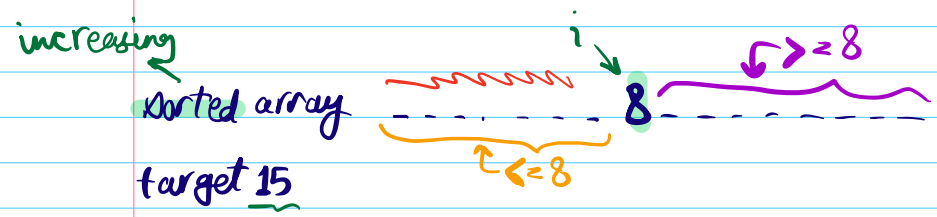
topic: find in sorted array
1- distinct ✓
2- repeating
3- one single element
4- global min
5- local min



P1 Given a sorted array of distinct elements, find index of a given target.

target 17 a: { 1, 3, 5, 7, 9, 10, 11, 13, 15, 17, 19, 30, 35, 40 }
indices: 0 1 2 3 4 5 6 7 8 9 10 11 12 13
l = 0, r = 13, m = 6

search space

Quiz
subjective

l	r	l+r/2 = mid	cmp
0	13	(0+13)/2 = 6	a[6] < 17
(mid+1) 7	13	(7+13)/2 = 10	a[10] > 17
7	(mid-1) 9	(7+9)/2 = 8	a[8] < 17
(mid+1) 9	9	(9+9)/2 = 9	a[9] == 17

index

```
int binSearch(int a[], target){
    n = a.len
    l = 0 ; r = n-1
    while(l <= r){
        mid = (l+r)/2 // floor
        if(a[mid] == target) ret mid
        if(a[mid] < target) l = mid+1
        else r = mid-1                    // a[mid] > target
    }
    ret -1 // not found
}
```

TC: O(log(n))
SC: O(1)

(z==2)!

runtime err

note: Short circute boolean terms

a || b || c

(x==2) ← x=2

if(a || b) → true

if(a && b) { → false (a is false)

if(arr.size!=0 && a[0]==3) → F

P2 Given a sorted array of integers, find first index of a given target.

a: { 2, 2, 5, 5, 5, 5, 8, 10, 10, 13, 13, 13 }
0 1 2 3 4 5 6 7 8 9 10 11

target ① 5 → 2
② 8 → 6

[5,10] → [2,8] range search

idea1

{ 1, 2, 2, 2, ..., 2, ..., 2, 2, ..., 2 } # 10 x 10^6

n/2 ~ O(n)

idea2

a: { 2, 2, 5, 5, 5, 5, 8, 10, 10, 13, 13, 13 }
0 1 2 3 4 5 6 7 8 9 10 11

l, r, m

target 5 (first index)

l	r	mid	
0	11	(0+11)/2 = 5	a[5] == 5
0	5-1=4	(0+4)/2 = 2	a[2] == 5

a[mid-1] == a[mid]
4 5

a[mid-1] == a[mid]
1 2
↓ Stop
found first index

pseudo code →

```
size = n
l=0  r=n-1
while(l<=r){
  mid = (l+r)/2
  if( a[mid]==target &&
      (mid==0 || a[mid-1]!=a[mid]) ){
    ret mid
  }
  if (a[mid]<targe) l = mid+1
  else{// a[mid]>target
    r=mid-1
  }
}
ret -1
```

TC: O(lag(n))
SC: O(1)
Quiz →

first ↑
[a, b] → last

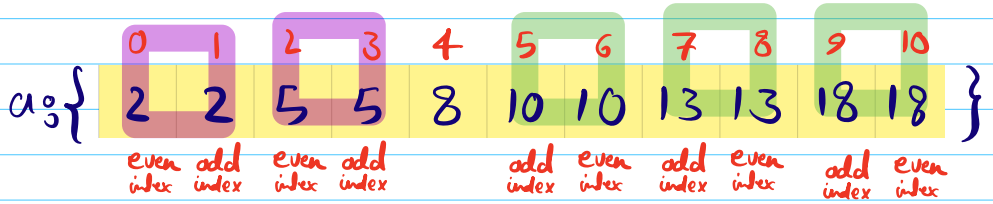
assignment:
① first | index
② last

P3 Given a sorted integer array where every element apears twice except for one element, find that unique element.

a: { 2, 2, 5, 5, 8, 10, 10, 13, 13, 18, 18 }
(0 1 2 3 4 5 6 7 8 9 10)

idea 1 XOR TC: O(N) SC: O(1)

idea 2

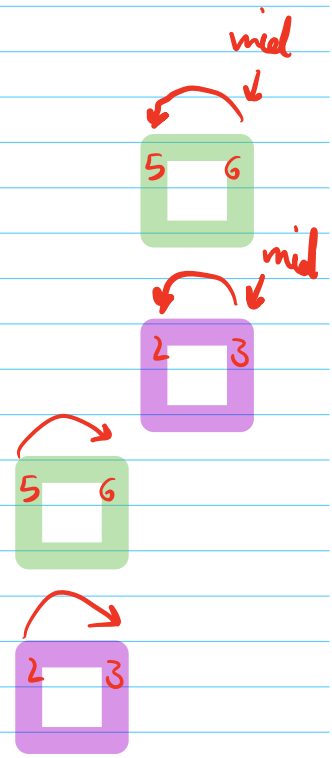



```
l = 0  r = n-1
while( l <= r){
    mid = (l+r)/2
    if( (mid==0 || (a[mid] != a[mid-1])) &&
        (mid==n-1 || (a[mid] != a[mid+1])) ){
        ret a[mid]
    }
    if( mid!=0 && a[mid] == a[mid-1] ){
        if(mid/2==0) r = mid-1
        else l = mid+1
    }else{ a[mid] == a[mid+1]
        if(mid/2==0) l = mid+1
        else r = mid-1
    }
}
ret -1
```

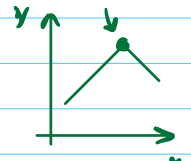
TC: O(log n)
SC: O(1)

a[mid-1]
a[mid]
a[mid+1]

? mid != n-1 assignment



P4 Given an increasing-decreasing array with distinct elements, find max elements.



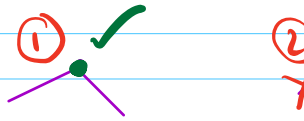
only one max exist

ans

$a = \{1, 3, 5, 2\}$

ans

$\{1, 3, 5, 7, 15, 12, 6\}$



O(n)

a[mid-1]
a[mid]
a[mid+1]

Quiz
TC: O(log n)
SC: O(1)

```
int findMaxInMountain(int a[]){
    n = a.Len
    l = 0        r = n-1
    while(l<=r){
        mid = (l+r)/2
①       if(( mid==0 || (a[mid] > a[mid-1]) ) &&
             ( mid==n-1 || (a[mid] > a[mid+1]) ) ){
            ret mid
        }
②       if(mid!=0 && a[mid] > a[mid-1]){
            l = mid+1
        }
③       else{
            r = mid-1
        }
    }
    ret -1
}
```

P5 Given a random array with distinct elements

find any one local minima in the array.

not all of mins, any one of them is acceptable

minmum

a: { 1, 3, 8, 5, 2, 6, 4 }

a[i-1] > a[i] < a[i+1]

O(n) → global min/max

only compare to imidiate neighbors if exist

Quiz a: { 1, , 8, 5, 2, 6, 4, 0, 9, 10, -1, -2, 100 }

you still can use bin search for a local minima, but not the global minma.

→ hint for assigment