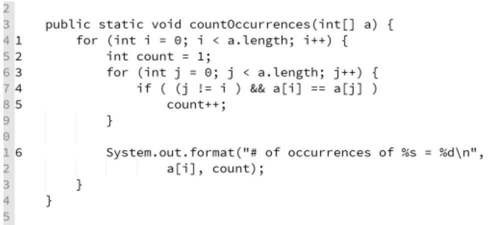
- Administrivia
  - HW2 due on Friday
  - Any other questions
  - Office hours canceled today to make sure we get enough time for grading
    - Will make up with 2x office hours at a high-demand time
- Last time
  - Talked about linked lists, something you learned about in depth in your lab
  - Checking in: things you should understand about abstract data types
    - They’re abstract! Implementation separate from function.
    - Function of Lists
    - How to implement an array-backed list
    - How to implement a linked list list
- Today: intro to algorithmic efficiency
  - Place add(E o), add (int index, E o), get(int index), E remove(int index), E set(int index, E o), int size() in terms of execution speed
    - 5 minutes
      - Right side of the room: ArrayLists or array-backed lsits
      - Left side of the room: Linked lists
      - Teams of two
      - Don’t want to see anyone on their laptop doing anything but this, this is important
      - Going to need a volunteer on each side to justify their decision
  - If the list gets bigger, what’s the effect at the top of the rankings, bottom?
- Capture what we’ve seen so far
  - Constant time
    - Size, get, set for array-backed
    - What for linked list
  - Linear time
    - Add, remove, indexOf/contains for array-backed
  - Two key concepts here
    - Problem size: n! (number of operations)
    - Growth functions or O
- Let’s get more complex
  - Hypothetical algorithm that implements countOccurences
  - SLIDE:



```
    public static void countOccurrences(int[] a) {
1       for (int i = 0; i < a.length; i++) {
2           int count = 1;
3           for (int j = 0; j < a.length; j++) {
4               if ( (j != i ) && a[i] == a[j] )
5                   count++;
            }

6           System.out.format("# of occurrences of %s = %d\n",
                    a[i], count);
        }
    }
```

  -
  - Walk through code
  - If n is the length of a, how many times does each line of code run?
    - 1 = N

        - 2 = N
        - 3 = N
        - 4 = N * N
        - 5 = N * (N-1)
        - 6 = N
    - Adding everything up
        - N^2 + N(N-1) + 4N
        - N^2 + N^2 – N + 4N
        - 2N^2 + 3N
    - We say O(N^2)
        - Count only the “highest order” term
            - Ignore lower order terms, including constants
        - Ignore multiplier on the highest order term
    - What if we said, only count up to five occurences? What would be the effect on O?
    - To be clear
        - Why are
            - 5N^2
            - 10N^2 + 1000N + 1,000,000
            - 0.0001N^2
        - The same? (Why do we just care about O(N^2)?)
        - And why is?
            - F1(n) = 1000N faster than
            - F2(n) = 0.01N^2
    - A: Big data
        - We only care about large problems (large N), and for large problems the higher order term dominates
            - For small problems, efficiency isn’t that important
- So, what do we have here in terms of the concepts we discussed earlier?
    - New growth function
        - Polynomial
- Challenge problems
    - 5-min: Design an algorithm that counts the number of occurences in a list in O(n)
    - 5-min: Returning to contains()…Introduce binary sort
        - Q: What is the efficiency of binary sort?