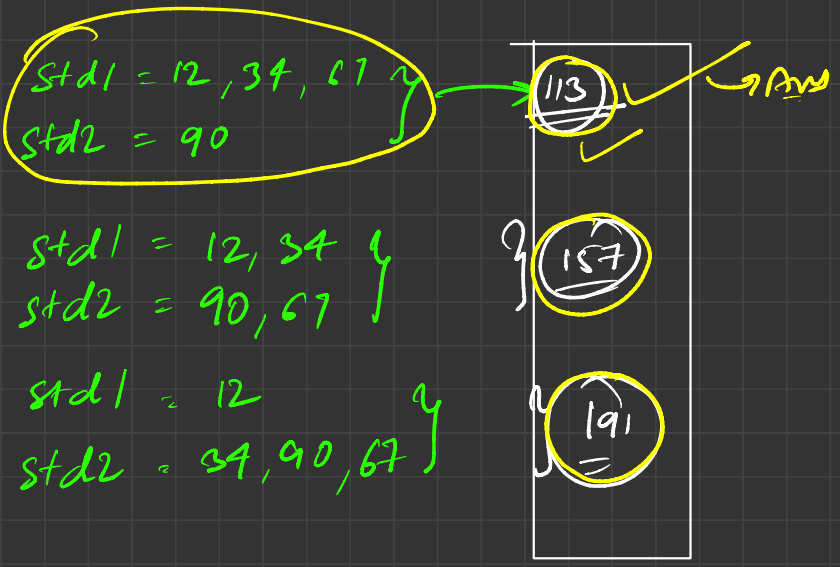

mid-1


1
1
3
0
10
2
10
4
1
3
2
?
w
10
mid

Allocate Minimum Number of Pages

book[] = [12, 34, 67, 90]

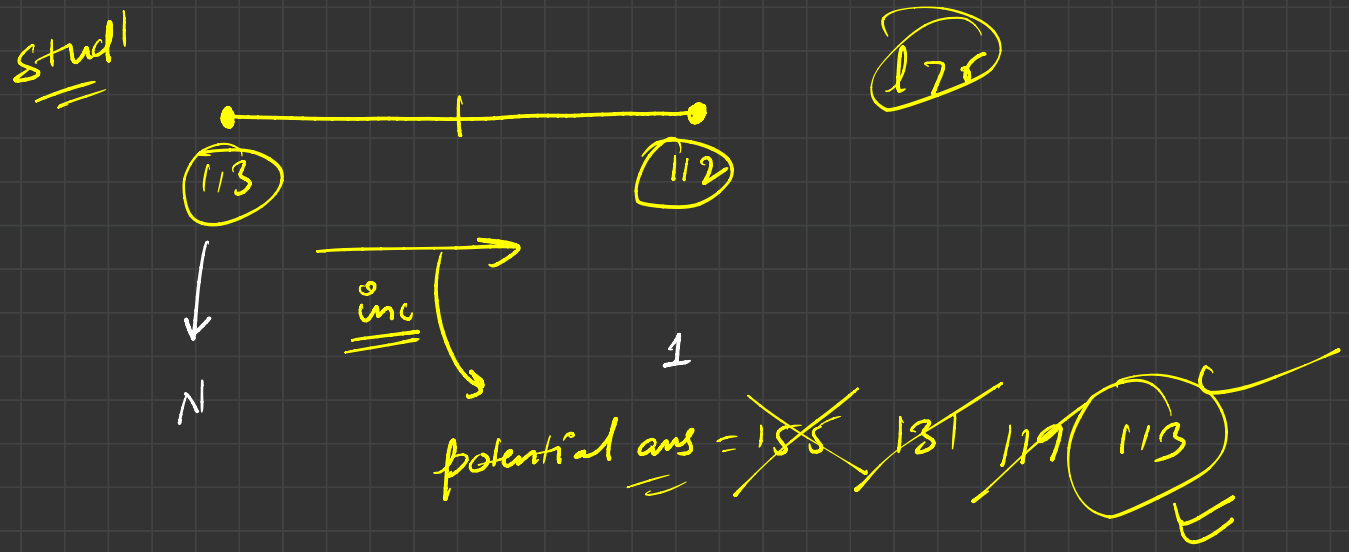
students = 2



① Each student should have minimum 1 book

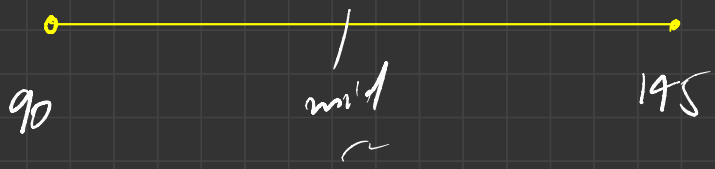
② Book allotment to a student should be contigeous Array.

book[] = [12 34 67 90] {student = 2}



books[] = {12, 34, 61, 90}

Stud = 2



pAns = ~~146~~ 117 (113)

N + N + N ---- log N times

N × log N

TC: O(N log N)

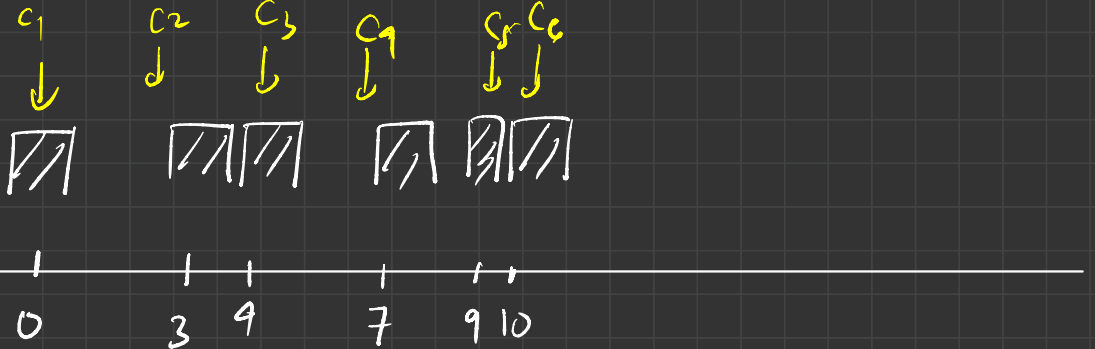
Aggressive Cows

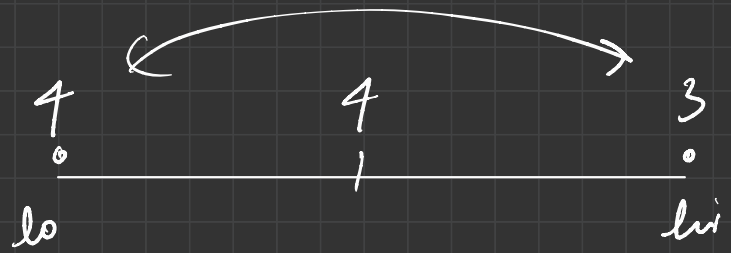
pos[] : {0, 3, 4, 7, 10, 9} cows = 4

2 cows

dist = 1 → min

dist = 10 → max!





low = 4

potentialAns = ~~2~~ 3

pos[] : {0, 3, 4, 7, 9, 10}

c_1 c_2 c_3

low = 3