**INTERNAL PRACTICAL EXAM**

**AIM:**

Design and implement searching algorithm to find given word from English dictionary using minimum number of comparisons. Also find out time complexity of algorithm.

**NOTE**:

* **FOR SIMPLICITY, I HAVE TAKEN THE DICTIONARY AS AN ARRAY WITH 22 ENTRIES.**
* **WE WILL USE ITERATIVE BINARY SEARCH ALGORITHM TO SOLVE THE PROBLEM.**
* **I AM USING ITERATIVE APPROACH AS NO APPROACH IS SPECIFIED.**

**PROGRAM CODE:**

//PERFORMED BY PARTH NITESHKUMAR PATEL

//ID: 19DCS098

//INTERNAL PRACTICAL EXAM OF [CS351] DAA

#include<iostream>

using namespace std;

int counter=0; //COUNTER VARIABLE TO COUNTER THE NUMBER OF PASSES

//binarySearch() is the main logic

int binarySearch(string arr[], string x, int n) {

   int lower = 0;

   int upper = n - 1;

   while (lower <= upper){

       counter++;

      int mid = lower + (upper - lower) / 2;  //FINDING THE MID ELEMENT

      int res;

      if (x == (arr[mid]))

         res = 0;

      if (res == 0)

         return mid;

      if (x > (arr[mid]))

         lower = mid + 1;

      else

         upper = mid - 1;

   }

   return -1;

}

int main () {

    // DICTIONARY

   string arr[] = {"America","Bangladesh","Chandigarh","Denmark","Egypt","France","Hungary","Germany",

   "India","Japan","London","Monaco","Nepal","Ottawa","Peru","Russia","Serbia","Tanzania",

   "Toronto","Uruguay","Venezuala","Zambia"};

   string name="";

    // TAKING THE INPUT

   cout<<"Enter the word you want to search : ";

   cin>>name;

   int n=sizeof(arr)/sizeof(arr[0]);

   int result = binarySearch(arr, name, n);

   if(result == -1)

      cout<<"ENTERED ELEMENT NOT PRESENT IN THE DICTIONARY"<<endl;

   else

      cout<<"YES, THE WORD EXISTS"<<endl;

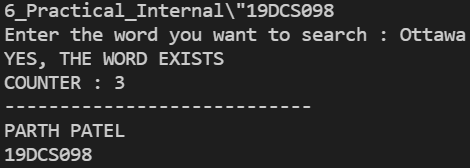
    cout<<"COUNTER : "<<counter<<endl;

    cout<<"----------------------------"<<endl;

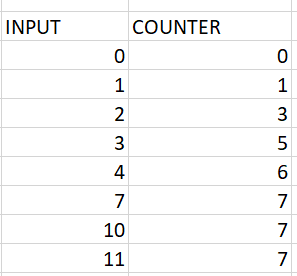
    cout<<"PARTH PATEL\n19DCS098"<<endl;

}

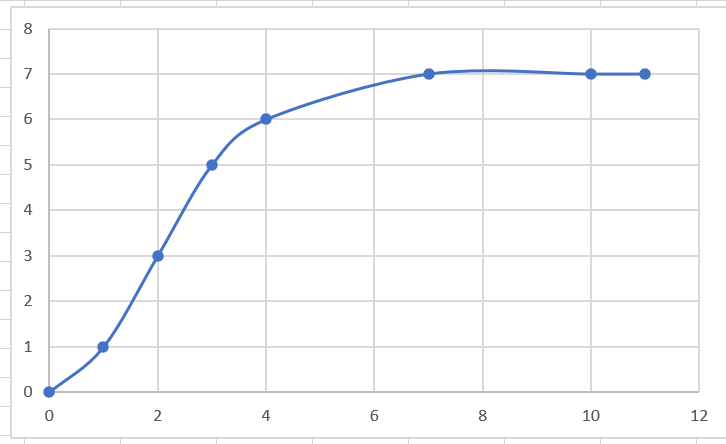
**OUTPUT:**



**TABLE:**



**GRAPH:**



**CONCLUSION:**

From the above graph, it is clear that :

* BEST CASE TIME COMPLEXITY OF BINARY SEARCH IS : O(1)
* WORST CASE TIME COMPLEXITY OF BINARY SEARCH IS : O(logN).