Induction

Dr. Mattox Beckman

Illinois Institute of Technology
Department of Computer Science

Objectives

- Review the parts of an inductive proof.
- Relate those parts to a recursive function.

Induction

A proof by induction works by making two steps do the work of an infinite number of steps. It's really a way of being very lazy!

- Pick a property $P(n)$ which you'd like to prove for all n.
- **Base case:** Prove $P(n)$, for $n = 1$, or whatever n's smallest value should be.
- **Induction Case:** You want to prove $P(n)$, for some general n. To do that, *assume* that $P(n-1)$ is true, and use that information to prove that $P(n)$ has to be true.

The idea is that there are an infinite number of n such that $P(n)$ is true. But with this technique you only had to prove two cases.

Induction Example

To Prove: Let $P(n)$ = "The sum of the first n odd numbers is n^2."

Base Case: Let $n = 1$. Then n^2 = 1, and the sum of the list $\{1\}$ is 1; therefore the base case holds.

Induction Case: Suppose you need to show that this property is true for some n. First, pretend that somebody else already did all the work of proving that $P(n-1)$ is true. Now use that to show that $P(n)$ is true, and take all the credit.

If $\{1, 3, 5, \ldots, 2n-3\} = (n-1)^2$, then add $2n-1$ …

$$\{1, 3, 5, \ldots, 2n-3, 2n-1\} = (n-1)^2 + 2n - 1$$
$$\Rightarrow n^2 - 2n + 1 + 2n - 1 \Rightarrow n^2$$

Recursion

A recursive routine has a similar structure. You have a base case, a recursive case, and a conditional to check which case is appropriate.

- Pick a function $f(n)$ which you'd like to compute for all n.
- **Base case:** Compute $f(n)$, for $n = 1$, or whatever n's smallest value should be.
- **Recursive Case:** Assume that someone else already computed $f(n-1)$ for you. Use that information to compute $f(n)$, and then take all the credit.

Iterating Recursion Example

Suppose you want a recursive routine that computes the *n*th square.

```
nthsq 0 = 0
nthsq n = 2*n-1 + nthsq (n-1)
```

- The conditional checks which case is active.
- Line 1 is the base case — it stops the recursion.
- Lines 2 is the recursive case.

Important things about recursion

```
nthsq 0 = 0
nthsq n = 2*n-1 + nthsq (n-1)
```

- Your base case has to stop the computation.
- Your recursive case has to call the function with a *smaller* argument than the original call.
- Your `if` statement has to be able to tell when the base case is reached.
- Failure to do any of the above will cause an infinite loop.