Elementary Sorting

- Have talked about using “built-in” sorting
  - Arrays.sort
  - Collections.sort

- These are “heavy-duty”, “industrial” sorts that are for general use. With the appropriate Comparator, they can meet most of your needs.

- However, you should know how to write a simple sort routine from scratch.

Selection Sort Example

- N items in an array named Data
  
  \[ 2 | 4 | 7 | 3 | 1 | 8 | 5 \]

- Find smallest of elements 1 thru N of Data
  
  \[ 1 | 4 | 7 | 3 | 2 | 8 | 5 \]

- Interchange this with 1st element of array Data
  
  \[ 1 | 2 | 7 | 3 | 2 | 8 | 5 \]

- Find smallest of elements 2 thru N of Data
  
  \[ 1 | 2 | 3 | 4 | 7 | 8 | 5 \]

- Interchange this with 2nd element of array Data
  
  \[ 1 | 2 | 3 | 4 | 5 | 8 | 7 \]

- ... 

- Find smallest of elements K thru N of Data
  
  \[ 1 | 2 | 3 | 4 | 5 | 7 | 8 \]

- Interchange this with Kth element of array Data

- Done when K = N

\[ 1 | 2 | 3 | 4 | 5 | 7 | 8 \]

Selection Sort Code

```java
public int locMin(int[] nums, int start){
    int loc = start;
    for (int k = start + 1; k < nums.length; k++){
        if (nums[k] < nums[loc])
            loc = k;
    }
    return loc;
}

public void SelectSort(int[] nums){
    for (int k = 0; k < nums.length; k++) {
        int minloc = locMin(nums, k);
        int temp = nums[k];
        nums[k] = nums[minloc];
        nums[minloc] = temp;
    }
}
```

Selection Sort

- Think about Selection Sort
- Loop Invariant
  - What can we say about our partially sorted list that is true each time around?

  ![Diagram](sorted list with final position k)

  - How efficient is it?
  - Develop measure of efficiency
  - We say it's performance is proportional to $N^2$
More Sorting

- **Other Simple Sorts**
  - Simple results in order $N^2$ performance
  - Bubble Sort? (XXX)
    - Worst of the $N^2$ sorts

- **Insertion Sort**
  - Develop Algorithm
  - (steps often used when updating a sorted list, one item at a time)
  - More complicated to program than selection sort
    - But, has some very nice properties

---

### Example

```
[5 | 4 | 6 | 9 | 3 | 8 | 1] 4
[5 | 4 | 6 | 9 | 3 | 8 | 1] 4
[5 | 5 | 6 | 9 | 3 | 8 | 1] 4
[4 | 5 | 6 | 9 | 3 | 8 | 1] 6
[4 | 5 | 6 | 9 | 3 | 8 | 1] 6
[4 | 5 | 6 | 9 | 3 | 8 | 1] 9
[4 | 5 | 6 | 9 | 3 | 8 | 1] 9
[4 | 5 | 6 | 9 | 3 | 8 | 1] 9
[4 | 5 | 6 | 9 | 3 | 8 | 1] 3
[4 | 5 | 6 | 9 | 3 | 8 | 1] 3
[4 | 5 | 6 | 9 | 3 | 8 | 1] 3
[4 | 5 | 6 | 9 | 3 | 8 | 1] 3
[3 | 4 | 5 | 6 | 9 | 8 | 1]
[3 | 4 | 5 | 6 | 9 | 8 | 1]
[3 | 4 | 5 | 6 | 9 | 8 | 1]
[3 | 4 | 5 | 6 | 9 | 8 | 1]
[3 | 4 | 5 | 6 | 9 | 8 | 1]
[3 | 4 | 5 | 6 | 9 | 8 | 1]
[3 | 4 | 5 | 6 | 9 | 8 | 1]
[3 | 4 | 5 | 6 | 9 | 8 | 1]
[3 | 4 | 5 | 6 | 9 | 8 | 1]
[3 | 4 | 5 | 6 | 9 | 8 | 1]
[3 | 4 | 5 | 6 | 9 | 8 | 1]
[3 | 4 | 5 | 6 | 9 | 8 | 1]
[3 | 4 | 5 | 6 | 9 | 8 | 1]
[3 | 4 | 5 | 6 | 9 | 8 | 1]
[3 | 4 | 5 | 6 | 9 | 8 | 1]
[3 | 4 | 5 | 6 | 9 | 8 | 1]
[3 | 4 | 5 | 6 | 9 | 8 | 1]
[3 | 4 | 5 | 6 | 9 | 8 | 1]
```

---

### Insertion Sort Code

```java
public void insertSort(int[] nums) {
    int j, k, temp;
    for (k = 1; k < nums.length; k++) {
        temp = nums[k];
        for (j = k; j > 0; j--) { // decrement!
            if (temp < nums[j - 1])
                nums[j] = nums[j - 1];
            else
                break;
        }
        nums[j] = temp;
    }
}
```

---

### Insertion Sort

- **Loop Invariant?**
  - For almost sorted?
- **How efficient?**
- **Is stable?**
  - What does that mean?