Binary Search

Given a sorted array arr[] of n elements, write a function to search a given element x in arr[].

A simple approach is to do [linear search](http://quiz.geeksforgeeks.org/linear-search/)**.**The time complexity of above algorithm is O(n). Another approach to perform the same task is using Binary Search.

**Binary Search:** Search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval to the lower half. Otherwise narrow it to the upper half. Repeatedly check until the value is found or the interval is empty.

The idea of binary search is to use the information that the array is sorted and reduce the time complexity to O(Log n).

We basically ignore half of the elements just after one comparison.

1. Compare x with the middle element.
2. If x matches with middle element, we return the mid index.
3. Else If x is greater than the mid element, then x can only lie in right half subarray after the mid element. So we recur for right half.
4. Else (x is smaller) recur for the left half.

**Recursive**implementation of Binary Search

|  |
| --- |
| // C++ program to implement recursive Binary Search  #include <bits/stdc++.h>  using namespace std;    // A recursive binary search function. It returns  // location of x in given array arr[l..r] is present,  // otherwise -1  int binarySearch(int arr[], int l, int r, int x)  {      if (r >= l) {          int mid = l + (r - l) / 2;            // If the element is present at the middle          // itself          if (arr[mid] == x)              return mid;            // If element is smaller than mid, then          // it can only be present in left subarray          if (arr[mid] > x)              return binarySearch(arr, l, mid - 1, x);            // Else the element can only be present          // in right subarray          return binarySearch(arr, mid + 1, r, x);      }        // We reach here when element is not      // present in array      return -1;  }    int main(void)  {      int arr[] = { 2, 3, 4, 10, 40 };      int x = 10;      int n = sizeof(arr) / sizeof(arr[0]);      int result = binarySearch(arr, 0, n - 1, x);      (result == -1) ? cout << "Element is not present in array"                     : cout << "Element is present at index " << result;      return 0;  } |

**Output :**

Element is present at index 3