CSCI 136
Data Structures &
Advanced Programming

Lecture 11

Fall 2017

Instructors: Bills

Administrative Details

- Lab 4 will be available online this afternoon
  - Partner? Submit 1 folder
- Problem Set 1 due Thursday by 11:00pm
  - In Instructor cubby outside of TCL 303

Last Time

- Comparing Complexity of List Operations on Vectors and Linked Lists
- Recursion and Induction

Today's Outline

- More about Mathematical Induction
  - For algorithm run-time and correctness
- More About Recursion
  - Recursion on arrays; helper methods
  - Recursion on Chains
- Strong Induction
- Linear and Binary Searching review

Mathematical Induction

Principle of Mathematical Induction (Weak)

Let $P(0)$, $P(1)$, $P(2)$, ... Be a sequence of statements, each of which could be either true or false. Suppose that

1. $P(0)$ is true, and
2. For all $n \geq 0$, if $P(n)$ is true, then so is $P(n+1)$.

Then all of the statements are true!

Note: Often Property 2 is stated as

2. For all $n > 0$, if $P(n-1)$ is true, then so is $P(n)$.

   Apology: I do this a lot, as you'll see on future slides!

Form of Induction Proof

Principle of Mathematical Induction (Weak)

Let P(0), P(1), P(2), ... Be a sequence of statements, each of which could be either true or false.

- Show that Base Case P(0) is true
- Show that for any $n \geq 0$
  - If P(n) is true (Induction Hypothesis)
  - Then P(n+1) must be true (Induction Step)

If this can be shown, then each P(n) ($n \geq 0$) is true

Mathematical Induction

- Prove: $\sum_{i=0}^{n} 2^i = 2^0 + 2^1 + 2^2 + \ldots + 2^n = 2^{n+1} - 1$

- Prove: $0^3 + 1^3 + \ldots + n^3 = (0 + 1 + \ldots + n)^2$

Proof: $0^3 + 1^3 + \ldots + n^3 = (0 + 1 + \ldots + n)^2$

Base case: n = 0

- LHS: $0^3 = 0$
- RHS: $(0)^2 = 0$ ✓

Induction Hypothesis: Assume that for some n > 0,

$$0^3 + 1^3 + \ldots + (n-1)^3 = (0 + 1 + \ldots + (\mathrm{n}-1))^2$$

Induction Step: Show that

$$0^3 + 1^3 + \ldots + n^3 = (0 + 1 + \ldots\ n)^2$$

Proof: $0^3 + 1^3 + \ldots + n^3 = (0 + 1 + \ldots + n)^2$

Note: I'm just doing the induction step: n-1 → n version

$$0^3 + 1^3 + \ \ldots + \ n^3 = (0^3 + 1^3 + \ \ldots + \ (n-1)^3 + n^3$$

Induction☞ $$= (0 + 1 + \ \ldots + (\mathrm{n} - 1))^2 + n^3$$

Algebra☞ $$= \left(\frac{(n-1)n}{2}\right)^2 + n^3$$

$$= n^2\left(\frac{(n-1)^2 + 4n}{4}\right)$$

$$= n^2\left(\frac{n^2 + 2n + 1}{4}\right)$$

$$= n^2\left(\frac{(n+1)^2}{4}\right)$$

$$= \left(\frac{n(n+1)}{2}\right)^2$$

$$= (0 + 1 + \cdots + n)^2$$

Form of Induction Proof

We don't have to start at $n = 0$!

Principle of Mathematical Induction (Weak)

Let $P(k)$, $P(k+1)$, $P(k+2)$, ... Be a sequence of statements, each of which could be either true or false.

- Show that Base Case $P(k)$ is true
- Show that for any $n \geq k$
  - If $P(n)$ is true (Induction Hypothesis)
  - Then $P(n+1)$ must be true (Induction Step)

If this can be shown, then each $P(n)$ $(n \geq k)$ is true

Examples (Try These at Home!)

Show that the angles of any n-sided polygon add up to $\pi(n-2)$.

Note: $n \geq 3$, so base case is n=3

Show that if there are at least 6 people at a party, then either there are 3 mutual acquaintances or three mutual strangers.

Base case is n = 6

The induction step should be trivial!

What about Recursion?

- What does induction have to do with recursion?
  - Same form!
    - Base case
    - Inductive case that uses simpler form of problem
- Example: factorial
  - Prove that fact(n) requires n multiplications
    - Base case: $n = 0$ returns 1, so 0 multiplications
    - Assume for some $n \geq 0$ that fact(n) requires n multiplications.
    - fact(n+1) performs one multiplication: (n+1)*fact(n).
    - We know that fact(n) requires n multiplications.
    - So fact(n+1) requires (exactly) n+1 multiplications.

Recursive contains() for Vector

```
public boolean contains(E elt) {
   return contains(elt, 0, size()-1); }

// Helper method: returns true if elt has index in range from..to
public boolean contains(E elt, int from, int to) {
   if (from > to)
        return false; // Base case: empty range
   else
        return elt.equals(elementData[from]) ||
               contains(elt, from+1, to);
}
```

- What's the time complexity of contains?
  - O(to – from + 1) = O(n) (n is the portion of the array searched)
  - Prove by induction on n
- Often recursive methods on arrays use helper methods
  - They pass a pair of indices as parameters

Design Decision: Chains vs Nodes

- SLL and DLL used a simple Node model
- We could push more of the work down to the “Node” level
- A Chain object contains a value and a reference to “the rest of the chain”
- We can now implement many methods recursively and elegantly
- Uses a “dummy” node for empty chain
  - So an empty Chain is not a null value
- Let’s look at some code....

A Proof About Chains

Prove: deleteDuplicates() is correct

- Base Case: n = 0: Empty List is returned ✓
- Induction Hypothesis: For some $n \geq 0$, the method is correct
- Induction Step: Show it is correct for n+1

```
Chain<E> result = rest.deleteDuplicates();
if(rest.contains(value)) return result;
else return new Chain<E>(value, result);
```

  - By I.H. result is rest without duplicates
  - If statement only includes (first) value if it is not a duplicate of something in rest. ✓

Counting Method Calls

- Example: Fibonacci
  - Prove that for $n \geq 0$ fib(n) makes at least F_n calls to fib(), where F_n is the n^{th} Fibonacci number
    - Base cases: n = 0: 1 call; n = 1; 1 call
    - Assume that for some $n \geq 2$, fib(n-1) makes at least F_{n-1} calls to fib() and fib(n-2) makes at least F_{n-2} calls to fib().
    - Claim: Then fib(n) makes at least F_n calls to fib()
      - 1 initial call: fib(n)
      - By induction: At least fib(n-1) calls for fib(n-1)
      - And as least fib(n-2) calls for fib(n-2)
      - Total: 1 + fib(n-1) + fib(n-2) > fib(n-1) + fib(n-2) = fib(n) calls
    - Note: Need two base cases!
  - One can show by induction that for n > 10: fib(n) > $(1.5)^n$
  - Thus the number of calls grows exponentially!

Mathematical Induction : Version 2

Principle of Mathematical Induction (Weak)

Let P_0, P_1, P_2, ... Be a sequence of statements, each of which could be either true or false. Suppose that

1. P_0 and P_1 are true, and
2. For all $n \geq 2$, if P_{n-1} and P_{n-2} are true, then so is P_n.

Then all of the statements are true!

Other versions:

- Can have $k > 2$ base cases
- Doesn't need to start at 0

Example: Binary Search

- Given an array a[] of positive integers in increasing order, and an integer x, find location of x in a[].
  - Take “indexOf” approach: return -1 if x is not in a[]

```
protected static int recBinarySearch(int a[], int value,
              int low, int high) {
   if (low > high) return -1;
   else {
       int mid = (low + high) / 2;        //find midpoint
       if (a[mid] == value) return mid;  //first comparison
                                          //second comparison
       else if (a[mid] < value)           //search upper half
       return recBinarySearch(a, value, mid + 1, high);
        else                              //search lower half
             return recBinarySearch(a, value, low, mid - 1);
   }
```

Binary Search takes O(log n) Time

Can we use induction to prove the following?

- Claim: If n = high - low + 1, then recBinSearch performs at most c (1+ log n) operations, where c is *twice* the number of statements in recBinSearch
- Base case: n = 1: Then low = high so only c statements execute (method runs twice) and c ≤ c(1+log 1)
- Assume that claim holds for some n ≥ 1, does it hold for n+1? [Note: n+1 > 1, so low < high]
- Problem: Recursive call is *not* on n---it's on n/2.
- Solution: We need a better version of the PMI….

Strong Mathematical Induction

Principle of Mathematical Induction (Strong)

Let P(0), P(1), P(2), ... Be a sequence of statements, each of which could be either true or false. Suppose that, for some $a \geq 0$

1. P(0), P(1), ... , P(a) are true, and
2. For every $n \geq a$, if P(1), P(2), ... , P(n) are true, then so is P(n+1).

Then all of the statements are true!

Form of Strong Induction Proof

Principle of Mathematical Induction (Strong)

Let P(0), P(1), P(2), ... Be a sequence of statements, each of which could be either true or false.

- Show that Base Cases P(0), P(1), … P(a) are true
- Show that for any $n \geq a$
  - If P(0), P(1), … P(n) are true (Induction Hypothesis)
  - Then P(n+1) must be true (Induction Step)

If this can be shown, then each P(n) ($n \geq 0$) is true

Binary Search takes O(log n) Time

Try again now:

- Assume that for some $n \geq 1$, the claim holds *for all* $k \leq n$, does claim hold for $n+1$?
- Yes! Either
  - $x = a[mid]$, so a constant number of operations are performed, or
  - RecBinSearch is called on a sub-array of size $n/2$, and by induction, at most $c(1 + \log (n/2))$ operations are performed.
    - This gives a total of at most $c + c(1 + \log(n/2)) = c + c(\log(2) + \log(n/2)) = c + c(\log n) = c(1 + \log n)$ statements