```
int x;
          x = middle;
          while(low < x && middle < high) {
               if(ar[low] < ar [middle]) {
                    low++;
               }
               else {
                    replace(low, middle, ar);
                    middle++;
               }
          }

          return ar;
```

O(n) is the best case because we are just comparing one element that is not in the correct order. In this case middle is n.

Worst case is when majority of the elements are in the other array and we need to move them to the front and that's when n*n o(n^2). In the merge process.

The sort is stable because we are merging the elements from the other array to the first and sorting them by value.