**package** midterm1\_problem1;

// in this problem I will use merge sort

**public** **class** sorting {

**public** **static** **void** main(String[] args){

**int**[] a = {15,9,60,44,12,1,4};

**int** size = a.length;

**int**[] result = *merge\_sort*(a,size);

System.***out***.println("Sorted array is:");

**for**(**int** i = 0; i < size; i++){

System.***out***.print(result[i] + " ");

}

}

**public** **static** **int**[] merge\_sort(**int**[] a, **int** size){

**int**[] result = **new** **int**[size];

*merge\_sort\_recursive*(a, result, 0, size-1);

**return** result;

}

**public** **static** **void** merge\_sort\_recursive(**int**[] arr, **int**[] result, **int** start, **int** end){

**if**(start == end){

**return**;

}

**int** mid = (start+end)/2;

**int** start1 = start;

**int** end1 = mid;

**int** start2 = mid+1;

**int** end2 = end;

**int** k = start;

*merge\_sort\_recursive*(arr, result, start1, end1);

*merge\_sort\_recursive*(arr, result, start2, end2);

**while**(start1 <= end1 && start2 <= end2){

**if**(arr[start1] < arr[start2]){

result[k] = arr[start1];

k++;

start1++;

}**else**{

result[k] = arr[start2];

k++;

start2++;

}

}

**while**(start1<= end1){

result[k] = arr[start1];

k++;

start1++;

}

**while**(start2 <= end2){

result[k] = arr[start2];

k++;

start2++;

}

**for**(**int** i = 0; i < arr.length; i++){

arr[i] = result[i];

}

}

}

1. I used merge sort.

*merge\_sort\_recursive*(arr, result, start1, end1);

*merge\_sort\_recursive*(arr, result, start2, end2);

The code here is to divide array into several parts until the array have been totally divided. And then compare each elements and put them into the result array.

1. The best case Time complexity is O(nlogn).
2. The worse case time complexity is O(ologn).

*merge\_sort\_recursive*(arr, result, start1, end1);

*merge\_sort\_recursive*(arr, result, start2, end2); O(logN)

No matter the best case or the worst case. Merge sort always divide the whole array into logN parts and compare them with each other and put the sorted array into the result array.

**for**(**int** i = 0; i < arr.length; i++){

arr[i] = result[i];

}

This process O(N)

So, the overall O(NlogN).