Merge Sort

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

A visual representation of the sequence of events taking place. The process is recursive, meaning that `sort(int arr[], int l, int r)` is recursively called inside of itself before previous calls to the function are finished.

Note:
- The boxes represent separate function calls.
- The argument values are array indices (0-indexed)

```
void sort(int arr[], int l, int r)
```
- Determines midpoint and splits the current array into 2 halves
- Recursively calls itself on 1st and 2nd halves
- Calls merge() – to sort and merge the 1st and 2nd halves, after they have been sorted by their recursive calls

```
void merge(int arr[], int l, int m, int r)
```
- Merges two subarrays that are assumed to already be sorted
- **"Merging"** two subarrays is basically taking two already-sorted lists, and then sorting between the two of them to merge them together. While the subarrays are already sorted, another sorting is needed to splice the two lists together.
- Similar to merging two lists/linked lists

Sort(arr, 0 ,6)

L = 0
R = 6
M = 3

Step 1

Sort(arr, 0, 3)

L = 0
R = 3
M = 1

Step 2

Sort(arr, 0, 1)

L = 0
R = 1
M = 0

Step 3

Sort(arr, 0, 0)

L = 0
R = 0

L < R == false
Return

Step 4

Sort(arr, 1, 1)

L = 1
R = 1

L < R == false
Return

Step 5

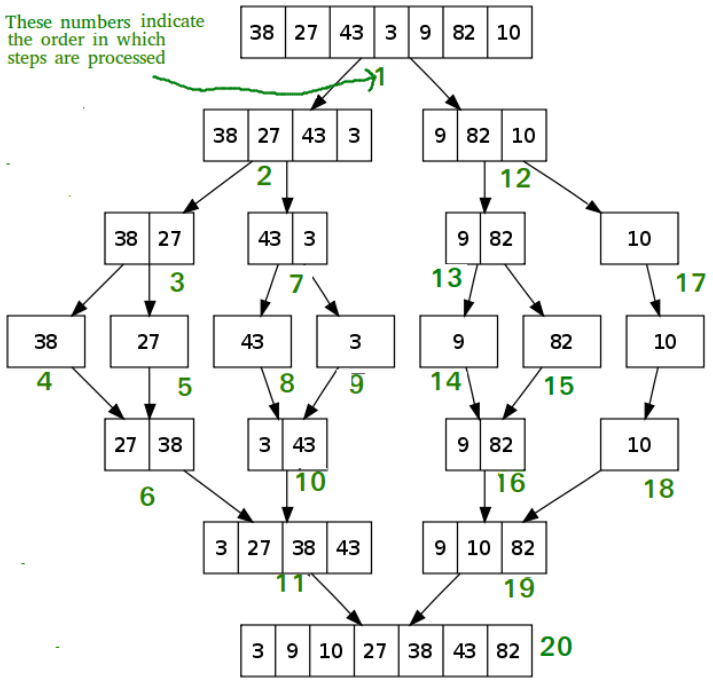
merge(arr, 0, 0, 1)

do the merge

Step 6

Sort(arr, 2, 3)

L = 2



```
void sort(int arr[], int l, int r)
    {
        if (l < r) {
            // Find the middle point
            int m =l+ (r-l)/2;

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

            // Merge the sorted halves
            merge(arr, l, m, r);
        }
    }
```

R = 3
M = 2

Step 7

Sort(arr, 2, 2)

L < R == false
Return

Step 8

Sort(arr, 3, 3)

L < R == false
Return

Step 9

merge(arr, 2, 2, 3)

do the merge

Step 10

Merge(arr, 0, 1, 3)

Do the merge

Step 11

Sort(arr, 4, 6)

L = 4
R = 6
M = 5

Step 12

Sort(arr, 4, 5)

L = 4
R = 5
M = 4

Step 13

Sort(arr, 4, 4)

L < R == false
Return

Step 14

Sort(arr, 5, 5)

L < R == false
Return

Step 15

merge(arr, 4, 4, 5)

do the merge

Step 16

Sort(arr, 6, 6)

L < R == false
Return

Step 17-18

merge(arr, 4, 5, 6)

do the merge

Step 19

Merge(arr, 0, 3, 6)

Do the merge

Step 20