Q → Given a rotated sorted array of distinct elements find index of element K.

Microsoft

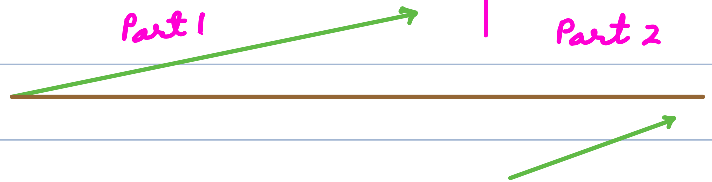
```
      0   1   2   3   4   5   6
A = [ 8  12  15  19   2   5   6 ]        K = 12
                                          Ans = 1
```

Brute force →

```
for i → 0 to N-1 {
    if (A[i] == K) return i
}
```

TC = O(N)   SC = O(1)

```
      0    1    2    3  |  4   5   6
A = [ (8)  12   15   19 |  2   5   6 ]
          Part 1        |  Part 2
```



∀ elements in Part 2 < ∀ elements in part 1.

Check if an element 'x' is in part 1 or 2 ?

```
if (x < A[0]) ⇒ 'x' is in Part 2
else 'x' is in part 1
```

```
// Search Space
l = 0        r = N-1
while (l <= r) {
    mid = (l+r)/2
    // check mid
    if (A[mid] == K) return mid
    // Decide left/right
    if (K < A[0]) {      // K is in part 2
        if (A[mid] < A[0]) {   // mid is in Part 2
```

```
            if (K < A[mid])   r = mid-1
            else      l = mid+1
        } else {   // mid is in Part 1
            l = mid + 1
        }
    } else {  // K is in part 1
        if (A[mid] >= A[0]) {   // mid is in Part 1
            if (K < A[mid])   r = mid-1
            else    l = mid+1
        } else {   // mid is in Part 2
            r = mid - 1
        }
    }
} return -1
```

$TC = \underline{O(\log_2(N))}$ $SC = \underline{O(1)}$

Q → Find square root of N (N → perfect sq)

N = 49   Ans = 7   N = 25   Ans = 5

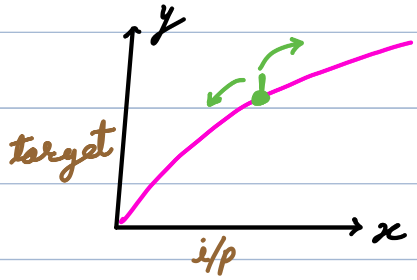
Bruteforce → Range of sqrt of N → [1 N]

```
for i → 1 to N {
    if ( i * i == N )  return i
}
```

$TC = \underline{O(\sqrt{N})}$

Search Space → Integers 1 to N

Target → $\sqrt{N}$



$N = 45$ $\quad x = 5$ $\quad 5 * 5 < N$

Binary Search on Answer

```
l = 1      r = N
while (l <= r) {
    mid = l + (r - l) / 2
    if ( mid * mid == N) return mid
    if (mid * mid < N) l = mid + 1
    else   r = mid - 1
}
```

$TC = \underline{O(\log_2(N))}$ $\quad SC = \underline{O(1)}$

Q → Find N^{th} magical number.

Magical No. → Divisible by either X or Y or both.

$X = 2$, $Y = 5$ } 2 4 5 6 8 10 12 . . .

$N = 3$ → Ans = $\underline{5}$

$N = 6$ → Ans = $\underline{10}$

$N = 4$

$X = 3$, $Y = 5$ } 3 5 6 $\underline{9}$

Bruteforce →

```
i = 1        cnt = 0
while (cnt < N) {
        if ( i % x == 0 || i % y == 0)
            cnt ++
    i ++
} return (i-1)
```

Smallest → $\underline{Min(X, Y)}$ Largest → $\underline{N * Min(X, Y)}$

# of multiples of $X <= K \Rightarrow \underline{K/X}$

$\left.\begin{array}{l} X = 6 \\ K = 50 \end{array}\right\}$ $50/6 = \underline{8}$

# of multiples of X or Y or both <= K →

$X = 3$ → 3 6 9 12 15 18 21 24 27 30

33 36 39

$Y = 10$ → 10 20 30 40

$K = 40$

$$\frac{K}{X} + \frac{K}{Y} - \frac{K}{\cancel{X * Y}}$$

$13 + 4 - 1 = \underline{16}$ Ans $= \underline{16}$

$X = 6$ → 6 12 18 24 30 36

$Y = 10$ → 10 20 30 40

$K = 41$

$\frac{K}{X} = 6$ $\frac{K}{Y} = 4$ $\frac{K}{lcm(x, y)} = 1$ Ans $= 6 + 4 - 1 = \underline{9}$

$\boxed{\text{\# of multiples}}$ of X or Y or both $<= \boxed{K}$ → $\boxed{\frac{K}{X} + \frac{K}{Y} - \frac{K}{lcm(X, Y)}}$

multiples ∝ K

$$\boxed{\frac{x * y}{gcd(x, y)} = lcm(x, y)}$$

```
l = min (X, Y)          r = N * min (X, Y)
while (l <= r) {
        mid = l + (r-l)/2
        // check mid
        cnt = mid/X + mid/Y - mid/lcm(X, Y)
        if (cnt == N && (mid % X==0 || mid % Y==0))
                return mid
        // Decide left/right
        if (cnt < N)      l = mid+1
        else      r = mid-1
}
```

TC = $\underline{O(\log(N * \min(X, Y)))}$

SC = $\underline{O(\log(\max(X, Y))}$ → gcd()

Median of Array → Middle element in sorted order.

$[2 \quad 8 \quad 1 \quad 10 \quad 5] \rightarrow 1 \quad 2 \quad \overset{\downarrow}{5} \quad 8 \quad 10$

$2 \quad 4 \quad \underbrace{5 \quad 7} \quad 8 \quad 12$

average → $\frac{5+7}{2} = \underline{6}$

Q → Find median of array.

Sol → 1) Sort the array
2) Ans = A[N/2]

TC = $\underline{O(N \log(N))}$

SC = $\underline{O(1)}$

Q → Given two sorted arrays, find median of combined data.

A = [6   8   11]

B = [1   7]   Ans = 7

A = [1   3   5   6]

B = [2   8   10]   Ans = 5

Sol 1 → 1) Merge two sorted arrays.

2) Ans = middle element.   TC = O(N+M)

SC = ~~O(N+M)~~ O(1)

Apply merge 2 sorted array without storing the result, just counting the selection.

Sol 2 →   A = [1   3   5   6   7   9]

B = [2   4   7   10   12]

↑ mid

cnt # element < A[mid] = x   → TC = O(log(N))

len = N+M   median position = (N+M)/2   (x + mid) + 1

(#element < K) $\propto$ (position of K in sorted order)

1) Pick any array & apply BS to find ans by counting #elements < A[i] in both arrays.

2) If ans not found, try with another array.

Pseudo-code → HW