Binary Search

Why Binary Search?

Where can we use binary search?

: In any monotonic sequence.

Use case of Binary Search

To determine the passing marks for a subject out of a total of 100, how many minimum steps are needed? (To be answered by 1st and 2nd years)

a) 1
b) 100
c) 25
d) Something else

Overview of Binary Search

BINARY SEARCH IS A POWERFUL ALGORITHM FOR SEARCHING IN SORTED DATASETS.

KEY CONCEPT: IGNORE HALF OF THE ELEMENTS IN EACH STEP, LEADING TO LOGARITHMIC TIME COMPLEXITY.

Why Binary Search?

- Time complexity: O(log(n))

Topics to be covered

Searching for an element in a sorted array.

Finding lower bound and upper bound in a multivalued sorted array.

Solving binary search on answer problems.

Case 1:
Basic Binary Search (Distinct Elements)

Example:

- Searching for an element x in a sorted array.
- n = length of the array
- arr = the sorted array

```cpp
// find index of element in sorted array
int findElement(vector<int> &vec, int x){
    int low=0;
    int high=vec.size()-1;
    int ans=-1;
    while(low<=high){
        int mid=(low+high)/2;
        if (vec[mid]==x){
            ans=mid;
            break;
        }
        else if (vec[mid]>x){
            high=mid-1;
        }
        else{
            low=mid+1;
        }
    }
    return ans;
}
```

Case 2: Multivalued Sorted Array (Example 1)

- Example: Find the first index of the element x in a sorted array.
- n = length of the array
- arr = the sorted array

```cpp
// first index of an element in multivalued sorted array
int firstIndex(vector<long long> &vec,long long x){
    int n=vec.size();
    int low=0;
    int high=n-1;
    int ans=-1;
    while(low<=high){
        int mid=(low+high)/2;
        if(vec[mid]==x){
            ans=mid;
            high=mid-1;
        }
        else if (vec[mid]>x){
            high=mid-1;
        }
        else{
            low=mid+1;
        }
    }
    return ans;
}
```

Case 2: Multivalued Sorted Array (Example 2)

- Example: Find the last index of the element x in a sorted array.
- n = length of the array
- arr = the sorted array

```cpp
// last index of element in multivalued sorted array
int lastIndex(vector<long long> &vec,long long x){
    int n=vec.size();
    int low=0;
    int high=n-1;
    int ans=-1;
    while(low<=high){
        int mid=(low+high)/2;
        if(vec[mid]==x){
            ans=mid;
            low=mid+1;
        }
        else if (vec[mid]>x){
            high=mid-1;
        }
        else{
            low=mid+1;
        }
    }
    return ans;
}
```

Exercise: Upper Bound in a Multivalued Sorted Array

- Find the index of the upper bound of an element x in a multivalued sorted array.
- Upper bound: First element which is greater than x.
- n = length of the array
- arr = the sorted array

```cpp
// upper bound

int upperBound(vector<int> &vec, int x){
    int n=vec.size();
    int low=0;
    int high=n-1;
    int ans=-1;
    while(low<=high){
        int mid=(low+high)/2;
        if (vec[mid]>x){
            ans=mid;
            high=mid-1;
        }
        else{
            low=mid+1;
        }
    }
    return ans;
}
```

Problems on Binary Search on Answers

Problem 1:

https://www.geeksforgeeks.org/problems/index-of-first-1-in-a-sorted-array-of-0s-and-1s4048/1

Problem 2:

https://www.geeksforgeeks.org/problems/square-root/1

Problem 3 (1st year can skip):

https://www.geeksforgeeks.org/assign-stalls-to-k-cows-to-maximize-the-minimum-distance-between-them/

```cpp
// first index of 1
int firstIndex(vector<int> &a, int n)
{
    int low=0;
    int high=n-1;
    int ans=-1;
    while(low<=high){
        int mid=(low+high)/2;
        if(a[mid]==1){
            ans=mid;
            high=mid-1;
        }
        else{
            low=mid+1;
        }
    }
    return ans;
}
```

```cpp
// sqrt

long long int floorSqrt(long long int x)
{
    // Your code goes here
    long long low = 1;
    long long high = x;
    long long ans = 1;

    while(low<=high){
        long long mid = (low+high)/2;

        if(mid*mid<=x){
            ans = mid;
            low = mid+1;
        }
        else{
            high = mid-1;
        }
    }

    return ans;
}
```

```cpp
// aggressive cows

int solve(int n, int k, vector<int> &stalls) {
    sort(stalls.begin(),stalls.end());
    int ans=-1;
    int low=0;
    int high=stalls[n-1]-stalls[0];
    while(low<=high){
        int mid = (low+high)/2;

        int temp = 1;
        int pre = stalls[0];
        for(int i=1;i<n;i++){
            if(stalls[i]-pre>=mid){
                temp++;
                pre = stalls[i];
            }
        }
        if(temp>=k){
            ans = mid;
            low = mid+1;
        }
        else{
            high = mid-1;
        }
    }
    return ans;
}
```