**# Binary Search:**

Binary Search is a searching technique, which works on the divide and conquers algorithm and it works only for sorted arrays.

[**Binary Search**](https://www.blogger.com/blog/post/edit/747603084220957601/4999044379100022913) is a searching algorithm. In each step, the algorithm compares the input element x with the value of the middle element in the array. If the values match, return the index of the middle. Otherwise, if x is less than the middle element, then the algorithm recurs for the left side of the middle element, else recurs for the right side of the middle element.

Suppose below is the array in which we want to search the key 53. So initially Low will be 0 and High will be (length - 1) length of below array is 21(number of items in array), So High will be 20.

So Mid = 10; //((Low + High) / 2) as Low is 0 and High is 20, So (0 + 20)/2 = 10

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 5 | 7 | 11 | 13 | 17 | 19 | 23 | 29 | 31 | 37 | 41 | 43 | 47 | 53 | 59 | 61 | 67 | 71 | 73 |

Low Mid High

Now we will compare the key 53 to A[Mid], which is 31 (A[10]) and 31 != 53 and key 53 is greater than 31 (A[Mid]), So our key should be in right of index 10 (Mid) as the array is sorted and key 53 is greater than A[Mid].

So now, we have to update our Low to (Mid + 1), So Low = 10 + 1 = 11;

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 5 | 7 | 11 | 13 | 17 | 19 | 23 | 29 | 31 | 37 | 41 | 43 | 47 | 53 | 59 | 61 | 67 | 71 | 73 |

Mid Low High

Now again calculate the Mid, So Mid = (Low + High)/2 = (11 + 20)/2 = 15,

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 2 | 3 | 5 | 7 | 11 | 13 | 17 | 19 | 23 | 29 | 31 | 37 | 41 | 43 | 47 | 53 | 59 | 61 | 67 | 71 | 73 |

Low Mid High

Now we will compare the key 53 with A[Mid], which is 53.

So Key (53) = A[Mid] (53), So our Key has been found, So now we can say that the Key(53) is found at index 15.

If suppose in the first iteration the key 53 is lesser than the Mid Value, than we have to update the High as High = Mid – 1, As in that case our Key should be in left Index of Mid.

**# Algorithm/Function for Binary Search using while loop:**

int **binarySearchUsingLoop**(struct Array objArr, int iKey)

{

int iHigh = objArr.iLength - 1;

int iLow = 0;

int iMid = 0;

int iReturnIndex = -1;

while (iHigh >= iLow)

{

iMid = floor( ( ( iHigh + iLow ) / 2 ) );

if (objArr.A[iMid] == iKey)

{

iReturnIndex = iMid;

break;

}

else if (iKey < objArr.A[iMid])

{

iHigh = iMid - 1;

}

else

{

iLow = iMid + 1;

}

}

return iReturnIndex;

}

**# Algorithm/Function for Binary Search using recursion:**

int **binarySearchUsingRecursion**(struct Array objArr, int iKey, int iLow, int iHigh)

{

int iMid = 0;

int iReturnIndex = -1;

while (iHigh >= iLow)

{

iMid = floor( ( ( iHigh + iLow ) / 2 ) );

if (objArr.A[iMid] == iKey)

{

iReturnIndex = iMid;

break;

}

else if (iKey < objArr.A[iMid])

{

return binarySearchUsingRecursion(objArr, iKey, iLow, (iMid - 1));

}

else

{

return binarySearchUsingRecursion(objArr, iKey, (iMid + 1), iHigh);

}

}

return iReturnIndex;

}

Note: Full program can be downloaded from the link below:

[https://github.com/VChaudhary854/LearnDataStructureWithVishal/tree/DataStructurePrograms](https://www.blogger.com/blog/post/edit/747603084220957601/4999044379100022913)