*IOSR Journal of Computer Engineering (IOSR-JCE)*
*e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 1, Ver. I (Jan. 2014), PP 20-22*
*www.iosrjournals.org*

# Alternate Sort

Syed Azher Nadeem Pasha
*Lecturer Department of Computer Science Bapuji Institute of Hi-tech Education Davangere - 577004 , Karnataka State , India*

***Abstract:*** *Sorting algorithms are the main concepts of the subject Data Structures and It's Applications. These algorithms are designed in arranging the data elements in the sorted order. If the data elements are arranged in sorted order , then the searching is very easier. Some algorithms are comparison sort and some are non-comparison sort. The choice of a algorithm is based on the efficiency of the algorithm. I have designed one algorithm called as Alternate Sort. The main aspect is that different technique of comparisons is involed. I have presented the algorithm , It's working and the examples and finally my paper is consisting of the program listing.*
***Keywords:*** *Alternate Array Efficiency Exchanges Sort*

## I. Introduction

The sorting algorithms are used in arranging the data elements in ascending order or descending order. Many sorting algorithms are existing like Selection sort , Insertion sort , Bubble sort , Quick sort , Shell sort , Radix sort etc. The choice of the algorithm depends on the efficiency of the algorithm. The efficiency of the algorithm depends on the number of comparisons , number of exchanges and the memory requirements for data storage.

The purpose of presenting my paper is to find the compatibility and different techniques of comparing the data elements. .

## II. Alternate Sort

### 2.1 Algorithm

```
K=0 ;
for (i=0 i<=n-1 ; ++i)
    begin
        for (j=k ; j<n-1 ; j=j+2)
            begin
               if a[j] > a[j+1]
                      a[j]<->a[j+1]
            end
   k=k+1
  if (k>1) then
             k=0 ;
end
```

#### 2.1.1 Working of above algorithm

This sort uses two loops . The outer loop controls the number of iterations . The inner loop is used to compare the data elements.

Step 1:
First time when the inner loop executes : The first two elements are compared . If they are out of order, the elements will be exchanged. Next the third and fourth elements are compared .If they are out of order the elements are exchanged. Process is repeated till the end of the list.When we reach the end of the list , if single element iremains then do not consider it for comparison , leave it as it is.

Step2 :
Second time when the inner loop executes : second and third elements are compared . If they are out of order the elements will be exchanged. Then fourth and fifth elements are compared . If they are out of order the elements are exchanged. Repeat the process till the end of the list. If one element remains do not consider it for comparison .

Step3 :
The same method as in Step1 is continued

Step 4 :
The same method as in step 2 is continued .

The complete process halts when the outer loop exits.

**2.2 Examples**

| Problem | Pass1 | Pass2 | Pass3 | Pass4 | Pass5 | Sorted Array |
|---|---|---|---|---|---|---|
| 5 | 5 | 4 | 4 | 2 | 2 | 1 |
| 4 | 4 | 5 | 2 | 4 | 1 | 2 |
| 3 | 3 | 2 | 5 | 1 | 4 | 3 |
| 2 | 2 | 3 | 1 | 5 | 3 | 4 |
| 1 | 1 | 1 | 3 | 3 | 5 | 5 |
| Comparisons | 2 | 2 | 2 | 2 | 2 | |
| exchanges | 2 | 2 | 2 | 2 | 2 | |

Total comparisons = 10 =5*4/2 n(n-1)/2 where n is number of elements in the array
Total exchanges = 10 = 5*4/2n(n0-1)/2

| Problem | Pass1 | Pass2 | Pass3 | Pass4 | Pass5 | Pass6 | Sorted Array |
|---|---|---|---|---|---|---|---|
| 6 | 6 | 5 | 5 | 3 | 3 | 1 | 1 |
| 5 | 5 | 6 | 3 | 5 | 1 | 3 | 2 |
| 4 | 4 | 3 | 6 | 1 | 5 | 2 | 3 |
| 3 | 3 | 4 | 1 | 6 | 2 | 5 | 4 |
| 2 | 2 | 1 | 4 | 2 | 6 | 4 | 5 |
| 1 | 1 | 2 | 2 | 4 | 4 | 6 | 6 |
| Comparisons | 3 | 2 | 3 | 2 | 3 | 2 | |
| exchanges | 3 | 2 | 3 | 2 | 3 | 2 | |

Total comparisons = 15=6*5/2=n*(n-1)/2 where n is number of element s in the array
Total exchanges =15=6*5/2=n*(n-1)/2

**2.3 Program listing :**

```
void main()
{
                        int i,j,k,n, temp , a[50] ;
clrscr () ;
printf("Enter the number of elements : ");
scanf("%d",&n) ;
printf("Enter array elements : \n") ;
for (i=0 ; i<=n-1 ; ++i)
                        {
                                scanf("%d",&a[i]) ;
                        }

                        k=0 ;
                        for (i=0 ; i<=n-1 ; ++i)
                                {
                                        for (j=k ; j<n-1 ; j=j+2 )
                                                {
                                                        if (a[j] > a[j+1] )
                                                                {
                                                                        temp = a[j] ;
                                                                        a[j] = a[j+1] ;
                                                                        a[j+1] = temp ;
                                                                }
                                                }

                                        k=k+1 ;
                                        if (k>1 )
```

```
                                        k= 0 ;

}

        printf("The array elements are : \n") ;
                        for (i=0 ; i<=n-1 ; ++i)
                                {
                                        printf("%d\n",a[i]) ;
                                }
                        getch() ;
}
```

## III. Efficiency :

The formula for number of comparisons is n*(n-1)/2
The formula for number of exchanges is n*(n-1)/2
The efficiency of the above algorithm is O(n2) w.r.t number of comparisons and also w.r.t number of exchanges.

## IV. Conclusion

The algorithm is simple to implement. It takes constant memory space. The efficiency is same as the efficiency of Bubble sort. This algorithm is used for relatively small sized arrays.

## REFERENCES

[1] Website referring the Sorting algorithm from Wickepedia