Binary Search

Q. Given an array arr of n integers and an integer x.
Write C++ program to find any position i such that arr[i] = x. If x is not present in the array, i = -1. Print i.

For eg:

Input:
arr=[2, 5, 1, 4, 7], x=4

Output:
3

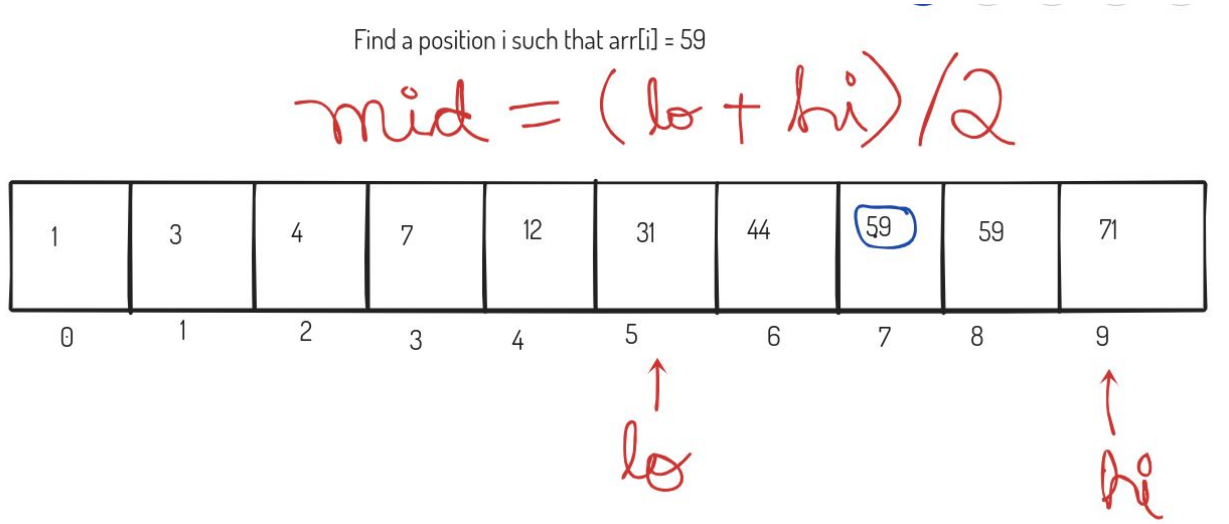
```
int pos=-1;
for(int i=0; i<n; i++)
{
    if(arr[i]==x)
   {
       pos=i;
       break;
   }
}
cout<<pos;
```

This is called **linear search**.

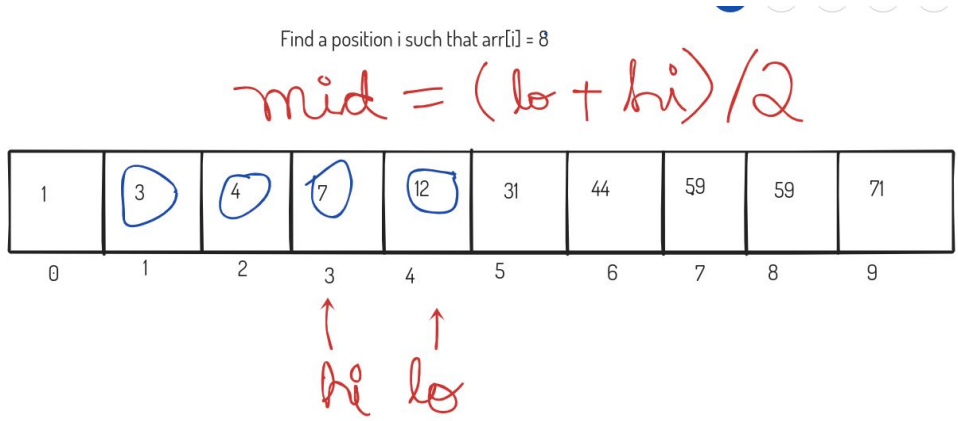
Time complexity: O(n) [Worst case time complexity]

Q. Given an array arr of n integers and an integer x. Array arr is sorted in non-decreasing i.e arr[i] <= arr[i+1].
Write C++ program to find any position i such that arr[i] = x. If x is not present in the array, i = -1. Print i.

mid = (0+9)/2 = 4
mid = (5+9)/2 = 7



(0+3)/2 = 1
(2+3)/2 = 2



Code:

```
int lo=0, hi=n-1;
int mid;
int pos=-1;
while(lo <= hi)
{
  mid = (lo+hi)/2;
  if(arr[mid]==x)
  {
    pos=mid;
    break;
  }
  else if(arr[mid]<x)
  {
      lo = mid +1;
  }
  else {
    hi = mid - 1;
  }
}
cout<<pos;
```

This is called **binary search**.

Time complexity: O(log N) [At every step, active region of search reduces by half]
n => n/2

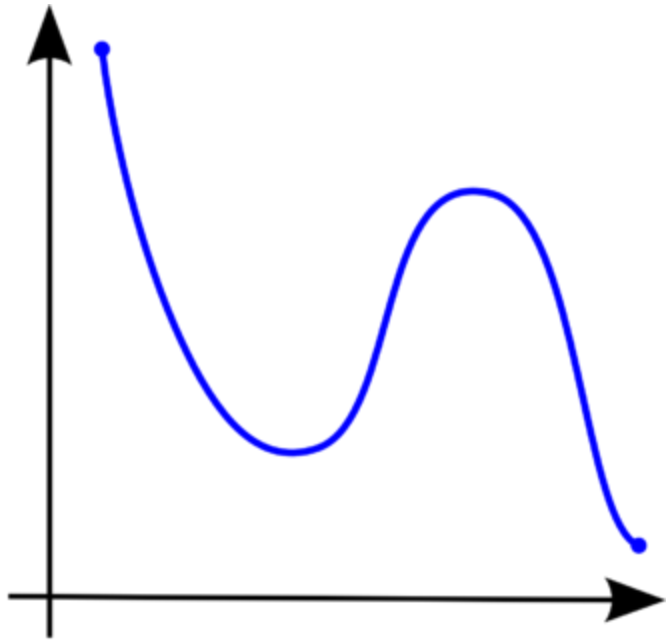
n/2 => n/4
n/4 => n/8
…..
1
n/(2^i) = 1
It will be O(log N)

To calculate the middle element, the above formula mid = (lo+hi)/2, then overflow may occur.

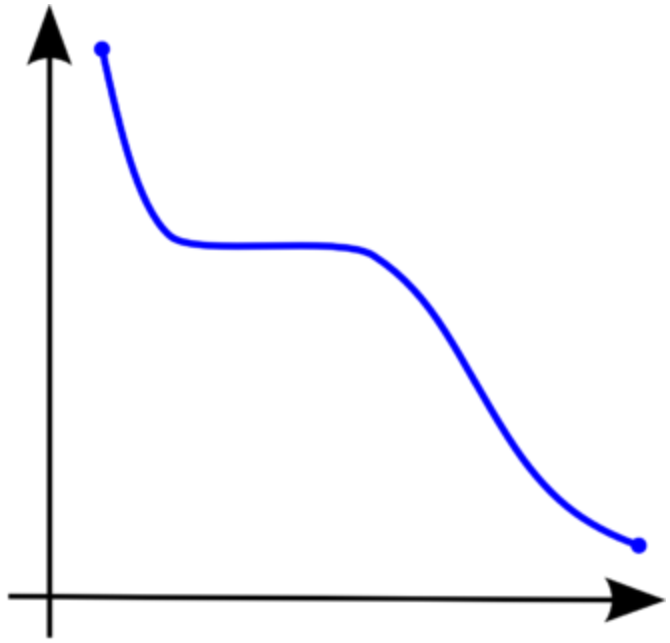
So, you should always use this formula to calculate middle element:

mid = lo + (hi-lo)/2;
or
mid = lo + (hi-lo+1)/2;

Q. Which of the following is a monotonic function?



Function a



Function b

Function b is a monotonic function because in the whole interval, it is non-increasing

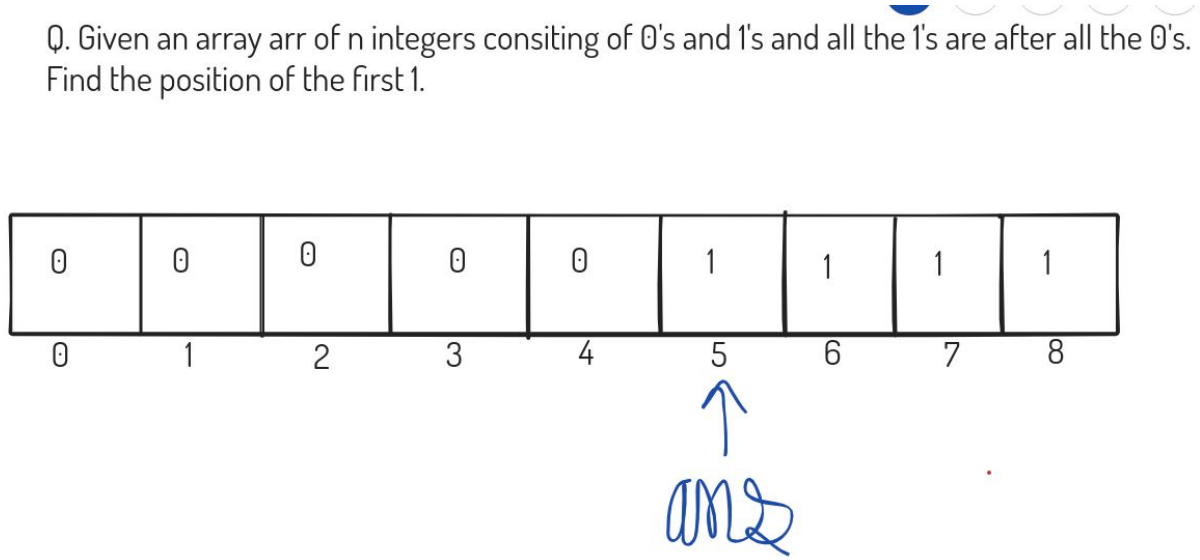
Function a is not monotonic because it first decreases, then increases and

then again decreases.

Note: Binary search can always be applied only on monotonic functions. [IMPORTANT]

Q. Given an array arr of n integers consiting of 0's and 1's and all the 1's are after all the 0's, find the position of the first 1.
The array elements are monotonically non-decreasing, so we can apply binary search here.

Eg:



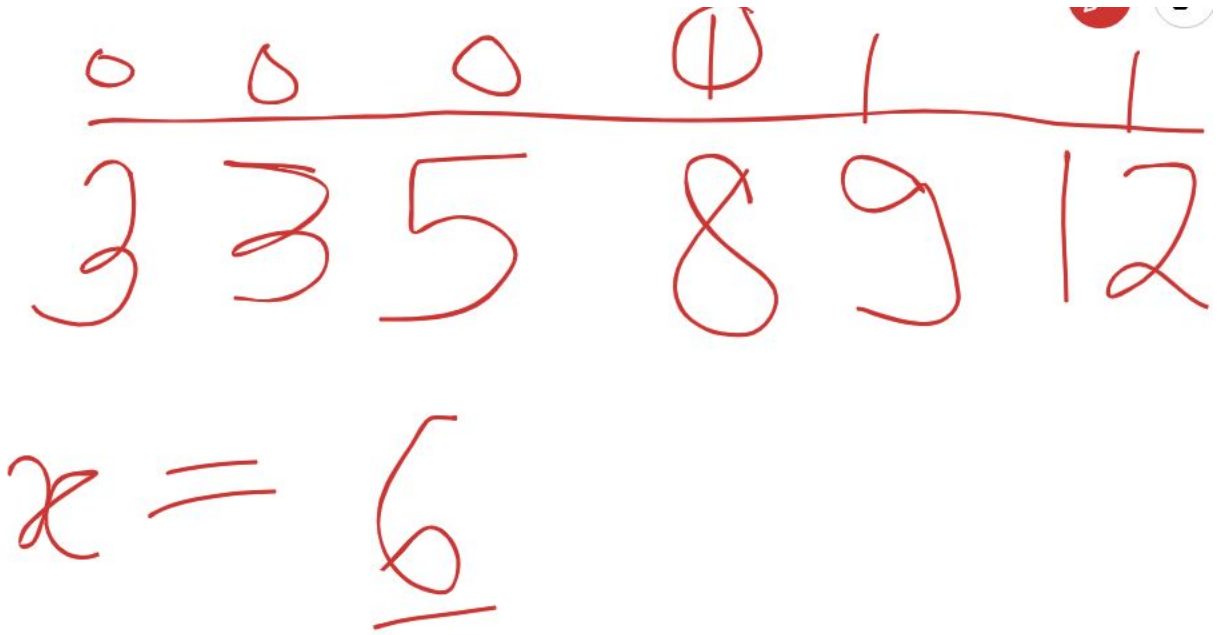
Code:

```
int lo=0,hi=n-1;
int mid;
int ans=-1;
while(lo<=hi)
{
   mid = lo + (hi-lo)/2;
   if (arr[mid] == 0)
```

```
    {
     lo = mid+1;
    }
    else
    {
        ans = mid;
        hi = mid-1;
    }
}
cout<<ans;
```

Q. https://codeforces.com/edu/course/2/lesson/6/1/practice/contest/283911/problem/C

Function : arr[i] >= x (Monotonic Function, since array is sorted)



$x = 6$

```cpp
#include <bits/stdc++.h>

using namespace std;

int main() {
 int n, k;
 cin >> n >> k;

 vector<int> vec(n);

 for (int i = 0; i < n; i++) {
   cin >> vec[i];
 }


 int x;
```

```
 while (k--) {
   cin >> x;
   int lo = 0, hi = n - 1;
   int mid;
   int ans = n;
   while (lo <= hi) {
     mid = lo + (hi - lo) / 2;
     if (vec[mid] < x) {
       lo = mid + 1;
     } else {
       ans = mid;
       hi = mid - 1;
     }
   }
   cout << ans + 1 << '\n';
 }

 return 0;
}
```

HW: (Solve the 4 problems given here)
https://codeforces.com/edu/course/2/lesson/6/1/practice

Q: https://www.spoj.com/problems/EKO/
N trees in a row: h1,h2,...hn
A woodcuter needs m length of wood
A[1...n] -> heights of trees
M
h-> height of the sawblade.

Approach:
Length of wood that mirka gets res= sum (max(0,a[i]-h)) ,1<=i<=n
If (res<m) -> h+1,h+2….. Are not possible
Else -> h can be your ans 1...h-1 are not required
l=0; -> cut all the trees completely
r=1e9; // log(10^9) = 9log10

Sol: n<=10^9
log(n)<log(1e9)
nlog(n)<nlog(1e9)

```
int l=0,r=1e9,ans=0; // nlog(1e9)  n<=10^6 logn<log(1e9)

    while(l<=r){
        int mid = l+(r-l)/2;
        int res=0;
        for(int i=0;i<n;i++){
            if(a[i]>mid){
```

```
            res+=a[i]-mid;
        }
    }
    if(res>=m){ //mid is a possible ans
        ans=mid;
        l=mid+1;
    }else{
        r=mid-1;
    }
}
cout<<ans;
```

You can also think of the problem as :
// Let us say height of sawblade = x
// Target : m metres of wood
// f(x) = 1, when you can get m metres of wood using a sawblade
// of height x and 0, otherwise

x: 0 1 2 ... 7 8 ... 1e9
f(x): 1 1 1… 0 0 …. 0 [Monotonically non-increasing]
We need to find the last value of x for which f(x)=1.

In case of binary search over floating point (decimal) values:

double lo, hi;
const double eps = 0.000001; // or 1e-6;
while (abs(hi-lo) > eps)
{
……

}

More practice problems:

1. Solve all the problems here:
 https://codeforces.com/edu/course/2/lesson/6/2/practice
2. https://www.spoj.com/problems/AGGRCOW/
3. https://codeforces.com/problemset/problem/1195/B
4. https://codeforces.com/problemset/problem/1119/B
5. https://www.spoj.com/problems/NOTATRI/
6. https://atcoder.jp/contests/abc174/tasks/abc174_e
7. https://www.spoj.com/problems/PIE/
8. https://www.spoj.com/problems/HACKRNDM/
9. https://codeforces.com/problemset/problem/760/B
10. https://www.spoj.com/problems/BOOKS1/