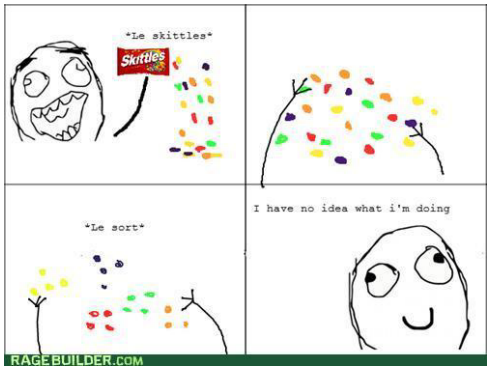
CSE 143

Computer Programming II

Sorting



Outline

Why Do We Care About Sorting?

Sorting is a fundamental idea that is used in many places.

There are also **many different sorting algorithms** which gives us an opportunity to compare them.

How Do We Compare Sorts?

- Timing? (That's not so great, because it varies.)
- Number of steps? (What is a step?)
- Idea: **number of comparisons**

Selection Sort

Idea

Go from left to right and find the next smallest element. Then, put that element in the right place. Now we know everything up to that element is sorted. Continue until we get to the end of the array.

Code

```
1  public static void selectionsort(List<Integer> list) {
2     for (int i = 0; i < list.size(); i++) {
3        Integer index = null;
4        Integer selection = null;
5        for (int j = i; j < list.size(); j++) {
6           int next = list.get(j);
7           if (selection == null || next < selection) {
8              selection = next;
9              index = j;
10          }
11       }
12       // Swap index and i
13       list.set(index, list.get(i));
14       list.set(i, selection);
15    }
16 }
```

Merging Two Sorted Lists

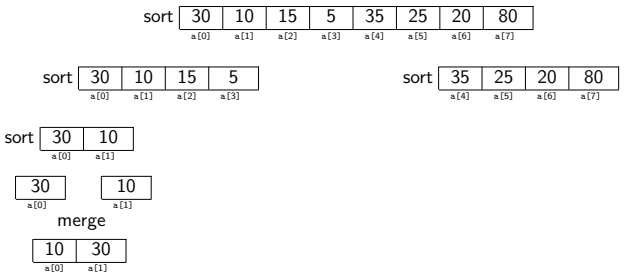
Idea

Keep a finger on the last element we've handled from each list. Figure out which of the elements we're pointing to is smaller. Put the smaller one in the result and move that finger right. Keep on going until both fingers are past the last elements of their lists.

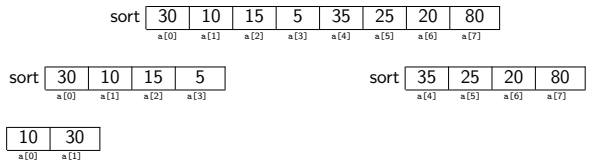
Code

```
1  public static List<Integer> merge(List<Integer> list1, List<Integer> list2) {
2     List<Integer> result = new ArrayList<Integer>(list1.size() + list2.size());
3     int i1 = 0;
4     int i2 = 0;
5     while (i1 + i2 < list1.size() + list2.size()) {
6        Integer a = null, b = null;
7        if (i1 < list1.size()) { a = list1.get(i1); }
8        if (i2 < list2.size()) { b = list2.get(i2); }
9        if (a != null && (b == null || a < b)) {
10          result.add(a);
11          i1++;
12       }
13       else {
14          result.add(b);
15          i2++;
16       }
17    }
18    return result;
19 }
```

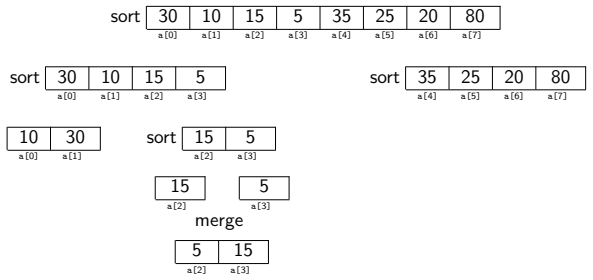
Merge Sort By Picture 4



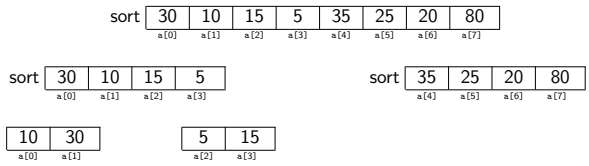
Merge Sort By Picture 5



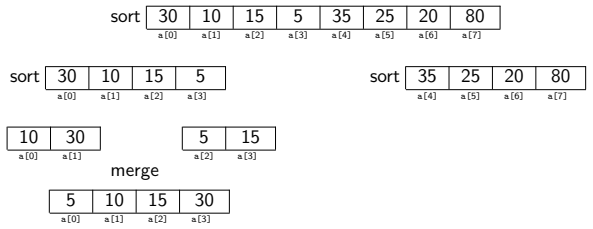
Merge Sort By Picture 6



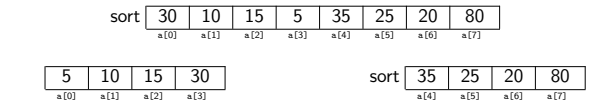
Merge Sort By Picture 7



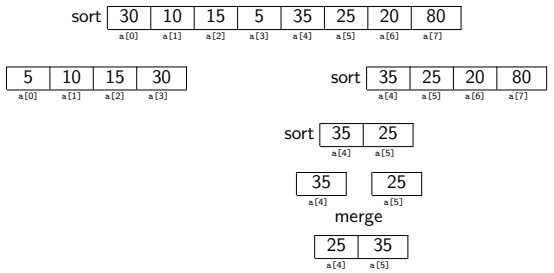
Merge Sort By Picture 8



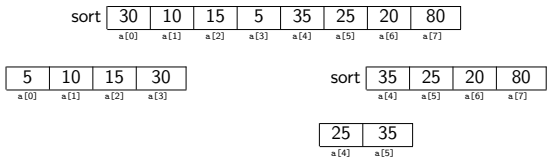
Merge Sort By Picture 9



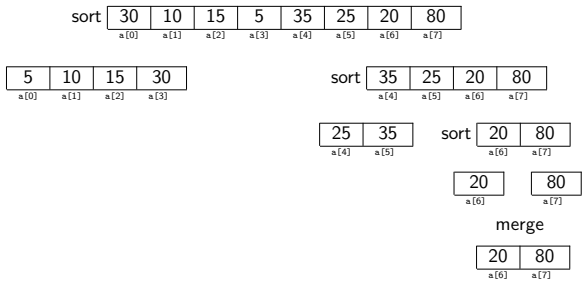
Merge Sort By Picture 10



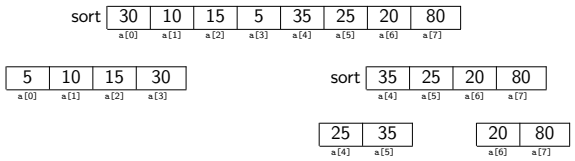
Merge Sort By Picture 11



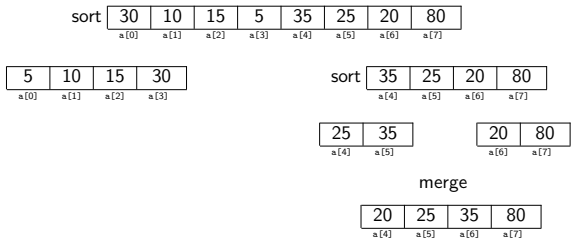
Merge Sort By Picture 12



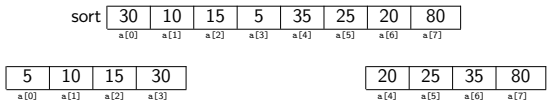
Merge Sort By Picture 13



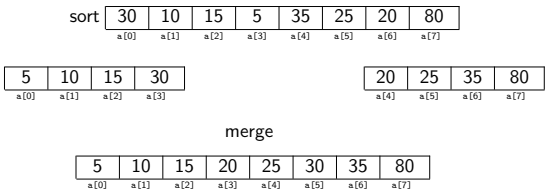
Merge Sort By Picture 14



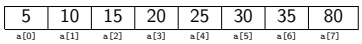
Merge Sort By Picture 15



Merge Sort By Picture



Merge Sort By Picture



Merge Sort

Idea

Break the list into two equal pieces. Sort the left piece; sort the right piece. Use our previous algorithm to merge the two sorted pieces together.

Code

```
1  public static void mergesort(List<Integer> list) {
2     if (list.size() > 1) {
3        int mid = list.size() / 2;
4        List<Integer> left = list.subList(0, mid);
5        List<Integer> right = list.subList(mid, list.size());
6
7        mergesort(left);
8        mergesort(right);
9
10       List<Integer> merged = merge(left, right);
11       for (int i = 0; i < merged.size(); i++) {
12          list.set(i, merged.get(i));
13       }
14    }
15 }
```