**Binary search**

**search space: the domain in which our answer is surely located.**

**The search space is initially the entire sequence. At each step, the algorithm compares the median value in the search space to the target value. Based on the comparison and because the sequence is sorted, it can then eliminate half of the search space. By doing this repeatedly, it will eventually be left with a search space consisting of a single element, the target value and that will lead to our answer.**

## Finding a value in a sorted sequence:

**Let the sequence be:**

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **values** | **0** | **5** | **13** | **19** | **22** | **41** | **55** | **68** | **72** | **81** | **98** |  |
| **index** | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** |  |

**Let we are searching for 55**

**it is sure that it is somewhere between index 0 and 10 , so it is our search space.**

**Let var l=0 and r=10 (our search space)**

**median=(0+10)/2=5,**

**now check if the median value is 55 greater than 55 or smaller than 55.**

**if median value is our key(55)**

**than we found our key. We will exit by printing be found the value.**

**If the median value is smaller than key.**

**as in our example median is 5 and value at index 5 is 41 than is smaller than our key(55). So we are sure that 55 will not be in range of indices 0 to 5. is it?.**

**So now our search space is from index 6 to 10 i.e.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **values** | **55** | **68** | **72** | **81** | **98** |
| **index** | **6** | **7** | **8** | **9** | **10** |

**Now, again calculate the median and check the median value.**

**Median=(6+10)/2=8.**

**If the median value is greater than key.**

**as in our example median is 8 and value at index 8 is 72 than is greater than our key(55). So we are sure that 55 will not be in range of indices 8 to 510 is it?.**

**So again we will reduce our search space**

|  |  |  |
| --- | --- | --- |
| **Values** | **55** | **68** |
| **Index** | **6** | **7** |

**In next step our median value will become (6+7)/2=6.**

**and value at index 6 is 55 so we found our answer.**

lld n;

cin>>n;

lld a[n];

for(int i=0;i<n;i++)

cin>>a[i];

sort(a,a+n);

lld key;

cin>>key;

lld l=0,r=n-1;

while(l<=r)

{

lld mid=(l+r)/2;

if(a[mid]==key)

{

cout<<"yahoo! found";

return 0;

}

if(a[mid]>key)

{

r=mid-1;

}

else

{

l=mid+1;

}

}

cout<<"not found";

**TIME COMPLEXITY OF BINARY SEARCH:**

**Let our search space initially is [1,100];**

**Mid=(100+1)/2=50, so either we will search in left part [1,50] or right part [50,100] point to be noted is search space become half in one operation.**

**In next step, either our domain would be [1,25] or [26,50] or [50,75] or [76,100]**

**So search space become one by fourth in second step.**

**We come out of loop when our search space is reduced to 1;**

**So our complexity would be number of steps to reduce search space to 1, if seach space is reduced to half in each step**

**i.e n -> n/2 -> n/4 -> n/8 -> n/16 …………1.**

**So it is GP with initial term n final term 1. To calculate number of tems apply formula**

**An~~=a(~~r)(n-1)**

**1=n\*(1/2)x-1**

**2(x-1)=n taking log both side we get**

**X= log2(n)+1**

**So our complexity becomes O(log2(n)).**

**Binary search in c++ standard template library:**

* **lower\_bound,**
* **upper\_bound,**
* **binary\_search and equal\_range,**

**it is best to use library function whenever possible. Implementing binary search on own is can be tricky, we will see it now.**

**Variants of binary search:**

**1) Contains (True or False)  
2) Index of first occurrence of a key  
3) Index of last occurrence of a key  
4) Index of least element greater than key  
5) Index of greatest element less than key**

**1) We have all ready discussed the first one how to know weather element exist or not, now we will learn how to shift the boundaries to obtain the answer.**

**2) Index Of first occurrence of key.**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Va- 13** | **14** | **15** | **15** | **16** | **16** | **16** | **17** |
| **In- 0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** |

**Let suppose we need to find the smallest index of 16**

**Domain where answer can be will be index [0,7]**

**L=0 smallest value of domain**

**R=7 largest value of domain.**

**Now, mid = (0+7)/2=3, so check value at index 3 can be in answer or nor so**

**“no” a[3]=15, now check where the answer would be weather left to the mid or right of the mid.**

**So 16 would be right of the mid, so be restrict our domain from index [4,7]**

**i.e mid+1 to r**

**now l=4 , r=7;**

**again calculate mid=(4+7)/2=5; a[5]=16;**

**now this index could be the first index but we are not confirmed, but we come to know our answer would not be any index greater than 5.**

**So again restrict your domain [4,5], r=mid;**

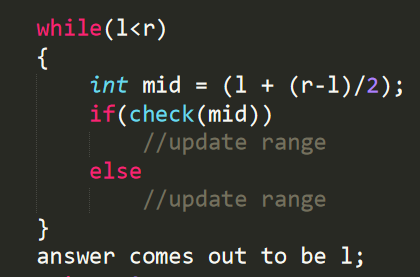
**Again calculate the mid=(4+5)/2=4, so same as above r=4(how?)**

**L==R , so l=4 would be our answer.**

**In same fasion try to implement for the rest parts.**

**Binary Search on real numbers:**

**So here we comes with the pseudo code:**

****

**We need to learn how to update the range and value of mid;**

**Let l=3,r=4;**

**Mid would be 3;**

**So if we update l=mid; then again l will be equal to three and we will get trap in infinite loop, so remember whenever we are updating l=mid , we will calculate mid as we do not get trap in infinite loop.**

**Mid=(l+(r-l+1))/2, so**

**//Try to contradict on your own.**

**Second thing we need to learn how to update the ranges :**

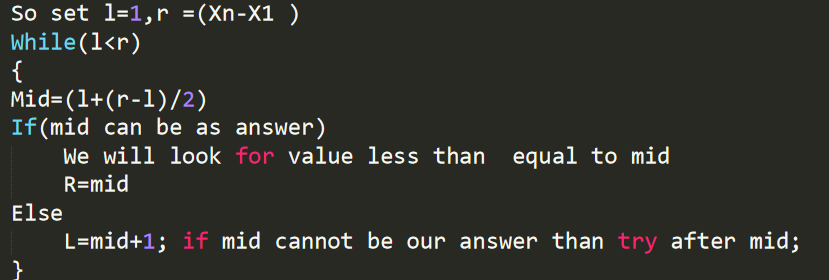
**So it comes by practice:**

**Here are some questions to practice for :**

**Spoj problem** [**AGGRCOW**](https://www.spoj.com/problems/AGGRCOW/)**:**

**So in this question what can be the maximum distance between two cow and minimum distance between cows?**

**Minimum distance 1.,Max distance could be (Xn – X1 )**

****

**L comes out as minimum distance.**

**Binary search on fractional domain(easy to understand )**

[**calculating square root up to fix precision using binary search**](https://www.tutorialspoint.com/find-square-root-of-number-upto-given-precision-using-binary-search-in-cplusplus)

[**spoj .pie**](https://www.spoj.com/problems/PIE/)

**Problem to practice:**

[**google : workout**](https://codingcompetitions.withgoogle.com/kickstart/round/000000000019ffc7/00000000001d3f5b)

[**codeforces div2 c**](https://codeforces.com/contest/689/problem/C)

[**codeforces D1**](https://codeforces.com/contest/670/problem/D1) **and practice more questions from** [**codechef prepare**](https://www.codechef.com/certification/data-structures-and-algorithms/prepare#foundation)**.**

Useful Resources:

[**topcoder**](https://www.topcoder.com/community/data-science/data-science-tutorials/binary-search/)

