1.Consider the following sequence of input 2,4,6,8,10,12,14,16,19,25. Select the searching algorithm (to search the element 8 in the given sequence) in such a way that it should find the element with minimum searching time (no. Of comparisons). Find the space and time complexity of your choice (of algorithm).

Since the given array is sorted, we will employ the **Binary Search Algorithm.**

Size of array= 10

Number to be searched =8

In the binary search algorithm, we compare z (value to be searched) with the middle element. If it is greater than the middle value, search the last half of the array. Otherwise, search the first half.

# Binary Search Algorithm:

* Step 1: Initialize beg=lower bound, end=upper bound
* Step 2: Repeat Steps 3 and 4 while beg<=end
* Step 3: Initialize mid = (beg+end)/2
* Step 4: IF arr[mid] = z (number to be searched) Return mid

Go to Step 6

ELSE IF arr[mid] < z beg = mid + 1

ELSE

end = mid - 1 [END OF IF] [END OF LOOP]

* Step 5: IF result = -1

PRINT "ELEMENT NOT FOUND" [END OF IF]

* Step 6: ELSE PRINT “Z FOUND AT INDEX” result
* Step 7: EXIT

# Binary Search Program:

#include <iostream> using namespace std;

int binarySearch (int arr[], int z, int beg, int end)

{

while (beg <= end)

{

int mid =(beg+end) / 2; if (arr[mid] == z)

return mid;

else if (arr[mid] < z) beg = mid + 1;

else

end = mid - 1;

}

return -1;

}

int main(void)

{

int arr[] = {2,4,6,8,10,12,14,16,19,25};

int z = 8;

int n = sizeof(arr) / sizeof(arr[0]);

int result = binarySearch(arr, z, 0, n - 1); if (result == -1)

cout<<”Element Not found”;

else

}

cout<<z<<” “<<”found at index”<<” “<<result;

# Time Complexity:

Let us assume that the loop runs for k iterations. In binary search we divide the array into two halves after each iteration. So, the question is, how many times can you divide N by 2 until you have 1? This means that we continue to do binary search (half the elements) until we find the number or reach the end of the array i.e.

1 = N / 2k multiply by 2k:

2k = N

Taking log2 both sides:

log2(2k) = log2 N k \* log2(2) = log2 N

k \* 1 = log2 N

**So the time complexity is O(log N).**

# Space Complexity:

Binary search requires three pointers to elements, which may be array indices or pointers to memory locations, regardless of the size of the array. These are the “lower bound of array” i.e. l, the “ upper bound of array” i.e. u and the “middle element pointer” i.e. mid. Thus the space is constant and **the space complexity of binary search is O(1).**