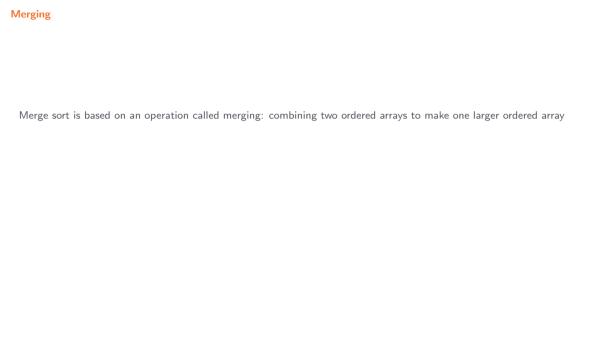


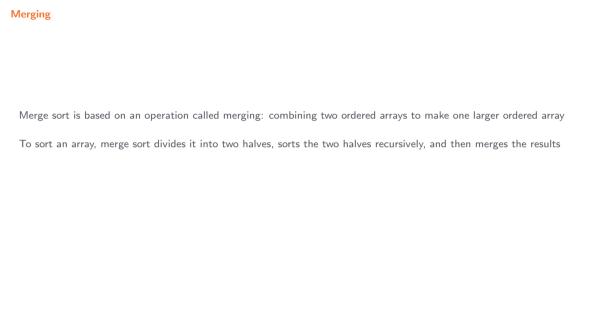
Outline

1 Merging

2 Sorting









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





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



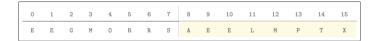


To sort an array, merge sort divides it into two halves, sorts the two halves recursively, and then merges the results



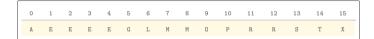


To sort an array, merge sort divides it into two halves, sorts the two halves recursively, and then merges the results





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





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



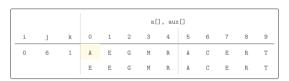


						a[], aux	:[]				
i	j	k	0	1	2	3	4	5	6	7	8	9
			E	E	G	М	R	A	C	E	R	Т

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

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







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



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



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

						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	Т
			Е	E	G	M	R	A	C	Е	R	Т

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

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

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

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

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

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

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

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

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

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

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

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



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. i = mid + 1:
        for (int k = lo; k <= hi; k++) {
            if (i > mid) {
                a[k] = aux[i++]:
            } else if (j > hi) {
                a[k] = aux[i++]:
            } else if (less(aux[i], aux[i])) {
                a[k] = aux[i++]:
            } 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, i = mid + 1:
        for (int k = lo: k <= hi: k++) {
            if (i > mid) {
                a[k] = aux[j++];
            } else if (i > hi) {
                a[k] = aux[i++];
            } else if (less(aux[i], aux[i], c)) {
                a[k] = aux[j++];
            } else {
                a[k] = aux[i++]:
```



										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
		М	Е	R	G	Е	S	0	R	Т	Е	Х	A	М	P	L	E



















										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4	7	Е	G	М	R	Е	S	0	R	Т	Е	Х	A	М	Р	L	Е

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4	5	Е	G	М	R	Е	S	0	R	Т	Е	Х	A	М	P	L	Е

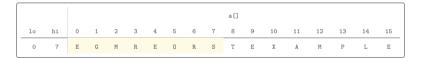
										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4	5	Е	G	М	R	Е	S	0	R	Т	Е	Х	A	М	P	L	Е

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	7	Е	G	M	R	Е	S	0	R	Т	Е	Х	A	М	Р	L	E

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	7	Е	G	M	R	Е	S	0	R	Т	Е	Х	A	М	Р	L	E

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	7	Е	G	М	R	Е	S	0	R	Т	Е	Х	A	М	Р	L	E

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
4	7	Е	G	М	R	Е	0	R	S	Т	Е	Х	A	М	Р	L	Е





										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	15	Е	Е	G	М	0	R	R	S	Т	Е	х	A	М	P	L	E

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	11	Е	Е	G	М	0	R	R	S	Т	Е	Х	A	М	Р	L	Е

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	9	Е	Е	G	М	0	R	R	S	Т	E	Х	A	М	P	L	E

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	9	Е	Е	G	М	0	R	R	S	Е	Т	Х	A	М	P	L	Е

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	11	Е	Е	G	М	0	R	R	S	Е	Т	Х	A	М	P	L	Е

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
10	11	Е	Е	G	М	0	R	R	S	Е	Т	A	Х	М	P	L	Е

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	11	Е	Е	G	М	0	R	R	S	Е	Т	A	Х	М	Р	L	Е

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	11	Е	Е	G	М	0	R	R	S	A	Е	Т	Х	М	P	L	E

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	15	Е	Е	G	M	0	R	R	S	Α	Е	Т	Х	М	P	L	E

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	13	Е	Е	G	М	0	R	R	S	A	Е	Т	Х	М	P	L	Е

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	13	Е	Е	G	М	0	R	R	S	A	Е	Т	Х	М	P	L	Е

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14	15	Е	Е	G	М	0	R	R	S	A	Е	Т	Х	М	Р	L	E

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
14	15	Е	Е	G	М	0	R	R	S	A	Е	Т	Х	М	Р	E	L

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	15	Е	Е	G	М	0	R	R	S	Α	Е	Т	Х	М	P	E	L

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
12	15	Е	Е	G	M	0	R	R	S	A	Е	Т	Х	Е	L	М	P

										a[]							
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	15	Е	Е	G	М	0	R	R	S	A	E	Т	Х	E	L	М	P

		a[]															
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
8	15	E	Е	G	M	0	R	R	S	A	E	E	L	М	P	Т	х





		a[]															
10	hi	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	15	A	Е	Е	Е	Е	G	L	М	М	0	P	R	R	S	Т	Х



```
☑ 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):
```

```
☑ 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):
```

$$T(n) = n \log n$$
 and $S(n) = n$