International Journal of Computer Applications (0975 – 8887)
Volume 148 – No.8, August 2016

10x Matrix Searching Algorithm

Ajay Kedia
B-Tech IT Engineering
Vellore Institute of Technology
Vellore

Shagun Dhingra
B-Tech IT Engineering
Vellore Institute of Technology
Vellore

ABSTRACT

Searching is the main fundamental concept using in computer Science to get the data from unsorted and sorted elements. There are a number of searching algorithms present in computer science. Here, we are going to introduce with new searching algorithms i.e. 10x Matrix Searching algorithms. 10x refer to a table which contains only 10 rows and x refer to the columns associated with each row. So, basically we are going to search an item from a collection of items. These item may be a part of database, may be a part of some formula or procedure which are going to use in our programs, like we are finding some solution of an equation and we need the root of an equation.

General Terms

Linear Search, Binary Search, Jagged Array, Indexing.

Keywords

Searching, Jagged Array, Index, 10x Matrix, Linked List.

1. INTRODUCTION

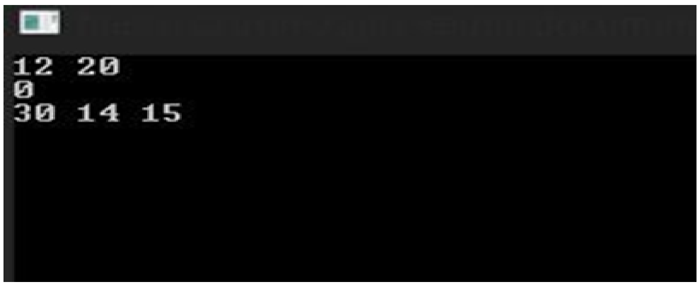
Finding an element from the sorted or unsorted list of elements has always been a hot topic in computer science. In an unsorted list of elements, elements are arranged in different order and in a sorted list, elements are arranged in either ascending or descending order. Here we are going to introduce a 10x Matrix searching algorithm. In this algorithm, we are going to insert elements in 10 rows whose indexes start from 0 to 9. Each index has its own meaning. If the elements whose remainder is 0 after dividing by 10 , then it will take whose index start from 0. Similarly, if an element has remainder 1 then it will insert into a row whose index is 1 and so on.

So basically we will have only 10 rows and column according to data inserted. So, here we are also working on memory management and also we are introducing jagged array in our algorithm. As we know a jagged array contains different no. of columns in different rows.

1.1 Introducing Jagged Array

A simple C# Program introducing jagged array, containing 3 rows and different number columns:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ConsoleApplication12
{
class Program
{
static void Main(string[] args)
{
 //Declaring a jagged array with 3 rows
 int[][] jagged_array = new int[3][];
 // Creating array inside the jagged array and assigning the value to it
 jagged_array[0] = new int[2];
 jagged_array[0][0] = 12;
 jagged_array[0][1] = 20;
 // we initializing the second row equal to zero
jagged_array[1] = new int[1];
// initializing the third row by array initializer
jagged_array[2] = new int[3] { 30, 14, 15 };
// writing all the elements present in jagged array
int i=0;
while(i<jagged_array.Length)
{
int[] inner_Array = jagged_array[i];
for (int a = 0; a < inner_Array.Length; a++)
{
        Console.Write(inner_Array[a] + " ");
}
Console.WriteLine();
i++;
}
Console.ReadKey();
}
}
}
```



```
12 20
0
30 14 15
```

Fig 1: Output of simple jagged array

2. RELATED WORK

2.1 10x Matrix Searching

A user enters some data in a Program
15,77,33,12,100,68,1004,93,101,80,76,90,219,82,99,2,7

Now user wants to search 219 from the given array. So if we are going to use normal search it will increase the running time of the compiler and hence not more effective but if we will use 10x Matrix Searching Algorithm, we can reduce the time complexity of the compiler.

First we are going to store these elements into a jagged array.

Table 1. Data Sorted in Jagged Array

0	100	80	90
1	101		
2	12	82	2
3	33	93	
4	1004		
5	15		
6	76		
7	77	7	
8	68		
9	219	99	

Elements are stored in their respective rows by dividing the element by 10 and store into its remainder as an index of row.

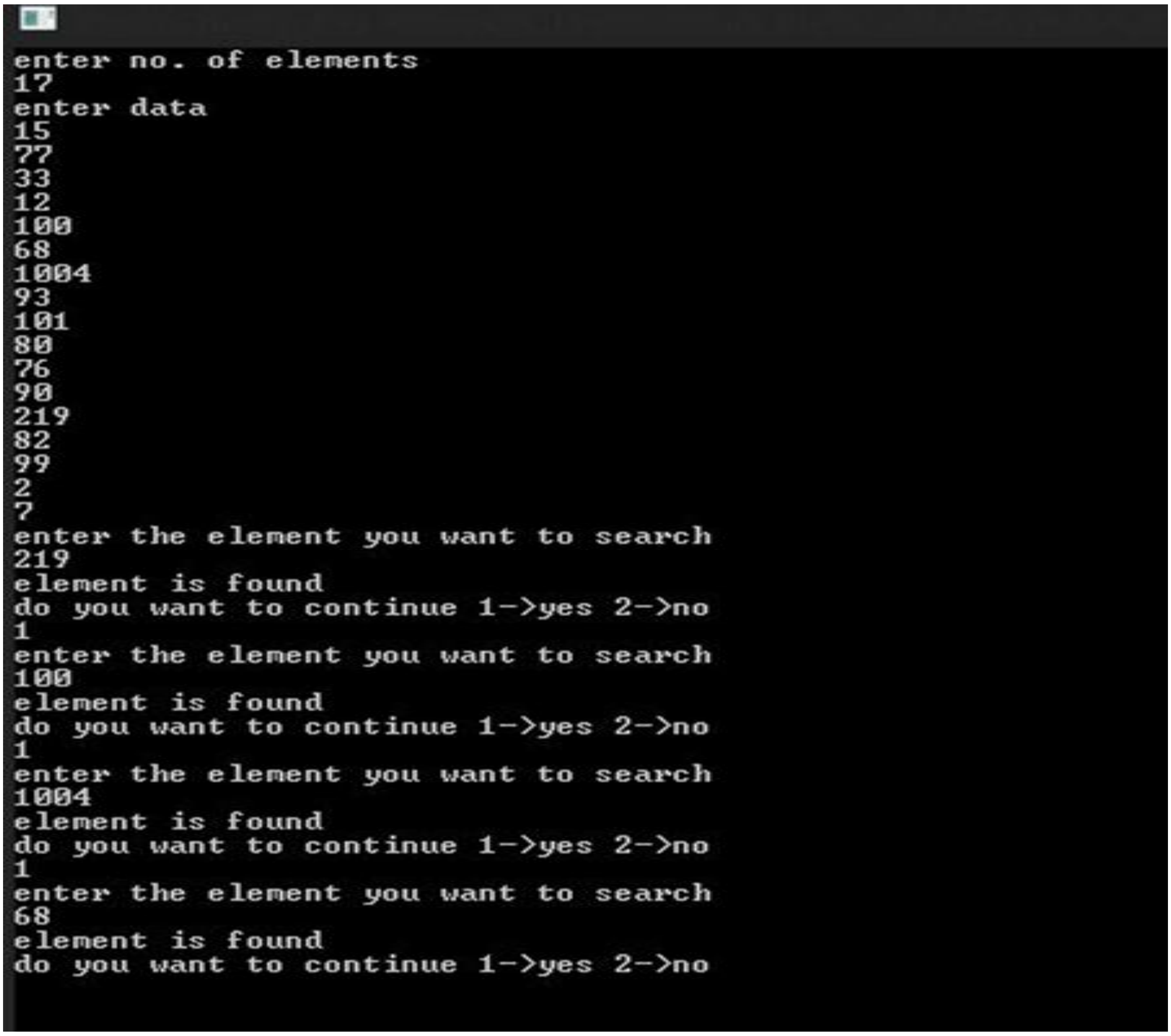
Row[index]=Array[index]%10;

Now we have to search 219, so 219%10=9

So in row 9 we have 219 at first position… so if we store it into simple array we have to till 13th index but in this case we search the element at first index.

2.2 Binding the whole example into a 10x Matrix Search Program

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace ConsoleApplication13
{
class Matrix10xSearching
{
static void Main(string[] args)
{
int n;
Console.WriteLine("enter no. of elements");
 n = Convert.ToInt32(Console.ReadLine());
 int[] array = new int[n];//inner array of jagged array
int[][] jagged_array = new int[10][];//creating Jagged array
int[] a = new int[10] {3,1,3,2,1,1,1,2,1,2};//assigning value to inner array
Console.WriteLine("enter data");
for (int i = 0; i < n; i++)
{
array[i] = Convert.ToInt32(Console.ReadLine());
//15,77,33,12,100,68,1004,93,101,80,76,90,219,82,99,2,7
}
//initialising each row of jagged array
int j = 0,remainder;
for (int i = 0; i < 10; i++)
{
jagged_array[i] = new int[a[i]];
j++;
}
//inserting the value into the array
int x1,x2,x3,x4,x5,x6,x7,x8,x9,x10;
x1 = x2 = x3 = x4 = x5 = x6 = x7 = x8 = x9 = x10 = 0;
for (int i = 0; i < array.Length; i++)
{
remainder = array[i] % 10;
switch (remainder)
{
case 0:                jagged_array[remainder][x1] = array[i];//row with index->0
x1++;
break;
case 1:    jagged_array[remainder][x2] = array[i];//row with
index->1
x2++;
break;
case 2:     jagged_array[remainder][x3] = array[i];//row with
index->2
x3++;
break;
case 3:     jagged_array[remainder][x4] = array[i];//row with
index->3
x4++;
break;
case 4:     jagged_array[remainder][x5] = array[i];//row with
index->4
x5++;
break;
case 5:     jagged_array[remainder][x6] = array[i];//row with
index->5
x6++;
break;
case 6:     jagged_array[remainder][x7] = array[i];//row with
index->6
x7++;
break;
case 7:     jagged_array[remainder][x8] = array[i];//row with
index->7
x8++;
break;
case 8:     jagged_array[remainder][x9] = array[i];//row with
index->8
x9++;
break;
case 9:     jagged_array[remainder][x10] = array[i];//row with
index->9
x10++;
break;
default:
break;
}//end of switch case
}//end of for loop
int ch;
do
{           Console.WriteLine("enter the element you want
to search");
int search;
search = Convert.ToInt32(Console.ReadLine());
remainder = search % 10;
int flag = 0;
for (int i = 0; i < a[remainder]; i++)
{
if (search == jagged_array[remainder][i])//searching happen
at particular index of row
{
flag = 1;
}
}
if (flag == 0)
{           Console.WriteLine("element doesnot found");
}
else
{           Console.WriteLine("element is found");
}
Console.WriteLine("do you want to continue 1->yes 2->no");
ch=Convert.ToInt32(Console.ReadLine());
}while(ch==1);
Console.ReadKey();
}//end Of main Method
}//end of Matrix10xSearching
}//end of Namespace
```



```
enter no. of elements
17
enter data
15
77
33
12
100
68
1004
93
101
80
76
90
219
82
99
2
7
enter the element you want to search
219
element is found
do you want to continue 1->yes 2->no
1
enter the element you want to search
100
element is found
do you want to continue 1->yes 2->no
1
enter the element you want to search
1004
element is found
do you want to continue 1->yes 2->no
1
enter the element you want to search
68
element is found
do you want to continue 1->yes 2->no
```

Fig 1: Output of simple 10x Matrix Algortihm

2.3 Pseudo code

Step1: Get number of element from the user and store into n.

Step2: Now store it into a jagged array whose row size is equal to 10 and number of columns will be calculated on the basis of element is going to store.

Step3: Now ask the user to enter the element which he/she want to search.

Step 4: Find the remainder =element % 10 and search in the index=remainder.

Step5: If element found then print ’okay’ otherwise ‘not found’.

Step6: If you want to search more, go back to step3 otherwise go to step7.

Step7: Exit

3. TIME COMPLEXITY AND SPACE COMPLEXITY

3.1 Time complexity

The time complexity of searching is depend upon length of inner array in a jagged array where we are going to search using the remainder of the searching element.

In above example, we are searching 219 , so the time complexity for searching 219 in 9th index of row is constant because element present at first position In jagged array. But if we go for simple searching or binary searching it will take time to search the element, even more than our proposed algorithm. Binary search algorithm will search the element on the basis of min or max, so it will take time to calculate the min max where as simple searching will check with each element and so time complexity will be more.

if a user wants to enter the elements whose remainder is same for all the element then our algorithm follow the same simple searching but all other row will be empty except the row whose remainder has been calculate.

We can also check that the number of remainder in the entered elements, according to that we can define the row.

3.2 Space complexity

Jagged array will take only that much memory which are needed to store the elements. So memory problem also handled. For better memory control we can use linked list in our searching.

3.3 A simple C# program to define jagged array

```
 using System;
  namespace ConsoleApplication13
  {
      class Node
      {
          public int data;
          public Node left;
          public Node right;
          public Node up;
          public Node down;
      }
  }
```

So here we have four mapping Left, right, up, down. The starting index of each row has three mapping among four mapping i.e. up, down and right but left one is null. Where as in 0th row the middle one contains left, right only, others will be null. The end node have only left mapping, others will be null.

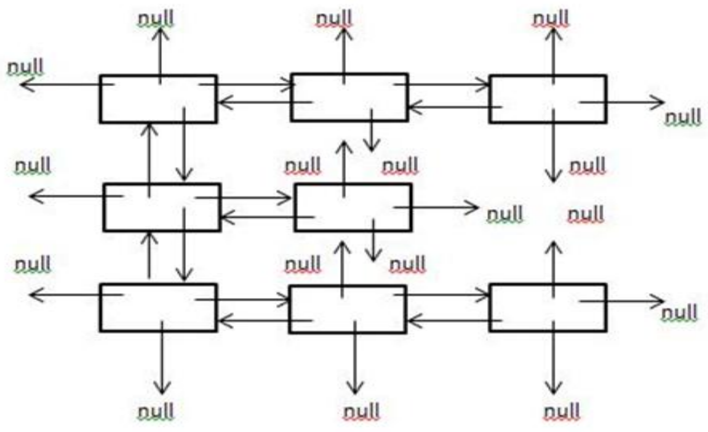


Fig 3: A Simple Digram of Linked List

So we can handle the memory at run time. So time complexity and space complexity both are preserved in 10x matrix searching algorithm. Thus, it enhances our model functionality from other searching models.

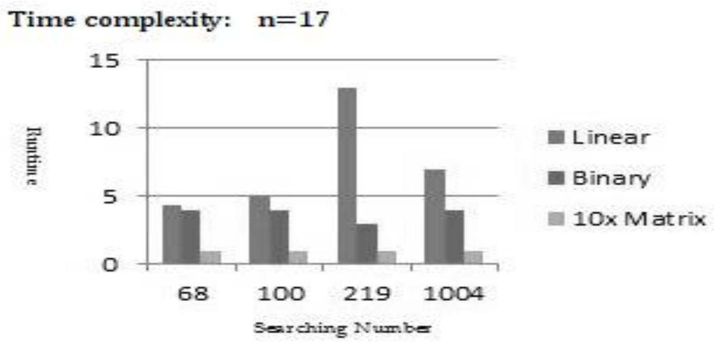


Fig 4: Difference between Linear searching, binary searching and 10x Matrix Searching

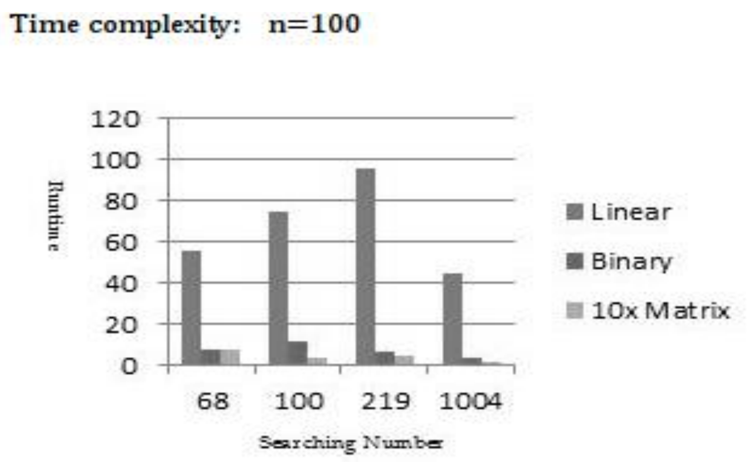


Fig 5: Graph between Linear search, Binary search and 10x Matrix search for n=100 data samples

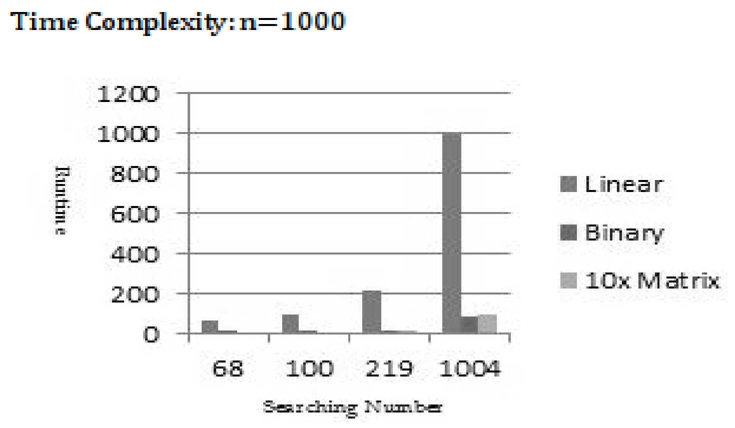


Fig 6: Graph between Linear search, Binary search and 10x Matrix search for n=1000 data samples

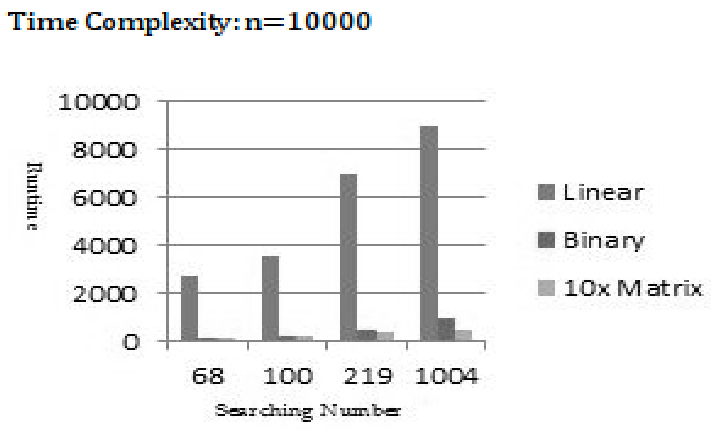
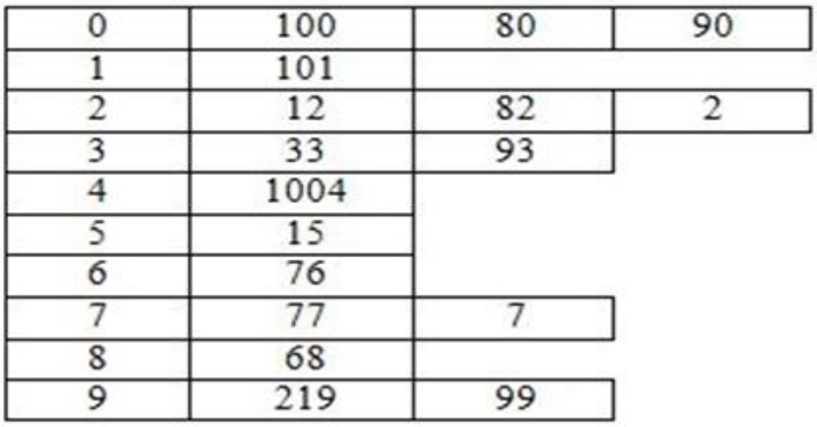


Fig 7: Graph between Linear search, Binary search and 10x Matrix search for n=10000 data samples

4. CONCLUSION

10x Matrix searching algorithm reduce the time complexity of the program and also space complexity. Data stored in 10x matrix package and search the element from a specific position of a row whose row is identify through its remainder, i.e. Element % 10 and store the value at that location. So, it helps us to search an element from a large amount of data. Unlike in linear searching we search the element by element. If it is found then it will be okay, otherwise it will continue till last. Whereas in binary searching, our searching element will jump b/w min and max and large number of calculation will do.Our algorithm can be better if we do binary search after storing the value into jagged array. So, binary search can be applied only on a subset of the array not on the whole array Macintosh, use the font named Times. Right margins should be justified, not ragged.



0	100	80	90
1	101		
2	12	82	2
3	33	93	
4	1004		
5	15		
6	76		
7	77	7	
8	68		
9	219	99	

Fig 8: Data arranged in jagged array

We have plotted a graph using 10x Matrix Searching algorithm using binary search and without binary search and the graph which we got is not the clear one. The graph swings b/w with and without binary search.

Sometimes it is good to use binary search and sometimes it is not so much good to carry out searching using binary search.

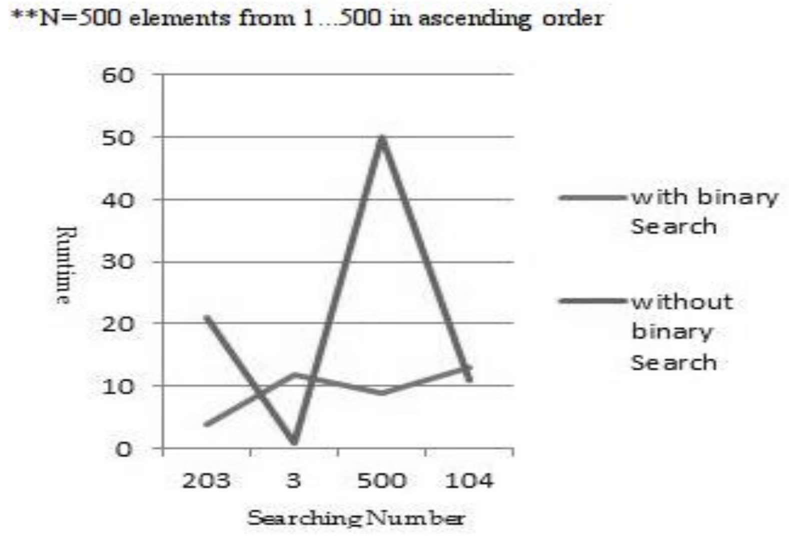


Fig 9: Graph to check runtime of 10x matrix searching using binary searching and without binary searching

5. ACKNOWLEDGMENTS

We express our sincere thanks to www.Google.com that help us through out and provide us the information to introduce this algorithm. A heart full and sincere gratitude to our parents who motivate us through out and help us in our bad time.

6. REFERENCES

[1] Narasimha Karumanchi, Data Structure and algorithms Made Easy in Java: Data structure and Algorithms.

[2] Jagged array example. Online http://www.dotnetperls.com/jagged-array .

[3] Linear and Binary searching example. Online www.cprogramming.com/discussionarticles/sorting_and_searching.html

[4] En.m.wikipedia.org/wiki/Linear_search.