BINARY SEARCH ALGOS

Page: Raj
Date: / /

Q. Lower Bound : find smallest no. in sorted array which is greater than "x"

```
int left = 0, right = nums.size()-1;
while (left < right)
{
   int mid = left + (right-left)/2;
   if (nums[mid] < x)
     { left = mid+1; }
   else.
   { right = mid
}
                  ↑ first index
return left ; of element >= x
```

Q Floor (largest elemt smaller than x) ceil (smallest elmt bigger than x)

```
{ sort (nums.begin(), nums.end());
   int left=0, right=num.size()-1; int floor=ceil=-1;
while (left <= right) {
{      int mid = left + (right-left)/2;
       if (nums[mid] == x) { floor=ceil=nums[mid]; break; }
   else if (nums[mid] < x) { floor=nums[mid]; left=mid+1; }
   else if (nums[mid > x)  { ceil=nums[mid]; right=mid-1; }
}

return {floor, ceil};
```

Count no. of occurances = last - first + 1

(optimal)

Raj

Page:

Date: / /

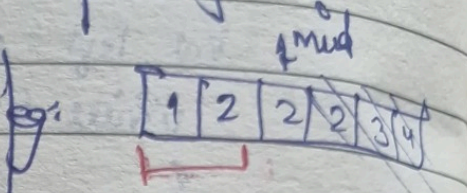
Q first & last position of "x" in sorted Array.

1) find first and last occurance separately

```
// first
while (left <= right)
  int mid = ...
if(nums[mid] == x
{
   first = mid
   right = mid-1
}
else if( nums < x)
{right = mid -1 }
else {left = mid+1 }
```

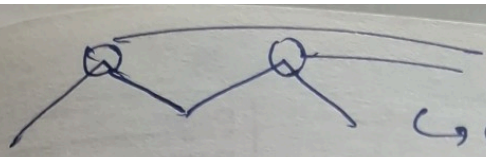
```
// last
while(left <= right)
int mid = ....
if(nums[mid] == x)
{
   last = mid
   left = mid+1
}
```

→ { same }



first will be here (eliminate right part)

opposite for last



Peak element
↳ almost sorted Arr
(use Binary Search)

optimal (logn)

Raj
Page:
Date: / /

PEAK ELEMENT : element greater than neighbours

```
int n = nums.size()-1;
~~while (left < right)~~
```

```
if (num.size() == 1) return 0;
if (nums[0] > nums[1]) return 0;
else if (nums[n] > nums[n-1]) return n;
```

3 manual cases

```
while (left < right)
{
    int mid = left + (right - left)/2;
    if (nums[mid] > nums[mid+1] && nums[mid] > num[mid-1])
    {
        return mid;
    }
    else if (nums[mid] < nums[mid+1])
    {
        left = mid+1;
    }
    else {
        right = mid-1;
    }
}
return left;
```

Peak when left == right

Raj
Page:
Date: / /

Q SINGLE ELEMENT in (SORTED) array where all element come twice

0	1	2	3	4	5	6	7	8	9	10
1	1	2	2	3	3	4	5	5	6	6

Idea

Index: e o e o e o | o e o e
nature

{ o = odd, e = even }

(even - odd) pair | Single | (odd - even) pair

polarity change after single element.

```
while (left < right)
{
    int mid = left + (right - left) / 2 ;

    if (mid % 2 != 0)   // odd index of Mid
        mid--

    Valid Pair & Single is on Right
    if (nums[mid] == nums[mid + 1])
    {
        left = mid + 2;
    }
    else{   // Invalid, single is on left (including Mid)
        right = mid
    }
```

eg: a a b c c d d

mid

(mid--) mid

a a b c c d d

l r

not valid

mid

a a b c c d d

l r

mid

a a b

Valid: Mid+2

a a b

l r

when l = r

then that's the single element

{(Distinct values)}

Page: Raj
Date: / /

Search in Rotated Sorted Array

Here we can't eliminate a particular half
- we need to know which side is sorted and then check on that side

```
while (left <= right) {
    int mid = left + (right - left) / 2;

    if (nums[left] <= nums[mid])   // target
    {
        return mid;
    }
    // Left is sorted
    else if (nums[left] <= nums[mid])
    {
        // lies in left Part
        if(target >= nums[left] && target <= nums[mid])
        {
            right = mid - 1
        }
        else {left = mid + 1 }
    }
    // Right is sorted
    else {
    if (target > nums[mid] && target <= nums[right])
        {
            left = mid + 1;
        }
    else
        {
            right = mid - 1;
        }
    }

    return -1;
```

target = 1 = t

7 8 9 1 2 3 4 5 6

sorted (2 to 6)

t lies in sorted part? NO

(NO, so eliminate

mid

7 8 9 1

l r

sorted

∵ 7 <= 8 ✓ (1 to 8)

t lies in sorted Half?

↳ NO, eliminate it

mid

9 1

l r

sorted

9 <= 9 'NO', eliminate

1 l == r (Got it)

l r

Search in Rotated Array {with duplicate Values}

We can't check sorted part just by nums[left] <= nums[mid]

↳ Becoz in case like

mid
| 2 | 3 | 2 | 1 | 1 | 2 |
l ↑ (at 2 | 3 | 2 | 1 | 1 | 2) ↑ r

here dono hi sorted as esse lgh

So apply a condⁿ

Duplicate check ;

```
if (nums[left] = = nums[mid] && nums[right] = nums[mid])
{
    left ++ ;
    right - - ;
}
```

Q How many times Array is Rotated

→ First find Mini element

- In BS we need to eleminate one half.
- If one part is sorted then other have the min element

if left is sorted then take left most for "minm" element and left = mid + 1

```
While (left < righ)   { int mid - - - - -   }

    if (nums [mid] > nums [right])   left = mid + 1
    else   right = mid
}

return nums [left]
```

Pattern II : Binary Search
(Find min / max)

Raj
Page:
Date: / /

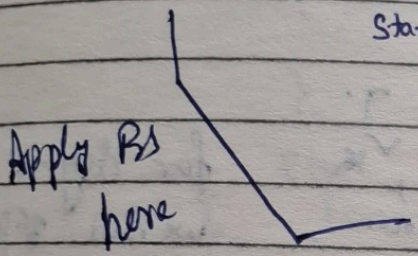
Q Find N^{th} root of Integer (Binary Search on answers)

Ex if num = 27 find N = 3 (3rd root) i.e 3

So

Possible answers 1 2 3 4 . . . 14 . . . 27

↑ Start ↑ Mid ↑ end



check if 14x14x14 < or > or == to 27

Q Min no. of day to make M bouquets (each bouquet have K flowers)

- i^{th} flower blooms on arr[i] day
- We need K adjacent flowers in each bouquet

① Impossible case

↳

On last day (max day) all flower blooms i.e 'n'
we need (m x k) flower for goal

if (m x k) > n → Impossible return -1

Page: Raj
Date: / /

Binary Search on Answer

eg: | 7 | 7 | 7 | 7 | 13 | 11 | 12 | 7 |

n = 2 k = 3

~~n = 3, k = 2~~

↳ here possible answer min to max days

∴ 7 8 9 10 11 12 13

low mid high

- Check for every mid

↳

7 7 7 7 13 11 12 9

✓ ✓ ✓ ✓ X ✓ ✓ ✓

3 adj flow = 1 boug.

3 adj flower = 1 boug

for day = 12 we need 6 flow

2 bouquet = M ✓

{

Use BS and for every mid, iterate a loop on array if it is possible to get k consecutive no. (m times) which are less than or equ to mid

In Bin. Search on Answers

1. decide min and max range.
2. Method to eliminate one half.
3. move left / Right pointer

```
int boughMade (vector<int> v, int mid, int k)
{
    int count = 0, boug = 0;
    for(i = 0 to v.size())
    {
        if(v[i] <= mid)
        { count++;
            if(count == k){boug++; count = 0;}
        }
        else
    }   count = 0;

    return b;


int minDays( vector, m, k)
{
    int left = * min_element(v.begin(), v.end());
    int right = * max_element(v.begin(), v.end());
{ while (left <= right)
    int mid = same;
    int fb = boughMade (v, mid, k)

    if (fb < m) // need more boug
        left = mid + 1

    else { minD = min(minD, mid);
           right = mid - 1;
    }
return minD;
```

Raj
Page:
Date: / /

Q. Find k^{th} missing no. in sorted array

eg: k=5 [2 | 3 | 4 | 7 | 11] ✓ ⇒ 9.

1 2 3 4 5 6 7 8 9 10 11

1st 2nd 3rd 4th 5th

- arr ⇒ 2 3 4 7 11
- missing no till that indx ⇒ 1 1 1 3 6

low mid high

↓ at end when high crosses low

arr[high = 7] . missing = 3

k - missing = 2

+1 → 8

+1 → 9 ✓

Raj
Page:
Date: / /

```
int low = 0 , high = arr.size() - 1;
while (low <= high)
{
    int mid = (low + high) / 2;
    int missing = arr[mid] - (mid + 1)
    if (missing < k)
    {
        low = mid + 1
    }
    else
    {
        high = mid - 1;
    }
}
return k + high + 1
```

arr[high] + more
+ (more k - missing)

missing = arr[high] - (high + 1)

arr[high] + more

Pattern III : Binary search on answers II

Find min of max vice versa

Raj
Page:
Date: / /

Q Aggressive cows : given array with coordinates of n stalles, place 'k' cows in so that min dist b/t any 2 cows are maximum

eg: | 0 | 3 | 4 | 7 | 10 | 9 |

cows (k) = 4

- min value = 1
- max value = (max - min)

Binary Search

Range

- Always fix cow-1 on index 0
- For every mid check if possible funct"

```
bool possible (vector &v, int dist, int k)
{
  count=1;  last = v[0];
  for (i=0 to v.size())
  {
    if ( v[i]-last >=dist )
      { last = v[i]
        count++;
      }
    else continue
    if (count >=k)  return True;
  }
  return false
}
```

REST APPLY BS for Range and check this funct" for all mid = dist

mindist

0 3 4 7 10 9

C_1, C_2, C_3, C_4 ✓(1)

$C_1, C_2 \to C_3, C_4$ ✓(3)

$C_1, C_2 \to C_3 \to C_4$ ✓(2)

C ✗(4)

So min dist of 1, 2, 3 are possible but after that we can't place all cows

ALLOCATE

Q Array repr by

- Each student get atleast

Sol

Range

max

- for
- Poss

ALLOCATE BOOKS

Page: Raj
Date: / /

Q Array represent no of pages, allocate book to students so that max pg to one students are minimum

- Each student should get atleast one Book
- Each book only allocated to one student
- Order of allocation be contiguous

Soln

Range

max possible pages = Σ arr[i] when students = 1

min possible (we need this as max poss) = Max of array

Binary search for all mid

- for every mid : check if that is possible or not
- Impossible : m > n m = no. of stud., n = size of arr

Possible

```
Stud = 1, pgstu = 0
for (i=0 to n)
{
  if (pgstu + arr[i] <= mid)
  {
    pgstu = pgstu + arr[i]
  }
  else {
    Stu++;
    pgstu = arr[i];
  }
```

BS on Range

```
if (Possible)
{ // Look for lesser mid
    right = mid - 1;
}
else
{
    left = mid + 1;
}
```

SPLIT ARRAY | Painters Partition

Page: ____ Date: / /

Q. Splice Array in "k" non empty sub arr so that largest sum is minimum.

(k = 2)

eg: 10 20 30 40

optimal soln → {10, 20, 30} {40}

low = max of Array
High = Σ summation of array. → (K=1)

Allot Book eg: {25, 46, 28, 49, 24} K = 4

K=5 (K=1)

Range 49 (max) ... 71 ↑ mid ... 172 (Σ arr[i])

if it is possible to allocate max 71 pages.

yes → Try to allocate lesser pages; right = mid - 1

NO → inc pages; left = mid + 1

NOTE
Q. Want to minimise the max Pages
So for our range every mid is max pages
. If able to satisfy toh kam karo max pg (mid)
. else max page (badao)

Raj
Page:
Date: / /

SPLIT ARRAY

Ideally

ans. (sub arr ka min.. largest sum) ans ↑

make

'K' subarrays K ↓

Agar for particular "mid" (max possible sum)

if no of subarr > K → means mid is less

So, to inc mid → left = mid + 1