Lecture 7

Proof by Induction

These are the steps we follow when doing proof by induction:

- Declare that we’re using induction
- Directly prove whatever base cases are necessary.
- State the assumption that the observation holds for all values from the base case up to but not including the $n^{th}$ case.
- Prove, from simpler cases, that the $n^{th}$ case also holds.
- Claim that, by mathematical induction on $n$, the observation is true for all cases more complex than the base case.

The classic example, with formal proof:

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

Proof (induction):

base case:
show that $\sum_{i=0}^{0} i = 0$, trivial by definition of summation.

inductive case: if $\sum_{i=0}^{k} i = \frac{k(k+1)}{2}$ for all $k = \{0, 1, \ldots, n-1\}$ then $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$

By the inductive hypothesis, $\sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}$

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

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

$$= \frac{n^2 + n}{2}$$

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

By induction on $n$, we have shown that $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$ for $n \geq 0$. 
Recursive Selection Sort

Recursive Selection Sort, basic algorithm:

• find largest item in whole list
• move largest item to last index
• sort list, not including last index

Use induction on size of list to prove correctness:

Base case:

recursive selection sort works if lastIndex <= 0

if lastIndex == 0, then list size is 1, and sorted by definition

Inductive case:

assume the recSelSort works on lists of size n-1.

On list of size n, we:

• Find largest item in whole list
• move largest item to lastIndex

  Now, all items in indices 0..lastIndex - 1 are smaller than or equal to the item in lastIndex

• sort items in list [0..lastIndex - 1] (we know this works by our assumption)

  Now, we have [0..lastIndex - 1] sorted, and item in lastIndex that is greater than or equal to all values in [0..lastIndex - 1]. Thus, we have sorted list.

Complexity:

We need to think to determine a good guess for the complexity.

Let’s reason,
for list of size 1, we do 0 comparisons in indexOfLargest
for list of size 2, we do 0 + 1 comparison in indexOfLargest
for list of size 3, we do 0 + 1 + 2 comparisons in indexOfLargest
... 
for list of size n, we do 0 + 1 + 2 + ... + n - 1 comparisons in indexOfLargest

We note that there is only a constant amount of work done outside indexOfLargest for each call to recSelSort

Our hypothesis: For list of size n, we do $\frac{n(n-1)}{2}$ comparisons.

Let’s prove our hypothesis.

Proof(induction on size of list n)
base case: list of size 1 requires \( \frac{1(0)}{2} = 0 \) comparisons. Code does 0 comparisons. Base case holds.

Inductive step:

Assume for list of size \( n - 1 \), do \( \frac{(n-1)(n-2)}{2} \) comparisons, show that for list of size \( n \), we do \( \frac{(n)(n-1)}{2} \) comparisons.

This is almost identical to our first example. Won’t repeat the work here.

**fastPower Example**

Another example. We can calculate a base raised to an exponent in the obvious way. For example, \( 2^4 \) can be written as \( 2 \times 2 \times 2 \times 2 \). This works great for small exponents, but when the exponent becomes bigger, it can be a lot of work.

fastPower algorithm gives a better way.

- if exponent is 0, return 1 (base case)
- When the exponent is even, return \( x^n = (x^2)^{n/2} \). (Now, the exponent is cut in half)
- When the exponent is odd, return \( x^n = x \times x^{n-1} \). (the exponent decreases by one)

This seems reasonable, but can we prove that it really works?

Proof by induction on size of the exponent \( exp \).

base case: exponent = 0, fastPower(base, 0) returns 1. \( base^0 = 1 \). holds.

inductive case: assume fastPower(base, exp) = \( base^{exp} \) for all \( exp < n \).

Show fastPower(base, n) = \( base^n \)

two cases: case 1, \( n \) is odd:

\[
fastPower(base, n) = base * fastPower(base, n - 1)
= base * base^{n-1} \text{ by inductive hypothesis (ie, our assumption)}
= base^n
\]

case 2: \( n \) is even:

\[
fastPower(base, n) = fastPower(base * base, n/2)
= (base^2)^{n/2} \text{ by inductive hypothesis (ie, our assumption)}
= base^n
\]