Code, Compile & Run

main.c

C (gcc 6.3)

```c
#include <stdio.h>

#define max 10

int a[11] = { 10, 14, 19, 26, 27, 31, 33, 35, 42, 44, 0 };
int b[10];

void merging(int low, int mid, int high) {
   int l1, l2, i;

   for(l1 = low, l2 = mid + 1, i = low; l1 <= mid && l2 <= high; i++)
   {
      if(a[l1] <= a[l2])
         b[i] = a[l1++];
      else
         b[i] = a[l2++];
   }

   while(l1 <= mid)
      b[i++] = a[l1++];

   while(l2 <= high)
      b[i++] = a[l2++];

   for(i = low; i <= high; i++)
      a[i] = b[i];
}

void sort(int low, int high)
```

223

Open File

Custom Input

Run

Status Successfully executed **Date** 2020-06-17 05:34:12 **Time** 0 sec **Mem** 9.424 kB

Output

```
List before sorting
10 14 19 26 27 31 33 35 42 44 0
List after sorting
0 10 14 19 26 27 31 33 35 42 44
```

Code, Compile & Run

main.c

C (gcc 6.3)

```
31      int mid;
32
33      if(low < high)
34      {
35          mid = (low + high) / 2;
36          sort(low, mid);
37          sort(mid+1, high);
38          merging(low, mid, high);
39      } else {
40          return;
41      }
42  }
43
44  int main()
45  {
46      int i;
47
48      printf("List before sorting\n");
49
50      for(i = 0; i <= max; i++)
51          printf("%d ", a[i]);
52
53      sort(0, max);
54
55      printf("\nList after sorting\n");
56
57      for(i = 0; i <= max; i++)
58          printf("%d ", a[i]);
59  }
```

Open File

Custom Input

Run

Status Successfully executed **Date** 2020-06-17 05:34:12 **Time** 0 sec **Mem** 9.424 kB

Output

```
List before sorting
10 14 19 26 27 31 33 35 42 44 0
List after sorting
0 10 14 19 26 27 31 33 35 42 44
```

C Program to implement merge Sort

Algorithm:

Step 1: Start

Step 2: MergeSort (arr[], l, r), where l is the index of the first element & r is the index of the last element.

Step 3: If r > l

Step 4: Find the middle index of the array divide it in two halves.

$m = (l + r)/2$

Step 5: Call Merge sort for second half:

mergesort (array, l, m)

Step 6: Call mergesort for second half:

mergesort (array, m+1, r)

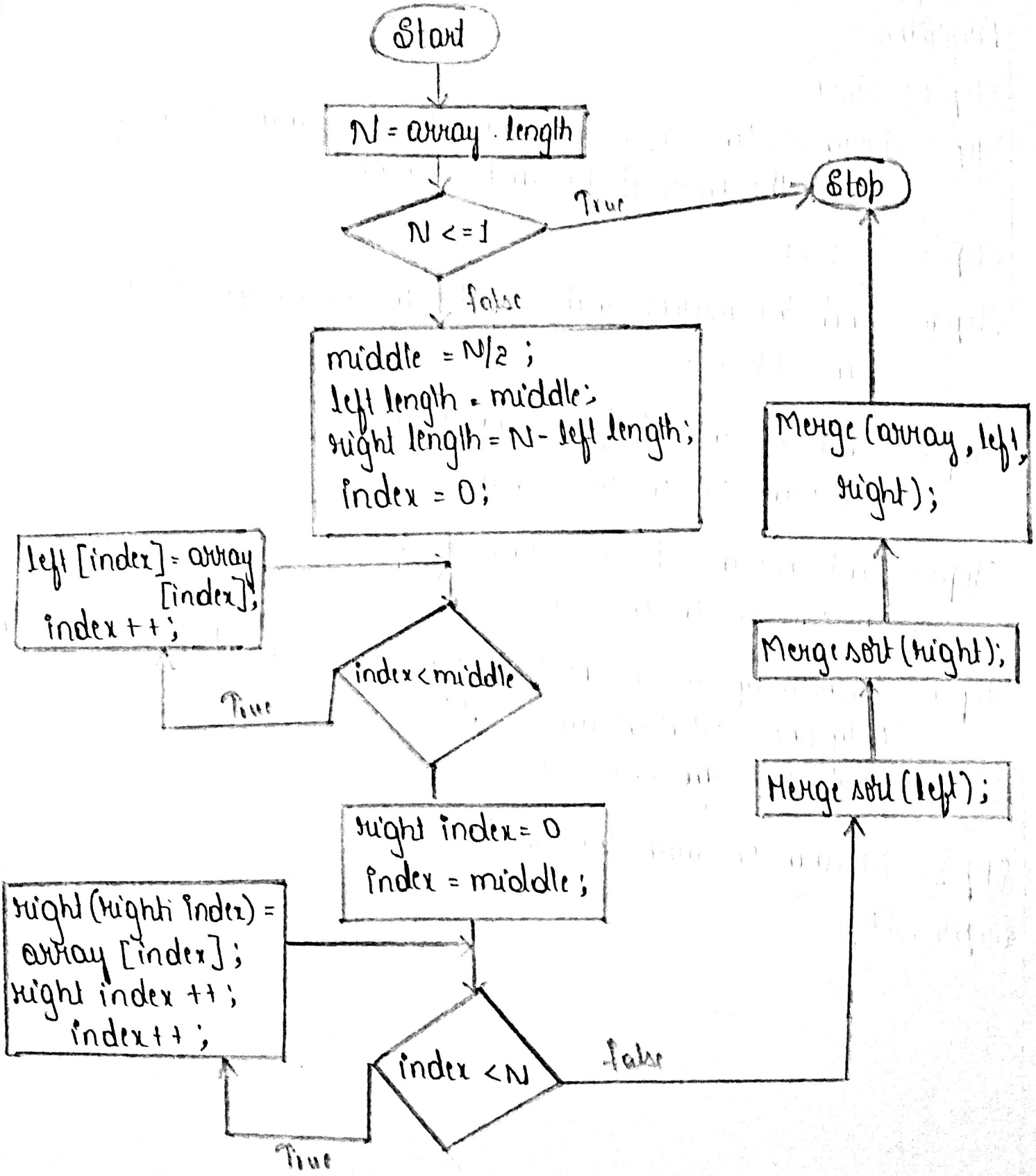
Step 7: Recursively, merge the two halves in a sorted manner, so that only one sorted array is left:

merge (array, l, m, r)

Step 8: Moving on with this article

Step 9: Stop

Flowchart:


Start
N = array . length
N <= 1
True
Stop
false
middle = N/2 ;
left length = middle;
right length = N - left length;
index = 0;
left [index] = array [index];
index ++;
index < middle
True
right index = 0
index = middle;
right (right index) = array [index];
right index ++;
index ++;
index < N
false
True
Merge sort (left);
Merge sort (right);
Merge (array, left, right);