MERGE SORT

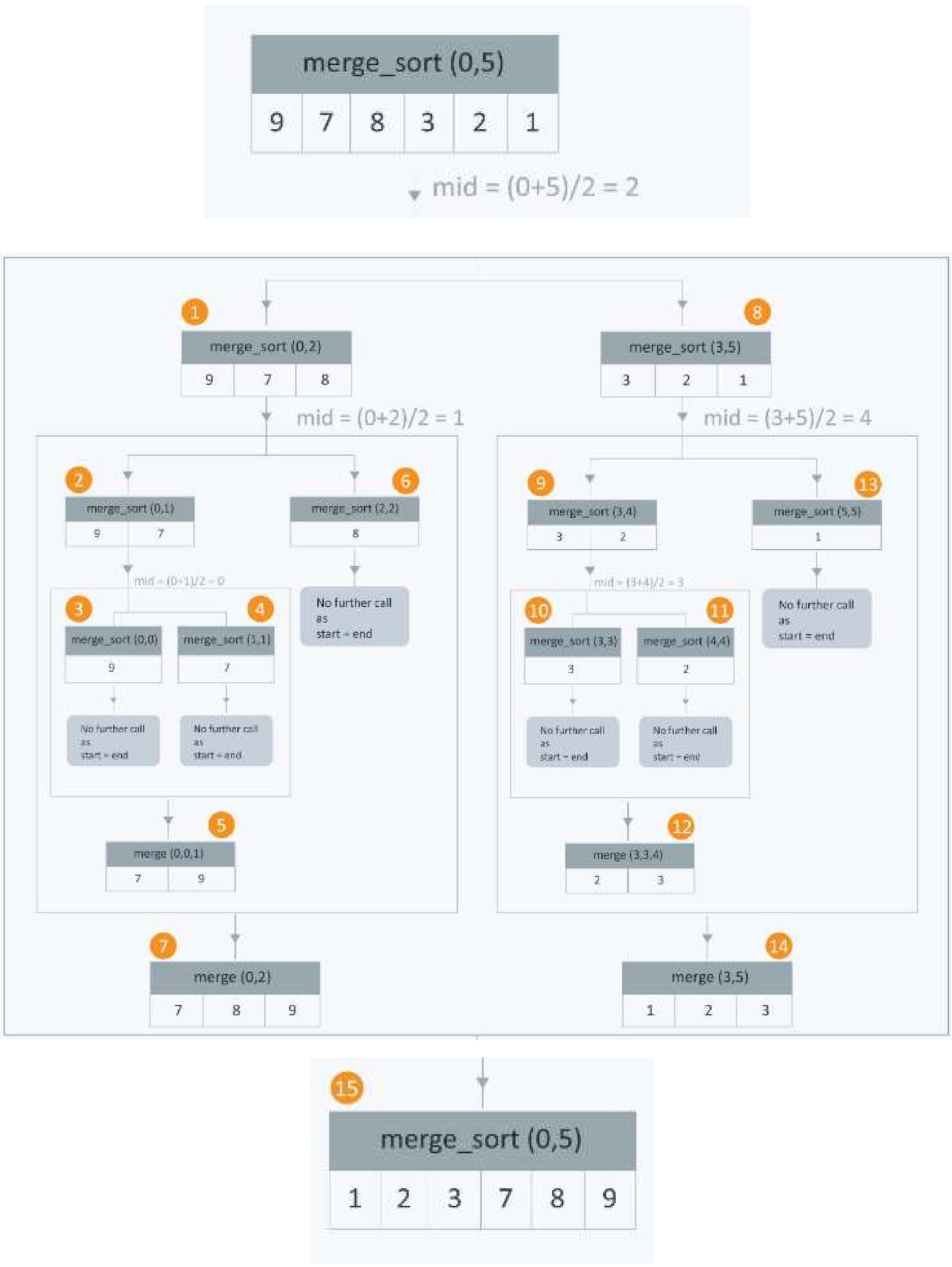
Merge sort is a divide-and-conquer algorithm based on the idea of breaking down a list into several sub-lists until each sublist consists of a single element and merging those sublists in a manner that results into a sorted list.

Idea:

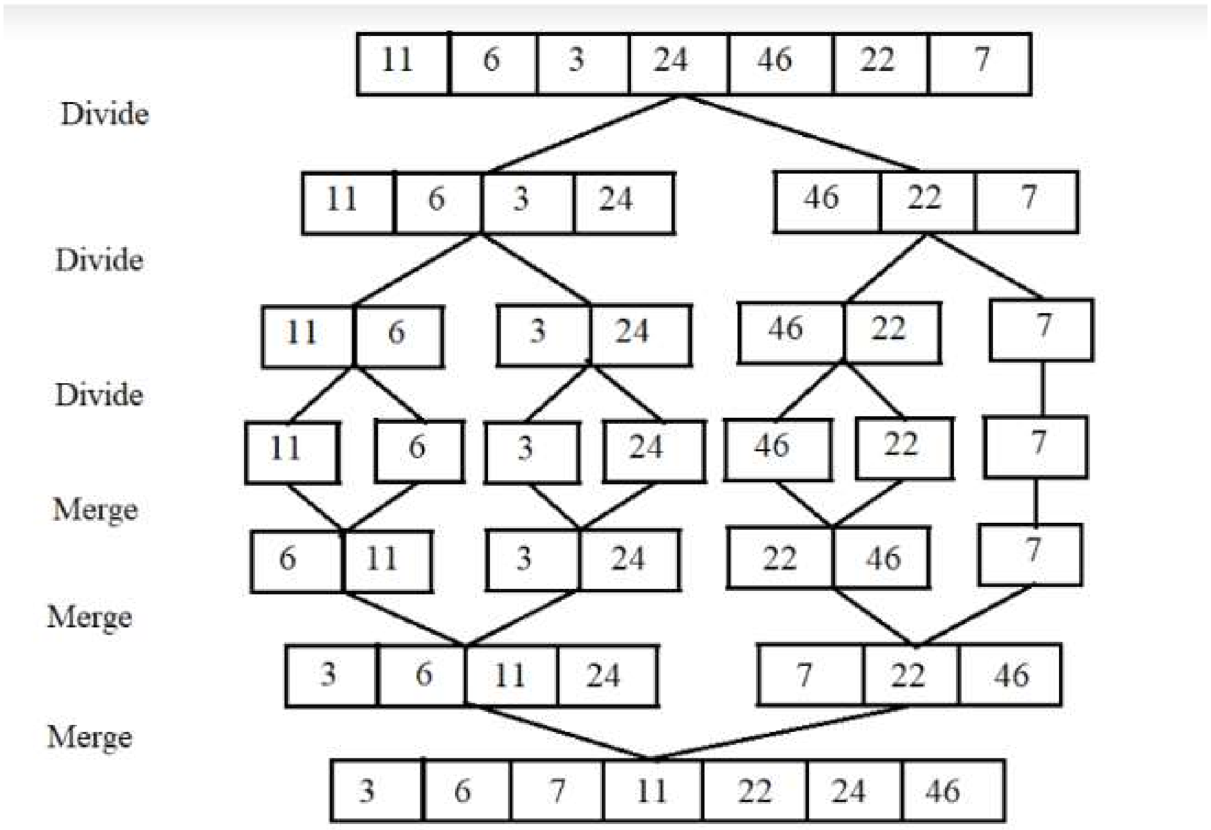
- Divide the unsorted list into N sublists, each containing 1 element.
- Take adjacent pairs of two singleton lists and merge them to form a list of 2 elements. N will now convert into N/2 lists of size 2.
- Repeat the process till a single sorted list of obtained.

While comparing two sublists for merging, the first element of both lists is taken into consideration. While sorting in ascending order, the element that is of a lesser value becomes a new element of the sorted list. This procedure is repeated until both the smaller sublists are empty and the new combined sublist comprises all the elements of both the sublists.

Consider the following example


merge_sort (0,5)
9 7 8 3 2 1
mid = (0+5)/2 = 2
1
merge_sort (0,2)
9 7 8
mid = (0+2)/2 = 1
8
merge_sort (3,5)
3 2 1
mid = (3+5)/2 = 4
2
merge_sort (0,1)
9 7
mid = (0+1)/2 = 0
6
merge_sort (2,2)
8
No further call as start = end
3
merge_sort (0,0)
9
No further call as start = end
4
merge_sort (1,1)
7
No further call as start = end
5
merge (0,0,1)
7 9
9
merge_sort (3,4)
3 2
mid = (3+4)/2 = 3
13
merge_sort (5,5)
1
No further call as start = end
10
merge_sort (3,3)
3
No further call as start = end
11
merge_sort (4,4)
2
No further call as start = end
12
merge (3,3,4)
2 3
7
merge (0,2)
7 8 9
14
merge (3,5)
1 2 3
15
merge_sort (0,5)
1 2 3 7 8 9

Consider another example



Pseudocode

```
void merge(int A[ ] , int start, int mid, int end) {
 //stores the starting position of both parts in temporary variables.
int p = start ,q = mid+1;
int Arr[end-start+1] , k=0;
for(int i = start ;i <= end ;i++) {
    if(p > mid)      //checks if first part comes to an end or not .
      Arr[ k++ ] = A[ q++] ;
   else if ( q > end)   //checks if second part comes to an end or not
       Arr[ k++ ] = A[ p++ ];
```

```
  else if( A[ p ] < A[ q ])     //checks which part has smaller element.
     Arr[ k++ ] = A[ p++ ];
  else
     Arr[ k++ ] = A[ q++];
 }
 for (int p=0 ; p< k ;p ++) {
  /* Now the real array has elements in sorted manner including both
       parts.*/
    A[ start++ ] = Arr[ p ] ;
 }
}
void merge_sort (int A[ ] , int start , int end )
   {
          if( start < end ) {
          int mid = (start + end ) / 2 ;          // defines the current array in 2 parts .
          merge_sort (A, start, mid);                // sort the 1st part of array.
          merge_sort (A, mid+1, end);              // sort the 2nd part of array.

         // merge the both parts by comparing elements of both the parts.
          merge(A, start , mid , end );
   }
}
```