Lecture 4: Proof Techniques

COSC242: Algorithms and Data Structures

Brendan McCane

Department of Computer Science, University of Otago

Proof by Contradiction

Suppose you are asked: Is $n = \Theta(n^2)$? It is easy to see that $n = O(n^2)$. Just take $c = n_0 = 1$. But we have to think carefully about whether $n^2 = O(n)$.

First, use your intuition: n^2 has a sharp upward curve, whereas n is a straight line with gradient 1. So surely n^2 can't scale up as well as n?

But to be convinced that $n^2 \neq O(n)$, we need more than intuition, so we use **proof by contradiction**.

Proof by Contradiction

There are exactly two cases: either $n^2 = O(n)$ or $n^2 \neq O(n)$. We eliminate one by a thought experiment:

Imagine for a moment that $n^2 = O(n)$.
Then c, n_0 would have to exist so that $n^2 \leq c \cdot n$ for all $n \geq n_0$.
Then $n^2/n \leq c$, so that $n \leq c$.

But this is absurd: c is a constant and n can be as big as you like, say $n = c + 1$.
Since $c + 1 \not\leq c$, we have reached a contradiction. This leaves only the other case:
$n^2 \neq O(n)$.

Proof by Contradiction

Proofs by contradiction are useful for proving that something is not the case.

They always have the same structure:

1. Assume the opposite of what you want to prove.
2. Use the rules of logic or math to arrive at a statement that can't be true (an absurdity, or a paradox).
3. If you've applied the rules correctly, then the only thing that could be wrong must be the assumption at the start. Therefore the opposite of the assumption must be true.

Other sorts of proof

It's harder to use proof by contradiction to prove something is the case. So what alternatives do we have?

Let's say we have some finite set, X, of integers, and we want to prove that all of it's elements are divisible by 3. How can I do that?

E.g. if $X = \{3, 6, 33, 66, 99, 321\}$

What about $X = \{3, 6, 9, 2023\}$?

How big is big enough?

We are usually interested in comparing algorithms that have a number of objects as input (e.g. integers, or names or ...).

We are interested in how the algorithms scale up with the number of input objects.

Since we are dealing with a "number of objects", the performance of the algorithms can be considered as a function of an integer (n, the number of objects), regardless of the type of object.

Should we put an upper limit on the number of objects that would be input? How big is big enough?

As an aside, one of the worst decisions in operating systems design was when Microsoft decided that 640KB of RAM was more than anyone would ever need.

Infinite Sets

Since we don't want to specify an upper limit on the number of items, we need a proof method that will work with *any* number of items.

Our sets of interest are infinite!

As an example: consider the set consisting of numbers $f(n) = n^3 + 2n$ where n is a positive integer. In other words we have:

$$X = \{3, 12, 33, \ldots\}$$

Are all the elements of X divisible by 3?

Notice that our input domain (the numbers n) is the set of positive integers.

Proof by Induction

Problem: Prove that all numbers, $f(n) = n^3 + 2n$ are divisible by 3 for n an integer and $n > 0$.

All proofs by induction follow the same recipe:

Base Case:	show the statement is true for the first element of the set
Inductive Step:	assume the statement is true for the k^{th} element, and show it must therefore be true for the $(k+1)^{th}$ element of the set.

If we can show the inductive step, then we have an effective procedure for showing that the statement must be true for all elements of the set: for a given element, say the k^{th} element, just start at the base element and apply the inductive step k times. The beauty of proofs by induction is that we don't have to apply the inductive step k times - just the once for an arbitrary element is enough.

Proof.

Base case: *first prove the statement is true for the first element of the set.*
$f(1) = 3$, which is divisible by 3.
Inductive step: *assume that the statement is true for arbitrary k, and show it must be true for $k+1$.*
Assume $f(k) = k^3 + 2k$ is divisible by 3. Show that $f(k+1) = (k+1)^3 + 2(k+1)$ is also divisible by 3:

$$\begin{aligned} f(k+1) &= (k+1)^3 + 2(k+1) \\ &= k^3 + 3k^2 + 3k + 1 + 2k + 2 \\ &= k^3 + 2k + 3k^2 + 3k + 3 \\ &= f(k) + 3(k^2 + k + 1) \end{aligned}$$

We have assumed that $f(k)$ is divisible by 3, and the second term must also be divisible by 3, therefore $f(k+1)$ must be divisible by 3.

Another Example (insertion sort)

Prove, by induction, that

$$1 + 2 + 3 + \ldots + n - 1 = n(n-1)/2$$

This example is slightly different because we have a left hand side and a right hand side, but the inductive proof follows the same pattern.

Let's call the LHS, l(n), and the RHS r(n). So:

$$\begin{aligned} l(n) &= 1 + 2 + 3 + \ldots + n - 1 \\ r(n) &= n(n-1)/2 \end{aligned}$$

And we want to show that $l(n) = r(n)$ for all $n \geq 1$.

Proof.

Base case: $l(1) = 0 = r(1)$, so the base case is true.
Inductive step: assume that $l(k) = r(k)$ and show that $l(k+1) = r(k+1)$.

$$\begin{aligned}
l(k+1) &= 1 + 2 + 3 + \ldots + k - 1 + k \\
&= l(k) + k = r(k) + k \\
r(k+1) &= (k+1)(k+1-1)/2 \\
&= (k+1)(k)/2 \\
&= (k^2 + k)/2 \\
&= (k^2 - k + 2k)/2 \\
&= (k^2 - k)/2 + 2k/2 \\
&= r(k) + k = l(k+1)
\end{aligned}$$