Administrative Details

- Lab 3 is due today at noon
- Run-time (big-O notation) in comments above methods
- Don’t forget to define “n” if you say O(n)
- I have to cancel office hours today…sorry! 😞
- I might be in tomorrow afternoon instead
- Check your email

Last Time

- Discussed runtime analysis techniques
- Reviewed and discussed recursion
  - Looked at factorial in class
  - Looked at digit sum and subset sum in lab

Today’s Outline

- Begin reviewing mathematical induction
- Begin learning about searching and sorting
  - Two of the most important classes of algorithms
- We’ll discuss two searches:
  - Linear search
  - Binary search

Recursion Tradeoffs

- Advantages
  - Often easier to construct recursive solution
  - Code is usually cleaner
  - Some problems do not have obvious non-recursive solutions
- Disadvantages
  - Overhead of recursive calls
  - Can use lots of memory (need to store state for each recursive call until base case is reached)

Mathematical Induction

- The mathematical equivalent of recursion is called induction
- Induction is a proof technique
- Comes from how natural numbers are defined:
  1. 0 is an element of A
  2. For each n, if 0, 1, 2, …, n-1 are in A, than n is in A.
Mathematical Induction

• Examples
  \[ P = \sum_{i=0}^{n} i = 0 + 1 + \ldots + n = \frac{n(n+1)}{2} \]

• Proof by induction:
  • Base case: P is true for 0
  • Inductive hypothesis: If P is true for all \( k < n \), then P is true for \( n \).

\[ \sum_{i=0}^{n} i^2 = 0^2 + 1^2 + 2^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6} \]

Proof: \( \sum_{i=0}^{n} i^3 = (0 + 1 + \ldots + n)^2 \)

• Base case: \( n = 0, 0^3 = (0)^2 \).

• Ind. Hyp.: For \( k < n \) assume true.
  • \( (0^3 + 1^3 + \ldots + k^3) = (0 + 1 + \ldots + k)^2 \)

• Ind. Step: Show true for \( n \).

What about Recursion?

• What does induction have to do with recursion?
  • Same form!
    • Base case
    • Inductive case that uses simpler form of problem

What about Recursion?

• Example: factorial
  • Prove that fact(n) requires (n-1) multiplications
    • Base case: \( n = 1 \) returns 1, 0 multiplications
    • Assume true for all \( k < n \), so fact(k) requires \( k-1 \) multiplications.
    • fact(n) performs one multiplication (n\times fact(n-1)). We know that fact(n-1) requires n-2 multiplications by inductive hypothesis. \( 1 + n-2 = n-1 \), therefore fact(n) requires n-1 multiplications.
Moving on…

Linear Search

- Where have we seen a linear search?
  - Dictionary.java!
  - [Look at SortSearchDemo]

```java
// post: returns the definition of word, or "" if not found.
public String lookup(String word) {
    for (int i = 0; i < words.length; i++) {
        Association a = words[i];
        if (a.getKey().equals(word)) {
            return (String)a.getValue();
        }
    }
    return "";
}
```

Linear Search

- Complexity analysis of linear search:
  - Best case: \( O(1) \)
  - Worst case: \( O(n) \)
  - Average case: \( O(n) \)

Searching

- What is searching?
  - Locate element in collection
  - Examples: Number in list, grade in gradebook, etc
  - Complexity analysis, induction, recursion
- Today’s algorithms
  - Linear search - move down line
  - Binary search - divide elements in half
- Next up
  - Sorting
  - Designing data structures to support other searching/sorting algorithms

Binary Search

- Find a name in the phonebook
- Guess a number between 1 and 100

• These are examples of binary search
• Why does it work?
  - Rule out as much of search space as possible with each guess
• What assumption (about the data) does it rely on?
  - Is it recursive? Let’s look at the code…

```
public class LinearSearchComp {
    // post: returns index of value in a, or -1 if not found
    public static <E> int linearSearch(E a[], E value) {
        for (int i = 0; i < a.length; i++) {
            if (a[i].equals(value)) {
                return i;
            }
        }
        return -1;
    }

    public static void main(String args[]) {
        // search a String array
        System.out.println(linearSearch(args, "cow"));

        // search a Linear array
        Integer odds[] = new Integer[] {1,3,5,7,9};
        System.out.println(linearSearch(odds, 7));
    }
}
```

Required in parameterized static methods! (Can also have parameterized classes)

```
public class LinearSearchComp {
    // post: returns index of value in a, or -1 if not found
    public static <E> int linearSearch(E a[], E value) {
        for (int i = 0; i < a.length; i++) {
            if (a[i].equals(value)) {
                return i;
            }
        }
        return -1;
    }

    public static void main(String args[]) {
        // search a String array
        System.out.println(linearSearch(args, "cow"));

        // search a Linear array
        Integer odds[] = new Integer[] {1,3,5,7,9};
        System.out.println(linearSearch(odds, 7));
    }
}
```

```
```