**PRACTICAL # 02**

**OBJECT:**

Run-time complexity comparison of linear search and binary search algorithms

**THEORY:**

The world is full of huge data. There is massive data even in invisible living cells. To process such enormous amounts of data, we need very heavy processing units and efficient algorithms to solve them.

The linear search algorithm is an algorithm designed to process a list of unsorted random data and find some required element in it. The number of steps required to process the data in worst case increases linearly with the increase in the size of the data or the number of elements in the list to process. This makes the worst case complexity of the algorithm of the order O(N), where N is the number of data elements to process.

The binary search is a searching algorithm that can search through data list in a much more efficient way than that of the linear search algorithm. But this algorithm has a condition to work properly. It requires array to be sorted to work properly.

The algorithm checks first and last element of the array. If number is not found, it finds the middle index of the array be taking mean of the upper and lower indexes. If the number to search is less than the middle array element, the number can be found in the lower sub-array with respect to the middle element of the array. This is because array is sorted so all elements in the upper half of the array would be greater than this element. Else the element can be in the upper sub-array. The array upper or lower index would be updated according to the case. This process continues until the element is found or the array is completely searched. Since the algorithm divides the array to half in each iteration, the resulting worst case run-time complexity is O(log N).

Comparing the complexity of linear search algorithm, O(N) with binary search algorithm O(log N), it shows that binary search algorithm significantly outperforms the linear search algorithm. This is one way of getting high performance that is by designing a more efficient algorithm to solve the same problem in less number of steps.

**CODE:**

The program below defines the two algorithms, the linear search and binary search in C++. The program calculates number of iterations executed in each of the algorithm and their individual execution time. The results show the efficiency of each algorithm. Note that the data (array) given in the program has only 10 elements and there is a significant difference in the number of steps executed in each algorithm which ultimately defines the execution time. If the size of the array is in some tens of thousands, the difference would be greater and noticeable making the binary search algorithm to be the best choice if the data is sorted.

*#include <iostream>*

*#include <time.h>*

*#include <iomanip>*

*using namespace std;*

*int main() {*

*clock\_t begin, end;*

*double time\_spent;*

*//set the precision of the real number*

*cout.precision(20);*

*//Linear Search Program*

*int arr[10] = { 1, 33, 66, 32, 11, 4, 25, 21, 18, 35 }; //the list of data*

*bool found = false; //check if the number is found*

*int sNum = 14; //number to be searched*

*int stepsCounter = 0; //number of times loop body executed*

*begin = clock();*

*for (int i = 0; i < 10; i++) {*

*stepsCounter++;*

*if ( sNum == arr[i]) { //check against each element in the array*

*cout << "num found";*

*found = true;*

*break;*

*}*

*}*

*//calculate execution time*

*end = clock();*

*time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;*

*if (!found)*

*cout << "num not found";*

*cout << "\n\nnum of iterations executed in Linear Search Algorithm = " << stepsCounter;*

*cout<<"\ntime taken = "<<(double)time\_spent <<" Seconds";*

*//Binary Search Program*

*int iLow =0 , iHi=9;*

*int iMid = 0, counter = 0;*

*//Start time counter*

*begin = clock();*

*while(iLow <= iHi){*

*counter++;*

*iMid = (iLow + iHi)/2;*

*if(arr[iMid] == sNum){*

*cout<<"num found at index " <<iMid;*

*break;*

*}*

*if(arr[iMid] < sNum){*

*iLow = iMid+1;*

*}else{*

*iHi = iMid-1;*

*}*

*}*

*//calculate execution time*

*end = clock();*

*time\_spent = (double)(end - begin) / CLOCKS\_PER\_SEC;*

*cout<<"\n\nNumber of iterations executed in Binary Search = "<<counter;*

*cout<<"\ntime taken = "<<(double)time\_spent <<" Seconds";*

*return 0;*

*}*

**ACTIVITY**

Write and execute the program given above in any C++ compiler. Compare the number of times the loop body is executed in both the linear search and the binary search algorithms.

You can also step run the program using a breakpoint to understand the working of the code.

**REVIEW QUESTIONS**

1. What is the necessary condition for the binary search algorithm?

1. What is the worst case complexity of binary search algorithm?

1. If size of data or number of elements in array (N) is 100, how many times the loop body will iterate in linear search and in binary search in the worst case?