

(A Constituent College of Somaiya Vidyavihar University)

Department of Computer Engineering

Batch:-B-2 Roll No:-16010122151

Experiment No:-2

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Title: Implementation of Binary search/Max-Min algorithm

Objective: To learn the divide and conquer strategy of solving the problems of different types

CO to be achieved:

CO 2 Describe various algorithm design strategies to solve different problems and analyse Complexity.

Books/ Journals/ Websites referred:

- 1. Ellis horowitz, Sarataj Sahni, S.Rajsekaran," Fundamentals of computer algorithm", University Press
- 2. T.H.Cormen ,C.E.Leiserson,R.L.Rivest and C.Stein," Introduction to algorithms",2nd Edition ,MIT press/McGraw Hill,2001
- 3. http://en.wikipedia.org/wiki/Binary_search_algorithm
- 4. https://www.princeton.edu/~achaney/tmve/wiki100k/docs/Binary_search_alg orithm.html
- 5. http://video.franklin.edu/Franklin/Math/170/common/mod01/binarySearchAlg.html
- 6. http://xlinux.nist.gov/dads/HTML/binarySearch.html
- 7. https://www.cs.auckland.ac.nz/software/AlgAnim/searching.html

Pre Lab/ Prior Concepts:

Data structures

Historical Profile:

Finding maximum and minimum or Binary search are few problems those are solved with the divide-and-conquer technique. This is one the simplest strategies which basically works on dividing the problem to the smallest possible level.

Binary Search is an extremely well-known instance of divide-and-conquer paradigm. Given an ordered array of n elements, the basic idea of binary search is that for a given element, "probe" the middle element of the array. Then continue in either the lower or upper



(A Constituent College of Somaiya Vidyavihar University)

Department of Computer Engineering

segment of the array, depending on the outcome of the probe until the required (given) element is reached.

New Concepts to be learned:

Number of comparisons, Application of algorithmic design strategy to any problem, Classical problem solving Vs Divide-and-Conquer problem solving.

```
Algorithm IterativeBinarySearch
int binary_search(int A[], int key, int imin, int imax)
//The algorithm takes as parameters an array A[1...n], the search key and lower-higher index
pair of the array.
// Output- The algorithm returns index of the search key in the given array, if it's present.
 // continue searching while [imin, imax] is not empty
 WHILE (imax >= imin)
           // calculate the midpoint for roughly equal partition
           int imid = midpoint(imin, imax);
           \mathbf{IF}(A[imid] == key)
              // key found at index imid
              return imid;
             // determine which subarray to search
           ELSE If (A[imid] < key)
              // change min index to search upper subarray
              imin = imid + 1:
             ELSE
              // change max index to search lower subarray
              imax = imid - 1;
 // key was not found
 RETURN KEY_NOT_FOUND;
 The space complexity of Iterative Binary Search:-
```

Code for Binary search Iterative Method:-

```
#include <iostream>
using namespace std;
int main() {
   cout << "Enter the size of the sorted array: ";
   int size;
   cin >> size;
   int sorted_array[size];
```

(A Constituent College of Somaiya Vidyavihar University)

```
cout << "Enter the sorted array elements: ";</pre>
for (int i = 0; i < size; ++i)</pre>
    cin >> sorted_array[i];
cout << "Enter the target value: ";</pre>
int target_value;
cin >> target_value;
int start = 0;
int end = size - 1;
int result = -1;
while (start <= end) {</pre>
    int mid = start + (end - start) / 2;
    if (sorted_array[mid] == target_value) {
        result = mid;
        break;
    } else if (sorted_array[mid] < target_value)</pre>
        start = mid + 1;
    } else
        end = mid - 1;
}
if (result != −1)
    cout << "Found "<< endl;</pre>
} else
{
    cout << " Not found " << endl;</pre>
return 0;
```

Output:-

For Iterative Method:-

```
Enter the size of the sorted array: 5
Enter the sorted array elements: 1 2 3 4 5
Enter the target value: 5
Found
```

```
Enter the size of the sorted array: 5
Enter the sorted array elements: 1 2 3 4 5
Enter the target value: 6
Not found
```

For Recursive Method:-

```
#include <iostream>
using namespace std;
int binary_search_recursive(int arr[], int target, int start, int end)
{
    if (start > end)
    {
        return -1;
    }

    int mid = start + (end - start) / 2;

    if (arr[mid] == target)
    {
        return mid;
    }
    else if (arr[mid] < target)
    {
        return binary_search_recursive(arr, target, mid + 1, end);
    }
    else
    {
        return binary_search_recursive(arr, target, start, mid - 1);
}</pre>
```

(A Constituent College of Somaiya Vidyavihar University)

```
}
int main() {
    cout << "Enter the size of the sorted array: ";</pre>
    int size;
    cin >> size;
    int sorted_array[size];
    cout << "Enter the sorted array elements: ";</pre>
    for (int i = 0; i < size; ++i)</pre>
        cin >> sorted_array[i];
    }
    cout << "Enter the target value: ";</pre>
    int target_value;
    cin >> target_value;
    int result = binary_search_recursive(sorted_array, target_value, 0, size -
1);
    if (result != -1)
        cout << "Target " << target_value << " found at index " << result <<</pre>
endl;
    } else
    {
        cout << "Target " << target_value << " not found in the array" << endl;</pre>
    return 0;
```

Output:-

For Recursive Method:-

Enter the size of the sorted array: 5
Enter the sorted array elements: 1 2 3 4 5
Enter the target value: 5
Target 5 found at index 4

Enter the size of the sorted array: 5
Enter the sorted array elements: 1 2 3 4 5
Enter the target value: 45
Target 45 not found in the array

Hydes Psessaglu 16 Dien 7749
Binady Scaleti 2000 HANNIA
Hexative method:
end= n-1; mil, end; end= n-1; mil, end; while (start L= end)
id (ass Cmid) == key)
(out al "Yes"; o it (all (mid) L key)
Start = mid +2; g else (1) 0 = $\frac{1}{\sqrt{1+x^2/9}}$ may?
end= mid-1;
Time complexity:- a) Best case: O(1)
Assuming mid is the ney

-	
	Contract of the second of the
h)	AMBAR (OSE: -0(1005 U)
	Assuming trey is at start of end
()	Aug (ase : o(log2n)
	tray is bet start at mid or bet mid bend
	(200) (200) (600) (610)
	50 we have seatch hald
	(1217) (1210) Sie) si
	50 0 ([0420)
	Schling hanhe
-	Space complexity = o(i)
	it - line 1503
	i fixalque mil
	a) Best case: 0(1)
	you got de Gin prima 22 A



Algorithm Recursive Binary Search

```
int binary_search(int A[], int key, int imin, int imax)
//The algorithm takes as parameters an array A[1...n], the search key and lower-higher index
pair of the array.
// Output- The algorithm returns index of the search key in the given array, if it's present.
 // test if array is empty
 IF (imax < imin)
  // set is empty, so return value showing not found
  RETURN KEY_NOT_FOUND;
 ELSE{
           // calculate midpoint to cut set in half
           int imid = midpoint(imin, imax);
           // three-way comparison
           \mathbf{IF}(A[imid] > key)
              // key is in □ lower subset
              RETURN binary_search(A, key, imin, imid-1);
           ELSE IF (A[imid] < key)
              // key is in ☐ higher subset
              RETURN binary_search(A, key, imid+1, imax);
           ELSE
              // key has been found
              RETURN imid;
          }
```

The space complexity of Recursive Binary Search:-

	Department of Computer Engineering
7777	SHI SEE
Mer	
1	
	Open
	Recarsive approach
	use (do sout)
	Int algos binary ((ass (7, Aust, end, reg)
	a start end, they
	is (Start > end)
	1
	80448000
	9
	0150
	else mid = Start + (end - Start) /2;
	W101 - 21947 + (6U) - 21047/15
	18 (hee = = add (mjd))
	g
	(vut cc en " ye ("-
	4
	1 (1 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 - 1 -
	id (key Lard Cmid)
	200 970 198 14
	at 90 ting
	Cart 10 1911CF
	al qu binasys (abbit 7, bry, start, mp-1)
	al que bioadys (addic7, bey, start, mid-1)
	else
	else
	else algo binasys (assil bey, start, m.d.) algo binasy (assil bey, midt 1 end)
	else algo binaby (abb(1, bey, start, m.d.) else algo binaby (abb(1, bey, midt 1 end)
	else algo binaby (abb(), bey, start, m.d.) else algo binaby (abb(), bey, midt) end)
	else algo binaby (abb(1, bey, start, m.d.) else algo binaby (abb(1, bey, midt 1 end)
	al qu binaby (abb(), bey, start, m.d.) else al qu binaby (abb(), bey, m.d.) graphing (umplexity
	al qu binaby (abb(), bey, start, m.d.) else al qu binaby (abb(), bey, m.d.) graphing (umplexity
a)	al qu binaby (abb(), bry, start, m.d.) else al qu binaby (abb(), bry, m.d.) graphing (bry) Time (umplexity)
a)	algo binasys (assill bry, start, m.s.) else algo binasy (assill, bry, m.s.) rime (umplexity A wosst :- o(logen)
a)	algo binary (abb(), bey, start, m.d.) else algo binary (abb(), bey, midtlend) grow (umplexity A word (to grow) In securcive method we to calculate
a)	algo binary (abb(), bey, start, m.d.) else algo binary (abb(), bey, midtlend) grow (umplexity A word (to grow) In securcive method we to calculate
a)	algo binasys (assill bry, start, m.s.) else algo binasy (assill, bry, m.s.) rime (umplexity A wosst :- o(logen)
a)	algo binary (abb(), bey, start, m.d.) else algo binary (abb(), bey, midtlend) grow (umplexity A word (to grow) In securcive method we to calculate
a)	algo binary (abb(), bey, start, m.d.) else algo binary (abb(), bey, midtlend) grow (umplexity A word (to grow) In securcive method we to calculate
a)	algo binary (abb(), bey, start, mb-1) else algo binary (abb(), bey, mid+1 end) rime (umplexity A word sto :- 6 (logon) In securcive method we to calculate

Time complexity of Binary Search:-

The

Octo .
(Charles)
Master method.
a=1, 5=1, H=0, q=0
1pg 21=0=1h=0, q=0
To The section
T(n) = 0 (nt luy")
= Q(n° (vg n)
=0((oqin)
=0(109(1))
1 Mandard 1 3 d 1 5
b) Be4 (ase = O(1)
when begis mid
- () Avg (ase :- 60 (lpg, n)
The self of the transfer of the self of th
Depends on how many times up
can the function.
Rus
(Mpa) 3 = 3 043 624 M 21
5 3 (10 10) at 90 by do no 9412 4000 at
le son enper de de fingens



K. J. Somaiya College of Engineering, Mumbai-77 (A Constituent College of Somaiya Vidyavihar University)

Department of Computer Engineering

Algorithm StraightMaxMin:

```
VOID StraightMaxMin (Type a[], int n, Type& max, Type& min)
// Set max to the maximum and min to the minimum of a[1:n].
\{ \max = \min = a[1]; \}
       FOR (int i=2; i<=n; i++)
              IF (a[i]>max) then max = a[i];
                      IF (a[i] < min) min = a[i];
}
 Algorithm: Recursive Max-Min
VOID MaxMin(int i, int j, Type& max, Type& min)
// A[1:n] is a global array. Parameters i and j are integers, 1 \le i \le j \le n.
//The effect is to set max and min to the largest and smallest values in a[i:i], respectively.
     IF (i == j) \max = \min = a[i]; // Small(P)
     ELSE IF (i == j-1) { // Another case of Small(P)
           IF (a[i] < a[j])
               max = a[j]; min = a[i];
           ELSE { max = a[i]; min = a[i];
       ELSE {
                   Type max1, min1;
 // If P is not small divide P into sub problems. Find where to split the set.
          int mid=(i+j)/2;
         // solve the sub problems.
         MaxMin(i, mid, max, min);
         MaxMin(mid+1, j, max1, min1);
       // Combine the solutions.
        IF (max < max1) max = max1;
        IF (min > min1) min = min1;
```

Code for MaxMin Iterative Method:-

```
#include <iostream>
using namespace std;
```

(A Constituent College of Somaiya Vidyavihar University)

```
int main() {
    cout << "Enter the size of the array: ";</pre>
    int size;
    cin >> size;
    int arr[size];
    cout << "Enter the array elements: ";</pre>
    for (int i = 0; i < size; ++i)</pre>
        cin >> arr[i];
    }
    int max_value = arr[0];
    int min_value = arr[0];
    for (int i = 1; i < size; ++i)</pre>
        if (arr[i] > max_value)
         {
             max_value = arr[i];
         if (arr[i] < min_value)</pre>
             min_value = arr[i];
         }
    }
    cout << "Maximum value in the array: " << max_value << endl;</pre>
    cout << "Minimum value in the array: " << min_value << endl;</pre>
    return 0;
```

```
Enter the size of the array: 5
Enter the array elements: 10 20 65 90 100
Maximum value in the array: 100
Minimum value in the array: 10
```

```
#include <iostream>
using namespace std;
```

(A Constituent College of Somaiya Vidyavihar University)

```
void find_max_min_recursive(int arr[], int start, int end, int& max_value, int&
min_value) {
    if (start == end)
        max_value = arr[start];
        min_value = arr[start];
        return;
    }
    int mid = start + (end - start) / 2;
    int max_left, min_left, max_right, min_right;
    find_max_min_recursive(arr, start, mid, max_left, min_left);
    find_max_min_recursive(arr, mid + 1, end, max_right, min_right);
    max_value = max(max_left, max_right);
    min_value = min(min_left, min_right);
int main()
 {
    cout << "Enter the size of the array: ";</pre>
    int size;
    cin >> size;
    int arr[size];
    cout << "Enter the array elements: ";</pre>
    for (int i = 0; i < size; ++i)</pre>
   {
        cin >> arr[i];
    }
    int max_value, min_value;
    find_max_min_recursive(arr, 0, size - 1, max_value, min_value);
    cout << "Maximum value in the array: " << max_value << endl;</pre>
```

(A Constituent College of Somaiya Vidyavihar University)

Department of Computer Engineering

```
cout << "Minimum value in the array: " << min_value << endl;
return 0;
}</pre>
```

Enter the size of the array: 5

Enter the array elements: 10 29 65 89 90

Maximum value in the array: 90 Minimum value in the array: 10

AND	
Hydre Paessada	
160101711 Domes	
Mio 1. May	
Min Comay	2
Myc	1
1) Iterative approach	
	1
min max (ac], max, min)	-
	1
$\frac{1}{1}$ (mdx = min = $\sigma(2)$)	-
for (i=2; (=0;)+1);	1
	1
id (a[i]>max).	
maxal max = a(j');	7
· Se Jan X & M	-
it (aCi] (max)	-
4 -9	-
min = a(i);1/-	7
4	-
Cago = voin	
TIDES NO	
Time (umplexity:-	1
1 2 1 1 2 11/2 1	
of Best = Avg = Wodst = O(n) becas are	-
have to seasin the whole adday	-
: any ciays sim is vomina	-
Monthax (mild) & max2, mino?	-
Coale (Ilm devidu - aci)	-
Spale (mapexity = o(1)	-
(Il cian) of (cian) o) bi	PIECE
	STATE OF THE PARTY
COINE OWN	
21.7	k



Time complexity for Max-Min:-

11 12/14
Was Wall Wheels
Page 2
Recappie appliant:
3794
Algo minmax (a[], 1, h, max, min).
ix (1==b) on o 100 volo o 100
max=min =aCD;
0 6
else it (b-1==1), 1
id (a(1))=a(h)),
6 A 6 2 6
max = a(1);
min = a(6);
else
d (:):
Max = a(b);
min > qC17;
g: white grant soul
else
mid: (14h)/2:
minmax (1, mid, max); minmax (midtl, b, max1, min1); ik (a (max) c a (max 1))
minmax (midtl. b, max1, min1);
is la [max 7 c a [max 2]
(1) of touch and area
may = max 2',
id (a(min) > a (min 1)
min zmin z
14 14

(A Constituent College of Somaiya Vidyavihar University)

Department of Computer Engineering
373743646
Time (omplexity:
of Best Case = worst = ang
$\frac{2+(3)+5}{2}$
$f(n) = o\left(3n - 1\right)$
= O(n)
Spare
6 (109E) Pau

CONCLUSION:- Binary Search, the iterative way is memory-efficient but less intuitive, while the recursive way is more readable but can use more memory. In finding max and min, the iterative method is straightforward and efficient, while the recursive approach, though elegant, might use more memory, especially for large datasets. The decision depends on readability, familiarity, and efficiency for the specific problem and data size.