Prove: For every n ∈ $\mathbf{Z}^+$, $\sum_{i=1}^{n} i = (n+\frac{1}{2})^2/2$

P(n) ≡ $\sum_{i=1}^{n} i = (n+\frac{1}{2})^2/2$
Base case: P(1) clearly holds
I.H.: Assume that P(k) holds for an arbitrary integer k

Inductive step: We will now show that P(k) → P(k+1)

- $1 + 2 + \ldots + k = (k + ½)^2/2$ by I.H.
- $1 + 2 + \ldots + k+1 = (k + ½)^2/2 + k + 1$
- $= (k^2 + 3k + 9/4)/2$
- $= (k + 3/2)^2/2$
- $= [(k+1) + ½]^2/2$

Conclusion: Since we have proved the base case and the inductive case, the claim holds by mathematical induction ❏

Prove: For every n ∈ $\mathbf{Z}^+$, if x,y ∈ $\mathbf{Z}^+$ and max(x,y) = n, then x=y

P(n) ≡ max(x,y) = n → x = y
Base case: P(1): If max(x,y) = 1, then x=y=1 since x,y ∈ $\mathbf{Z}^+$
I.H.: Assume that P(k) holds for an arbitrary integer k

Inductive step: We will now show that P(k) → P(k+1)

- Let max(x,y) = k + 1
- Then, max(x-1,y-1) = k, so by the I.H. x - 1 = y - 1
- It thus follows that x = y

Problem: Our induction is on the variable k, so we have no guarantee that x-1 or y-1 are positive integers, only that k-1 is a positive integer…

Conclusion: Since we have proved the base case and the inductive case, the claim holds by mathematical induction ❏

Recall that mathematical induction let us prove universally quantified statements

Goal: Prove $\forall x \in \mathbf{N}\ P(x)$.

Intuition: If P(0) is true, then P(1) is true. If P(1) is true, then P(2) is true…

Procedure:

1. Prove P(0)
2. Show that $P(k) \rightarrow P(k+1)$ for any arbitrary k
3. Conclude that P(x) is true $\forall x \in \mathbf{N}$

$$\frac{\begin{array}{l} P(0) \\ P(k) \rightarrow P(k+1) \end{array}}{\therefore \forall x \in \mathbf{N}\ P(x)}$$

Strong mathematical induction is another flavor of induction

Goal: Prove $\forall x \in \mathbf{N}\ P(x)$.

Procedure:

1. Prove P(0)
2. Show that $[P(0) \wedge P(1) \wedge \ldots \wedge P(k)] \rightarrow P(k+1)$ for any arbitrary k
3. Conclude that P(x) is true $\forall x \in \mathbf{N}$

$$\frac{\begin{array}{l} P(0) \\ [P(0) \wedge P(1) \wedge \ldots \wedge P(k)] \rightarrow P(k+1) \end{array}}{\therefore \forall x \in \mathbf{N}\ P(x)}$$

So what's the big deal?

Recall: In mathematical induction, our inductive hypothesis allows us to assume that P(k) is true and use this knowledge to prove P(k+1)

However, in strong induction, we can assume that $P(0) \land P(1) \land \ldots \land P(k)$ is true before trying to prove P(k+1)

For certain types of proofs, this is much easier than trying to prove P(k+1) from P(k) alone.

For example…

Show that if n is an integer greater than 1, then n can be written as the product of primes

$P(n) \equiv$ n can be written as a product of primes

Base case: P(2): $2 = 2^1$ ✓

I.H.: Assume that $P(2) \land \ldots \land P(k)$ holds for an arbitrary integer k

Inductive step: We will now show that $[P(2) \land \ldots \land P(k)] \rightarrow P(k+1)$

- Two cases to consider: k+1 prime and k+1 composite
- If k+1 is prime, then we're done
- If k+1 is composite, then by definition, k+1 = ab
- Since $2 \leq a < k+1$ and $2 \leq b < k+1$, a and b can be written as products of primes by the I.H.
- Thus, k+1 can be written as a product of primes

Conclusion: Since we have proved the base case and the inductive case, the claim holds by strong induction ❑

Is strong induction somehow more powerful than mathematical induction?

The ability to assume $P(0) \wedge P(1) \wedge \ldots \wedge P(k)$ true before proving $P(k+1)$ seems more powerful than just assuming $P(k)$ is true

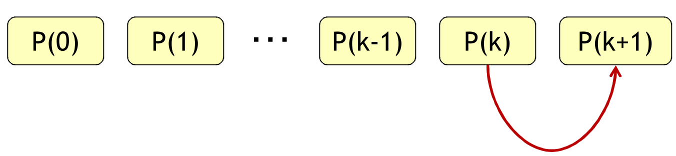
Perhaps surprisingly, mathematical induction and strong induction are all equivalent!

That is, a proof using one of these methods can always be written using the other two methods

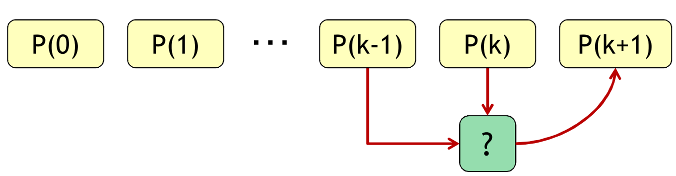
This may not be easy, though!

So when should we use strong induction?

If it is straightforward to prove $P(k+1)$ from $P(k)$ alone, use mathematical induction



If it would be easier to prove $P(k+1)$ using one or more $P(j)$ for $0 \leq j < k$, use strong induction



There are many uses of induction in computer science!

Proof by induction is often used to reason about:

- Algorithm properties (correctness, etc.)
- Properties of data structures
- Membership in certain sets
- Determining whether certain expressions are well-formed
- …

To begin looking at how we can use induction to prove the above types of statements, we first need to learn about recursion

Sometimes, it is difficult or messy to define some object explicitly

Recursive objects are defined in terms of themselves

We often see the recursive versions of the following types of objects:

- Functions
- Sequences
- Sets
- Data structures

Let's look at some examples…

Recursive functions are useful

When defining a recursive function whose domain is the set of natural numbers, we have two steps:

1. Basis step: Define the behavior of $f(0)$
2. Recursive step: Compute $f(n+1)$ using $f(0), \ldots, f(n)$

Doesn't this look a little bit like strong induction?

Example: Let $f(0) = 3$, $f(n+1) = 2f(n) + 3$

- $f(1) = 2f(0) + 3 = 2(3) + 3 = 9$
- $f(2) = 2f(1) + 3 = 2(9) + 3 = 21$
- $f(3) = 2f(2) + 3 = 2(21) + 3 = 45$
- $f(4) = 2f(3) + 3 = 2(45) + 3 = 93$
- …

Some functions can be defined more precisely using recursion

Example: Define the factorial function $F(n)$ recursively

1. Basis step: $F(0) = 1$
2. Recursive step: $F(n+1) = (n+1) \times F(n)$

Note:
$$\begin{aligned} F(4) &= 4 \times F(3) \\ &= 4 \times 3 \times F(2) \\ &= 4 \times 3 \times 2 \times F(1) \\ &= 4 \times 3 \times 2 \times 1 \times F(0) \\ &= 4 \times 3 \times 2 \times 1 \times 1 = 24 \end{aligned}$$

The recursive definition avoids using the "…" shorthand!

Compare the above definition our old definition:

- $F(n) = n \times (n-1) \times \ldots \times 2 \times 1$

It should be no surprise that we can also define recursive sequences

Example: The Fibonacci numbers, $\{f_n\}$, are defined as follows:

- $f_0 = 1$
- $f_1 = 1$
- $f_n = f_{n-1} + f_{n-2}$

This is like strong induction, since we need more than f_{n-1} to compute f_n.

Calculate: f_2, f_3, f_4, and f_5

- $f_2 = f_1 + f_0 = 1 + 1 = 2$
- $f_3 = f_2 + f_1 = 2 + 1 = 3$
- $f_4 = f_3 + f_2 = 3 + 2 = 5$
- $f_5 = f_4 + f_3 = 5 + 3 = 8$

This gives us the sequence $\{f_n\}$ = 1, 1, 2, 3, 5, 8, 13, 21, 34, …

Recursively defined sets are also used frequently in computer science

Simple example: Consider the following set S

1. Basis step: $3 \in S$
2. Recursive step: if $x \in S$ and $y \in S$, then $x + y \in S$

Claim: The set S thus contains every multiple of 3.

Intuition: $3 \in S$, $6 \in S$ (since 3 and 3 are in S), $9 \in S$ (since 3 and 6 are in S), …

We'll show how we can prove this claim during the next lecture…

Recursion is used heavily in the study of strings

Let: Σ be defined as an alphabet

- Binary strings: $\Sigma = \{0, 1\}$
- Lower case letters: $\Sigma = \{a, b, c, \ldots, z\}$

We can define the set Σ^* containing all strings over the alphabet Σ as follows:

1. Basis step: $\lambda \in \Sigma^*$
2. Recursive step: If $w \in \Sigma^*$ and $x \in \Sigma$, then $wx \in \Sigma^*$

λ is the empty string containing no characters

Example: If $\Sigma = \{0, 1\}$, then $\Sigma = \{\lambda, 0, 1, 01, 11, \ldots\}$

This recursive definition allows us to easily define important string operations

Definition: The length $l(w)$ of a string can be defined as follows:

1. Basis step: $l(\lambda) = 0$
2. Recursive step: $l(wx) = l(w) + 1$ if $w \in \Sigma^*$ and $x \in \Sigma$

Example:
$$\begin{aligned} l(1001) &= l(100) + 1 \\ &= l(10) + 1 + 1 \\ &= l(1) + 1 + 1 + 1 \\ &= l(\lambda) + 1 + 1 + 1 + 1 \\ &= 0 + 1 + 1 + 1 + 1 \\ &= 4 \end{aligned}$$

We can define sets of well-formed formulae recursively

This is often used to specify the operations permissible in a given formal language (e.g., a programming language)

Example: Defining propositional logic

1. Basis step: **T**, **F**, and *s* are well-formed propositional logic statements (where *s* is a propositional variable)
2. Recursive step: If E and F are well-formed statements, so are
   - $(\neg E)$
   - $(E \wedge F)$
   - $(E \vee F)$
   - $(E \rightarrow F)$
   - $(E \leftrightarrow F)$

Example

Question: Is $((p \wedge q) \rightarrow (((\neg r) \vee q) \wedge t))$ well-formed?

- Basis tells us that p, q, r, t are well-formed
- 1st application: $(p \wedge q)$, $(\neg r)$ are well-formed
- 2nd application: $((\neg r) \wedge q)$ is well-formed
- 3rd application: $(((\neg r) \vee q) \wedge t)$
- 4th application: $((p \wedge q) \rightarrow (((\neg r) \vee q) \wedge t))$ is well-formed

✓

Final Thoughts

- Strong induction lets us prove universally quantified statements using this inference rule:

$$\begin{array}{l} P(0) \\ [P(0) \land P(1) \land \ldots \land P(k)\ ] \rightarrow P(k+1) \\ \hline \therefore \forall x \in \mathbf{N}\ P(x) \end{array}$$

- We can construct recursive
  - Sets
  - Sequences
  - Grammars