

A simplified approach in Sorting method: Bubble Sort, Selection Sort and Merge Sort

Lakshamma. T

Assistant Professor, Department of Computer Applications,
International Institute of Business Studies.

Abstract

Sorting is a method of arranging the elements in an order. There are different sorting algorithms available to sort the elements. In this article bubble sort, selection sort and merge sort algorithm is explained. The working process, the algorithm, and the C program version of these 3 sorting techniques are explained. Bubble sort technique sorts the elements by comparing each pair of adjacent elements from left to right. Swapping of elements is done if the elements are not in order. Selection sort method sorts the elements by selecting the smallest element at each pass and keeping it in the appropriate position. Merge sort method uses the divide and conquer technique to sort the list. The list is divided at each step until the list cannot be divided further. Then we combine the pair of one element lists into two-element lists, sorting them in the process.

Keywords: - Sorting, C Program for Bubble sort, C Program for Selection sort, C Program for Merge sort, Algorithm for Bubble sort, Algorithm for Selection sort, Algorithm for Merge sort

Date of Submission: 16-12-2023

Date of Acceptance: 26-12-2023

I. INTRODUCTION

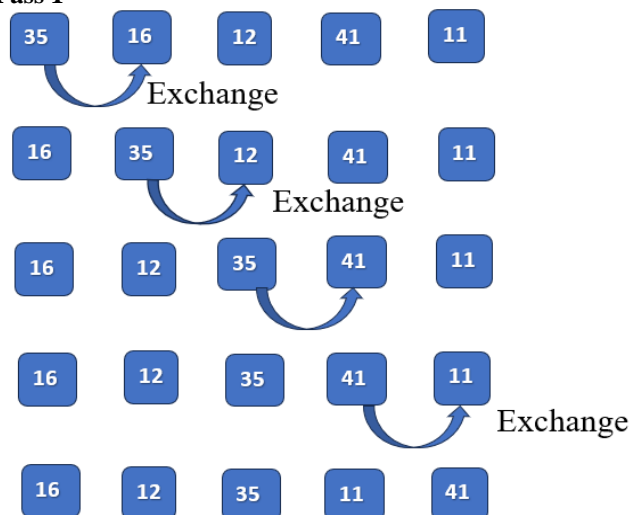
Sorting means arranging the elements in a particular order usually in ascending order. There are different sorting algorithms available like bubble sort, selection sort, insertion sort, quick sort, merge sort and so on. All these algorithms follow different techniques to sort the elements of the list. In this article the Bubble sort, Selection sort and Merge sort algorithm along with the C programming version of the algorithm is studied.

BUBBLE SORT

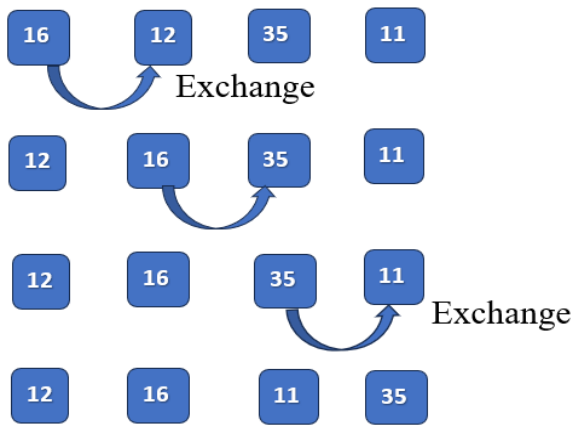
Bubble sort is a simple sorting technique. In this method sorting is done by comparing each pair of adjacent elements from left to right. Swapping of elements is done if the elements are not in order. During each pass the larger element "bubbles" to the top of the list. Due to this reason this sorting method is called Bubble Sorting.

The working of bubble sort can be explained for the list of numbers 35,16, 12,41,11 as follows.

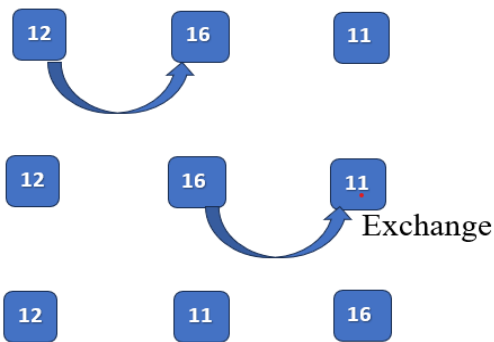
Pass 1



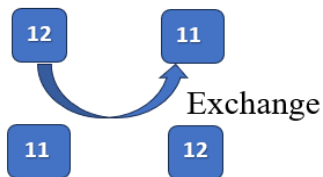
Pass 2



Pass 3



Pass 4



After Pass 4, the final list of sorted lists will be.



Algorithm for bubble sort

BubbleSort (arr[0...n-1])

Here **arr** is an array of **n** elements to be sorted

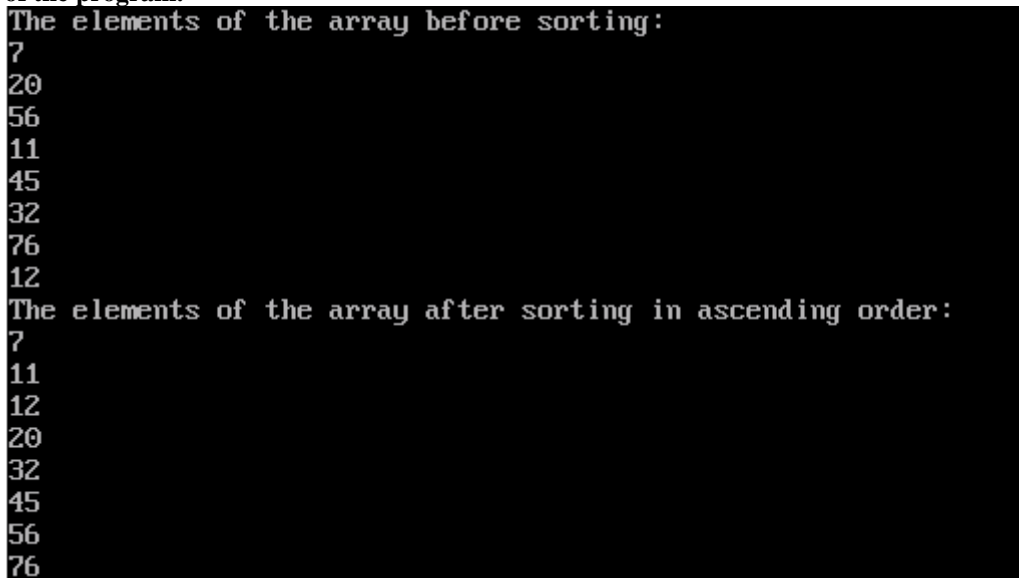
- Step 1: for i in 1 to n-1 do
- Step 2: for j in 1 to n-i-1 do
- Step 3: if arr[j] > arr[j+1] then
- Step 4: temp = arr[j]
- Step 5: arr[j] = arr[j+1]
- Step 6: arr[j+1] = temp
- Step 7: end for
- Step 8: end for

C program for bubble sort:

```
#include <stdio.h>
int main()
{
int array[8]={7,20,56,11,45,32,76,12};
```

```
int n=8,i,j,temp;
printf("The elements of the array before sorting:\n");
for (i= 0; i < n; i++)
printf("%d\n", array[i]);
for (i = 0 ; i < n-1; i++)
{
for (j = 0 ; j < n-i-1; j++)
{
if (array[j] > array[j+1]) /* For decreasing order use < */
{
temp = array[j];
array[j] = array[j+1];
array[j+1] = temp;
}
}
}
printf("The elements of the array after sorting in ascending order:\n");
for (i= 0; i < n; i++)
printf("%d\n", array[i]);
return 0;
}
```

Output of the program:



```
The elements of the array before sorting:
7
20
56
11
45
32
76
12
The elements of the array after sorting in ascending order:
7
11
12
20
32
45
56
76
```

Selection sort

Selection sort is another simplest algorithm. In this algorithm, the array is divided into two parts, the first list is the sorted part, and another list is the unsorted part. Initially, the sorted part of the array is empty, and the unsorted part is the given array. Sorted list is placed at the left side, while the unsorted list is placed at the right side. During each pass the smallest value among the unsorted elements of the array is selected and it is placed to its appropriate position into the array. In selection sort, during the first pass the first smallest element is selected and placed at the first position. Next during the second pass, the second smallest element is selected and placed in the second position. This process is continued until the array is completely sorted. This algorithm is not efficient for large arrays.

Let us take an array Arr[0], Arr [1], .. Arr [N-1] of elements. The whole process of selection sort is given below.

Pass 1: 1. Search the smallest element from Arr [0]----- Arr [N-1]

2. Interchange Arr [0] with smallest element

Result: Arr [0] is sorted.

Pass 2: 1. Search the smallest element from Arr [1]----- A[N-1]

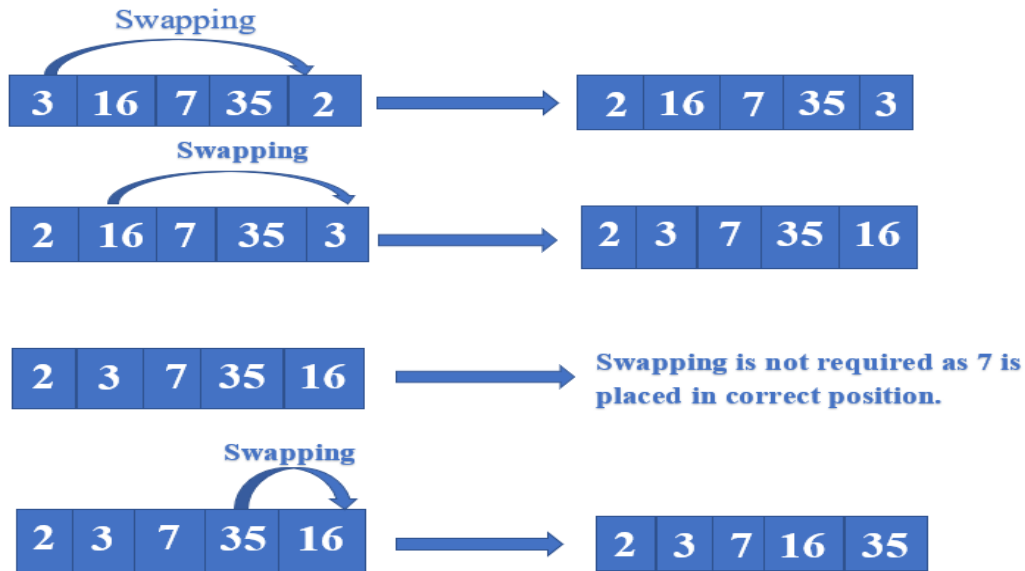
2. Interchange Arr [1] with the smallest element

Result: Arr [0], Arr [1] are sorted.

Pass N-1: 1. Search the smallest element from Arr [N-2]----- Arr [N-1]

2. Interchange Arr [N-2] with the smallest element.

The working of selection sort can be explained for the list of numbers 3,16, 7,35,2 as follows.



Algorithm for selection sort:

Step 1 – Set SMALL to location 0

Step 2 – Search the minimum element in the list

Step 3 – Swap the element with value at location SMALL

Step 4 – Increment SMALL to point to next element

Step 5 – Repeat until list is sorted

C program for selection sort:

```
#include<stdio.h>
void selectionsort(int array[], int n)
{
    int i,j,small,temp;
    for(i=0;i<n-1;i++)
    {
        small=i;
        for(j=i+1;j<n;j++)
        {
            if(array[j]<array[small])
                small=j;
        }
        temp=array[small];
        array[small]=array[i];
        array[i]=temp;
    }
    printf("\nArray elements after sorting:");
    for(i=0;i<n;i++)
        printf("\n%d",array[i]);
}
int main()
{
    int array[]={75,8,1,16,48,3,7,0}, i,n=8;
    printf("Array elements before sorting:");
    for(i=0;i<n;i++)
```

```
printf("\n%d",array[i]);  
selectionsort(array,n);  
return 0;  
}
```

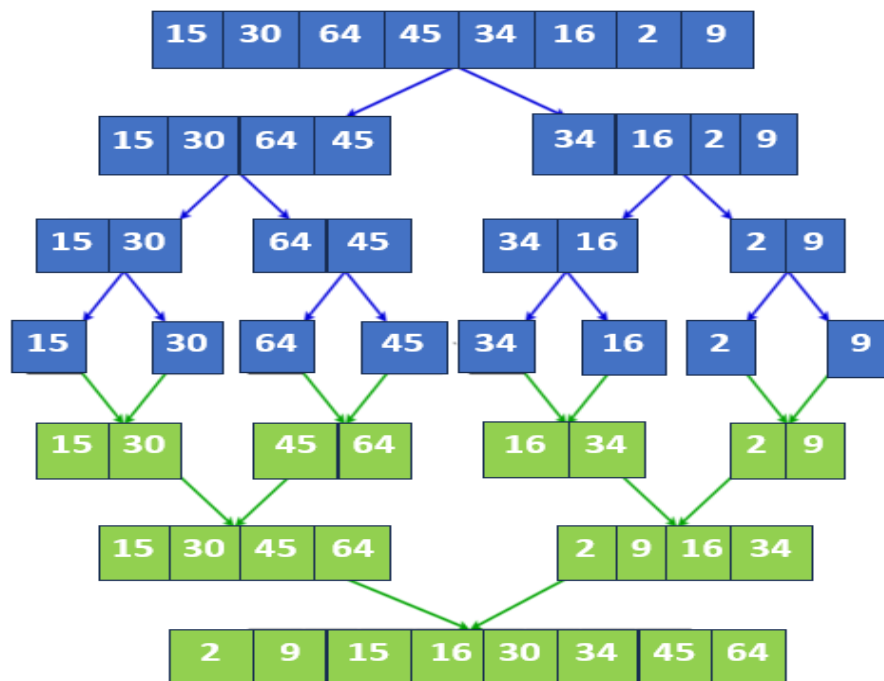
Output of the program:

```
Array elements before sorting:  
75  
8  
1  
16  
48  
3  
7  
0  
Array elements after sorting:  
0  
1  
3  
7  
8  
16  
48  
75_
```

Merge sort:

Merge sort method uses divide and conquer approach to sort the elements. It divides the given list into two equal halves, calls itself for the two halves and then merges the two sorted halves. The sub-lists are divided again and again into halves until the list cannot be divided further. Then we combine the pair of one element lists into two-element lists, sorting them in the process. The sorted two-element pair is merged into the four-element lists, and so on until we get the sorted list.

The working of merge sort can be explained for the list of numbers 15,30,64,45,34,16,2,9 as follows.



Merge sort algorithm

Algorithm MergeSort(arr, beg, end) :

In this algorithm **arr** is the given array, **beg** is the starting element of the array, and **end** is the last element of the array.

Step 1: if beg < end then

Step 2: mid = (beg + end)/2

Step 3: MergeSort(arr, beg, mid)

Step 4: MergeSort(arr, mid + 1, end)

Step 5: Merge(arr, beg, mid, end)

Step 6: end if

Algorithm: Merge(arr, beg, mid, end)

Here arr is the array with (end -beg+1) elements. The variables beg, mid and end are used to identify the position of elements in an array for each partition.

Step 1: Initialize i = beg, j = mid +1, k = beg

Step 2: While ((i <= mid) and (j <= high)) do Step 3

Step 3: if(arr[i] < arr[j])

c[k] = arr[i]

k = k+1

i = i+1

else

c[k] = arr[j]

k = k+1

j = j+1

end if

End While (step 2)

Step 4: While (i <= mid)

c[k] = arr[i]

k = k+1

i = i+1

End While (Step 4)

Step 5: While (j <= high)

c[k] = arr[j]

k = k+1

j = j+1

End While (Step 5)

Step 6: for i = low to k -1

arr[i] = c[i]

End for (Step 6)

Step 7: Return

C program for Merge sort:

```
#include <stdio.h>
```

```
void printArray(int A[], int n)
```

```
{
```

```
int i;
```

```
for (i = 0; i < n; i++)
```

```
printf("\t%d ", A[i]);
```

```
}
```

```
void merge(int A[], int mid, int low, int high)
```

```
{
```

```
int i, j, k, B[100];
```

```
i = low;
```

```
j = mid + 1;
```

```
k = low;
```

```
while (i <= mid && j <= high)
```

```
{
```

```
if (A[i] < A[j])
```

```
{
```

```
B[k] = A[i];
```

```
i++;
```

```
k++;
}
else
{
B[k] = A[j];
j++;
k++;
}
}
while (i <= mid)
{
B[k] = A[i];
k++;
i++;
}
while (j <= high)
{
B[k] = A[j];
k++;
j++;
}
for ( i = low; i <= high; i++)
{
A[i] = B[i];
}
}
void mergeSort(int A[], int low, int high)
{
int mid;
if(low<high)
{
mid = (low + high) /2;
mergeSort(A, low, mid);
mergeSort(A, mid+1, high);
merge(A, mid, low, high);
}
}
int main()
{
int A[] = {19, 21, 44, 56, 24, 95, 76,10};
int n = 8;
printf(" Array elements before sorting:\n");
printArray(A, n);
mergeSort(A, 0, 7);
printf("\n Array elements after sorting:\n"),
printArray(A, n);
return 0;
}
```

Output of the program

```
Array elements before sorting:
 19   21   44   56   24   95   76   10
Array elements after sorting:
 10   19   21   24   44   56   76   95 _
```

II. CONCLUSION

In this paper 3 different sorting techniques are discussed briefly. The sorting methods discussed are Bubble sort, Selection sort and Merge sort. Simplified examples for each of the sorting methods are explained.

The algorithms and the C Programming version of the different sorting methods are discussed. The C program is modified by including the input values in the program itself, so that we can concentrate on the logic of the sorting method and to run the program faster. This makes understanding of the program easier. The example given makes the concept of each sorting method clearer.

REFERENCES

- [1]. Srikanth S. Data Structures Using C, Skyward Publishers , 2021
- [2]. E Balagurusamy, Programming In Ansi C, Mcgraw Hill Education, 2019.
- [3]. Narasimha Karumanchi, Data Structures And Algorithms Made Easy: Data Structures And Algorithmic Puzzles, 2023
- [4]. <https://www.educba.com/sorting-in-c/>
- [5]. <https://www.javatpoint.com/selection-sort>
- [6]. <https://www.javatpoint.com/merge-sort>
- [7]. https://www.codewithharry.com/videos/data-structures-and-algorithms-in-hindi-59/#google_vignette
- [8]. <https://www.tutorialspoint.com/explain-the-sorting-techniques-in-c-language>
- [9]. <https://www.geeksforgeeks.org/c-program-to-sort-an-array-in-ascending-order/>
- [10]. <https://www.programiz.com/dsa/sorting-algorithm>