Lecture :- Binary Search 2

Agenda

- Search in rotated sorted array
- sqrt(n)
- Ath magical no.
- Median of sorted array.

Qu Search in a rotated sorted array (VVVVI)

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

, target = 7

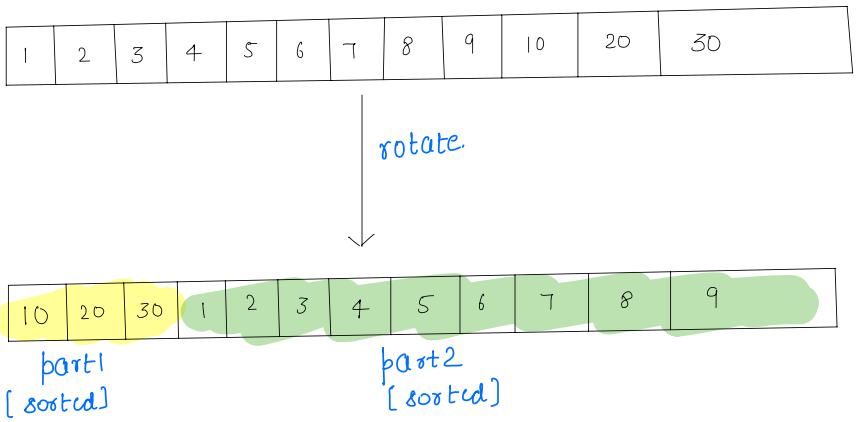
ans = true.

Rotated sorted array example

1	2	3	4	5	sorted
5	1	2	3	4	rotated sorted array.
4	5	1	2	3	
3	4	5	1	2	

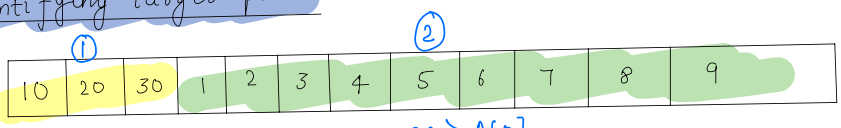
Brute force O(n) — Linear search.

Approach 2



Part1 > Part2.

Identifying target part



target = 20 part1 20 > A[0]

target = 4 part2 4 < A[0]

target = 8 part2 8 < A[0]

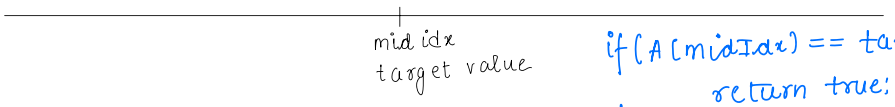
target = 30 part1. 30 > A[0]

Generalisation

```
if( target < A[0]) {
    target - part2
} else {
    target - part1.
}
```

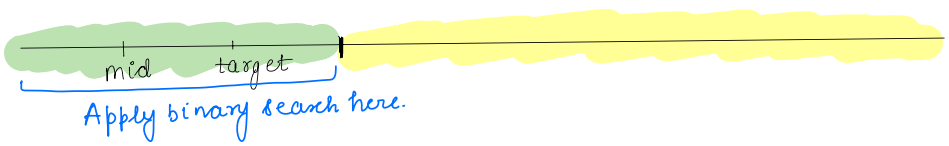
Challenge2: A point which separates part1 and part2.

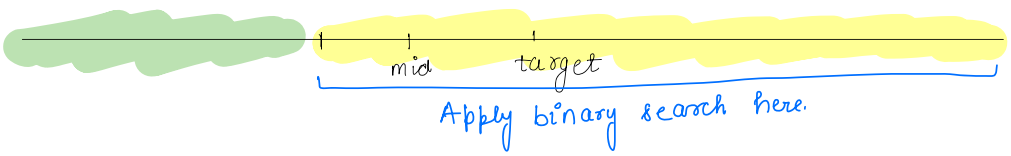
Case1:



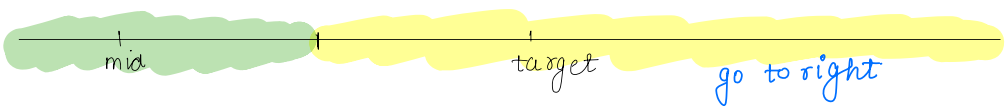

```
if(A[midIdx] == target) {
    return true;
}
```

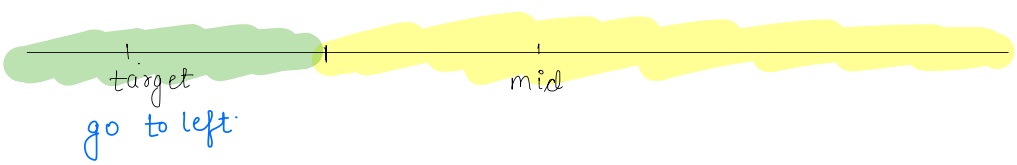
Case2



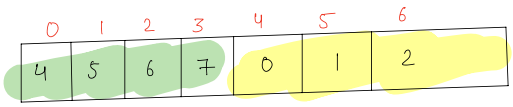


Case3





Dry run:



target = 0

start	end	mid idx	mid value	Action
0	6	3	7	target = part2 , target < A[0] (0 < 4) midvalue = part1. midvalue >= A[0] (7) midvalue = part1 target = part2 } go to right start = midIdx+1 = 4 end = 6.
4	6	5	1	target = part2 , target < A[0] (0 < 4) midvalue = part2 midvalue < A[0] (1 < 4) midvalue target } part2. Apply binary search in part2.

Algorithmic code

```
function rotatedSortedArraySearch( A , target) :
    start = 0;
    end = A.length-1;
    while( start <= end) {
        mid = (start + end) / 2 ;
        if( A[mid] == target) {
            return mid;
        }
        target — part2.
        if( target < A[0]) {
            midvalue — part1.
            if(A[mid] >= A[0]) {
                start = mid +1;
            }
            midvalue — part2.
            else {
                Binary search in part2.
                if( A[mid] < target) :
                    start = mid +1;
                else:
                    end = mid -1;
            }
        target — |
        else :
            midvalue — part2
            if(A[mid] < A[0]) {
                end = mid-1;
            }
            midvalue — part1
            else {
                Binary search in part1.
                if( A[mid] < target) :
                    start = mid +1;
                else:
                    end = mid -1;
            }
    }
```

TC: O(logn)
SC: O(1)

Qu finding square root of a number.

ip	output (sqrt(n))
16	4
28	5
39	6
50	7
48	6

Brute force:

```
i = 1;
ans = 1;
while(i*i <= n) :
    ans = i;
    i++;
}
return ans;
```

Dry run:

n = 10

i=1	ans = 1	1*1<=10 ans = 1 i=2
i=2	ans = 1	2*2<=10 ans = 2 i=3
i=3	ans = 2	3*3<=10 ans = 3 i=4
i=4	ans = 3	4*4<=10 [break]

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

SC: O(1)

Binary search

- Condition
- Search space

n —— sqrt(n) —— start = 1, end = n

n = 10 ⟶ sqrt(10) [1-10]

n = 18 ⟶ " (18) [1-18]

n = 1 ⟶ " (1) [1-1]

Approach 2 n = 50 range [1 - 50]

sqrt(n) = 7

start	end	mid	Action
1	50	25	25 * 25 > 50 Left 625 end = mid - 1;
1	24	12	12 * 12 > 50 Left 144 end = 12 - 1 = 11
1	11	6	6 * 6 < 50 right ans = 6 start = 7
7	11	9	9 * 9 > 50 left end = 9 - 1 = 8
7	8	7	7 * 7 < 50 Right ans = 7 left = mid + 1 7 + 1 = 8
8	8	8	8 * 8 > 50 Left end = 8 - 1 = 7
8	7		break.

TC: log(n)

SC: O(1)

Algorithmic code

```
function sqrt (num):
    if (target==0 || target==1) {
                num        num
        return target;
                    num
    }
    start=1;
    end = target, num
    ans=0;
    while( start<= end) :
        mid = (start + end) / 2;
        if (mid * mid ==num) {
            return mid;
        } else if ( mid * mid < num) {
            start = mid+1;
            ans = mid;
        }
        else {
            end = mid-1;
        }
    }
    return ans:
}
```

TC: $O(\log_2 n)$
SC: $O(1)$

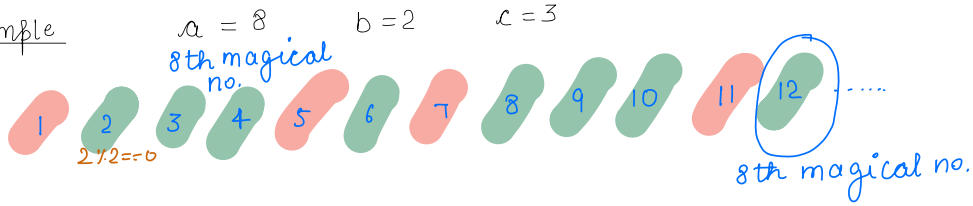
Break: 8:23 – 8:33

Qu Ath magical number.

Magical number: A magical no. is a +ve integer that is divisible by either b or c both. In other words, if a number x is a magical number, it means that x % b == 0 or x % c == 0 or both conditions hold

Task: find ath magical no. based on conditions mentioned above.

Example a = 8 b = 2 c = 3



Bruteforce: Linear fashion — Try it.

Observation a b c

Example: A = 8
B = 2
C = 3

Min possible ans ⇒ 2 [min(b, c)]

Max possible ans ⇒ a * min(b, c) ✓
max(b, c)

Range ⇒ [Min(b, c) —— a * min(b, c)]

Qu How many multiples of B, C will lie b/w range [min(b,c) to a * min(b,c)] ?

Ans

Example: a = 5
b = 3
c = 4

Range = [3 , 15]

3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15

7 no. which are multiple of B (3) and C (4)

[3 - 15] —— Multiples of 3 = $\frac{15}{3}$ = 5 { 3, 6, 9, 12, 15 }

[3 - 15] —— " " 4 - $\frac{15}{4}$ = 3 [4, 8, 12]

Count of multiples of 3 and 4 b/w [3-15) ⟹

$\frac{15}{3} + \frac{15}{4} - \frac{15}{\text{LCM}(3,4)}$

(5) + (3) − 1 $\frac{15}{12}$ = 1

(7)

Dry run / Approach

a = 4
b = 5
c = 7

min = 5 [min(b,c)]
max = 20

5, 7, 10, 14, 15, 20, 21 - — -
↑ 5th magical no.

mid = 16 $\frac{16}{5} + \frac{16}{7} - \frac{16}{35}$ = 5th ←

15 ✓

start	end	mid	Action
5	20	12	Check if 12 is potential 4th magical no or not? $\frac{12}{5} + \frac{12}{7} - \frac{12}{lcm(5,7)}$ 2 + 1 − 0 = 3 [potential 3rd magical no] 5, 7, 10, 14 - - - - - - - - - - start = 13
13	20	16	Check if 16 is potential 4th magical no or not? $\frac{16}{5} + \frac{16}{7} - \frac{16}{lcm(5,7)}$ 3 + 2 − 0 = 5 [potential 5th magical no] 5, 7, 10, 14, - - - - - - - - - - Left [e = 15]
13	15	14	$\frac{14}{5} + \frac{14}{7} - \frac{14}{35}$ ⇒ 2 + 2 − 0 ⇒ 4 ans = 4 Left (e = mid − 1)
13	13	13	$\frac{13}{5} + \frac{13}{7} - \frac{13}{35}$ = 2 + 1 − 0 = 3 right (s = 14)
14	13		break

Pseudocode

```
int count(x, b, c) {
    return x/b + x/c - x/lcm(b,c) ;
}
```

$a * b = lcm(a*b) * gcd(a*b)$;

```
int AthMagicalNo(A, B, C) {
    s = min(b,c)    // s = 1.      ans = 0;
    e = a*s         // e = 10^9.
    while(s<=e) {
        mid = (s+e)/2;
        int cnt = count(mid, b, c);
        if(cnt > A) {
            end = mid-1;
        } else if(cnt<A) {
            start = mid+1;
        } else{   cnt==A
            ans = mid;
            end = mid-1;
        }
    }
    return ans;
}
```

TC: $O(\log_2 n)$
SC: $O(1)$

Qu: find median of two sorted arrays.

A = | -5 | 3 | 6 | 12 | 15 |

B = | -12 | -10 | -6 | -3 | 4 | 10 |

median ⇒

| -12 | -10 | -6 | -5 | -3 | 3 | 4 | 6 | 10 | 12 | 15 |

A = | 2 | 3 | 5 | 8 |

B = | 10 | 12 | 14 | 16 | 18 | 20 |

median ⇒

| 2 | 3 | 5 | 8 | 10 | 12 | 14 | 16 | 18 | 20 |

avg $\left(\frac{10+12}{2}\right)$ = (11) ~~Any~~

Brute force: O(n).

Approach 2 $O(\log_2 n)$

A =

1	3	4	7	10	12

B =

2	3	6	15

Thankyou ☺

Observations

- Sorted array.
- Search space
- Midpoint el.
- Stopping cond^n
- Divide & conquer
- Boundary handling