Week 3 Lecture 7

Theory

What's in this lecture?

- Browser-less JavaScript with NodeJS
- Sorting!

NodeJS

- JavaScript debugging in FireFox is tricky
- NodeJS packages the lightning-fast V8 JavaScript engine into a useful command-line
- Let's use NodeJS this week to make things easier

Hello Node

```
var console = require(“console”);
console.log(“hello node!”);
console.log([1, 2, 3]);
console.log({“my_key”:”my_value”});
```

Algorithms

- An *algorithm* is "a specific way of doing a general task"
- For example, the "cleaning laundry" task has "washing machine", "washboard," and "dry cleaner" algorithms

Sorting

- A *sort* algorithm takes a collection of elements and returns an ordered collection from least to greatest
- This requires a comparison function compare(a, b) that is valid for all a, b in the collection
- Do these make sense? “apple” < “banana”, 111 < 222, “10” < “9”

Useful Reference

- Check out this site for interactive sorting demos animated with JavaScript:
- http://www.sorting-algorithms.com/

Inversions

- Consider the list: [2, 1, 3]
- Sorted, it would be: [1, 2, 3]
- We say that there is an *inversion* in the original list (2, 1) because 2 > 1
- A sorted list has no inversions

Bubble Sort

- The "bubble" sort algorithm works by "bubbling up" inversions repeatedly
- It repeatedly swaps adjacent positions where there is an inversion

Swap

```
// swaps elements at place i and j in the array
function swap(a_array, i, j) {
  var tmp = a_array[i];
  a_array[i] = a_array[j];
  a_array[j] = tmp;
}
```

Bubble Sort

```
function bubble_sort(a_array) {
  var n = a_array.length;
  var found_inversion = true;

  while (found_inversion) {
    found_inversion = false;

    for (var i = 1; i < n; i++) {
      if (a_array[i - 1] > a_array[i]) {
        found_inversion = true;
        swap(a_array, i - 1, i);
      }
    }
  }
  return a_array;
}
```

Insertion Sort

- Insertion Sort works by processing each element in the array and moving it backwards to its correct place
- The invariant is that the portion of the array “left of i” is always sorted

Insertion Sort

```
function insertion_sort(a_array) {
  var n = a_array.length;
  for (var j = 1; j < n; j++) {
    var key = a_array[j];
    var i = j;
    while (i > 0 && a_array[i - 1] > key) {
      a_array[i] = a_array[i - 1];
      i = i - 1;
    }
    a_array[i] = key;
  }
  return a_array;
}
```

Exercises

- Read Intro to Algorithms, 3rd Edition, Chapters 1 & 2
- Modify these 2 sorting functions to reverse the numeric sort order
- Make it so that the sorting functions take in a first-class compare(a,b) function that *you* write