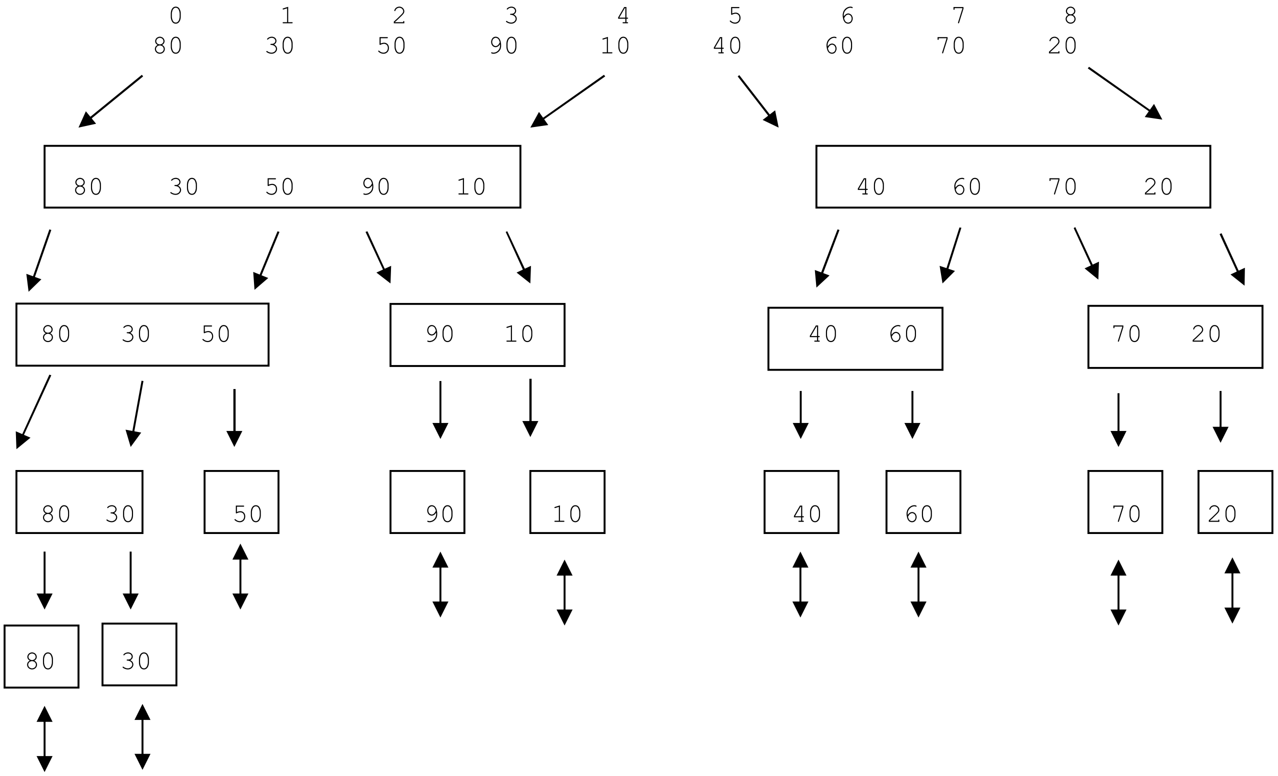
Merge sort

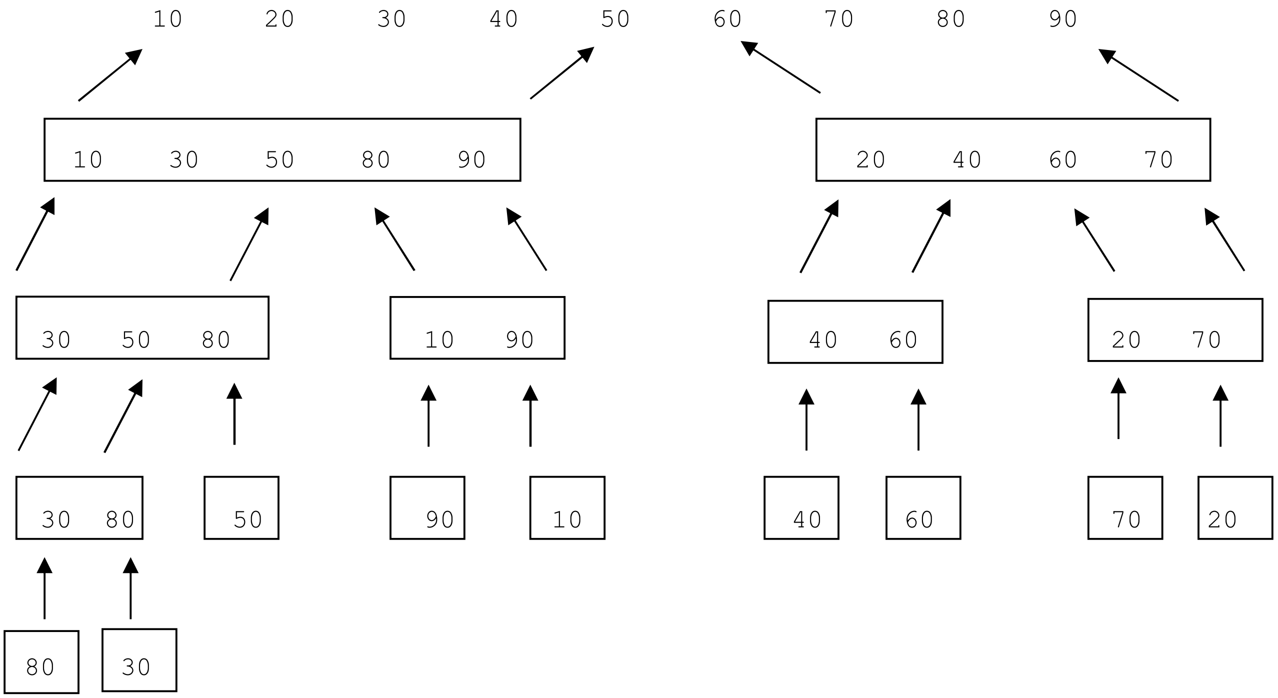
Merge sort is another divide-and-conquer algorithm. In general, merge sort works in a manner similar to hoare sort -- splits the data into two subsets, solves the problem on the smaller sets, and then combines the sets again. Merge sort is more straightforward -- breaks the array into two subsets by cutting it in the middle, sorts both subsets (recursively), and then combines them again.

```
//------------------------------------------------------------------------
// merge
// merge two sorted arrays into one long sorted array

void merge(int a[], int low, int mid, int high) {
   int temp[high - low + 1];

   int low1 = low;
   int high1 = mid;
   int low2 = mid + 1;
   int high2 = high;

   // As long as both lists still have elements, add the next.
   int index;
   for (index = 0; (low1 <= high1) && (low2 <= high2); index++) {
      if (a[low1] < a[low2]) {
         temp[index] = a[low1];
         low1++;
      }
      else {
         temp[index] = a[low2];
         low2++;
      }
   }

   // One of the lists still has elements, so add them now.
   for (; low1 <= high1; index++, low1++)
      temp[index] = a[low1];

   for (; low2 <= high2; index++, low2++)
      temp[index] = a[low2];

   // Copy back into the array.
   for (index = 0; index < high - low + 1; index++)
      a[index + low] = temp[index];
}

//------------------------------------------------------------------------
// mergeSort
// Break the array into two subsets by cutting it in the middle,
// sorts both subsets and combine

void mergeSort(int a[], int low, int high) {
   if (low < high) {
      int mid = (low + high) / 2;
      mergeSort(a, low, mid);
      mergeSort(a, mid + 1, high);

      merge(a, low, mid, high);
   }
}
```

When executing the merge sort, on the way down of the recursion, the items are split into parts, but the actual work of merging the smaller sets into sorted bigger sets is done on the way back up from the recursion. These execution trees show calling the recursive mergeSort, the "down", and the returning, the "up", separately.



0 1 2 3 4 5 6 7 8
80 30 50 90 10 40 60 70 20
80 30 50 90 10
40 60 70 20
80 30 50
90 10
40 60
70 20
80 30
50
90
10
40
60
70
20
80
30



10 20 30 40 50 60 70 80 90
10 30 50 80 90
20 40 60 70
30 50 80
10 90
40 60
20 70
30 80
50
90
10
40
60
70
20
80
30