CSE 143
Computer Programming II

Outline

1. Selection Sort
2. Merge Sort

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

```java
public static void selectionsort(List<Integer> list) {
    for (int i = 0; i < list.size(); i++) {
        Integer index = null;
        Integer selection = null;
        for (int j = i; j < list.size(); j++) {
            int next = list.get(j);
            if (selection == null || next < selection) {
                selection = next;
                index = j;
            }
        }
        // Swap index and i
        list.set(index, list.get(i));
        list.set(i, selection);
    }
}
```

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

```java
public static List<Integer> merge(List<Integer> list1, List<Integer> list2) {
    List<Integer> result = new ArrayList<Integer>(list1.size() + list2.size());
    int i1 = 0;
    int i2 = 0;
    while (i1 + i2 < list1.size() + list2.size()) {
        Integer a = null, b = null;
        if (i1 < list1.size()) { a = list1.get(i1); }
        if (i2 < list2.size()) { b = list2.get(i2); }
        if (a != null && (b == null || a < b)) {
            result.add(a);
            i1++;
        } else {
            result.add(b);
            i2++;
        }
    }
    return result;
}
```
### Merge Sort By Picture

#### 10

<table>
<thead>
<tr>
<th>sort</th>
<th>30 10 15 5 35 25 20 80</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20 4 5 12 15 17 20</td>
</tr>
<tr>
<td></td>
<td>5 10 15 30</td>
</tr>
</tbody>
</table>

#### 11

<table>
<thead>
<tr>
<th>sort</th>
<th>30 10 15 5 35 25 20 80</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20 4 5 12 15 17 20</td>
</tr>
<tr>
<td></td>
<td>5 10 15 35</td>
</tr>
</tbody>
</table>

#### 12

<table>
<thead>
<tr>
<th>sort</th>
<th>30 10 15 5 35 25 20 80</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20 4 5 12 15 17 20</td>
</tr>
<tr>
<td></td>
<td>25 35</td>
</tr>
</tbody>
</table>

#### 13

<table>
<thead>
<tr>
<th>sort</th>
<th>30 10 15 5 35 25 20 80</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20 4 5 12 15 17 20</td>
</tr>
<tr>
<td></td>
<td>25 35</td>
</tr>
</tbody>
</table>

#### 14

<table>
<thead>
<tr>
<th>sort</th>
<th>30 10 15 5 35 25 20 80</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20 4 5 12 15 17 20</td>
</tr>
<tr>
<td></td>
<td>20 25 35 80</td>
</tr>
</tbody>
</table>

#### 15

<table>
<thead>
<tr>
<th>sort</th>
<th>30 10 15 5 35 25 20 80</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20 4 5 12 15 17 20</td>
</tr>
<tr>
<td></td>
<td>20 25 35 80</td>
</tr>
</tbody>
</table>
Merge Sort 16

Sort 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

```java
public static void mergesort(List<Integer> list) {
    if (list.size() > 1) {
        int mid = list.size() / 2;
        List<Integer> left = list.subList(0, mid);
        List<Integer> right = list.subList(mid, list.size());
        mergesort(left);
        mergesort(right);
        List<Integer> merged = merge(left, right);
        for (int i = 0; i < merged.size(); i++) {
            list.set(i, merged.get(i));
        }
    }
}
```

Merge Sort 17

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

```java
public static void mergesort(List<Integer> list) {
    if (list.size() > 1) {
        int mid = list.size() / 2;
        List<Integer> left = list.subList(0, mid);
        List<Integer> right = list.subList(mid, list.size());
        mergesort(left);
        mergesort(right);
        List<Integer> merged = merge(left, right);
        for (int i = 0; i < merged.size(); i++) {
            list.set(i, merged.get(i));
        }
    }
}
```