Review

- Single-source shortest paths (last week Wednesday and Friday)
- Review for test (**Today**)
- Test 1 (Wednesday)
- Begin binary search trees (Friday)

Today:

- What I want you to know
  - Correctness
  - Efficiency
  - ADTs and data structures (including case studies)
  - Graphs
- What kind of questions to expect

Consider the following code fragment from an implementation of counting sort:

```
def counting_sort(sequence):
    max_val = sequence[0]
    i = 1
    while i < len(sequence):
        if max_val < sequence[i] :
            max_val = sequence[i]
        i = i + 1
    counts = [0 for i in range(max_val + 1)]
    i = 0
    while i < len(sequence) :
        counts[sequence[i]] += 1
        i += 1
    ...
```

Let n be the length of `sequence`.
Give a useful loop invariant for the first loop.
Give the running time of the first loop as a big-Oh category.
Give a useful loop invariant for the second loop.
Give the running time of the second loop as a big-Oh category.
What abstract data type is the `counts` array effectively acting as?

Consider this implementation of binary search:

```
public static int binarySearch(List<String> seq, String item) {
    int low = 0,
        high = seq.size(),
        mid = (low + high) / 2;
    int compare = item.compareTo(seq.get(mid));
    while (compare != 0 && high - low > 1) {
        if (compare < 0) high = mid;
        else low = mid;
        mid = (low + high) / 2;
        compare = item.compareTo(seq.get(mid));
    }
    if (compare == 0) return mid;
    else return -1;
}
```

Fill-in a chart indicating the worst-case for each item forlisted as a big-oh category, considering the case when `seq` is a `LinkedList` and when it is an `ArrayList`. Let *n* be the number of items in `seq`.
Running time of call `seq.size()`, running time of each call `seq.get(mid)`, number of iterations of the while loop, running time of entire method.

Implement a bag using a map as the internal representation. Fill-in the key and value types for the internal map and the implementations for the methods `add()`, `count()`, and `remove()`. (8 points total)

```
public class MapBag<E> implements Bag<E> {
    Map<          ,          > internal;   // <------- Fill in those blanks
    // assume there is a constructor that instantiates some class
    // implementing Map to initialize internal

    // Add an item to the bag, increasing its count if it's already there
    public void add(E item) {


    }
    // How many times does this bag contain this item?
    public int count(E item) {


    }
    // Remove (all occurrences of) an item
    public void remove(E item) {


    }
}
```

Coming up:

Do **SSSP** *project (suggested by Friday, Oct 14)*

Due **Fri, Oct 21** *(class time)*
Read Sections 5.(1 & 2)
Do Exercises 5.(2 & 6)
Take BST quiz