Intro to Sorting

- **Sorting**
  - "Ideal" Computer Science Topic
  - Theory and Practice meet
  - Efficient Sorting Saves Money

- First look at some simple (quick and dirty?) algorithms

- **Selection Sort**
  1. Find smallest; swap with element [0]
  2. Consider rest of list [1], [2], ..., find smallest, swap with element [1]
  3. Continue process until you get to end

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

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

- **Complexity**
  - What is the big Oh?
  - Develop relationship for Selection Sort
More Sorting

- **Other Simple Sorts**
  - Simple results in O(N^2)
  - Bubble Sort? (XXX)
    - Worst of the O(N^2)

- **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]
- [5 4 6 9 3 8 1]
- [5 4 6 9 3 8 1]
- [5 4 6 9 3 8 1]
- [5 4 6 9 3 8 1]
- [5 4 6 9 3 8 1]
- [5 4 6 9 3 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?**

- **Complexity?**
  - For almost sorted?

- **Is stable?**
  - What does that mean?