Volume 4, Issue 12, December 2014 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Divide and Combine Sorting Algorithm

Shaik. Kareem Basha, Shaik Jaheda Begum, Fouzia Bano
CSE Department HITAM,
Hyderabad, India

Abstract— ***Sorting is a process of arranging data in a specific order according to requirements of Application. In complex software applications, sorting plays a major role to reduce searching time complexity of data items. Efficient sorting brings orderliness in the data. Many sorting methods are advocated. Each method has its own advantages and disadvantages. Often a designer or developer is puzzled as to which method is best and efficient. The efficiency could be measured in terms of memory usage, time taken to sort, cpu usage etc. In this paper an efficient sorting algorithm called as Divide and Combine Sorting Algorithm is proposed. We followed the various stages of Software Development Life Cycle to demonstrate the proposed sorting algorithm. Section I will give introduction about proposed sorting algorithm. In section II, proposed Algorithm is Analyzed and Designed. In section III, proposed algorithm is Implemented using C programming Language. In section IV, we will test the implementation of proposed algorithm using different test cases. In section V, we conclude the proposed Algorithm.***

Keywords— queue, left sub list, right sub list, output array list, output linked list, input array list, divide list, sort, combine

I. Introduction

Sorting is a process of arranging data in a specific order according to requirements of application. Many sorting algorithms are developed. Each algorithm has its own advantages and disadvantages. Often a developer is puzzled as to which algorithm is best and efficient. The efficiency can be measured in terms of memory usage, time taken to sort, cpu usage etc. In merge sort algorithm, the input list is recursively divided into left sub list and right sub list based upon the mid element. The proposed algorithm also divides the given input list into left sub list and right sub list based upon the mid element. But in this algorithm, we are using queue to store starting index of left sub list, ending index of left sub list, starting index of right sub list and ending index of right sub list, where as merge sort algorithm uses stack to store the starting and ending indices of both left sub list and right sub list. The proposed algorithm is designed by using Divide and Conquer problem solving strategy, in which the complex problem is divided into smaller sub problems, those are solved individually, at last the solutions of smaller sub problems are combined to form solution of complex problem. In proposed algorithm we are not dividing the input list recursively, but we are dividing the input list into smaller un breakable lists. At last we are combining the smaller sub lists into output list. We have designed two variations of proposed algorithm based upon the output sorted list. The input list must be un sorted array list, which is divided into many individual sub lists. The two variations of proposed algorithm are ArrayListCombine algorithm and LinkedListCombine algorithm, where as the DivideList algorithm is used to divide un sorted input array list. The time taken by DivideList algorithm to divide unordered input array list is O(n), where as the time taken by ArrayListCombine algorithm to combine individual unbreakable lists into a single sorted output array list is $O(n^2)$, the time taken by LinkedListCombine algorithm to combine unbreakable lists is O(n).

II. Analysis and Design of Proposed Algorithm

Consider the following input unsorted list of elements { 10 7 5 6 4 1 3 8 2 9}. This list is divided into two sub lists based upon the mid element 1, The left sub list contains elements from starting index to mid index, where as the right sub list contains elements from mid+1 index to ending index. After division left sub list is {10 7 5 6 4}, right sub list is {1 3 8 2 9}, the starting index of left sub list, ending index of left sub list are placed into queue, again starting index of right sub list, ending index of right sub list are placed into queue. After placing the indices of left sub list and right sub list into queue. Left sub list is selected and it is divided into two sub lists. The above process is continued until all the lists becomes unbreakable.

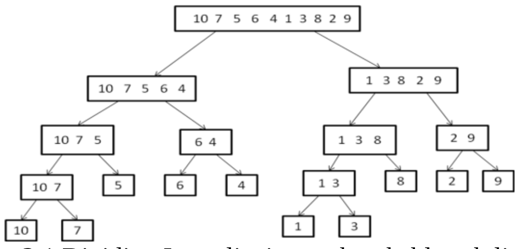


Fig 2.1 Dividing Input list into unbreakable sub lists

The proposed DivideList Algorithm is as follows:

```
Algorithm DivideList(input[],low,high) // divides the input array list into un breakable sub lists
// low is the starting index of input array list
// high is the ending index of input array list
{
 while(true) {
  if(low<high) { // if list is breakable
mid=(low+high)/2; // Dividing the list into two sub lists based upon mid index
rear=rear+1;
queue[rear].low=low; // storing starting index of left sub list into queue
queue[rear].high=mid;// storing ending index of left sub list into queue
rear=rear+1;
queue[rear].low=mid+1;//storing  starting index of right sub list into queue
queue[rear].high=high;  // storing ending index of right sub list into queue  }
else if (low==high) // if list is unbreakable
ArrayListCombine(ouput,k,input[low]);//combine the unbreakable list into output array list
(OR) LinkedListCombine(output,input[low]);  // combine the unbreakable list into linked list
if(front<rear) { // if queue is not empty
front=front+1;
low=queue[front].low; // Assign starting index to low
high=queue[front].high; // Assign ending index to high }
else  break;  } // end of while statement
} // end of DivideList Algorithm
```

Algorithm 2.1 Dividing unsorted input list into unbreakable sub lists

In Algorithm 2.1, An unsorted input array list is taken as input, it is divided into left sub list and right sub list based upon the mid element. The indices of left sub list and right sub list are placed into queue. If queue is not empty then taking indices of list and repeating the above steps until the queue becomes empty.

The time taken by DivideList Algorithm is as follows:

$T1(n) = 2T1(n/2) + C$

$T1(n)=2(2T1(n/4) + C) + C = 4T1(n/4)+C(2+1)$

$T1(n)=4(2T1(n/8)+C)+C(2+1)$

$T1(n)=2^3T1(n/8)+C(2^2+2+1)$

$T1(n)=2^kT1(n/2^k)+C(2^{k-1}+\text{------------}+2+1)$

$T1(n)=2^kT1(n/2^k)+C(2^k-1)$

Let $n=2^k$ $k=\log n$

$T1(n)=n+C(n-1)$

$T1(n)=O(n)$

The time taken to divide the input list into unbreakable sub lists is O(n).

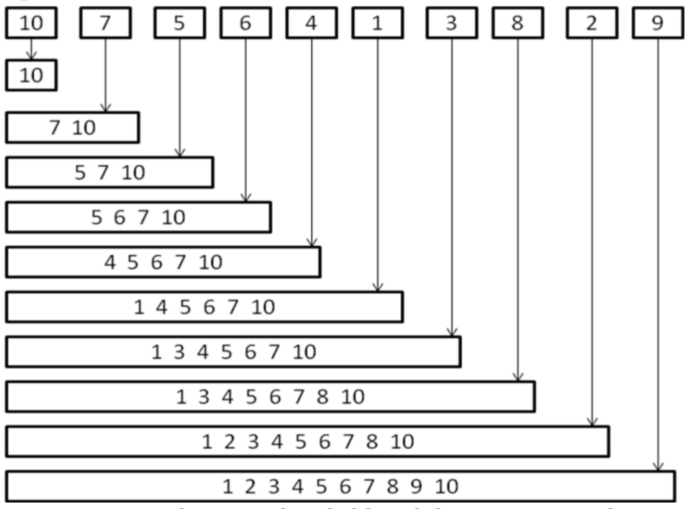


Fig 2.2 Combining unbreakable sub lists into output list

The proposed ArrayListCombine() algorithm is as follows:

```
Algorithm ArrayListCombine(output[],k,ele)
// output is the array list to which un breakable sub lists are combined.
//k is the size of output array list.
//ele is the element of un breakable sub list
{
 pos=k+1;
for(i=1;i<=k;i++) // selecting the suitable position of ele into output array list
 if(ele<output[i]) {
 pos=i;
break; } // end of if statement
for(i=k;i>=pos;i--)// Adjusting elements of output array list to place ele
output[i+1]=output[i];
output[pos]=ele;// placing ele into suitable position within output array list.
k=k+1;//incrementing the size of output array list
}// end of ArrayListCombine Algorithm
```

Algorithm 2.2 Combining unbreakable sub lists into output sorted Array List

In algorithm 2.2 three operations are performed on output array list. They are selecting suitable position for ele within array list, adjusting elements of output array list and placing ele into suitable position within output array list.

The time taken by ArrayListCombine Algorithm is as follows:

$T2(n)=1C+2C+3C+------+nC = (1+2+3+--------+n)C$

$T2(n)=(n(n+1)/2)C$

$T2(n)=(n^2+n)C$

$T2(n)=O(n^2)$

The proposed LinkedListCombine() algorithm is as follows:

```
Algorithm  LinkedListCombine(output,ele)
// output is linked list to combine un breakable sub lists.
// ele is the element of un breakable sub list, which is to be combined.
{
temp=new node; //  new node is created
temp->data=ele; // element is placed into data field of new node
temp->next=NULL;
if(output==NULL)  // if output list is empty{
 output=temp; // make new node as starting node of output list.
return; }
if(ele<output->data)  // if element is smaller than first element of output list{
temp->next=output; // make starting  node of output list as next node of new node
output=temp;// make new node as starting  node of output list
return;  }
t=ouput;
while(t->next!=NULL) // traverse the output list to find suitable position for new node  {
   if(ele<t->next->data) {
temp->next=t->next;
t->next=temp;
return; }
t=t->next; }
If(t->next==NULL) // if suitable position for new node is not found within output list{
t->next=temp; // make new node as ending node of output list
}
}
```

Algorithm 2.3 Combining un breakable sub lists into output sorted Linked List

In algorithm 2.3 a new node is created and element is placed into it. If output list is empty then make new node as starting node of output list, other wise traverse the output list to find suitable position for new node, if suitable position is found then place new node into output list, other wise make new node as last node of output list.

The time taken by LinkedListCombine Algorithm is as follows:

$T3(n)=C+C+C-------+C=nC$

$T3(n)=O(n)$

If input and output lists are array lists then Time taken by the proposed algorithm is:

$T(n)=T1(n)+T2(n)$

$T(n)=O(n)+O(n^2)$

$T(n)=O(n^2)$

If input list is array list and output list is linked list then Time taken by the proposed algorithm is:

$T(n)=T1(n)+T3(n)$

$T(n)=O(n)+O(n)$

$T(n)=O(n)$

III. IMPLEMENTATION OF PROPOSED ALGORITHM USING C

The Proposed Algorithm is implemented by using C Programming Language, which is a robust, structure oriented programming language, which provides different concepts like functions, pointers, structures, arrays and so on to solve complex problems.

```
/* Program to sort n elements using Proposed Algorithm (Divide and Combine Sorting Algorithm array list variation) */
# include <stdio.h>
# include <conio.h>
struct queue  {
  int low;
  int high;
 }; // end of structure queue
 struct queue q[20]; // queue to store starting and ending indices of sub lists
 int rear=0,front=0;
int input[100],output[100],k=0;
void display(int low,int high) { // display the elements of ouput sorted list
 int i;
 for(i=low;i<=high;i++)
 printf("%d ",output[i]);
}// end of display function
void combine(int ele) { // combining sorted sub lists into output list
  int i,pos=k+1;
  for(i=1;i<=k;i++)
  if(ele<output[i]) {
   pos=i;
   break; } // end of if statement
  for(i=k;i>=pos;i--)
  output[i+1]=output[i];
  output[pos]=ele;
  printf("\n");
  display(1,k);
  k++;  } // end of combine function
void divide(int low,int high) { // dividing the input list into un breakable sub lists
 int mid;
 while(1)  {
  if(low<high)  { Dividing the list into two sub lists
   mid=(low+high)/2;  // finding the mid index of a list
   rear++;
   q[rear].low=low; //storing start index of left sub list into queue
   q[rear].high=mid;//storing end index of left sub list into queue
   rear++;
   q[rear].low=mid+1;//storing start index of right sub list into queue
   q[rear].high=high;// storing end index of right sub list into queue
  }
  else if(low==high)// if the list is un breakable
  combine(input[low]);//combining un breakable list into output list
  if(front<rear) { //if queue is not empty
   front++;
   low=q[front].low; //assigning start index of sub list to low
   high=q[front].high;//assigning end index of sub list to high
  }
  else break; // if queue is empty
  }// end of while statement
 }// end of divide function
  void main() {
   int i,n;
   clrscr();
   printf("\nImplementation of Divide and Combine Sorting Algorithm");
   printf("\nEnter number of elements:\n");
   scanf("%d",&n);
   printf("\nEnter %d elements:\n",n);
   for(i=1;i<=n;i++)
   scanf("%d",&input[i]);
   divide(1,n);
   printf("\nAfter Sorting:\n");
   display(1,n);
   getch();
  }// end of program
```

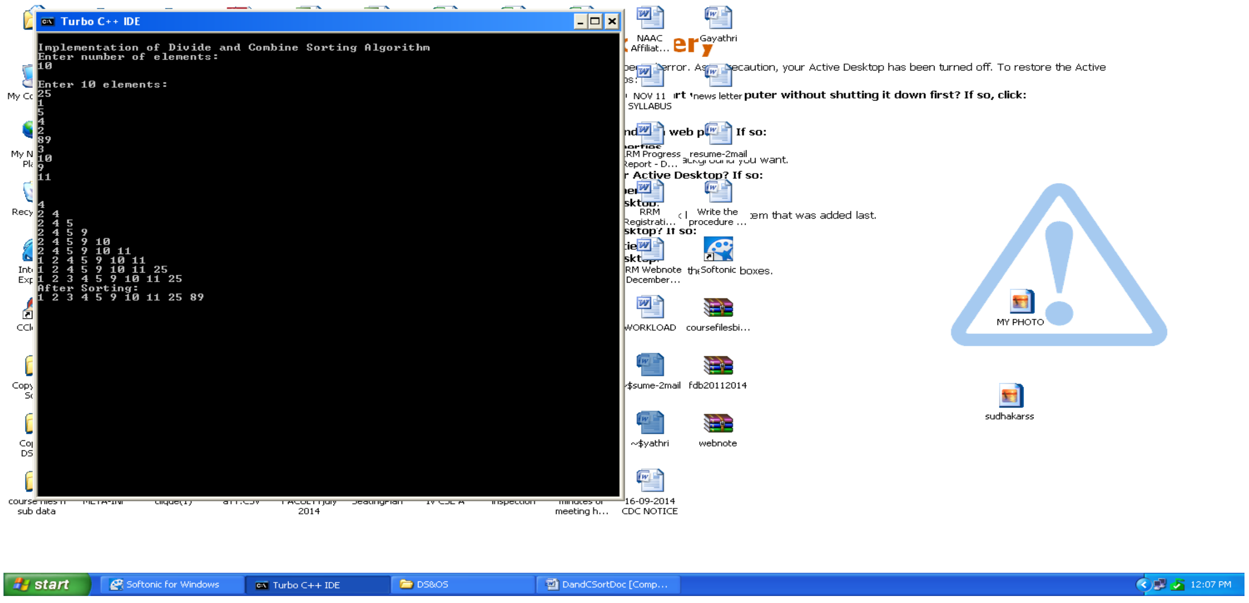
Program 3.1 Output Array List variation of proposed Algorithm

```
/* Program to sort n elements using Proposed Algorithm (Divide and Combine Sorting Algorithm Linked list variation)
*/
# include <stdio.h>
# include <conio.h>
struct queue {
  int low;
  int high;
 }; // end of queue data structure
 struct node{
   int data;
   struct node *next;
  }; // end of node structure
 struct queue q[20]; // queue to store starting and ending indices of  sub lists
 int rear=0,front=0;
int input[100]; // Input array list
struct node *output=NULL; // output linked list
void display(struct node *p) { // displaying the elements of output linked list
 printf("\n");
 while(p!=NULL) {
  printf("%d ",p->data);
  p=p->next;
 }
}
void combine(int ele)  { // combining un breakable sub lists into output linked list
 struct node *t,*temp;
 temp=(struct node *)malloc(sizeof(struct node)); // new node is created
 temp->data=ele; // element is placed into new node
 temp->next=NULL;
 if(output==NULL) // if output list is empty{
  output=temp; // make new node as starting node of output linked list
  display(output);
  return;
 }
 if(ele<output->data) // if element is smaller than first element of output list {
  temp->next=output; // make starting node of output list as next node of new node
  output=temp; // make new node as starting node of output list
  display(output);
  return;
 }
 t=output;
 while(t->next!=NULL) // traverse the output list to find suitable position for ele{
    if(ele < t->next->data)  {
     temp->next=t->next;
     t->next=temp;  // place the new node at suitable position within output list
     display(output);
     return;
    }
  t=t->next;
 }
 t->next=temp; // if suitable position not found, then make new node as last node of output list.
 display(output);
 }
void divide(int low,int high) {
 int mid;
 while(1)  {
  if(low<high) // if list is breakable {
   mid=(low+high)/2; // break the list into two sub lists based on mid index
   rear++;
   q[rear].low=low; // store starting index of left sub list into queue.
   q[rear].high=mid; //store ending index of left sub list into queue.
 rear++;
 q[rear].low=mid+1; //store starting index of right sub list into queue.
 q[rear].high=high; // store ending index of right sub list into queue.
}
else if(low==high)  // if list is un breakable
combine(input[low]); // combine un breakable list into output list
if(front<rear)  // if queue is not empty {
 front++;
 low=q[front].low; // select starting index of sub list to divide
 high=q[front].high; //select ending index of  sub list to divide
}
else break;
}
}
void main() {
 int i,n;
 clrscr();
 printf("\nImplementation of Divide and Combine Sorting Algorithm");
 printf("\nEnter number of elements:\n");
 scanf("%d",&n);
 printf("\nEnter %d elements:\n",n);
 for(i=1;i<=n;i++)
 scanf("%d",&input[i]);
 divide(1,n);
 printf("\nAfter Sorting:");
 display(output);
 getch();
 }
```

Program 3.2 Output Linked List variation of proposed Algorithm

IV. TESTING OF PROPOSED ALGORITHM

Different Test cases are considered to test the result of Program 3.1 and Program 3.2. The following are the screen shot, which shows the result of proposed algorithm. Consider the following test case of 10 elements {25 1 5 4 2 89 3 10 9 11}. The status of output list is {4},{2 4},{2 4 5},{2 4 5 9},{2 4 5 9 10},{2 4 5 9 10 11},{1 2 4 5 9 10 11}, {1 2 4 5 9 10 11},{1 2 4 5 9 10 11 25} ,{1 2 4 5 9 10 11 25 89}.



```
Implementation of Divide and Combine Sorting Algorithm
Enter number of elements:
10

Enter 10 elements:
25
1
5
4
2
89
3
10
9
11


4
2 4
2 4 5
2 4 5 9
2 4 5 9 10
2 4 5 9 10 11
1 2 4 5 9 10 11
1 2 4 5 9 10 11 25
1 2 3 4 5 9 10 11 25
After Sorting:
1 2 3 4 5 9 10 11 25 89
```

Fig 4.1 screen shot of Input and Output Lists

V. CONCLUSION

We conclude that in the proposed Algorithm, we are dividing the given unordered list into left sub list and right sub list based upon mid element. The indices of left sub list and right sub list are placed into queue. If queue is not empty, then indices of a list are taken from queue and the above process is repeated until queue becomes empty. Two variations of Combine algorithm ie, LinkedListCombine() algorithm and ArrayListCombine() algorithm are proposed and compared by deriving time complexities. DivideList() algorithm and two variations of Combine algorithm are implemented using C.

REFERENCES

[1] Aho, Alfred V. and Jeffrey D. Ullman [1983]. Data Structures and Algorithms. AddisonWesley, Reading, Massachusetts.

[2] Cormen, Thomas H., Charles E. Leiserson and Ronald L. Rivest [1990]. Introduction to Algorithms. McGraw-Hill, New York.

[3] Knuth, Donald. E. [1998]. The Art of Computer Programming, Volume 3, Sorting and Searching. Addison-Wesley, Reading, Massachusetts.