Stacks, Queues, ListNodes...
Selection sort

- **selection sort**: Orders a list of values by repeatedly putting the smallest or largest unplaced value into its final position.

  The algorithm:
  - Look through the list to find the smallest value.
  - Swap it so that it is at index 0.
  - Look through the list to find the second-smallest value.
  - Swap it so that it is at index 1.
  - ... 
  - Repeat until all values are in their proper places.
Bogo sort

- **bogo sort**: Orders a list of values by repetitively shuffling them and checking if they are sorted.
  - name comes from the word "bogus"

The algorithm:
- Scan the list, seeing if it is sorted. If so, stop.
- Else, shuffle the values in the list and repeat.

- This sorting algorithm (obviously) has terrible performance!
  - What is its runtime?
Bogo sort code

// Places the elements of a into sorted order.
public static void bogoSort(int[] a) {
    while (!isSorted(a)) {
        shuffle(a);
    }
}

// Returns true if a's elements are in sorted order.
public static boolean isSorted(int[] a) {
    for (int i = 0; i < a.length - 1; i++) {
        if (a[i] > a[i + 1]) {
            return false;
        }
    }
    return true;
}
Bogo sort code, cont'd.

// Shuffles an array of ints by randomly swapping each
// element with an element ahead of it in the array.
public static void shuffle(int[] a) {
    for (int i = 0; i < a.length - 1; i++) {
        // pick a random index in [i+1, a.length-1]
        int range = a.length - 1 - (i + 1) + 1;
        int j = (int) (Math.random() * range + (i + 1));
        swap(a, i, j);
    }
}

// Swaps a[i] with a[j].
public static void swap(int[] a, int i, int j) {
    if (i != j) {
        int temp = a[i];
        a[i] = a[j];
        a[j] = temp;
    }
}
Similar algorithms

- **bubble sort**: Make repeated passes, swapping adjacent values
  - slower than selection sort (has to do more swaps)

- **insertion sort**: Shift each element into a sorted sub-array
  - faster than selection sort (examines fewer values)
Merge sort

- **merge sort**: Repeatedly divides the data in half, sorts each half, and combines the sorted halves into a sorted whole.

  The algorithm:
  - Divide the list into two roughly equal halves.
  - Sort the left half.
  - Sort the right half.
  - Merge the two sorted halves into one sorted list.

- An example of a "divide and conquer" algorithm.
  - Invented by John von Neumann in 1945
Merge sort example

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>22</td>
<td>18</td>
<td>12</td>
<td>-4</td>
<td>58</td>
<td>7</td>
<td>31</td>
<td>42</td>
</tr>
</tbody>
</table>

```
22 18 12 -4
  
22 18
  
22
  
merge

18 22
  
merge

12 -4
  
merge

split

split

58 7 31 42
  
58 7
  
merge

split

split

7 58
  
merge

split

split

31 42
  
merge

merge

-4 7 12 18 22 31 42 58
```
# Merging sorted halves

<table>
<thead>
<tr>
<th>Subarrays</th>
<th>Next include</th>
<th>Merged array</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 1 2 3</td>
<td>0 1 2 3</td>
<td>0 1 2 3</td>
</tr>
<tr>
<td>14 32 67 76</td>
<td>14 from left</td>
<td>14</td>
</tr>
<tr>
<td>i1</td>
<td></td>
<td>i</td>
</tr>
<tr>
<td>i2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 32 67 76</td>
<td>23 from right</td>
<td>14 23</td>
</tr>
<tr>
<td>i1</td>
<td></td>
<td>i</td>
</tr>
<tr>
<td>i2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 32 67 76</td>
<td>32 from left</td>
<td>14 23 32</td>
</tr>
<tr>
<td>i1</td>
<td></td>
<td>i</td>
</tr>
<tr>
<td>i2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 32 67 76</td>
<td>41 from right</td>
<td>14 23 32 41</td>
</tr>
<tr>
<td>i1</td>
<td></td>
<td>i</td>
</tr>
<tr>
<td>i2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 32 67 76</td>
<td>58 from right</td>
<td>14 23 32 41 58</td>
</tr>
<tr>
<td>i1</td>
<td></td>
<td>i</td>
</tr>
<tr>
<td>i2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 32 67 76</td>
<td>67 from left</td>
<td>14 23 32 41 58 67</td>
</tr>
<tr>
<td>i1</td>
<td></td>
<td>i</td>
</tr>
<tr>
<td>i2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 32 67 76</td>
<td>76 from left</td>
<td>14 23 32 41 58 67 76</td>
</tr>
<tr>
<td>i1</td>
<td></td>
<td>i</td>
</tr>
<tr>
<td>i2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14 32 67 76</td>
<td>85 from right</td>
<td>14 23 32 41 58 67 76 85</td>
</tr>
<tr>
<td>i1</td>
<td></td>
<td>i</td>
</tr>
<tr>
<td>i2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Merge sort

- **merge sort**: Repeatedly divides the data in half, sorts each half, and combines the sorted halves into a sorted whole.

  The algorithm:
  - Divide the list into two roughly equal halves.
  - Sort the left half.
  - Sort the right half.
  - Merge the two sorted halves into one sorted list.

- An example of a "divide and conquer" algorithm.
  - Invented by John von Neumann in 1945