ISSN No. 0976-5697

Volume 2, No. 4, July-August 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Divide method of sorting records or numbers

Br. Sukumar (‘Sukumar Shil”)
Bramhachari, Ramakrishna Mission,Narendrapur
Ramakrishna Mission Residential College (Autonomous),
Narendrapur Kolkata, India
sukumarshil@gmail.com

Abstract: To sort a list of numbers there are few methods whereas the Divide and sort method at first divides the list of numbers into three groups rather two arrays and a variable. The numbers are compared with respect to the first element. The numbers lesser to the first element are kept in an array. The greater numbers are kept in a separate array and the numbers which are equal to the first element are counted and the value is set in a variable. Now the two arrays are sorted separately by any other method. Now the lists are merged accordingly, list of lesser numbers are kept first, then the numbers which are equal to first element are kept and then the numbers which are greater than the first element are kept after it. Then the list is printed or kept in an array of variables. As the numbers are divided before sorting the method is named as “Divide and Sort” method. The complexity of this process is 1/3rd of other methods of quadratic running time. Best case is when all the numbers are same. That case can be checked by this method.

Keywords: divide and sort method, time complexity, sorting method, sorting cost, three variables

I. INTRODUCTION

The sorting operation if performed on an array, will sort it in a specified order (ascending/descending) [8]. The divide and sort method is simply different from the Divide-and-Conquer Sorting as described in [5]. The idea was developed considering the case when all the numbers are same. To make zero the first number was subtracted from all numbers. Then the case when all the numbers are not same considered. The author then kept the negative numbers in a separate first array and which were positive in a second array. The zeroes were counted in a variable. But momentarily, it was found that there no necessity of subtracting the numbers rather what we can do is compare the first element with the others and kept the lesser numbers in an array and the larger numbers in another. Automatically the running time reduced. But momentarily the author found that there no need of subtracting first elements but only compare and keep the lesser ones in the b list and the upper greater ones in the c list and equal elements are only counted to m. If three variables get equal numbers of elements, in the average case the running time becomes less than 1/3rd of other cases where time complexity O(n^2). The proof is shown the method portion that dividing the list gives less time complexity. The program in this article is written in C with the help of Turbo C++ compiler and Windows 2000 OS was used.

II. ALGORITHM

Algorithm: The algorithm is better method for understanding a program. An algorithm is a finite set of instructions which, if followed, accomplish a particular task [7]. As following the steps of it program can be written in any computer language. We use array as data structure. An array is a collection of similar elements [4]. The steps of performing the method are given below:

Steps: Divide method of sorting

```
a.  Input the array numbers{a[0],……a[n-1]}
b.  Compare the numbers with the first element
c.  initialize m=1, j=0, k=0;
d.  for(i=1,i<n;i++)
e.  {
f.  if(a[0]>a[i])
g.  b[j]=a[i]; increment j by 1;
h.  else if(a[0]<a[i])
i.  c[k]=a[i]; increment k by 1;
j.  else
k.  increment m by 1;
l.  }
m.  sort (b,j)
n.  sort (c,k)
o.  for(i=0;i<j;i++)
p.  {
q.  e[i]=b[i];
r.  }
s.  middle=j+m-1;
t.  for(i=j;i<middle;i++)
u.  {
v.  e[i]=a[0];
w.  }
x.  o=middle;
y.  for(i=0;i<k;i++)
z.  {
aa. e[o]=c[i]; increment by 1;
bb. }
cc. Now print the e list that is the final sorted list
```

Subprogram: Sort()

```
Sort(int n, int f[])
{
int t,i,j;
for(i=0;i<n;i++)
for(j=0;j<n;j++)
{
t=f[i];
f[i]=f[i+1];
f[i+1]=t;
}
}
```

III. METHOD

Say a list of numbers{23,4,5,7,6,1,7,8,9,0}. We have to sort this list of numbers. First 23 is compared with other elements. There will be one list only b={4,5,7,6,1,7,8,9,0} and 23 is saved in a variable. So m will be 1.Now b is sorted and put in an array e then 23 is added with that array.

Now take another array {4,67,8,5,3,2,1,0,4}.On separating the list on the basis of larger and smaller than first element two lists become b={3,2,1,0} and c={67,8,5} and m will be 2 as 4 lies two times in the list. Now sort the lists separately and make an another array putting the lists in the order b,4,4,c. On sorting b becomes {0,1,2,3} and c becomes {5,8,67} and m is 2. So the sorted final list is e={0,1,2,3,4,4,5,8,67}.

When all the numbers are same b and c lists will have no numbers. Only variable m will count how many times the first element occurs. So there will be no sorting cost rather putting the numbers as many times as they occur. There only takes the time for comparing. Here m will be n. So in this case the time complexity becomes O (n), linear.

When the numbers are already sorted in that case the first element will be taken out from the list and c array will contain the rest of elements. It is then sorted and the first element is placed at the first position.

IV. COMPLEXITY ANALYSIS

The running time of this algorithm is one third of n^2 on the average. If the list can be divided into equal three groups then complexity becomes 1/3rd or even less than that of other methods of O(n^2). When we have only an asymptotic upper bound, we use O-notation. For a given function g(n), we denote by O(g(n)) the set of functions O(g(n))={f(n): there exist positive constants c and n_0 such that

$o \le f(n) \le c(g(n))$ for all $n \ge n_0$.}[9].

For best case when all the numbers are same then it gives minimum running time of the order of n-time for just comparing the elements. The complexity then becomes O(n), linear. When all the numbers are already sorted then other than the first element we have to sort whole list. Only c array will contain (n-1) numbers. So the running time complexity will be n*(n-1).

It is a stable sort.

Stable sort: A list of unsorted data may contain two or more equal data. If a sorting method maintains the same relative position of their occurrences in the sorted list, then it is called Stable Sort.[10]

V. PROGRAM IN C LANGUAGE

(Program written in C is efficient and fast [3])

A. *//"divide and sort" method*

```
#include<stdio.h>
#include<conio.h>
void main()
{
 void sort(int,int[]);
int a[100],b[100], c[100];
int n,i,j;
int d,k,m;
int o;
int e[100];
int middle;
clrscr();
printf("-----------------------------------");
printf("\n\nTo sort an array using divide and sort method");
printf("\n\n-----------------------------------");
printf("\n array input");
printf("\n no. of element would you like to sort=");
scanf("%d",&n);
/*input*/
printf("\n enter the numbers:");
for(i=0;i<n;i++)
{
printf("\ndata%d:",i);
scanf("%d",&a[i]);
}
printf("\n print the numbers:\n");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
}
```
//divide the numbers in three bags comparing with the first element
```
  d=a[0];
  k=0;
  j=0;
  m=1;                          //the first element
for(i=1;i<n;i++)
{
            if(d>a[i])
            {b[j]=a[i];
             j=j+1;}
            else if(d<a[i])
{           c[k]=a[i];
             k=k+1;}
            else
            m=m+1;              //number of first equal elements
 }
  printf("\ni=%d,j=%d,k=%d,m=%d\n",i,j,k,m);
  printf("\nThe lists are\n");
  for(i=0;i<j;i++)
  printf("%d\t",b[i]);
  printf("\n");
  for(i=0;i<k;i++)
  printf("%d\t",c[i]);
  printf("\nsort differently two separate lists \n");
  sort(j,b);
  printf("Print sorted b list\n");
  for(i=0;i<j;i++)
  printf("%d\t",b[i]);
  sort(k,c);
  printf("\nPrint the sorted c list:\n");
  for(i=0;i<k;i++)
  printf("%d\t",c[i]);
```
//Now merge the sorted lists in order b,a,c
```
  {for(i=0;i<j;i++)             //j number of elements are lesser than a[0]
  e[i]=b[i];                    //elements below the first element are placed at first
            }
             middle=j+m;
  for(o=i;o<middle;o++)
  e[o]=d;                         //first element is placed at the
middle

  for(i=0;i<k;i++) //k number of last elements
            e[o+i]=c[i];  //elements greater than last
```
/*show after sorting*/
```
printf("\n\n now the elements in ascending order::");
for(i=0;i<n;i++)
            {
            printf("\t%d",e[i]);
            }
getch();
}
```
//Subprogram
```
 void sort(int l,int v[])
 {
 int i,p,t;
 for(i=0;i<l;i++)
                        {
                         for(p=0;p<l-1;p++)
                                    {
                                    if(v[p]>v[p+1])
                                                {
                                                t=v[p];
                                                v[p]=v[p+1];
                                                v[p+1]=t;
                                                }
                                    }
                        }

 }
```

*B. /*Output*/*

```
/*------------------------------------
To sort an array using divide and sort method
------------------------------------
 array input
 no. of element would you like to sort=6
 enter the numbers:
data0:3
data1:23
data2:32
data3:34
data4:2
data5:1
 print the numbers:
3      23      32      34      2      1
i=6,j=2,k=3,m=1
The lists are
2      1
23      32      34
sort differently two separate lists
Print sorted b list
1      2
Print the sorted c list:
23      32      34

 Now the elements in ascending order::  1      2      3      23
32          34*/
```

 /*Output*/

With zero inputs

```
/*  ------------------------------------
To sort an array using divide and sort method
------------------------------------
 array input
 no. of element would you like to sort=0
 enter the numbers:
 print the numbers:
i=1,j=0,k=0,m=1
The lists are
sort differently two separate lists
Print sorted b list
Print the sorted c list:
 now the elements in ascending order::
                                        */
```

*C. /*Output When All Numbers are Same*

```
-----------------------------------
To sort an array using divide and sort method
------------------------------------
 array input
 no. of element would you like to sort=5
 enter the numbers:
data0:12
data1:12
data2:12
data3:12
data4:12
 print the numbers:
12      12      12      12      12
I=5,j=0,k=0,m=5
The lists are
sort differently two separate lists
Print sorted b list
Print the sorted c list:
now the elements in ascending order::
 12      12      12      12      12
*/
```

Table 1 : Comparison of running time among the algorithms

Method	Running time	Best Case	Worst Case	Characteristic
Selection Sort	O(n^2)[2]	O(n^2)[2]	O(n^2)[2]	Stable[1]
Insertion Sort	O(n^2)[2]	O(n)-when all the numbers are sorted[1]	O(n^2)[2]	Stable[1]
Bubble Sort	O(n^2)[2]	O(n^2)[1]	O(n^2)[2]	Stable[1]
Divide and Sort	<O(n^2)/3 on the average case	O(n) when all the numbers are same	O(n^2)	Stable
Alternate Sort[11]	O(n^2/2)	O(n) when applied with Divide and Sort method	O(n^2/2)	Stable

VI. CONCLUSION

This method can be applied on the limitation of array spaces. The machine is also responsible for the amount of numbers that it can sort. For normal computers it can sort near about 1000 elements. If large number of array spaces can be obtained then large number of arrays can be sorted. The best case of time complexity lies in the case when all the numbers are same. So the first comparison can be applied before starting for all kinds of sorts. This is the best method. Worst case of this method is when all the numbers are sorted in that case running time reaches maximum. Heap sort, that takes lesser time for sorting, is suitable only for contiguous lists [5]. It is not good for short lists [6]. Whereas the Divide and sort method applies to all cases giving better result. It can be found that some methods are good for less numbers of elements and some are good for greater number of elements. Insertion sort is good for less than 23 elements than quick sort [2]. It is also good than heap sorts for less than ten elements because of the constant terms in its time complexity. It can be applied in combination with all types of sorting. Alternate Sort method is a better method and when applied with Divide and Sort method will give a good result.

VII. ACKNOWLEDGMENT

I pay thanks to all the Staff members and Monastic members of Ramakrishna Mission, Narendrapur. I am thankful to all who knowingly or unknowingly have supported my work of research. I pay homage to Thakur (Ramakrishna Parahamsadev), Maa (Sarada), Swamiji(Swami Vivekananda), the holy trio of our worship and meditation. I specially thank Souvik Maharaj who was our teacher and Physics Dept. and Chemistry Dept. and above all Swami Suparnanandaji Maharaj. Lastly, I feel pleased to acknowledge my gratitude to two Colleges Ramakrishna Mission Residential College, Narendrapur and St. Xavier's college where from I learnt Computer Science and Indian Association of Cultivation of Science where I started learn Computer. I also thank Rakesh, Sujay, Pinaki, Shubhendu, Manasda, Amitda, Satyada, Suman for co-operating me in making the paper. Mr. Subrata Shil, my brother, also deserves thanks for helping me in many ways. Dr. Arun Kumar Tewari and Dr. Malay Purkait and Dr. Prashanta Ghosh of R K Misssion Residential College are very helpful and Dr. K P Ghatak of Electronics Dept., CU, who is a person with lot of research interest and it encouraged me. I also thank to Publication Company and the readers who will read it. At last I thank Swami Puratananandaji Maharaj and Jagatjyoti Banerjee and Mohitosh Giri. I also thank Arnab Datta, an MSc student for his collaboration. I also thank for it my mother, father and sister and Swami Divyanandaji Maharaj.

VIII. REFERENCES

[1] http://www.scribd.com(doc/48642850/The-Heroic-Tales-of-Sorting-Algorithms

[2] Horowitz-Sahni, Data Structure in Pascal (Chapter-Inter Sorting, P-(360-371). Ed. June 1983.

[3] Programming in ANSI C (2nd Ed.)-E. Balagurusamy . P-1

[4] Let Us C -Yashwant Kanetkar, 5th Ed., P-274

[5] Data Structures and Program Design in C-Robert Kruse, C.L. Tondo, Bruce Leung(2nd Ed.)-P.324.

[6] Data Structures and Program Design in C-Robert Kruse, C.L. Tondo, Bruce Leung(2nd Ed.)-P-327.

[7] Horowitz-Sahni, Data Structure in Pascal (Chapter-Inter Sorting, P-2. Ed. June 1983.

[8] Classic Data Structures (2nd Ed.)- Debasis Samanta. P-529

[9] Introduction to Algorithms (2nd Ed.)- T.H. CORMEN, C. E. LEISERSON, R. L. RIVEST, C.STEIN. P-44

[10] Classic Data Structures (2nd Ed.)- Debasis Samanta. P-530

[11] http://en.wikipedia.org/wiki/User:BrSukumar