Q. Find a peak element

Given an array of unique integers

Ex:

	0	1	2	3	4	5	6
a[] =	3	2	1	5	7	4	8
	✓	✗	✗	✗	✓	✗	✓

Return any one.

Brute Force :

Distinct elements
max ele will always be a peak ele.

TC $O(n)$ SC $O(1)$

What is better than $O(n)$? $\rightarrow O(\log n)$
↳ Binary Search
[But oops! The array is unsorted, so you can apply BS, right ?]

say we land on mid

0	1	2	3	4	5	6
3	2	1	5	7	4	8

start	end	mid	A[mid]
0	6	3	5 [not an ans]

How to decide where to move ?

If u know where your answer is, then BS can be applied.

↳ if in left, go to left

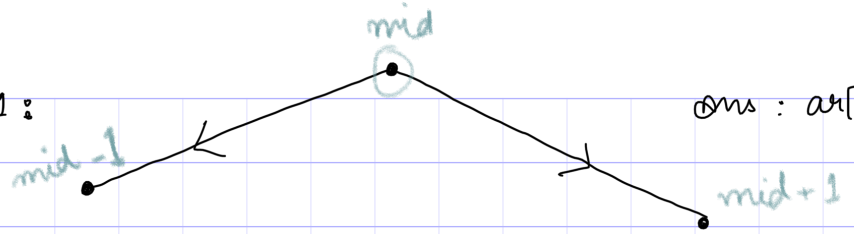
↳ if in right, go to right

But how to know where answer is ?

ar[mid - 1] ar[mid] ar[mid + 1]

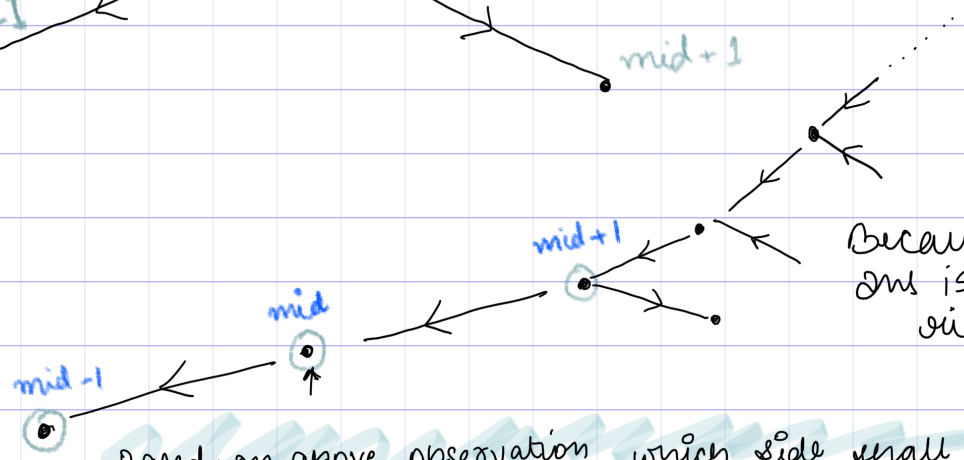
Possible scenarios

case 1 :



ans : ar[mid]

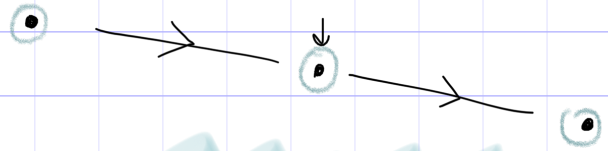
case 2 :



Because one ans is on right

Based on above observation, which side shall I go? — Right

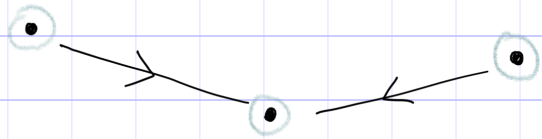
case 3 :



what about in this case?

Left side

case 4 :



Here?

— Any side

Note : There's always a peak element on greater side!

```
int peak_ele ( A , n) {
    s = 0 , e = n-1
    while ( s <= e) {
        mid = (s+e) /2
        if ( A[mid] < A[mid-1] && A[mid] > A[mid+1])
            return A[mid]

        if (A[mid] < A[mid+1]) {
            s = mid + 1
        }
        else e = mid - 1
    }
```

```
if(n==1) return A[0]
if ( A[0] > A[1] ) return A[0]
if (A[1] > A[0]) return A[1]
```

} Edge cases

TC = O(log n)

SC → O(1)

return -1

}

Do you need sorted array to apply BS?

—NO

Conditions

1. Search space
2. Some condition to go to left or right

[Google]

Painter's Partition

Given N tasks & K workers & time taken to complete each task.
Find min time in which we can complete all tasks.

Note:

A single worker can only do a continuous set of tasks.

Ex

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
3	5	1	7	8	2	5	3	10	1	4	7	5	4	6

K = 3 → $w1 = 9$ $\quad w_2 = 25$ $\quad w_3 = 37$

Total time taken : 37 → we have to minimize this val

$w_1 = 24$ $\quad w_2 = 21$ $\quad w_3 = 26$

$w_1 = 24$ $\quad w_2 = 25$ $\quad w_3 = 22$

$\boxed{\text{Ans} = 25}$

→ say ans: 30

able to assign worker with not more than 30 pages?

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
3	5	1	7	8	2	6	3	10	1	4	7	5	4	6

w_1 26 $\quad w_2$ 24 $\quad$ 22 w_3

You can do task in 30, can u do in 31, 32, ... ?

30	31	32	...
T	T	T	...

update
go to left

Say ans = $\boxed{20}$

$\boxed{\text{workers} = 3}$

possible to assign a worker less than 20 pages?

3	5	1	7	8	2	6	3	10	1	4	7	5	4	6

w_1 $\quad w_2$ $\quad w_3$ $\quad w_4$ $\quad w_5$

...	18	19	20
...	F	F	F

go to right

Can we apply BS now? — YES

On what are we applying? TIME

Search space

low	high
[max(time)]	[sum of all elements]

DRY RUN

3 5 1 7 8 2 6 3 | 10 1 4 7 5 4 6

W_1 W_2

W_1 W_2 W_3 W_4

W_1 W_2 W_3

start	end	mid	
10	72	41	possible | store | go to left
10	40	25	given_w < required | go to right
26	40	33	store | go left
26	32	29	

Pseudocode

```
int painter( A , n , K) {
    s = max(A) ,  e = sum(A)
    while(s<=e){
        m = (s+e)/2
        if ( possible(A , n, K, m)){
            ans = m
            e = m-1
        }
        else s = m+1
    }
    return -1
}
```

```
bool possible ( A , n, K, mid) {
    assigned = 1
    sum = A[0]
    for ( int i=1 ; i<n ; i++) {
        sum += A[i]
        if (sum > K) {
            assigned ++
            sum = A[i]
        }
    }
    if ( assigned > K) return false
    return true
}
```

Q. Aggressive Cows

Given N cows & M stalls. All M stalls are on x-axis, at different locations. Place all cows such a way that min dist. b/w any two cows is maximized. $M > N$

In one stall we place one cow

Ex

Stall 5	1	3	4	8	9

cows 3

$C_1 \xleftrightarrow{2} C_2 \xleftrightarrow{1} C_3$

min distance → Can you place cows s.t this min diff. ↑ ?

$C_1 \xleftrightarrow{3} C_2 \xleftrightarrow{5} C_3$

3 → Final ans

Ex:

Stalls	2	6	11	14	19	25	30	39	43
cows	C_1				C_2			C_3	

$C_1 \xleftrightarrow{4} C_2 \xleftrightarrow{5} C_3 \xleftrightarrow{4} C_4$

$C_1 \xleftrightarrow{9} C_2 \xleftrightarrow{14} C_3 \xleftrightarrow{18} C_4$

$C_1 \xleftrightarrow{12} C_2 \xleftrightarrow{16} C_3 \xleftrightarrow{13} C_4$

Brute Force

Choose 4 stalls from 9 stalls

$\Rightarrow$ $^{\text{stalls}}C_{\text{cows}} * N$

Observation

Possible to achieve a distance of 7? – YES

→ Possible to place cows at a min distance of 7?

SHOW

3	4	5	6	7
T	T	T	T	T

→ Better choice

update & go

Possible to place at a min dist of 15 ? – NO

15	16	17	18	. . . .
F	F	F	F	

Combine above 2

DRY RUN

Start	end	mid	
1	41	21	check by placing all cows at min dis 21 if placed < given cows // go to left
1	20	10	check again // store & go right
11	20	15	// go left
11	14	12	// store & go right
13	14	13	// go left
13	12	— stop —	

2	6	11	14	19	25	30	39	43

```
int aggressivecows (stalls, cows, n){

    s = 1      e = max(stalls) - min(stalls)      ans = 1
    while( s <= e){
        m = (s+e)/2          //overflow
        if( possible(stalls, cows, n, m)){
            ans = m
            s = m+1
        }
        else  e = m-1
    }
    return ans
}
```

$e \to$ not overflow

$e - s \to$ can't overflow

$\frac{e-s}{2} \to$ " "

$\underbrace{s + \left(\frac{e-s}{2}\right)}_{m} <= e$

```
bool possible (stalls, cows, n, m){
    cows_placed = 1 , last_placed = stalls[0]
    for(int i=1 ; i<n ; i++){
        if( stalls[i] - stalls[prev] >= m){
            cows_placed ++
            last_placed = stalls[i]
            if (c>=cows) return true
        }
    }
    if(c < cows) return false
    return true
}
```

TC: BS iterations × check func

total size: (end - start)/2 × O(n)

$\log_2((end - start)/2)$

Total TC:

$O(\log_2((end-start)/2) * n)$

Q. Maximum subarray sum less than S ele ≥ 0

A = [1 20 3 7] S = 25

K=1	K=2	K=3	K=4
[1] = 1	[1, 20] = 21	[1 20 3] = 24	[1 20 3 7] = 31
[20] = 20	[20, 3] = 23	[20 3 7] = 30	
[3] = 3	[3, 7] = 10		
[7] = 7			

Find max size of the subarrays such that all the subarrays of that size have sum < S [Maximize the size of such subarrays]

Brute Force

For all sizes of subarray, check if all subarrays of that size has sum less than S.

$O(n^3)$

```
for(K = 1 to n){
    flag = true
    for(i = 0 to n-1){
        j = i+K-1
        for(p = i to j){
            sum += A[p]
        }
        if(sum > S){
            flag = false
            break
        }
    }
}
```

Prefix array

0	1	2	3
1	20	3	7

0	1	2	3	4
0	1	21	24	31

```
ans = 0
for(K = 1 to n){
    flag = true
    for(i = K to n){
        sum = P[i] - P[i-K]
        if(sum > S){
            flag = false
            break
        }
    }
    if(flag){
        ans = K
    }
}
return ans
```

$O(n^2)$

* For some window of size K, can we check if subarrays of size K obeys the condition.

Observation

we are linearly iterating over value of K

Search space
↳ where we search for the answer
0 to n

Say I apply BS on 0 —— mid —— n

Eg size = 6
What to check as per question?
→ if all subarrays of size 6 have sum < S

NO: 6 7 8 9 ...
F F F F
go to left

YES: 4 5 6
... T T T
maximize val of K
store K
go to right

```
int subarraySum( A , S ){
    s = 0    e = A.size()    ans = 0
    while (s <= e) {
        m = s + (e-s)/2
        if( satisfy( A , S , m)) {
            ans = m
            s = m+1
        }
        else e = m-1
    }
    return ans
}
```

// prefix array ⇒ P , starts with P[0]=0

```
bool satisfy( A , S , K){
    for( i=K ; i<n ; i++ ){
        sum = P[i] - P[i-K]
        if (sum > S){
            return false
        }
    }
    return true
}
```