- Finding mid
- Q1. Search in Rotated Sorted Array
- Q2. Finding square root of N
- Q3. Median of 2 sorted arrays

9 Dec
↓
23 Dec

25 Dec
↓
holiday

27 Dec

3 Jan
↓
class

Best practice to compute Mid

Let's assume that we have a datatype → dtype which has a range -100 to 100.

Array of length 100 ⇒ 0 to 99 indices

① dtype $l = 0$, $r = 99$

dtype $mid = \frac{l+r}{2} = \frac{0+99}{2} = 49$

⇒ Go right $l = mid + 1$

② $l = 50$ $r = 99$

overflow

$mid = \frac{l+r}{2} = \frac{50+99}{2} = \frac{149}{2}$

✓

mid

$Mid = \frac{l+r}{2} = \boxed{l + \frac{(r-l)}{2}}$

↓

$l + \frac{r}{2} - \frac{l}{2} = \frac{l}{2} + \frac{r}{2}$

$= \frac{(l+r)}{2}$

$l = 50$ $r = 99$

$mid = l + \frac{(r-l)}{2} = 50 + \frac{(99-50)}{2} = 50+24$

$= 74$

1. Given an array of unique elements which was initially sorted, but someone rotated it at an unknown index, so it is a rotated sorted array.

Given k, check if k is present in array or not.

4 5 6 7 8 9 1 2 3

K = 8 true

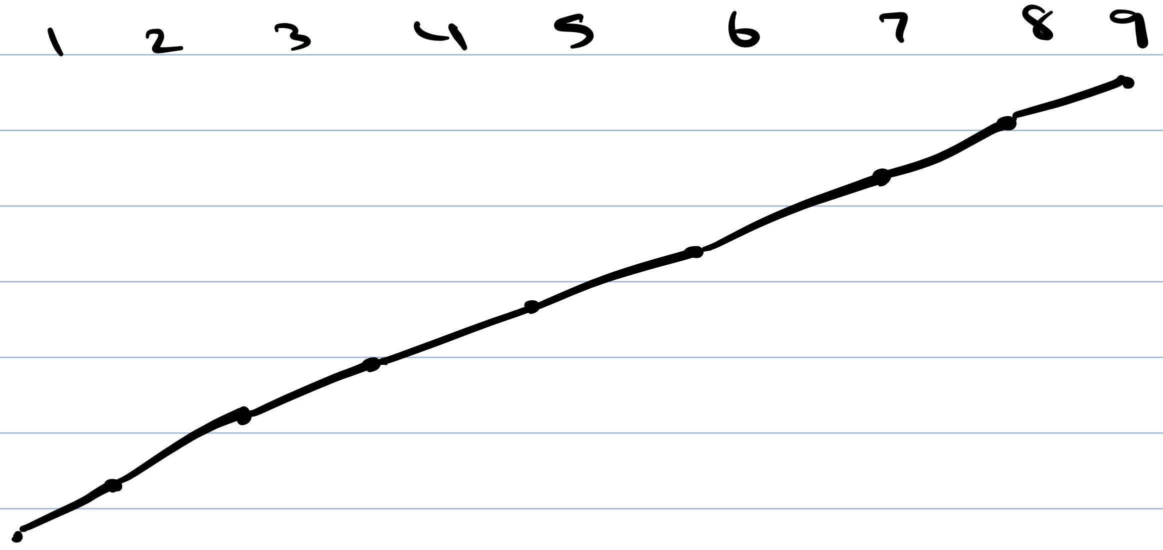
K = 11 false

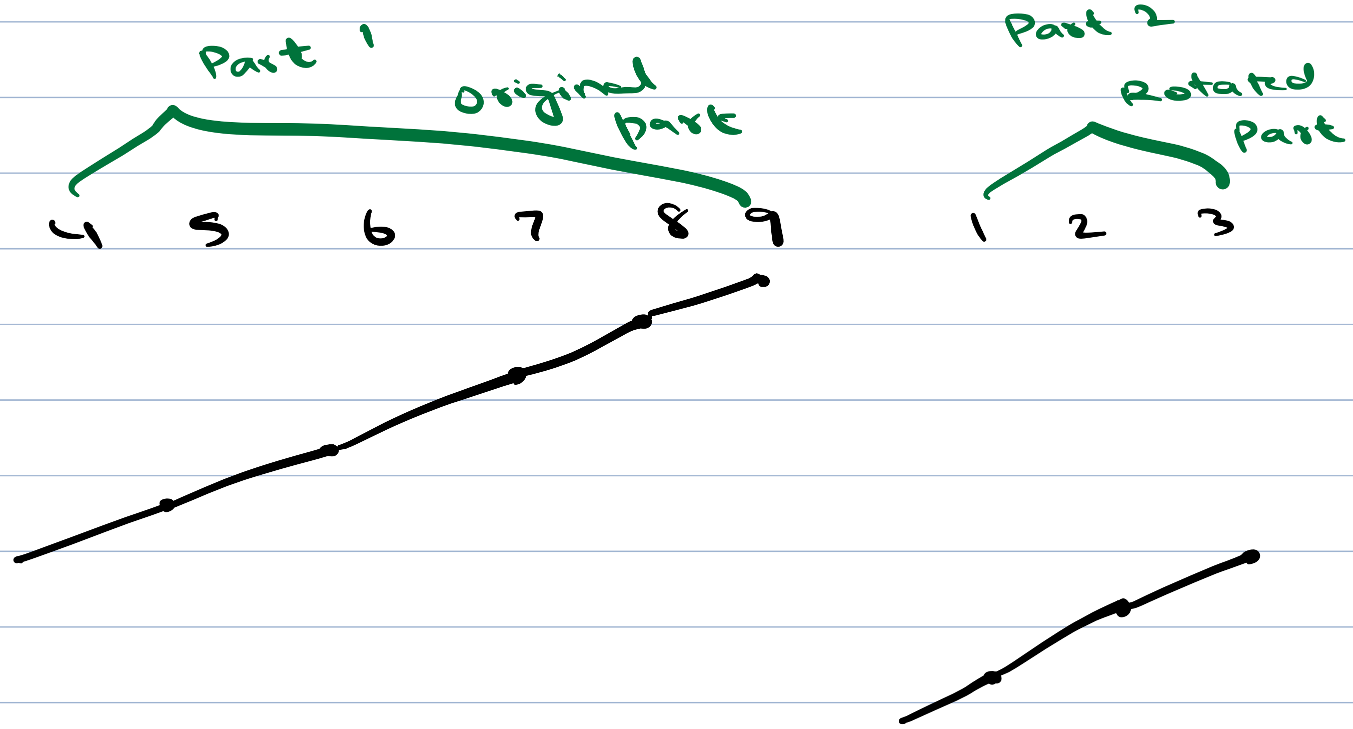
Brute Force: Do a linear search

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

K=2

0	1	2	3	4	5	6	7	8
4	5	6	7	8	9	1	2	3





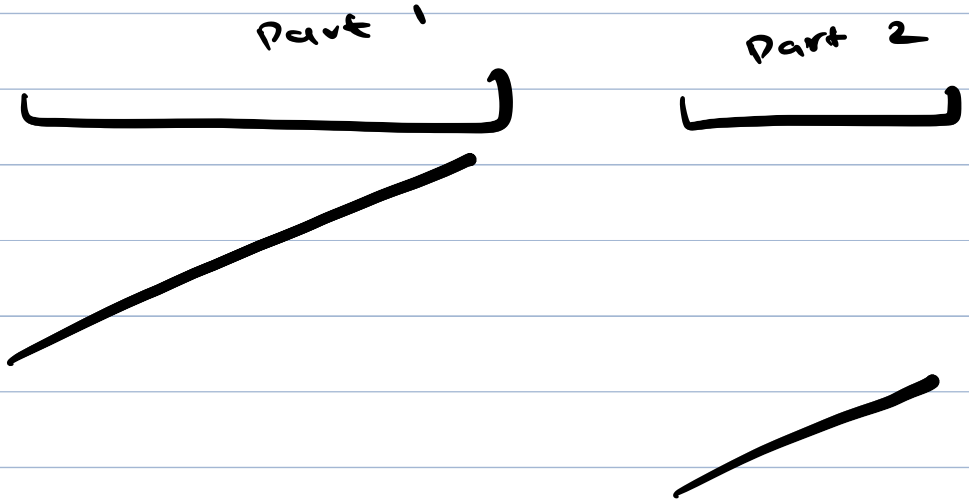
$$\begin{matrix}0\quad 1\quad 2\quad 3\quad 4\quad 5 & 6\quad 7\quad 8\\ \underbrace{4\quad 5\quad 6\quad 7\quad 8\quad 9}_{\text{Part 1}} & \underbrace{1\quad 2\quad 3}_{\text{Part 2}}\end{matrix}$$

If ele < 4 → Part 2

else Part 1

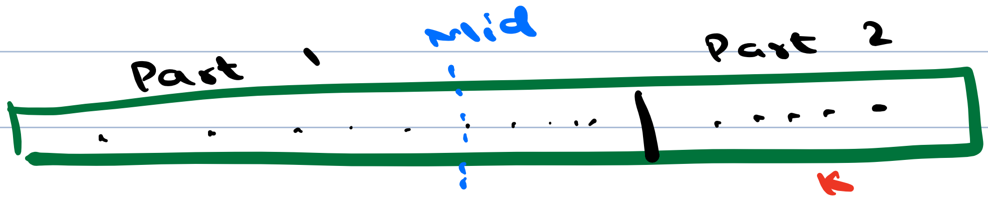
If ele < A[0] → Part 2

else Part 1



① Mid is in first part

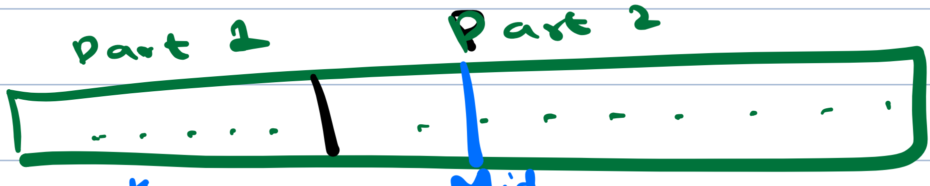
ⓐ K is in second part



Go right, l = mid+1

ⓑ K is in first part

compare to normal rule of BS



K Mid

② mid is in second part

ⓐ K is in first part

Go left $r = mid - 1$

② K is in second part

compare to normal rule of BS

$$\begin{array}{cccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \end{array}$$

$$A = \underbrace{4, 5, 6, 7}_{\text{Part 1}}, \underbrace{0, 1, 2}_{\text{Part 2}} \qquad k = 0$$

l	r	mid	mid is in part?	k is in part?	move?
0	6	3	part 1	Part 2	Go right $l = mid+1$
4	6	5	Part 2	Part 2	$K < a[mid]$ $0 < 1$ Left $r = mid-1$
4	4	4	$a[mid] = target$ stop		

```
bool isKPresent (int a[], int N, int k) {

    int l=0, r = N-1
    while (l <= r) {

        int mid = l + (r-l)/2

        if (A[mid] == K)   return true

        if (A[mid] < A[0]) {
                                 // Mid is in part 2

            if (K < A[0]) { // K is in part 2
                if (K < A[mid])           // Normal
                                             BS
                        r = mid-1

                else
                        l = mid+1
            }
            else {      // K is in part 1
                // left       r = mid - 1
            }
        }
        else {  // Mid is in part 1

            if (K < A[0]) { // K is in part 2
                // right     l = mid+1
            }
            else {        // k is in part 1
                if (k < A[mid])            // Normal
                                              BS
                        r = mid-1
                else
                        l = mid+1
                }
            }
        }
        return false
}
```

TC : $O(\log_2 N)$

2. Given N, find integer part of $\sqrt{N}$.

$N = 36$ ans $= 6$ $\sqrt{36}$

$N = 50$ ans $= 7$ $\sqrt{50}$

$N = 65$ ans $= 8$

$7 \times 7 = 49$

$\underline{7.} \times \underline{7.} = 50$

$8 \times 8 = 64$

$\underline{8.} \times \underline{8.} = 65$

$9 \times 9 = 81$

$N = 36$

i	i^2
1	→ 1
2	→ 4
3	→ 9
4	→ 16
5	→ 25
6	→ 36

$i^2 = N$ $i^2 < N$

$N = 50$

i	i^2
1	→ 1
2	→ 4
3	→ 9
4	→ 16
5	→ 25
6	→ 36
7	→ 49
8	→ 64

$\boxed{i^2 \le N}$

```
int ans
for (i= 1 ; i ≤ N ; i++) <
    if (i×i ≤ N)   ans = i
    else    break
```

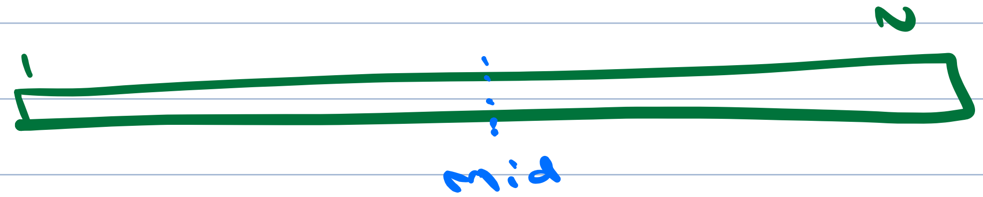
↳ return ans TC : $O(\sqrt{N})$

Binary search :

Target : x who is sqrt of N
⇒ $x \times x <= N$

Search space : 1 to N

Condition :



① mid × mid = N
return mid

② mid × mid > N
Left r = mid - 1

③ mid × mid < N
ans = mid
Right l = mid + 1

N = 65

l	r	mid	mid^2	
1	65	33	1089 > N	Left r = mid - 1

1	32	16	256 > N	Left r = mid-1
1	15	8	64 < N ans = 8	Right l = mid+1
9	15	12	144 > N	Left r = mid-1
9	11	10	100 > N	Left r = mid-1
9	9	9	81 > N	Left r = mid-1
9	8	l > r	stop	

```
int   sqrt (int  N) {
    l=1 , r=N  , ans=0
    while (l <=r) {
        long
        ~~int~~   mid  =  l+ (r-l)/2

        if (mid * mid ==N)
                               return   mid

        else  if (mid * mid >N)
                         r =mid-1        // left

        else {      // mid* mid <N
            ans = mid
             l = mid+1         // right
        }
    }
    return   ans
}
```

TC : $O(\log_2 N)$

SC : $O(1)$

10:40

$N = 65$

l	r	mid	mid^2