Algorithms

18. Binary Search

Handwritten by pankaj kumar

Binary Search

Date

Page No. 265.

① Binary Search :

```
int search (vector<int>& nums, int target){
    int start = 0;
    int end = nums.size()-1;
    while (start <= end){
        int mid = start + (end-start)/2;
        if (target == nums[mid]) return mid;
        else if (target < nums[mid]) end = mid-1;
        else start = mid+1;
    }
    return -1;        /// if target not found return -1.
}
```

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

② Binary Search on reverse sorted array :

```
while (start <= end){
    int mid = start + (end-start)/2;
    if (target == nums[mid]) return mid;
    else if (target < nums[mid]) start = mid+1;
    else end = mid-1;
}
return -1;
```

TC : O(logn)

③ Order not known BS or, Agnostic BS :-

Let's we have array 'nums'

First check if array element is in ascending or, in descending then apply Binary search as previous two question

```
if (nums[0] > nums[1]){
        Apply descending order Binary search
}else{
        Apply ascending order Binary search
}
```

(4) First and last occurrence of an element:-

ex:- Input: A[] = [-1, 1, 2, 2, 2, 5, 6, 12, 12], target = 2
Output: 2, 4
(2 ↑ first occurrence, 4 ↖ last occurrence)

Code:-

```
vector<int> searchRange(vector<int>& nums, int target){
    vector<int> res;
    int start=0, first=-1, last=-1, end=nums.size()-1;
    while(start<=end){
        int mid = start + (end-start)/2;
        if(target==nums[mid]){
            first=mid;
            end=mid-1;
        }
        else if(target<nums[mid]) end=mid-1;
        else start=mid+1;
    }
    start=0, end=nums.size()-1;
    while(start<=end){
        int mid = start + (end-start)/2;
        if(target==nums[mid]){
            last=mid;
            start=mid+1;
        }
        else if(target<nums[mid]) end=mid-1;
        else start=mid+1;
    }
    res.push_back(first);
    res.push_back(last);
    return res;
}
```

(5) Count of an element in a sorted array:-

ex:- ~~Input: A[] = [-1, 1, 2, 2, 2, 5, 6, 12, 12], target = 2~~
output: 3

Code:- in previous question after finding first & last ans will be equal to (last - first + 1) for that ele.

```
if(last==-1 && first==-1)
    return 0;
else
```

Date
Page No. 268.

(6) Find minimum in Rotated sorted array.

for example an array nums = [0,1,2,3,4,5,6,7] might become
- [4,5,6,7,0,1,2] or [0,1,2,4,5,6,7] → unique element
  ↳ rotated by 4 times ↳ rotated by 7 times

so, in Input we have given, rotated sorted array, and we have to find min/m element.

ex:- input nums = [3,4,5,1,2]
o/p: 1

Code:-

```
int FindMin(vector<int>& nums){
    int n=nums.size();
    int start=0,end=n-1;
    if(n==1) return nums[0];
    while(start<=end){
        int mid=start+(end-start)/2;
        int prev=(mid+n-1)%n;
        int next=(mid+1)%n;
        if(nums[mid]<nums[prev]&&nums[mid]<nums[next])
            return nums[mid];
        else if(nums[end]<nums[mid]) start=mid+1;
        else end=mid-1;
    }
    return -1;
}
```

(7) Search in Rotated sorted array.

ex:- (i) nums = [4,5,6,7,0,1,2], target = 6
o/p:- 2 (index)

(ii) nums = [4,5,6,7,0,1,2], target = 3
o/p: -1

Note:- all element are distinct

Code:-

```
int search (vector<int>& nums, int target){
    int n= nums.size();
    int i=0, j=n-1;
    while (i<=j){
        int mid = i+(j-i)/2;
        if (nums[mid]==target) return mid;
        if (nums[i]<=nums[mid]){
            if (target>=nums[i] && target<=nums[mid])
                j=mid-1
            else i=mid+1;
        }
        else if (target>=nums[mid] && target<=nums[j])
                i=mid+1;
            else j=mid-1;
    }
    return -1;
}
```

(8.) Searching in Nearly Sorted array

Given a sorted array arr[] of size N, some elements of array are moved to either of the adjacent positions, i.e arr[i] may be present at arr[i+1] or arr[i-1] i.e arr[i] can only be swapped with either arr[i+1] or arr[i-1]. The task is to search for an element in this array. if element not present then return -1, else its index.

ex:- arr[] = {10, 3, 40, 20, 50, 80, 70}, key=40,
o/p: 2

code:-
```
int search (int arr[], int key){
    int start=0, end= arr.size()-1;
    while (start<=end){
        int mid = start+(end-start)/2;
        if (arr[mid]==key) return mid;
        if (mid>start && arr[mid-1]==key) return mid-1;
        if (mid<end && arr[mid+1]==key) return mid+1;
        if (arr[mid]>key) end=mid-2;
        else start=mid+2;
    }
    return -1;
}
```

(9) find floor of an element in sorted array:

Given a sorted array and a value 'x', the floor of 'x' is the largest element in the array smaller than or equal to x. Find the floor of x.

ex:- (i) I/P: arr[] = {1, 2, 8, 10, 10, 12, 19}, x = 5
O/P: 2

(ii) I/P: arr[] = {1, 2, 8, 10, 10, 12, 19}, x = 20
O/P: 19

Code:

```
int FindFloor(vector<int> arr, int x){
    int start=0, end=arr.size()-1, res=-1;
    while (start<=end){
        int mid = start + (end-start)/2;
        if (x==arr[mid]) return mid;
        else if (x<arr[mid]) end=mid-1;
        else{
            res=arr[mid];
            start=mid+1;
        }
    }
    return res;
}
```

(10) find ceil of an element in sorted array:-

ceil value of 'x' is the smallest element in the array larger than or equal to 'x'.

ex:- (i) I/P: arr[] = {1, 2, 8, 10, 10, 12, 19}, x = 5, O/P: 8

Code:-

```
while (start<=end){
    int mid = start + (end-start)/2;
    if (x==arr[mid]) return mid;
    else if (x>arr[mid]) start=mid+1;
    else{
        res = arr[mid];
        end = mid-1;
    }
}
```

(11.) Next Alphabetical element :-

I/P :- letters = ["D", "J", "K"], target = "B", O/P :- "D"

I/P :- letters = ["h", "n", "s"], target = "t", O/P :- "h"

↳ (letter also wrap around)

code :-

```
char nextGreaterLetter(vector<char>& letters, char target){
    int start=0, end = letters.size()-1, res= letters[start];
    while (start<=end){
        int mid = start + (end-start)/2;
        if (target == letters[mid]) start=mid+1;
        else if (target< letters[mid]){
            res = letters[mid];
            end = mid-1;
        }
        else
            start= mid+1;
    }
    return res;
}
```

here we don't need to return matched

(12) Find the pos of an element in infinite sorted array :-

```
int binarySearch(int arr[], int l, int r, int x){
    while (l<=r){
        int mid = l+ (r-l)/2;
        if (arr[mid]==x) return mid;
        if (arr[mid]>x) r=mid-1;
        else l=mid+1;
    }
    return -1;
}
int FindPos(int arr[], int key){
    int l=0, h=1;
    int val= arr[0];
    while (val<key){
        l=h;
        h=2*h;
        val= arr[h];
    }
    return binarySearch(arr, l, h, key);
}
```

$T(i) \mid \log n + \log n = \log n$

(13) Find the index of 1st 1's in a Binary sorted array

Code:-

```
int FirstIndex (int a[], int n){
    int start=0;
    int end = n-1;
    while (start<=end){
        int mid = start+(end-start)/2;
        if (a[mid]==1){
            if (a[mid-1]==1)
                end=mid-1;
            else
                return mid;
        }
        else
            if (a[mid]==0)
                start=mid+1;
    }
    return -1;
}
```

(14) Find Peak Element

Given an array, find a peak element (print any one, if many are found). A peak element is one such that it is either greater than or equal to its neighbours. For the first & last element, it is enough to look at its only one neighbour.

ex:- (1) I/p: arr={3,5,4,1,1}, O/p: 5

(2) I/p: arr={2,6,3,7,8,9}, O/p: 6

Note:- here 9 is also a peak element.

*Approach:-

- start mid end: [3|5|4|1|1] — since 4 is not peak, 5 is greater ∴ end=mid-1
- mid end: [3|5|4|1|1] — 3 is not a peak, 5 is greater ∴ start=mid+1
- mid: [3|5|4|1|1] — 5 is peak

Code :-

```
int findPeakElement (int arr[], int n) {
    int start=0, end=n-1;
    while (start<end) {
        int mid = (start+end)/2;
        if (mid==0)
            return arr[0]>=arr[1] ? arr[0] : arr[1];
        if (mid==n-1)
            return arr[n-1]>=arr[n-2] ? arr[n-1] : arr[n-2];
        if (arr[mid]>=arr[mid-1] && arr[mid]>=arr[mid+1])
            return arr[mid];
        if (arr[mid]<arr[mid-1]) end=mid-1;
        else start=mid+1;
    }
    return arr[start];
}
```

(15) Find maxm element in Bitonic array.

Bitonic array :- monotonic ↑ → monotonic ↓

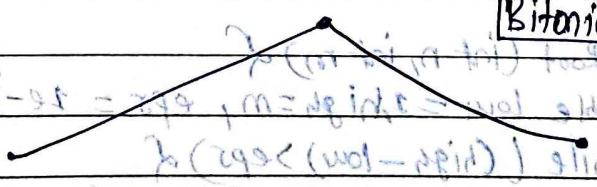
monotonic ~~increasing~~ (incr/dec) array is in which arr[i] != arr[i+1].

ex:- arr[] : [1|3|8|12|4|2], O/P :- (12)

increasing decreasing

Code :-

Bitonic array maxm ≡ Find Peak element



So, this problem is same as Find peak element.

Note :- here only one peak element are present in Bitonic array and in previous question there can be more than one peak element

(16) Search an element in Bitonic array.

ex:- arr[] = | 1 | 3 | 8 | 12 | 4 | 2 | (indices 0 1 2 3 4 5), key = 4

o/p:- 4.

Approach :—

(1) Using Find peak element / maxm element in Bitonic array Find maxm element

(2) check if maxm element is not the target element then (A) apply binary search on left side of maxm element (B) apply reverse BS on right side of maxm element

(17) Nth root of a number using binary search

- Given two no. N and m Find the Nth root of m.
- nth root of m is defind as a no. x, when raised to power N equals m i.e $x^N = M$ or, $x = \sqrt[N]{m}$

ex:- N=3, m=27
o/p:- 3

ex:- N=2, m=16
o/p:- 4

ex:- N=5, m=243
o/p:- 3
i.e $3^5 = 243$

Code :—

```
double multiply (double number, int n) {
    double ans = 1.0;
    for (int i=1; i<=n; i++) {
        ans = ans * number;
    }
    return ans;
}

double getNthRoot (int n, int m) {                    → (10-7)
    double low = 1, high = m, eps = 1e-7;
    while ((high - low) > eps) {
        double mid = (low + high) / 2.0;
        if (multiply (mid, n) < m)
            low = mid;
        else
            high = mid;
    }
    return low;
}
```

Dry Run

n=2, m=16
mid = 8
8*8 = 64 > m
∴ high = 8
now, mid = 5,
5*5 = 25 > 16
∴ high = 5
low, mid = 3.5

TC: $N * \log(m * 10^d)$ → digit