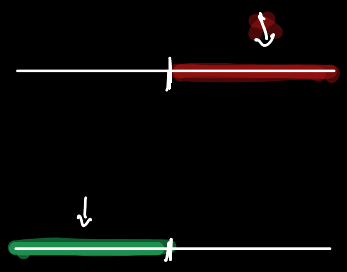
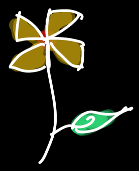
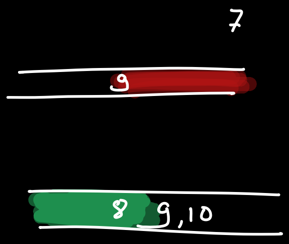






Q Every element appears twice except one element.

Amazon Find the single no.

(All pairs of duplicates will always be adjecent to each other)

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 3 | 3 | 1 | 1 | 8 | 8 | 10 | 10 | 6 | 2 | 2 | 4 | 4 |

6

Before the single no → 1st occurrance of all pairs is on even index

After the single no → 1st occurrance of all pairs is on odd index

TC: O(logN)

mid is 1st occ of A[mid]

mid is even

Google
Facebook
Amazon
LinkedIn
Goldman Sachs
Flipkart
* Snapdeal
MS
Adobe
Target
⋮

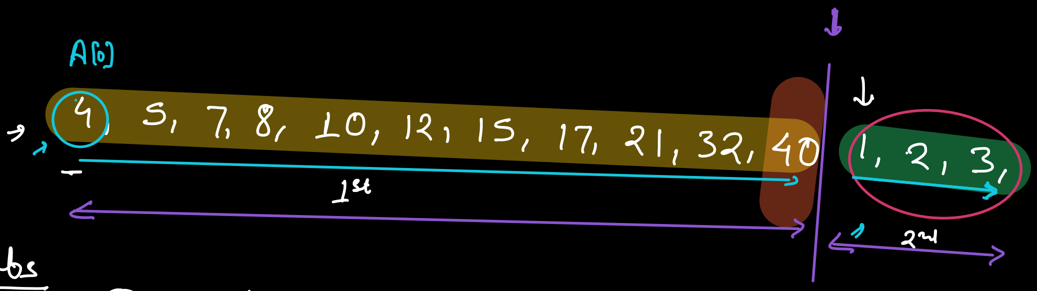
Q Search in a sorted but rotated array.
of distinct elements
(No repetition)

A : $\overset{0}{4},\ \overset{1}{5},\ \overset{2}{8},\ \overset{3}{10},\ \overset{4}{1},\ \overset{5}{2},\ \overset{6}{3}$

T = 2 ⟶ 5

T = 8 ⟶ 2

T = 20 ⟶ -1

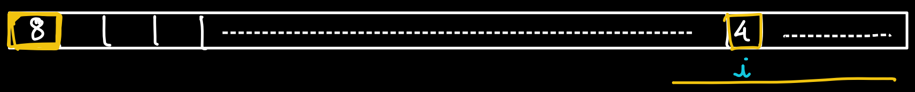


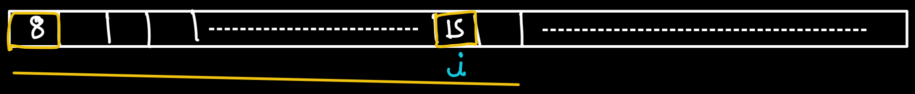
Obs

① Sorted and rotated array ⇒ Concatenation of 2 sorted arrays.

② All elements of 1st part will be greater than all elements of 2nd part.

⇒ A[0] will be greater than all elements of second half.

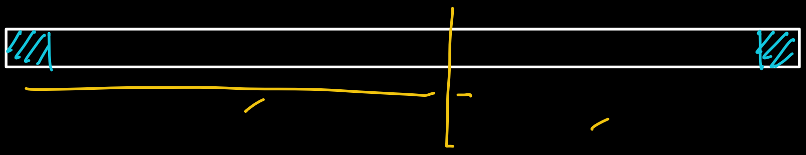




if $A[i] < A[0] \longrightarrow$ 2nd Part

$A[i] > A[0] \longrightarrow$ 1st Part.

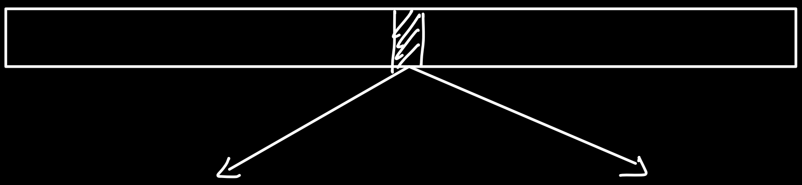
We want to find the pos at which the 1st part ends & 2nd begins. ( boundries of the 2 parts )



## Find the pivot point

Find : Starting index of second half.

⇒ index of smallest value of 2nd half.



if $A[mid] >= A[0]$

mid is on 1st Part

//Move to right

$l = mid + 1;$

if $A[mid] < A[0]$

mid is on 2nd Part

//Can mid be the ans ?

Yes → Store the ans

Move to left

$pivot = mid;$

$r = mid - 1$

r ↓ (over index 2), l ↓ (over index 3)

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 10, | 20, | 30, | 1, | 2, | 3, | 4, | 5, | 6, | 7, | 8, | 9 |

| l | r | mid | A[mid] | Pivot | move to |
|---|---|---|---|---|---|
| 0 | 11 | 5 | 3 | ~~3~~ 5 | left |
| 0 | 4 | 2 | 30 | ~~3~~ 5 | right |
| 3 | 4 | 3 | 1 | ~~1~~ 3 | left |
| 3 | 2 | → Break | | | |

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 5, | 8, | 10, | 14, | 18, | 22, | -15, | -9, | -6, | 0, | 2 |

| l | r | mid | A[mid] | Pivot (idx) | move to |
|---|---|---|---|---|---|
| 0 | 10 | 5 | 22 | - | right |
| 6 | 10 | 8 | -6 | 8 | left |
| 6 | 7 | 6 | -15 | 6 | left |
| 6 | 5 | → Break. | | | |

|  |  | ↓ | ↓ |
|---|---|---|---|
| 0 | 1 | 2 | 3 |
| 1, | 2, | 3, | 4 |

-1/-∞/0

| l | r | mid | A[mid] | pivot (idx) | move to |
|---|---|---|---|---|---|
| 0 | 3 | 1 | 2 | - | right |
| 2 | 3 | 2 | 3 | - | right |
| 3 | 3 | 3 | 4 | - | right |
| 4 | 3 | → | → | → | Break.. |

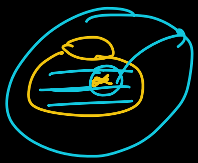
if no update in pivots value ⟶ No rotations.

Step I   Find pivot index (using BS) ⟹ O(log N)
↳ p

Step II
if (target < A[0])
↳ Search on right part
↳ BS(target, P, N-1)

if (target >= A[0])
↳ Search on left part
↳ BS(target, 0, P-1)

goldfish



TC : O(log N)

SC : O(1)

HW Implement the solution using a single iteration of B S. [TC: O(log N)]

Q Given a no N. Find the sqrt (N)

↳ floor ( sqrt(N))
(int part)

sqrt(25) = 5

sqrt (20) = 4

Sqrt (10) = 3

$N > 0$

linear search

```
int sqrt ( N) {
  ⇒ i = 1;
    While ( i × i <= N) {
      ans = i;
      i++;
    }
    ret ans;
}
```

TC : $O(\sqrt{N})$

N = 36

| i | i × i | ans |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 4 | 2 |
| 3 | 9 | 3 |
| 4 | 16 | 4 |
| 5 | 25 | 5 |
| ⇒ 6 | 36 | 6 |
| 7 | 49 → Break | |

{Find max val of i for which $i \times i <= N$

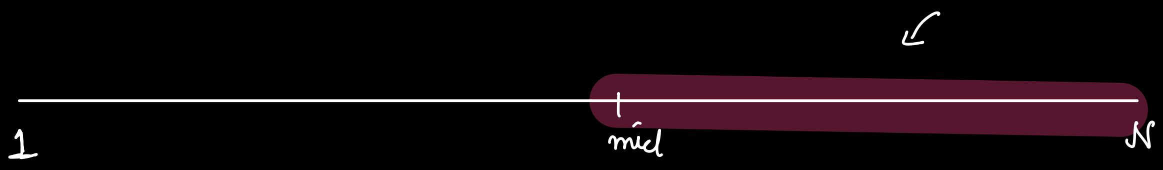
Min ans → 1

Max ans → N

[1, N]

Searching for ans. ←

in [1, N] ←



mid x mid > N ⟶ Move to left

mid x mid < N ⟶ Store the ans
Move to right.

Q Given an array of +ve no. Find the max length K. Such that → there exists no subarray of length K with sum >= B.

Google

→ All subarrays of length K have $\boxed{\text{sum} < B}$

A : $\overset{0}{3}, \overset{1}{2}, \overset{2}{5}, \overset{3}{4}, \overset{4}{6}, \overset{5}{3}, \overset{6}{7}, \overset{7}{2}$

B = 20

20

⇒ 3

K = 5 ✗

K = 4 ✗

K = 3 ✓

## Brute force

$K_{min} \longrightarrow 0$

$K_{max} \longrightarrow N$

Iterate from $K_i = N \longrightarrow 0$

→ Check if Ki satisfies the condition
{ • All subarrays of length K have sum < B }

Find sum of all subarray of length K

PS Approach

TC : O(N) + O(N) ⇒ O(N)

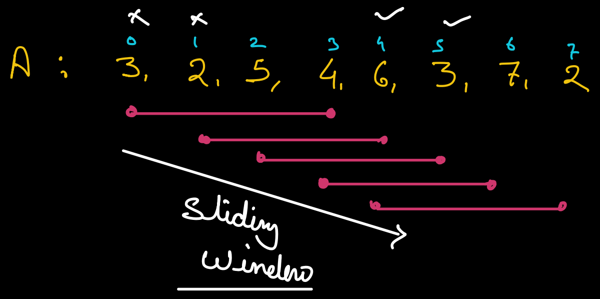
↑ Build PS ↓

SC : O(N)

Sliding window

TC : O(N)

SC : O(1)



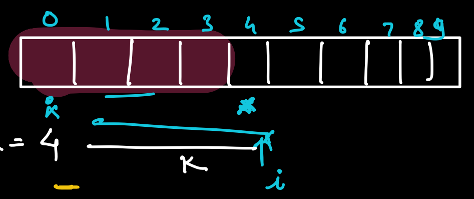
Write a fn which returns true if a given length K satisfies the condition (all subarrays of size K have sum < B)

```
boolean check(A[], K, B){

     // Get sum of first window of size K → [0, K-1] ⇒ sum
     if(sum >= B)
               ret false;

     for(i=K; i<N; i++){

          sum += A[i];
          sum -= A[i-K];
          if (sum >= B)
                ret false;
     }
   ret true;
}
```



| | | |
|---|---|---|
| 1st window | 0 - 3 | $S_1$ |
| 2nd win | 1 - 4 | $S_1 - A[0] + A[4]$ |
| 3rd win | 2 - 5 | $S_2 - A[1] + A[5]$ |
| 4th win | 3 - 6 | $S_3 - A[2] + A[6]$ |
| 5th win | 4 - 7 | $S_4 - A[3] + A[7]$ |
| | | ⋮ |
| | | N- |

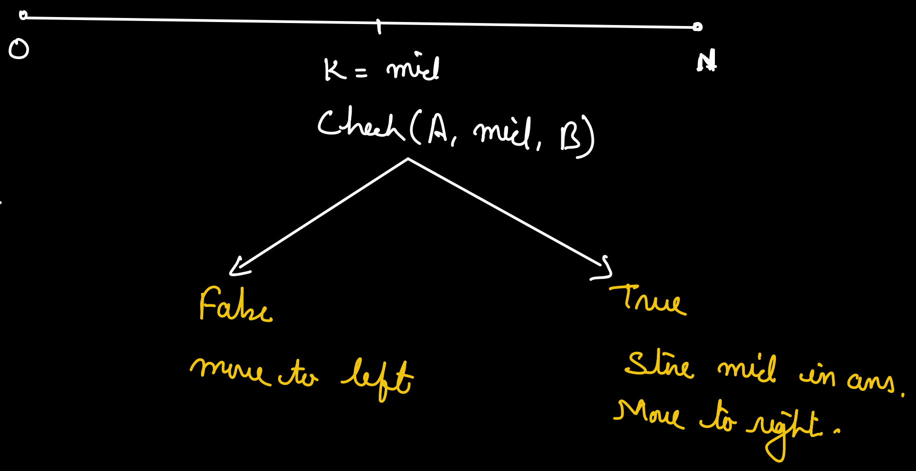
linear iteration

```
for (K= N;  K >=0;  K--){
        if( check(A, K, B))
                 ret K;
}
```

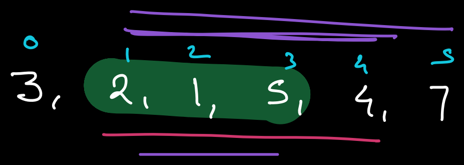
TC : $O(N^2)$

target : $K_{max}$

Search space : [0, N]



mid



K = 3

B = 6

```
int   BS on Ans ( A[], B ) {
    // define range of search space
      l = 0;    r = N;    // All possible values of K
                                                  (length of subarray)
      while ( l<=r ) {
            mid = (l+r)/2;
            if ( check ( A, mid, B ) ) {
                    ans = mid;
                     l  = mid +1;
            } else {
                     r = mid -1;
            }
      }
     ret ans;
}
```

TC : $O(N \log N)$

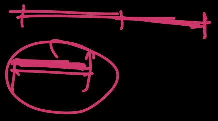
SC : $O(1)$

0 1 2 3 4 5

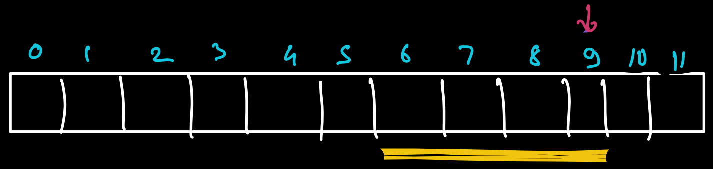
3, 2, 1, 5, 4, 7 11 <=12

B=12

7, 5, 4, 3, 2, 1



7, 12 16, 19, 21, 22



5x