

Binary Search Application:

Technical Guide

INTRODUCTION:

It is a Visual Basic.Net application to search for an element in an list using Binary Search Algorithm and show how the Binary Search Algorithm works.

USAGE:

People who want to search for a value in a list of similar values can use this application to get their job done instantaneously. Beginners who are interested in the Binary Search Algorithm, can use this application to see step-by-step process of how the Binary Search Algorithm works on different inputs.

BINARY SEARCH ALGORITHM:

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.


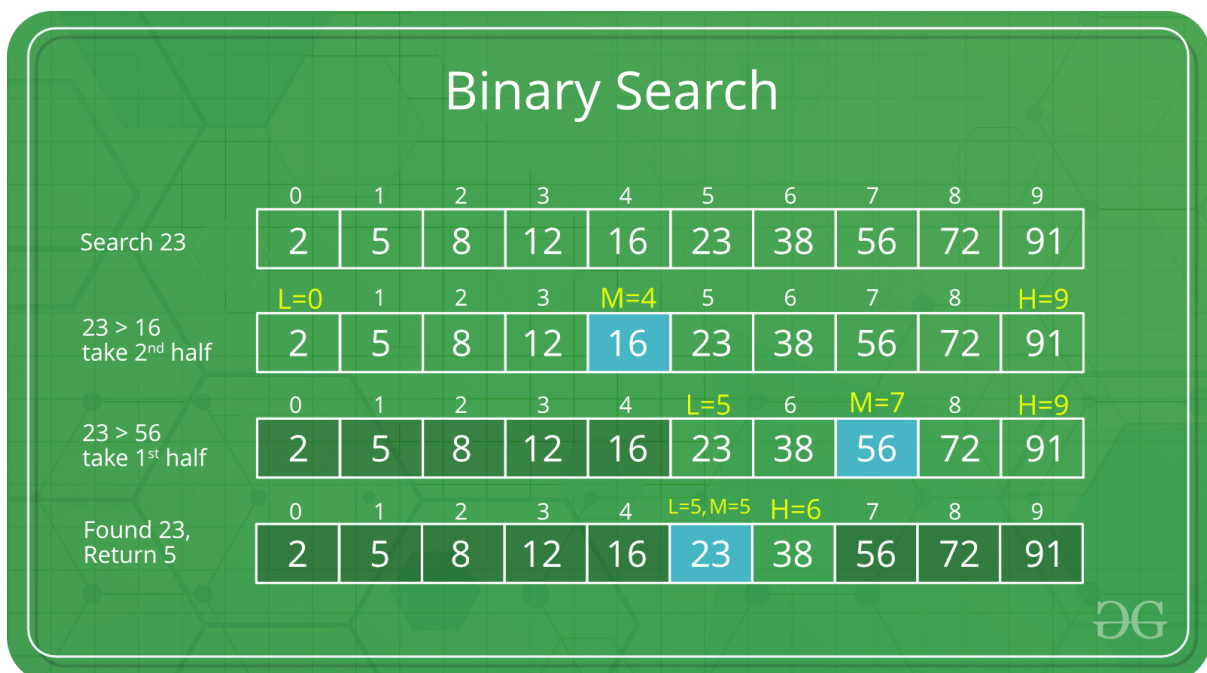
Binary Search

	0	1	2	3	4	5	6	7	8	9
Search 23	2	5	8	12	16	23	38	56	72	91

L=0	1	2	3	M=4	5	6	7	8	H=9	
23 > 16 take 2 nd half	2	5	8	12	16	23	38	56	72	91

	0	1	2	3	4	L=5	6	M=7	8	H=9
23 > 56 take 1 st half	2	5	8	12	16	23	38	56	72	91

	0	1	2	3	4	L=5, M=5	H=6	7	8	9
Found 23, Return 5	2	5	8	12	16	23	38	56	72	91



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.
 4. Else (x is smaller), then x can only lie in the left half.
- The idea of binary search is to use the information that the array is sorted and reduce the time complexity to $O(\log n)$, where 'n' is the number of elements in the array.

For more information about Binary Search, <https://www.geeksforgeeks.org/binary-search/>.

Here is a C++ code of Binary Search Algorithm:

```
int arr[n];
int low=0, high=n-1, mid;
string result = "NOT FOUND";
while ( low < high ) {
    mid = ( low + high ) / 2;
    if ( arr[mid] == x ) {
        low = mid + 1;
        high = mid - 1;
    }
    else if ( arr[mid] > x )
        high = mid - 1;
    else
        low = mid + 1;
}
mid = ( low + high ) / 2;
If ( arr[mid] == x )
    result = "FOUND";
cout<<result<<endl;
```

INTERFACE:

Binary Search

BINARY SEARCH

Enter Value To Be Added In The Array: **ADD**

Number Of Elements In The Array: **CLEAR**

Select Input Text File: **SELECT**
No File Selected

Enter Value To Be Searched: **SEARCH**

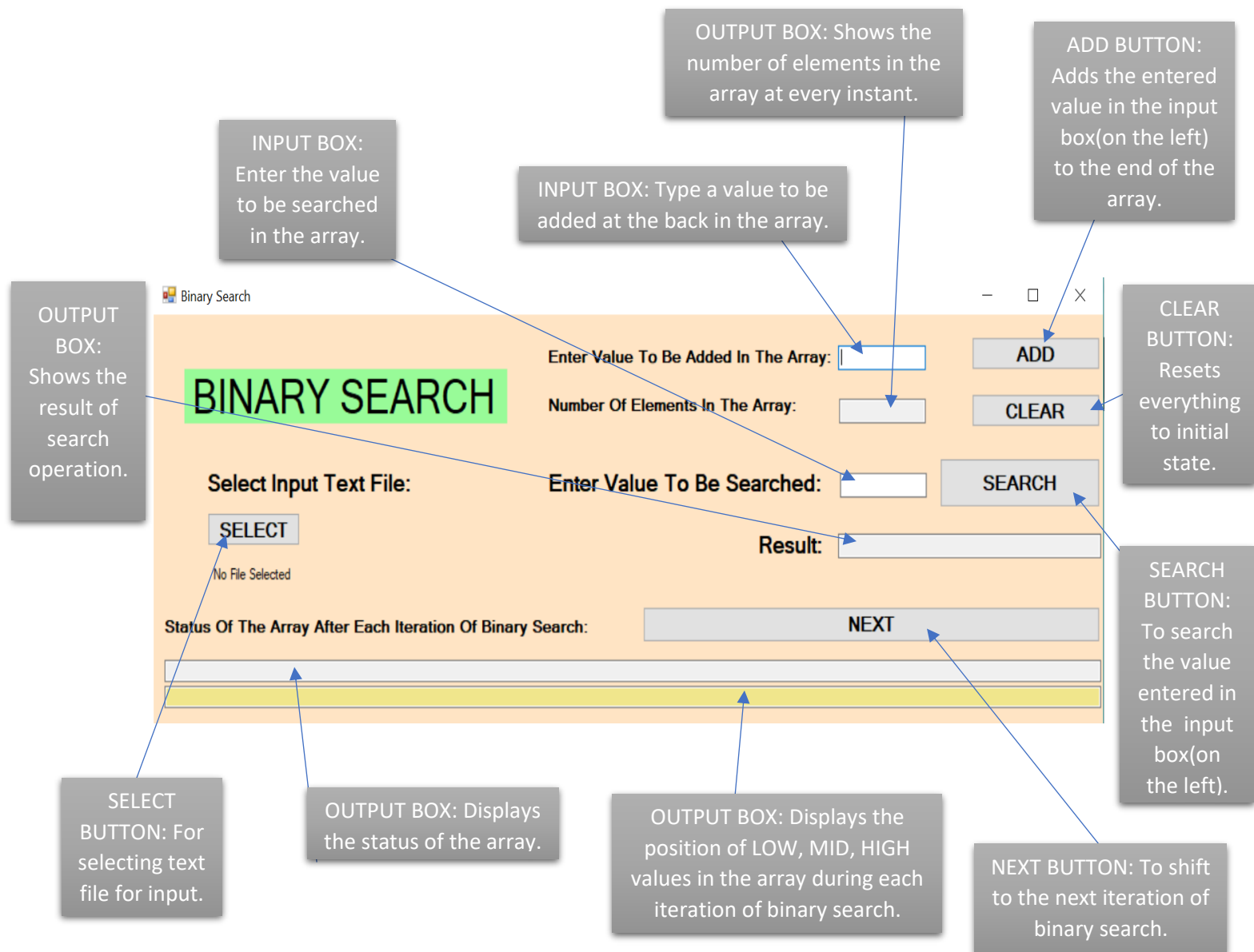
Result:

Status Of The Array After Each Iteration Of Binary Search: **NEXT**

(Window Size: 1459, 568)

As can be seen in the diagram that the application contains three elements:

1. **INPUT BOX:** User can write values in it and the application reads those values.
2. **BUTTONS:** Each button has a specific task and those tasks are performed when the user presses the button.
3. **OUTPUT BOX:** Displays some values. User can not write in an output box, as these are text boxes with “Read-Only” as “TRUE”.



INPUT BOX:

The input boxes can have either a string or a real number. Also, the input should be in sorted order. So, if you try to enter a value which is less than the current last element of the array, then also, a message box appears, warning the user about the invalid input.

CLEAR BUTTON:

This button resets the state of the application, i.e., brings it to the initial state when the array was empty.

OpenFileDialog

The **OpenFileDialog** control prompts the user to open a file and allows the user to select a file to open. The user can check if the file exists and then open it. The OpenFileDialog control class inherits from the abstract class **FileDialog**.

LABELS USED:

Label 1: "Enter Value To Be Added In The Array:"

Label 2: "Number Of Elements In The Array:"

Label 3: "Enter Value To Be Searched:"

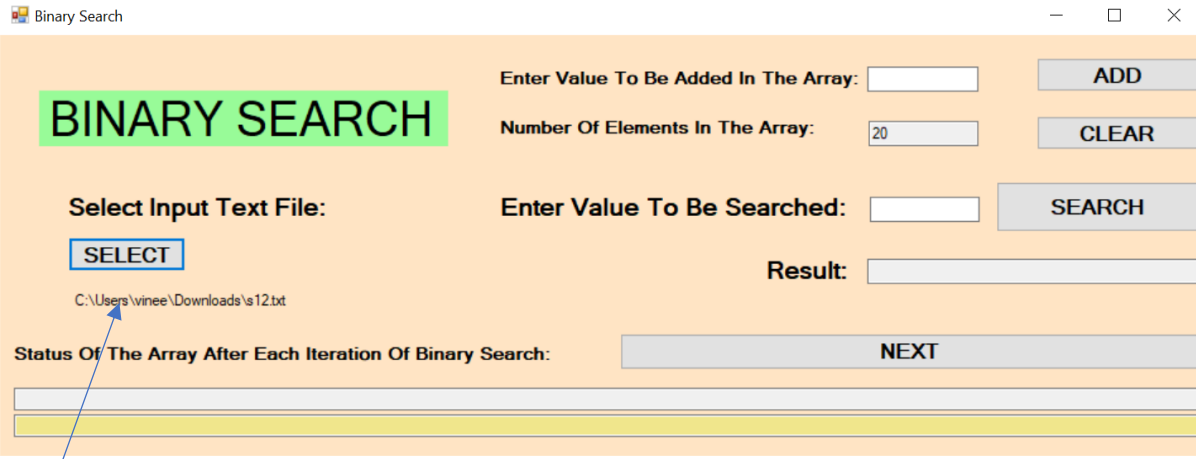
Label 4: "Status Of The Array After Each Iteration Of Binary Search:"

Label 5: "Result:"

Label 6: "Select Input Text File:"

Label 7: "BINARY SEARCH"

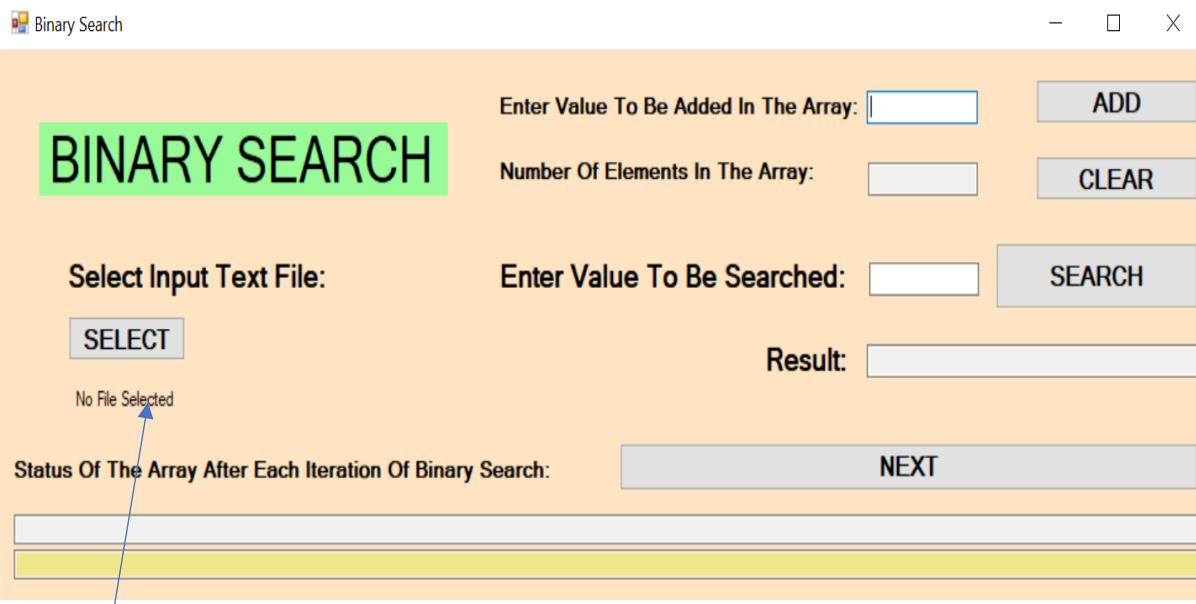
Label 8: Initially, when no file is selected, the label shows “No File Selected”, after you select a file it shows the path to that file.



The screenshot shows a window titled "Binary Search". On the left, a green box contains the text "BINARY SEARCH". Below it, the label "Select Input Text File:" is followed by a "SELECT" button. A text field below the button displays the file path "C:\Users\winee\Downloads\s12.txt". To the right, there are input fields for "Enter Value To Be Added In The Array:" and "Enter Value To Be Searched:", each with an "ADD" or "SEARCH" button respectively. Below these is a "Result:" label and a text field. At the bottom, a label "Status Of The Array After Each Iteration Of Binary Search:" is followed by a "NEXT" button. A blue arrow points from a grey callout box to the "SELECT" button.

This is the file which has been selected for

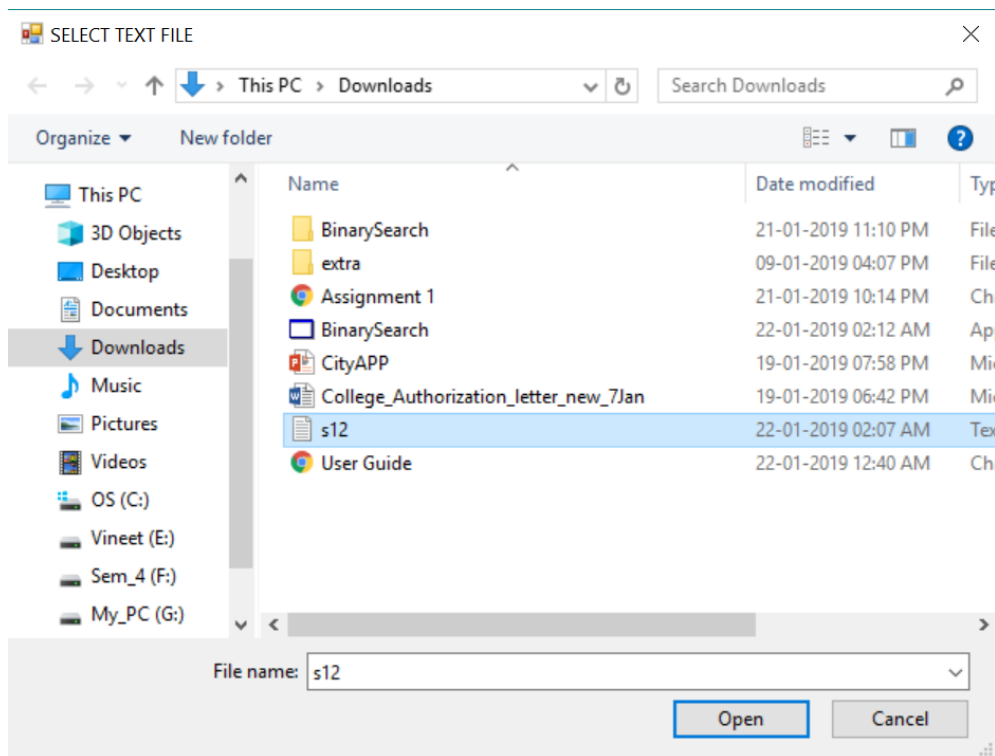
SELECT BUTTON:



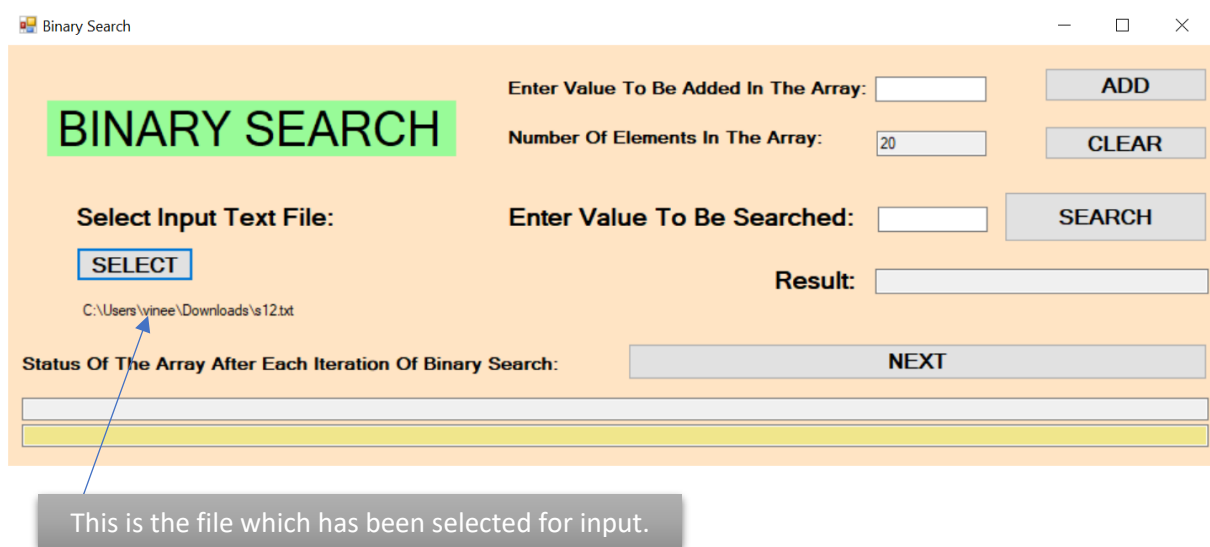
The screenshot shows the same "Binary Search" window, but the text field under "Select Input Text File:" now displays "No File Selected". The other elements, including the "SELECT" button, the input fields for adding values and searching, the "Result:" field, and the "NEXT" button, remain the same. A blue arrow points from a grey callout box to the "SELECT" button.

This shows that no file is selected yet for the input.

When you click on the select button a new window appears:



Now choose the text file with inputs, and click “Open”.



Limitations/Precautions of the Application:

1. Array size cannot be greater than 1,000,000.
2. Input array should be sorted. The program won't work for an unsorted array.
3. Array values should be of same type.
4. Selected file should be a text file, with each array value written on a separate line.
5. The visualization feature will only be helpful for arrays with number of elements ≤ 20 , due to the limitation of the window size.

BINARY SEARCH VISUALIZATION:

This application uses 3 symbols: L , M , H to represent the state after each iteration of binary search.

If search is done for a value present in the array, then the result Output box shows "FOUND", and the L , M , H converges to a single value after some operations.

Binary Search

BINARY SEARCH

Enter Value To Be Added In The Array: **ADD**

Number Of Elements In The Array: **CLEAR**

Select Input Text File: **SELECT**

Enter Value To Be Searched: **SEARCH**

Result:

Status Of The Array After Each Iteration Of Binary Search:

12	23	34	45	56	67	78	89	90	101	110	120	130	140	150	160	170	180	190	200
LMH																			

NEXT

If search is done for a value not present in the array, then the result Output box shows “NOT FOUND”, and the L , M , H never converges to a single value, no matter how many times you press the “NEXT” button.

Binary Search

BINARY SEARCH

Enter Value To Be Added In The Array: **ADD**

Number Of Elements In The Array: **CLEAR**

Select Input Text File: **SELECT**

Enter Value To Be Searched: **SEARCH**

Result:

Status Of The Array After Each Iteration Of Binary Search:

12	23	34	45	56	67	78	89	90	101	110	120	130	140	150	160	170	180	190	200
LM										H									

NEXT