CSCI 136
Data Structures &
Advanced Programming

Mathematical Induction

Mathematical Induction

For best results: Review the materials discussing recursion!

Recursive Contains

Recall our recursive contains method for a Singly-Linked List

```
// Pre: value is not null
public static boolean contains(Node<String> n, String v) {
   if( n == null ) return false;
   return v.equals(n.value()) || contains(n.next(), v);
}
```

How could we convince ourselves it's correct?

- Does it work on an empty list? [n is null]
- Does it work on a list of size 1? [n.next() is null]
- Does it work on a list of size 2? [n.next() is a list of size 1]

Key Observation:

- *Assuming* that contains works on *<u>all lists of size n</u>*, (for *<u>any n ≥ 0</u>*)
- Allows us to *conclude* that it works for *<u>all lists of size n+1</u>* !
- And since contains works on all lists of size 0…It always works!

Mathematical Induction

- The mathematical sibling of recursion is induction
- Induction is a proof technique
- Reflects the structure of the natural numbers
- Used to simultaneously prove an infinite number of theorems! For example:
  - `Contains` functions correctly for all lists of size 0
  - `Contains` functions correctly for all lists of size 1
  - `Contains` functions correctly for all lists of size 2
  - ....

Mathematical Induction

Let's make this notion formal and precise

Given: Boolean statements P_0, P_1, …, P_n, … . That is

- Each statement P_i is either true or false (boolean)
- There is a statement P_n for each integer $n \geq 0$

We would like to prove that each statement is true.

We do this by

- Directly showing that P_0 is true
- Then showing that *whenever* P_n is true for some $n \geq 0$, then P_{n+1} is also true

We can then conclude that all of the statements are true!

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 *every* $n \geq 0$, *if* P_n is true, *then* P_{n+1} is true

Then all of the statements are true!

Notes

- Often Property 2 is stated as
  2. For *every* $n > 0$, *if* P_{n-1} is true, *then* P_n is true
- We call Step 1 *Verifying the base case(s)* and Step 2 verifying the *induction step* (or the *induction hypothesis*)

Mathematical Induction

- Example: Prove that for every n ≥ 0

$$P_n : \ 0 + 1 + \ \ldots + n = \frac{n(n+1)}{2}$$

- Proof by induction:
  - Base case: P_n is true for n = 0 (just check it!)
  - Induction step: If P_n is true for some n≥0, then P_{n+1} is true.

$$P_{n+1}: 0 + 1 + \ \ldots + n + (n + 1) = \frac{(n + 1)\big((n + 1) + 1\big)}{2} = \frac{(n + 1)(n + 2)}{2}$$

Is P_{n+1} true?

Check: $0 + 1 + \ \ldots + n + (n + 1) = \frac{n(n+1)}{2} + (n + 1) = \frac{(n+1)(n+2)}{2}$

- First equality holds by assumed truth of P_n!

An Aside: Summation Notation

A sum of the form $a_0 + a_1 + \cdots a_n$ is frequently shortened to $\sum_{i=0}^{n} a_i$

Using this notation, the induction step of our previous proof would look like

- Induction step: If P_n is true for some $n \geq 0$, then P_{n+1} is true.

$$P_{n+1}: \sum_{i=0}^{n+1} i = \frac{(n+1)\big((n+1)+1\big)}{2} = \frac{(n+1)(n+2)}{2}$$

Is P_{n+1} true?

Check:

$$\sum_{i=0}^{n+1} i = \left(\sum_{i=0}^{n} i\right) + (n+1) = \frac{n(n+1)}{2} + (n+1) = \frac{(n+1)(n+2)}{2}$$

The second equality holds by assumed truth of P_n!

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

Proof: Using summation notation

- Base case: $n = 0$
  - LHS: $\sum_{i=0}^{0} 2^i = 2^0 = 1$
  - RHS: $2^{0+1} - 1 = 2 - 1 = 1$ ✓
- Induction Step: Show that, for $n \geq 0$, whenever

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

- Then

$$\sum_{i=0}^{n+1} 2^i = 2^{(n+1)+1} - 1$$

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

Induction Step: Show that, for $n \geq 0$, whenever

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

Then

$$\sum_{i=0}^{n+1} 2^i = 2^{(n+1)+1} - 1 = 2^{n+2} - 1$$

Well,

$$\sum_{i=0}^{n+1} 2^i = \left(\sum_{i=0}^{n} 2^i\right) + 2^{n+1} = (2^{n+1} - 1) + 2^{n+1} = 2^{n+2} - 1 \quad \checkmark$$

Mathematical Induction

Prove: $1^3 + 2^3 + \cdots + n^3 = (1 + 2 + \cdots + n)^2$

Note: This starts at n=1, not n=0. Is this a problem?

- No. We just
  - Make our base case n=1, and
  - Show that whenever the property holds for some n≥1 then it holds for n+1

Base Case: n = 1

LHS: $1^3 = 1$ and RHS: $1^2 = 1$ ✓

Induction step: Assume that for some n ≥ 1

$$1^3 + 2^3 + \cdots + n^3 = (1 + 2 + \cdots + n)^2$$

Now show that

$$1^3 + 2^3 + \cdots + (n+1)^3 = (1 + 2 + \cdots + (n+1))^2$$

IS: $1^3 + 2^3 + \cdots + (n+1)^3 = (1 + 2 + \cdots + (n+1))^2$

$$1^3 + 2^3 + \cdots + (n+1)^3 = (1^3 + 2^3 + \cdots + n^3) + (n+1)^3$$

Induction☞

$$= (1 + 2 + \cdots + n)^2 + (n+1)^3$$

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

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

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

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

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

$$= (1 + 2 + \cdots + (n+1))^2 \quad ✓$$

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

```
public static int fact(int n) {
      if (n==0) return 1;
      else return n*fact(n-1);
}
```

- Example: factorial
  - Prove that fact(n) requires n multiplications
    - Base case: n = 0 returns 1, using 0 multiplications ✓
    - Assume true for some $n \geq 0$, so fact(n) requires n multiplications.
    - fact(n+1) performs one multiplication (n+1)*fact(n). But, by induction, fact(n) requires n multiplications. Therefore fact(n) requires 1+n multiplications. ✓

Recursive Contains

Recall again our recursive contains method for a Singly-Linked List

```
// Pre: value is not null
public static boolean contains(Node<String> anode, String v) {
   if( aNode == null ) return false;
   return v.equals(aNode.value()) || contains(aNode.next(), v);
}
```

Claim: `contains` works correctly for any list of size n $\geq$ 0

- Base Case: n=0 [`aNode` is `null`]
  - The `if` statement immediately returns `false`—the correct answer ✓
- Induction step
  - Suppose `contains` works correctly on all lists of size n, for some n $\geq$ 0.
  - Show that it works correctly on all lists of size n+1
- Proof: If n $\geq$ 0, then n+1 $\geq$ 1, so the first call to `contains` will execute the final line of the method.
  - If `v.equals(aNode.value()` is `true`, then correct result is returned
  - Otherwise, `contains` is called on a list of size n, which by assumption returns the correct result (our *induction hypothesis*) ✓

Counting Method Calls

- Example: Fibonacci
  - Prove that fib(n) makes at least fib(n) calls to fib()
    - Base cases: n = 0: 1 call; n = 1; 1 call ✓
    - Assume that for some $n \geq 2$, fib(n-1) makes at least fib(n-1) calls to fib() and fib(n-2) makes at least fib(n-2) calls to fib().
    - Claim: Then fib(n) makes at least fib(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!
- Aside: Can show by induction that for n > 10: fib(n) > $(1.5)^n$
  - Thus the number of calls grows exponentially!
  - Verifying our empirical observation that computing fib(45) was slow!

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 recBinSrch(int a[], int value,
                int low, int high) {
   if (low > high) return -1;
   else {
    int mid = (low + high) / 2;                    //mid index
    if (a[mid] == value) return mid;
    else if (a[mid] < value)                       //look high!
        return recBinSarch(a, value, mid + 1, high);
    else                                           //look low!
        return recBinSarch(a, value, low, mid - 1);
   }
}
```

Binary Search takes O(log n) Time

Can we use induction to prove this?

- Induction on size of slice : $n = high - low + 1$
- Claim: If $n > 0$, then recBinSrch performs at most $c\ (1+ \log n)$ operations
  - where c is *twice* the number of statements in recBinSrch
    - All logs are base 2 unless specified differently
    - Recall : $\log 1 = 0$
- Base case: $n = 1$: Then low = high so only c statements execute (method runs twice) and $c \leq c(1+\log 1)$ ✓
- Assume that claim holds for some $n \geq 1$, 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….

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 $k \geq 0$

1. $P_0, P_1, ..., P_k$ are true, and
2. For <u>*every*</u> $n \geq k$, *if* $P_0, P_1, ..., P_n$ are true, *then* P_{n+1} is true

Then *all* of the statements are true!

Binary Search takes O(log n) Time

Try again now:

- Assume that for some $n \geq 1$, the claim holds *for all* $i \leq n$, 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)) operations
    - We want to show that this is at most c(1 + log(n+1))….

Binary Search takes O(log n) Time

This gives a total of at most $c + c\left(1 + \log_2 \frac{n}{2}\right)$ operations

- c statements in original call to recBinSrch, and
- $c\left(1 + \log_2 \frac{n}{2}\right)$ statements in recursive calls

So

$$
\begin{aligned}
c + c\left(1 + \log_2 \frac{n}{2}\right) &= c + c\left(\log_2 2 + \log_2 \frac{n}{2}\right) \\
&= c + c\left(\log_2 2 \cdot \frac{n}{2}\right) \\
&= c + c \log_2 n \\
&= c(1 + \log_2 n)
\end{aligned}
$$

which is what we wanted to show ✓

In Summary

- Two versions of the principle of mathematical induction
  - Strong: Given the truth of a *fixed number* of base cases $P_1, \ldots, P_k$, if we can show that *for every $n \geq k$*:
    - *If* $P_1, \ldots, P_n$ are true, *then* P_{n+1} is true

    Then all of the statements are true
  - Weak: Given the truth of a *fixed number* of base cases $P_1, \ldots, P_k$, if we can show that *for every $n > k$*:
    - *If* the k statements $P_{n-k}, P_{n-(k-1)}, \ldots, P_{n-1}$ are true, *then* P_n is true

    Then all of the statements are true
    - That is, if *for every $n > k$* we can show that whenever the k statements immediately preceding statement P_n are true, *then* P_n is true
- Strong induction is needed when a problem is being decomposed into subproblems much smaller size