

## Merge Sort

## Outline

1 Merging

2 Sorting

## Merging

## Merging

Merge sort is based on an operation called merging: combining two ordered arrays to make one larger ordered array

## Merging

Merge sort is based on an operation called merging: combining two ordered arrays to make one larger ordered array

To sort an array, merge sort divides it into two halves, sorts the two halves recursively, and then merges the results

## Merging

Merge sort is based on an operation called merging: combining two ordered arrays to make one larger ordered array

To sort an array, merge sort divides it into two halves, sorts the two halves recursively, and then merges the results

Example

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
M	E	R	G	E	S	O	R	T	E	X	A	M	P	L	E

# Merging

Merge sort is based on an operation called merging: combining two ordered arrays to make one larger ordered array

To sort an array, merge sort divides it into two halves, sorts the two halves recursively, and then merges the results

Example

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
M	E	R	G	E	S	O	R	T	E	X	A	M	P	L	E

## Merging

Merge sort is based on an operation called merging: combining two ordered arrays to make one larger ordered array

To sort an array, merge sort divides it into two halves, sorts the two halves recursively, and then merges the results

Example

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
E	E	G	M	O	R	R	S	T	E	X	A	M	P	L	E



## Merging

Merge sort is based on an operation called merging: combining two ordered arrays to make one larger ordered array

To sort an array, merge sort divides it into two halves, sorts the two halves recursively, and then merges the results

Example

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
E	E	G	M	O	R	R	S	A	E	E	L	M	P	T	X

## Merging

Merge sort is based on an operation called merging: combining two ordered arrays to make one larger ordered array

To sort an array, merge sort divides it into two halves, sorts the two halves recursively, and then merges the results

Example

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	E	E	E	E	G	L	M	M	O	P	R	R	S	T	X

## Merging

Merge sort is based on an operation called merging: combining two ordered arrays to make one larger ordered array

To sort an array, merge sort divides it into two halves, sorts the two halves recursively, and then merges the results

Example

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
A	E	E	E	E	G	L	M	M	O	P	R	R	S	T	X

## Merging

## Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
			E	E	G	M	R	A	C	E	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
0	5		E	E	G	M	R	A	C	E	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
0	5	0	E	E	G	M	R	A	C	E	R	T
			E	E	G	M	R	A	C	E	R	T



Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
0	6	1	A	E	G	M	R	A	C	E	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
0	6	1	A	E	G	M	R	A	C	E	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
0	7	2	A	C	G	M	R	A	C	E	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
0	7	2	A	C	G	M	R	A	C	E	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
1	7	3	A	C	E	M	R	A	C	E	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
1	7	3	A	C	E	M	R	A	C	E	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
2	7	4	A	C	E	E	R	A	C	E	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
2	7	4	A	C	E	E	R	A	C	E	R	T
			E	E	G	M	R	A	C	E	R	T



Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
2	8	5	A	C	E	E	E	A	C	E	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
2	8	5	A	C	E	E	E	A	C	E	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
3	8	6	A	C	E	E	E	G	C	E	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
3	8	6	A	C	E	E	E	G	C	E	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
4	8	7	A	C	E	E	E	G	M	E	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
4	8	7	A	C	E	E	E	G	M	E	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
5	8	8	A	C	E	E	E	G	M	R	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
5	8	8	A	C	E	E	E	G	M	R	R	T
			E	E	G	M	R	A	C	E	R	T



Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
5	9	9	A	C	E	E	E	G	M	R	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
5	9	9	A	C	E	E	E	G	M	R	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
5	10	10	A	C	E	E	E	G	M	R	R	T
			E	E	G	M	R	A	C	E	R	T

Merging

			a[], aux[]									
i	j	k	0	1	2	3	4	5	6	7	8	9
			A	C	E	E	E	G	M	R	R	T
			E	E	G	M	R	A	C	E	R	T

## Merging

## Merging

✎ Merge.java

```
public class Merge {
    private static void merge(Comparable[] a, Comparable[] aux, int lo, int mid, int hi) {
        for (int k = lo; k <= hi; k++) {
            aux[k] = a[k];
        }
        int i = lo, j = mid + 1;
        for (int k = lo; k <= hi; k++) {
            if (i > mid) {
                a[k] = aux[j++];
            } else if (j > hi) {
                a[k] = aux[i++];
            } else if (less(aux[j], aux[i])) {
                a[k] = aux[j++];
            } else {
                a[k] = aux[i++];
            }
        }
    }

    private static void merge(Object[] a, Object[] aux, int lo, int mid, int hi, Comparator c) {
        for (int k = lo; k <= hi; k++) {
            aux[k] = a[k];
        }
        int i = lo, j = mid + 1;
        for (int k = lo; k <= hi; k++) {
            if (i > mid) {
                a[k] = aux[j++];
            } else if (j > hi) {
                a[k] = aux[i++];
            } else if (less(aux[j], aux[i], c)) {
                a[k] = aux[j++];
            } else {
                a[k] = aux[i++];
            }
        }
    }
}
```

## Sorting

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		M	E	R	G	E	S	O	R	T	E	X	A	M	P	L	E



Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	M	E	R	G	E	S	O	R	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	M	E	R	G	E	S	O	R	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	3	M	E	R	G	E	S	O	R	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	M	E	R	G	E	S	O	R	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	1	E	M	R	G	E	S	O	R	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	3	E	M	R	G	E	S	O	R	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
2	3	E	M	G	R	E	S	O	R	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	3	E	M	G	R	E	S	O	R	T	E	X	A	M	P	L	E



Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	3	E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4	7	E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4	5	E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4	5	E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	7	E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	7	E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	7	E	G	M	R	E	S	O	R	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4	7	E	G	M	R	E	O	R	S	T	E	X	A	M	P	L	E



Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	E	G	M	R	E	O	R	S	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	7	E	E	G	M	O	R	R	S	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	15	E	E	G	M	O	R	R	S	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	11	E	E	G	M	O	R	R	S	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	9	E	E	G	M	O	R	R	S	T	E	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	9	E	E	G	M	O	R	R	S	E	T	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	11	E	E	G	M	O	R	R	S	E	T	X	A	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	11	E	E	G	M	O	R	R	S	E	T	A	X	M	P	L	E



Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	11	E	E	G	M	O	R	R	S	E	T	A	X	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	11	E	E	G	M	O	R	R	S	A	E	T	X	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	15	E	E	G	M	O	R	R	S	A	E	T	X	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	13	E	E	G	M	O	R	R	S	A	E	T	X	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	13	E	E	G	M	O	R	R	S	A	E	T	X	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14	15	E	E	G	M	O	R	R	S	A	E	T	X	M	P	L	E

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14	15	E	E	G	M	O	R	R	S	A	E	T	X	M	P	E	L

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	15	E	E	G	M	O	R	R	S	A	E	T	X	M	P	E	L



Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	15	E	E	G	M	O	R	R	S	A	E	T	X	E	L	M	P

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	15	E	E	G	M	O	R	R	S	A	E	T	X	E	L	M	P

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	15	E	E	G	M	O	R	R	S	A	E	E	L	M	P	T	X

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	E	E	G	M	O	R	R	S	A	E	E	L	M	P	T	X

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	A	E	E	E	E	G	L	M	M	O	P	R	R	S	T	X

Sorting

		a[]															
lo	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	A	E	E	E	E	G	L	M	M	O	P	R	R	S	T	X

## Sorting

## Sorting

✎ Merge.java

```
public class Merge {  
    public static void sort(Comparable[] a) {  
        Comparable[] aux = new Comparable[a.length];  
        sort(a, aux, 0, a.length - 1);  
    }  
  
    public static void sort(Object[] a, Comparator c) {  
        Object[] aux = new Object[a.length];  
        sort(a, aux, 0, a.length - 1, c);  
    }  
  
    private static void sort(Comparable[] a, Comparable[] aux, int lo, int hi) {  
        if (hi <= lo) {  
            return;  
        }  
        int mid = lo + (hi - lo) / 2;  
        sort(a, aux, lo, mid);  
        sort(a, aux, mid + 1, hi);  
        merge(a, aux, lo, mid, hi);  
    }  
  
    private static void sort(Object[] a, Object[] aux, int lo, int hi, Comparator c) {  
        if (hi <= lo) {  
            return;  
        }  
        int mid = lo + (hi - lo) / 2;  
        sort(a, aux, lo, mid, c);  
        sort(a, aux, mid + 1, hi, c);  
        merge(a, aux, lo, mid, hi, c);  
    }  
}
```



## Sorting

✎ Merge.java

```
public class Merge {  
    public static void sort(Comparable[] a) {  
        Comparable[] aux = new Comparable[a.length];  
        sort(a, aux, 0, a.length - 1);  
    }  
  
    public static void sort(Object[] a, Comparator c) {  
        Object[] aux = new Object[a.length];  
        sort(a, aux, 0, a.length - 1, c);  
    }  
  
    private static void sort(Comparable[] a, Comparable[] aux, int lo, int hi) {  
        if (hi <= lo) {  
            return;  
        }  
        int mid = lo + (hi - lo) / 2;  
        sort(a, aux, lo, mid);  
        sort(a, aux, mid + 1, hi);  
        merge(a, aux, lo, mid, hi);  
    }  
  
    private static void sort(Object[] a, Object[] aux, int lo, int hi, Comparator c) {  
        if (hi <= lo) {  
            return;  
        }  
        int mid = lo + (hi - lo) / 2;  
        sort(a, aux, lo, mid, c);  
        sort(a, aux, mid + 1, hi, c);  
        merge(a, aux, lo, mid, hi, c);  
    }  
}
```

$$T(n) = n \log n \text{ and } S(n) = n$$