University of Information Technology and Sciences

Assignment

On

# *Measuring the time complexity of*

# *Linear Search and Binary Search*

Course code: CSE 214

Course Title: Algorithms Lab

Semester : Spring 2019

Date : 04/02/2019

**Submitted By:** **Submitted To:**

Anik Barua Habibur Rahman

ID No:17151029 Lecturer, Dept. of CSE

Dept. of CSE

Introduction:::

Linear search is a very basic and simple search algorithm. In Linear search, we search an element or value in a given array by traversing the array from the starting, till the desired element or value is found.

It compares the element to be searched with all the elements present in the array and when the element is **matched** successfully, it returns the index of the element in the array, else it return -1.

Linear Search is applied on unsorted or unordered lists, when there are fewer elements in a list.

Binary Search is used with sorted array or list. In binary search, we follow the following steps:

1. We start by comparing the element to be searched with the element in the middle of the list/array.
2. If we get a match, we return the index of the middle element.
3. If we do not get a match, we check whether the element to be searched is less or greater than in value than the middle element.
4. If the element/number to be searched is greater in value than the middle number, then we pick the elements on the right side of the middle element(as the list/array is sorted, hence on the right, we will have all the numbers greater than the middle number), and start again from the step 1.
5. If the element/number to be searched is lesser in value than the middle number, then we pick the elements on the left side of the middle element, and start again from the step 1.

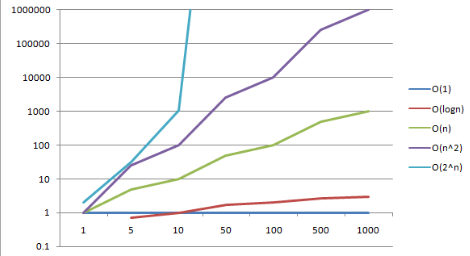
Binary Search is useful when there are large number of elements in an array and they are sorted.

So a necessary condition for Binary search to work is that the list/array should be sorted.

Table: Comparing the time elapsed by different input sizes: (time in ms)

|  |  |  |
| --- | --- | --- |
| Input size | Linear Search | Binary Search |
| 10 | 43 | 6 |
| 100 | 67 | 11 |
| 1000 | 76 | 13 |
| 10000 | 156 | 18 |
| 100000 | 226 |  |
| 1000000 | 2527 |  |

# Graph: Complexity graph for O (N), O(N^2), O( log N ), O(N log N)



CODE :::

/Generate input/

#include <stdio.h>

#include <stdlib.h>

int main()

{

freopen("Output.text","w",stdout);

long long i;

for(i=0;i<=1000000;i++)

{

printf("%ld ",rand());

}

printf("Hello\n");

return 0;

}

/………..END………./

Code of O(N):

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n;

long long arr[100001];

cin>>n;

freopen("Output.text","r",stdin);

for(int i=1;i<=n;i++)

{

scanf("%lld ",&arr[i]);

}

return 0; }

Code of O(N^2):

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n;

long long arr[100000];

cin>>n;

freopen("Output.text","r",stdin);

for(int i=1;i<=n;i++)

{

for(int j=1;j<=n;j++)

{

scanf("%lld ",&arr[i]);

}

}

return 0;

}

Code of O(Log N):

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n;

long long arr[100001];

cin>>n;

freopen("Output.text","r",stdin);

for(int i=1;i\*i<=n;i++)

{

scanf("%lld ",&arr[i]);

}

return 0;

}

Code of O(N Log N):

#include <bits/stdc++.h>

using namespace std;

int main()

{

int n;

long long arr[100001];

cin>>n;

freopen("Output.text","r",stdin);

for(int i=1;i<=n;i++)

{

for(int j=1;j\*j<=n;j++)

{

scanf("%lld ",&arr[i]);

}

}

return 0;

}