Bansilal Ramnath Agarwal Charitable Trust's
Vishwakarma Institute of Information Technology

Department of
Artificial Intelligence and Data Science

Name: Siddhesh Dilip Khairnar		
Class: TY	Division: B	Roll No: 372028
Semester: V	Academic Year: 2023-2024	
Subject Name & Code: Design and Analysis of Algorithm: ADUA31202		
Title of Assignment: Write a program to perform binary search on an unsorted random list of at least 5000 elements. The key element should be user input. Use the Divide & Conquer method to implement this program.		
Date of Performance: 04-08-2023	Date of Submission: 11-08-2023	

ASSIGNMENT NO. 1

Theory:
Binary search is a fast search algorithm with run-time complexity of O(log n). This search algorithm works on the principle of divide and conquer. For this algorithm to work properly, the data collection should be in the sorted form.
Binary search looks for a particular item by comparing the middle most item of the collection. If a match occurs, then the index of item is returned. If the middle item is greater than the item, then the item is searched in the sub-array to the left of the middle item. Otherwise, the item is searched for in the sub-array to the right of the middle item. This process continues the sub-array as well until the size of the subarray reduces to zero.

How Binary search works?

1) **Divide and Conquer Principle:**
   Binary search is based on the "divide and conquer" approach, which means it breaks down a larger problem into smaller subproblems until a solution is found. In the case of binary search, the algorithm repeatedly divides the search space in half, eliminating half of the remaining elements at each step, thus significantly reducing the number of elements to search through.

2) **Preconditions:**
   1. The collection of data must be sorted in ascending (or descending) order. This is crucial because binary search relies on comparing elements to decide whether to search the left or right half.
   2. The data structure should allow for efficient random access to elements, such as arrays or lists with indexed access.

3) **Searching Process:**
   a. **Initialization:**
      1. Initially, you have the entire sorted collection as your search space.
      2. You maintain two pointers: one pointing to the start of the current search space and the other pointing to the end.

   b. **Comparison and Subdivision:**
      1. Calculate the middle index of the current search space: (start + end) / 2.
      2. Compare the element at the middle index to the target element you're searching for.
      3. If the middle element matches the target, you've found it and can return the index.
      4. If the middle element is greater than the target, this means the target must be in the left half, so you update the 'end' pointer to be the middle index minus one.
      5. If the middle element is less than the target, the target must be in the right half, so you update the 'start' pointer to be the middle index plus one.

   c. **Iteration or Termination:**

1. The process is repeated until the search space is reduced to zero (start becomes greater than end), indicating that the target element is not present in the collection.
2. If the target is found, its index is returned. If not, the algorithm terminates with a signal that the target is not in the collection.

Time Complexity:

The key reason for binary search's efficiency is its logarithmic runtime complexity, which is O(log n). With each iteration, the search space is effectively halved, leading to rapid convergence even with large datasets. This is in stark contrast to linear search (O(n)), where each step eliminates only one element from consideration.

Page No.
Date / /

DAA Assignment no 1

* Problem statement : →
write a program to perform binary search on an unsorted random array of atleast 5000 element. The key element should be user input. use divide & conquer method to implement program

* Objective : →
1) To learn about divide & conquer method.
2) To learn about binary search algorithm.
3) Analze the binary search algorithm.
4) Perform binary search algorithm using divide & conquer method.

* Algorithm : →
1) Calculate the mid element
   mid = (low + high)/2
2) Compare the mid element with key element.
(i) If key = mid, return mid.
(ii) if key > mid element, set low = mid + 1 & go back to step 1.
(iii) if key < mid element, set high = mid - 1 & go back to step 1.
3) Return -1 indicating that the key element is not present inside array

* Pseudocode : →
```
static int binsearch(int [] arr, int key) {
        int first = 0.
        int last = arr.length - 1
        while (first <= last) {
            int mid = (first + last)/2
```

```
        if (key = arr[mid]) : return mid
        else if (key > arr[mid]) : first = mid+1
        else : last = mid-1
        }
    return -1
}
```

* Example:

Consider array arr[] = {2, 5, 8, 12, 16, 23, 38, 56, 72, 91} & key = 23

i) $mid = \frac{0+9}{2} = 4$

arr[mid] = 16

key > arr[mid] => low = mid+1 = 5

ii) $mid = \frac{5+9}{2} = 7$

arr[mid] = 56 > key

∴ high = mid-1 = 6

iii) $mid = \frac{5+6}{2} = 5$

arr[mid] = 23 = key ⇒ return mid

∴ key found at 5^{th} index

~~Conclusion~~

Conclusion: We studied & implemented Binary search using divide & conquer method.

OUTPUT:

```
#include<iostream>
using namespace std;
int main()
{
    int i, num, first, last, middle;
    int x, y,size, temp;
    int sz;

    cout<<"Enter the size of array::";
    cin>>sz;
    int randArray[sz];
    int arr[5000];
    for(int i=0;i<sz;i++)
        randArray[i]= 1 + rand()%5000; //Generate number between 0
to 5999


    cout<<"The generated array : ";
    for(int i=0;i<sz;i++)
        cout<<randArray[i]<<" ";

    for(int x=0;x<sz;x++)
        arr[x]=randArray[x];


    //Ascending order
    for (x = 0; x < sz; x++){
        for (y = x; y < sz; y++){
            if (arr[x] > arr[y+1]){
                temp = arr[x];
                arr[x] = arr[y+1];
                arr[y+1] = temp;
            }
        }
    }

    //Output
    cout <<"\n\nElements sorted in the ascending order are : ";
    for (x = 1; x <= sz; x++){
        cout << arr[x]<<" ";
    }


    cout<<"\n\nEnter Element to be Search: ";
    cin>>num;
    first = 0;
    last = sz-1;
```

```
    middle = (first+last)/2;
    while(first <= last)
    {
        if(arr[middle]<num)
            first = middle+1;
        else if(arr[middle]==num)
        {
            cout<<"\nThe number, "<<num<<" found at Position
"<<middle;
            break;
        }
        else
            last = middle-1;
        middle = (first+last)/2;
    }
    if(first>last)
        cout<<"\nThe number, "<<num<<" is not found in given Array";
    cout<<endl;
    return 0;
}
```