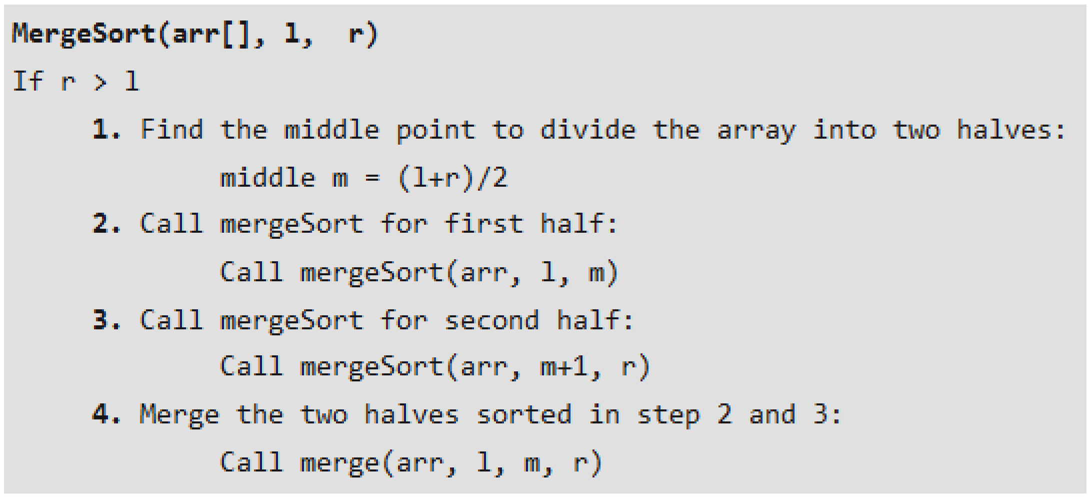
Merge Sort

Merge Sort is a Divide and Conquer algorithm. It divides input array in two halves, calls itself for the two halves and then merges the two sorted halves. **The merge() function** is used for merging two halves. The merge(arr, l, m, r) is key process that assumes that arr[l..m] and arr[m+1..r] are sorted and merges the two sorted sub-arrays into one.

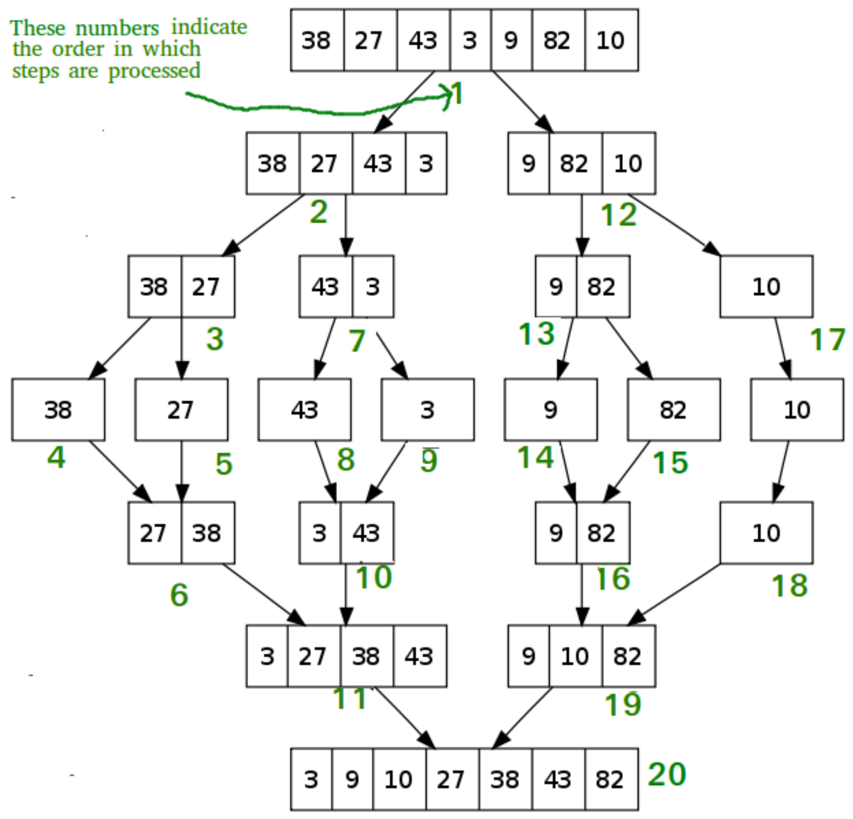
The following diagram from wikipedia shows the complete merge sort process for an example array {38, 27, 43, 3, 9, 82, 10}. If we take a closer look at the diagram, we can see that the array is recursively divided in two halves till the size becomes 1. Once the size becomes 1, the merge processes comes into action and starts merging arrays back till the complete array is merged.

Implementation of the merge sort algorithm:



```
MergeSort(arr[], l,  r)
If r > l
     1. Find the middle point to divide the array into two halves:
             middle m = (l+r)/2
     2. Call mergeSort for first half:
             Call mergeSort(arr, l, m)
     3. Call mergeSort for second half:
             Call mergeSort(arr, m+1, r)
     4. Merge the two halves sorted in step 2 and 3:
             Call merge(arr, l, m, r)
```

https://www.geeksforgeeks.org/merge-sort/


These numbers indicate the order in which steps are processed

To simplify the illustration above:

Divide (Just divide the sub-arrays but maintain order):

[38,27,43,3,9,82,10]

[38,27,43,3] [9,82,10]

[38,27] [43,3] [9,82] [10]

[38] [27] [43] [3] [9] [82] [10]

Merge (Sort the elements then merge):

[27,38] [3,43] [9,82] [10]

[3,27,38,43] [9,10,82]

[3,9,10,27,38,43,82]

https://www.geeksforgeeks.org/merge-sort/

```c
/* C program for Merge Sort */

#include<stdlib.h>
#include<stdio.h>

// Merges two subarrays of arr[].
// First subarray is arr[l..m]
// Second subarray is arr[m+1..r]
void merge(int arr[], int l, int m, int r)
{
	int i, j, k;
	int n1 = m - l + 1;
	int n2 = r - m;

	/* create temp arrays */
	int L[n1], R[n2];

	/* Copy data to temp arrays L[] and R[] */
	for (i = 0; i < n1; i++)
		L[i] = arr[l + i];
	for (j = 0; j < n2; j++)
		R[j] = arr[m + 1+ j];

	/* Merge the temp arrays back into arr[l..r]*/
	i = 0; // Initial index of first subarray
	j = 0; // Initial index of second subarray
	k = l; // Initial index of merged subarray
	while (i < n1 && j < n2)
	{
		if (L[i] <= R[j])
		{
			arr[k] = L[i];
			i++;
		}
		else
		{
			arr[k] = R[j];
			j++;
		}
		k++;
	}

	/* Copy the remaining elements of L[], if there
	are any */
	while (i < n1)
```

```
        {
                arr[k] = L[i];
                i++;
                k++;
        }

        /* Copy the remaining elements of R[], if there
        are any */
        while (j < n2)
        {
                arr[k] = R[j];
                j++;
                k++;
        }
}

/* l is for left index and r is right index of the
sub-array of arr to be sorted */
void mergeSort(int arr[], int l, int r)
{
        if (l < r)
        {
                // Same as (l+r)/2, but avoids overflow for
                // large l and h
                int m = l+(r-l)/2;

                // Sort first and second halves
                mergeSort(arr, l, m);
                mergeSort(arr, m+1, r);

                merge(arr, l, m, r);
        }
}

/* UTILITY FUNCTIONS */
/* Function to print an array */
void printArray(int A[], int size)
{
        int i;
        for (i=0; i < size; i++)
                printf("%d ", A[i]);
        printf("\n");
}
```

https://www.geeksforgeeks.org/merge-sort/

```
/* Driver program to test above functions */
int main()
{
	int arr[] = {12, 11, 13, 5, 6, 7};
	int arr_size = sizeof(arr)/sizeof(arr[0]);

	printf("Given array is \n");
	printArray(arr, arr_size);

	mergeSort(arr, 0, arr_size - 1);

	printf("\nSorted array is \n");
	printArray(arr, arr_size);
	return 0;
}
```

Below is a link for the merge sort tutorial:

https://www.youtube.com/watch?v=JSceec-wEyw&feature=emb_logo

https://www.geeksforgeeks.org/merge-sort/

https://www.geeksforgeeks.org/merge-sort/