ISSN No:-2456-2165

Multi-Tier Search-A Novel Way of Searching 4D Array

Himani Tyagi
Computer Science and Engineering
B.S. Anangpuria Institute of Technology and Management
Faridabad, India

Puneet Bhardwaj
Computer Science and Engineering
B.S. Anangpuria Institute of Technology and Management
Faridabad, India

Abstract:- The linear search is the basic searching algorithm to search an item among the set of different items. Second basic algorithm is binary search that performs better than linear search. both are unstable sorting algorithms. Today many searching algorithm are present that are very fast to search an item. In this paper, a new approach for searching an item is presented. The comparison of proposed algorithm with Linear, Binary is also shown. The use of C++ for the implementation of this algorithm and analysis of CPU time taken by both is also preferred. Results have shown that proposed algorithm is working well for all the input values and it takes lesser time than sequential, binary searching in all aspects.

***Keywords**:- Binary search, Linear Search, Layer Search, Searching techniques, Time Complexity, unstable searching algorithm.*

I. INTRODUCTION

The searching algorithms is a step by step procedure to locate a specific data among the collection of data. This is the fundamental procedure in computing. The speed of searching algorithm is the major concern as it should be as fast as possible and provide accurate results .In computer science, The criteria for selection of a best algorithm for searching is the accurate results and speed. Reflecting it to everyday life of a human, it is impossible to be without searching any concept or anything. There are certain famous algorithm that works behind this process. Matrix is nothing but rectangular representation of data, numbers, expressions, symbols. In this documentation a new searching technique named *"Multi-tier search*" based on the limitation of linear and binary search is been shown.The layer division based on the structure of 4-dimensional and data arrangement is been shown in the following figure. There are three layer division and in each layer matrics is arranged and selection of layer is first performed. Then, matrics is chosen and last element of row compared with the searched data and in this manner the search continues.

[0,0]	[0,1]	[0,2]
[1,0]	[1,1]	[1,2]
[2,0]	[2,1]	[2,2]

[0,0]	[0,1]	[0,2]
[1,0]	[1,1]	[1,2]
[2,0]	[2,1]	[2,2]

[0,0]	[0,1]	[0,2]
[1,0]	[1,1]	[1,2]
[2,0]	[2,1]	[2,2]

[0,0]	[0,1]	[0,2]
[1,0]	[1,1]	[1,2]
[2,0]	[2,1]	[2,2]

[0,0]	[0,1]	[0,2]
[1,0]	[1,1]	[1,2]
[2,0]	[2,1]	[2,2]

[0,0]	[0,1]	[0,2]
[1,0]	[1,1]	[1,2]
[2,0]	[2,1]	[2,2]

Fig 1:- Multi-tier architecture

[0,0]	[0,1]	[0,2]
[1,0]	[1,1]	[1,2]
[2,0]	[2,1]	[2,2]

[0,0]	[0,1]	[0,2]
[1,0]	[1,1]	[1,2]
[2,0]	[2,1]	[2,2]

[0,0]	[0,1]	[0,2]
[1,0]	[1,1]	[1,2]
[2,0]	[2,1]	[2,2]

II. PROPOSED METHODOLOGY

In 4D array, layer is selected first by comparing last element of layer with searching element and then from a selected layer, a matrix (which is a 2D array) is selected. In 2D array searching traverses sequentially from left to right(row major) or top to down(column major).when it finds the element equal to the searched item the search terminates and the index of the element is returned, if unsuccessful then returns 0.in the proposed document the data is assumed to be in a sorted and arranged in a 4-dimensional array and the searching takes less time when compared to linear and binary search. The four dimensions of array indicates the horizontal layer, matrix in a particular layer, and rows and columns in that particular matrix chosen for the search. After the text edit has been completed, the paper is ready for the template. Duplicate the template file by using the Save As command, and use the naming convention prescribed by your conference for the name of your paper. In this newly created file, highlight all of the contents and import your prepared text file. You are now ready to style your paper; use the scroll down window on the left of the MS Word Formatting toolbar.

➢ *Algorithm*

Let a 4-dimensional matrix be a[w][x][y][z] having w*x*y*z numbers. Let S be the number to be searched. In matrix a[w][x][y][z] (˜as per the structure of 4d matrix), there are w horizontal layers, x be the number of 2dimensional matrix in each layer, y & z be the number of rows and columns of 2d be matrices respectively. w,x,y,z be some integer values.

(Assume matrix is completely filled in a sorted manner).
The proposed algorithm gives better results when compared with binary and linear.
The algorithm and code are given below.

ISSN No:-2456-2165

➢ *Algorithm:*

Step1: (selecting horizontal layer)

For i=0 to w:

Compare a[i][x][y][z] with S.

If S is greater than a[i][x][y][z] then i=i+1 and repeat step 1.

Else step 2.

Step 2: (selecting matrix in selected layer)

for j=0 to x

Compare a[i][j][y][z] with S.

If S is greater than a[i][j][y][z] then j=j+1 and repeat step 2.

Else step 3.

Step3: (selecting row in selected matrix)

for k=0 to y

Compare a[i][j][k][z] with S.

If S is greater than a[i][j][k][z] then k=k+1 and repeat step 3.

Else step 4

Step4: (finally matching selected row to find number exist or not) for l=0 to z

Compare a[i][j][k][l] with S.

If S is greater than a[i][j][k][l] then l=l+1 and repeat step 4.

Else if S==a[i][j][k][l] then print “number found”.

III. MULTI-TIER SEARCH

```
#include <iostream>
#include <time.h>
using namespace std;
int main()
{
int
a[13][13][13][13],i,j,k,l,s=28560,p=1,e=12,f=12,g=12,h=12,
w=13,x=13,y=13,z=13;
cout<<"Enter the data in array";
for(i=0;i<13;i++)
{
for(j=0;j<13;j++)
{
for(k=0;k<13;k++)
{
for(l=0;l<13;l++)
{
a[i][j][k][l]=p;
cout<<a[i][j][k][l]<<"   ";
p=p+1;
}
}
}
}
cout<<"\nEnter number to be searched\n ";
cout<<" "<<s;
double start_s=clock();
for(e=0;e<w;e++)
{
if(s<=a[e][f][g][h])
{
for(f=0;f<x;f++)
{
if(s<=a[e][f][g][h])
{
for(g=0;g<y;g++)
{
if(s<=a[e][f][g][h])
{
for(h=0;h<z;h++)
{
if(s==a[e][f][g][h])
{
cout<<"number found\n";
}
}
}
}
}
}
}
}
double stop_s=clock();
cout<<"time "<<(stop_s-start_s)/CLOCKS_PER_SEC;
return 0;
}
```

Results

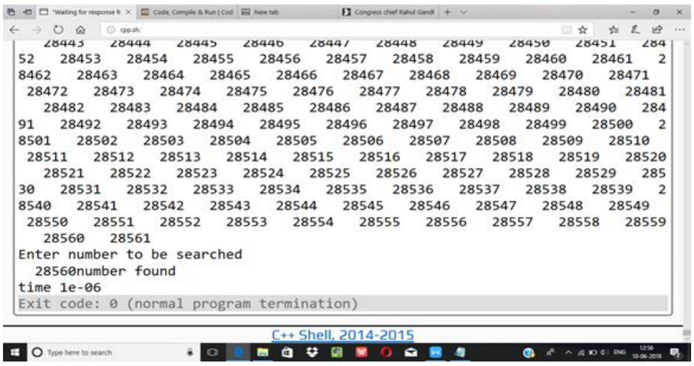

Fig 2:- Output by proposed methodology

IV. LINEAR SEARCH

```
#include <iostream>
#include <time.h>
using namespace std;
int main()
{
int a[28561],i, s=28560,p=1,n=28561;
cout<<"Enter the data in array";
for(i=0;i<n;i++)
{
a[i]=p;
cout<<a[i]<<"   ";
p=p+1;
}
```

```
cout<<"\nEnter number to be searched\n ";
cout<<" "<<s;
double start_s=clock();
for(i=0;i<n;i++)
{
   if(s==a[i])
   {
      cout<<"number found\n";
      break;
}
}
double stop_s=clock();
cout<<"time "<<(stop_s-start_s)/CLOCKS_PER_SEC;
return 0;
}
```

Results

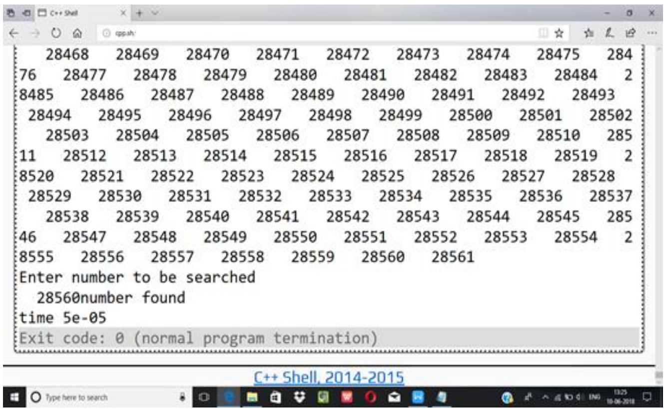

Fig 3:- Output by linear search

V. BINARY SEARCH

```
#include <iostream>
#include <time.h>
using namespace std;
int main()
{
int a[28561],first,last,middle,i,s=28560,p=1,n=28561;
cout<<"Enter the data in array";
for(i=0;i<n;i++)
{
a[i]=p;
cout<<a[i]<<"   ";
p=p+1;
}
cout<<"\nEnter number to be searched\n ";
cout<<" "<<s;
double start_s=clock();
first = 0;
        last = n-1;
        middle = (first+last)/2;
        while (first <= last)
        {
                if(a[middle] < s)
                {
                        first = middle + 1;

                }
                else if(a[middle] == s)
                {
                        cout<<"number found\n";
                        break;
                }
                else
{
                        last = middle - 1;
                }
                middle = (first + last)/2;
        }
        if(first > last)
        {
                cout<<"Not found! "<<s<<" is not present
in the list.";
        }

double stop_s=clock();
cout<<"time "<<(stop_s-start_s)/CLOCKS_PER_SEC;
return 0;
}
```

Result

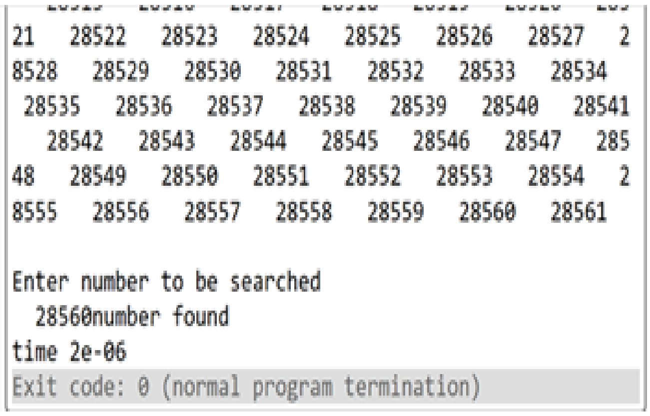

Fig 4:- Output by binary search

➢ *Performance analysis and comparison*

All the searching algorithms (linear, binary, Multi-Tier) are implemented in C++ using …… compiler,32-bit operating system having Intel core i5 and installed memory (RAM) 8.00GB.

VI. CONCLUSION

The time required for searching using the proposed algorithm is shown in the following table, that shows the time required by proposed method is less as compared to binary and linear search.

Table 1. Execution time for searching algorithms (for 28561 inputs)

Search algorithm	Linear search	Binary search	Multi-tier search
Execution time	5e-05 s = 0.05 ms	2e-06 s= 0.002 ms	1e-06 s= 0.001 ms

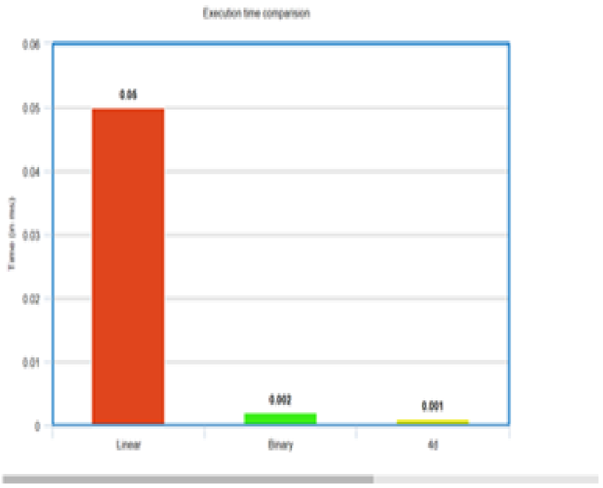

Fig 5:- Graph indicating difference in linear and proposed method.

➢ *Future Scope*

The proposed algorithm is faster than linear, binary search which are supposed to be basic and fast algorithm. Multi-tier search is fruitful with searching among large amount of data as the pixel tracing algorithms that are derived from sequential algorithms.

REFERENCES

[1]. E. Horowitz and S. Sahni, fundamental of Data Structure Rockville, MD: Computer Science Press, 1982.
[2]. Nitin Arora, Two Way Linear Search Algorithm, Vol. 107-No. 21, December 2014.
[3]. Debadrita Roy, A comparative Analysis of Three Different Types of Searching Algorithms in Data Structures, Vol. 3, Issue 5, May 2014.
[4]. Jyotirmayee Rautaray, Hashed Based Searching Algorithm, Vol. 2, Issue 2, February 2013.
[5]. Alfred V., Aho J., Horroroft, Jeffrey D.U. (2002) Data Structures and Algorithms.
[6]. Seymour Lipschutz (2009) Data Structure with C, Schaum Series, Tata McGraw-Hill Education.
[7]. Rehan Guha, GRID SEARCHING Novel way of Searching 2D Array, International Journal of Computer Applications Technology and Research Volume 5–Issue 1, 26 - 33, 2016, ISSN:- 2319–8656.