Induction versus recursion

- Recursion: method m(n): implement m for any n in some domain (e.g. n>=0) by testing for a base case and returning result or creating the result using the solution of a smaller problem, reducing e.g. from n to n-1.
- Induction: predicate P(n): show P(n) is true for any n in a domain (e.g. n>=0) by showing P(0) holds (base) and that P(n) → P(n+1). i.e. extending from n to n+1
Motivation

- Many mathematical statements have the form:
  \[ \forall n \in \mathbb{N}, P(n) \]
  \( P(n) \): Logical predicate

- Example: For every positive value of \( n \),
  \[ 1 + 2 + \ldots + n = \frac{n(n + 1)}{2}. \]

- **Predicate** – propositional function that depends on a variable, and has a truth value once the variable is assigned a value. An assert statement checks a predicate in your program.

- Mathematical induction is a proof technique for proving such predicates
Proving $P(3)$

- Suppose we can show that $P(1)$ holds and that
  \[ P(n) \rightarrow P(n + 1) \quad \forall n \geq 1. \]

  as in recursion, we call $P(1)$ the base case
  We call $P(n)$ the induction hypothesis and
  $P(n) \rightarrow P(n + 1)$ the implication

Proof:

1. $P(1)$ [base]
2. $P(1) \rightarrow P(2)$. [base implies $P(2)$]
3. $P(2)$. [therefore $P(2)$] ( if $p$ and $p \rightarrow q$ then $q$)
4. $P(2) \rightarrow P(3)$.
5. $P(3)$. [therefore $P(3)$] (etcetera)

We can construct a proof for every finite value of $n$
Example: \[ 1 + 2 + \ldots + n = \frac{n(n + 1)}{2}. \]

- Let \( F(n) = 1 + 2 + \ldots + n \)
- **Predicate** \( P(n) \): \( F(n) = \frac{n(n + 1)}{2} \)
- **Verify base case**: \( P(1) \): \( 1(1 + 1)/2 = 1 \).
- **Implication**: Show that \( P(n) \Rightarrow P(n+1) \):

\[
F(n) = \frac{n(n + 1)}{2} \Rightarrow F(n + 1) = \frac{(n + 1)(n + 2)}{2}.
\]

So we can use \( F(n) = \frac{n(n + 1)}{2} \) in our proof, we call \( F(n) = \frac{n(n + 1)}{2} \) the **induction hypothesis**

\[
F(n + 1) = 1 + 2 + \ldots + n + (n + 1)
\]

\[
= F(n) + n + 1 = \frac{n(n + 1)}{2} + n + 1
\]

\[
= \frac{n(n + 1)}{2} + 2(n + 1)/2 = \frac{(n + 1)(n + 2)}{2}.
\]
A Geometrical interpretation

1: 

2: 

3: 

Put these blocks, which represent numbers, together to form sums:

1 + 2 = 

1 + 2 + 3 =
A Geometrical interpretation

Area is \( \frac{n^2}{2} + \frac{n}{2} = \frac{n(n + 1)}{2} \)
The Principle of Mathematical Induction

- Let $P(n)$ be a statement that, for each natural number $n$, is either true or false.

- To prove that $\forall n \in \mathbb{N}, P(n)$, it suffices to prove:
  - $P(1)$ is true. (base case)
  - $\forall n \in \mathbb{N}, P(n) \rightarrow P(n + 1)$. (inductive step)

- This is not magic.

- It is a recipe for constructing a proof for an arbitrary $n \in \mathbb{N}$. 
Mathematical Induction and the Domino Principle

If

the 1st domino falls over

and

the nth domino falls over implies that domino \((n + 1)\)

falls over

then

domino \(n\) falls over for all \(n \in \mathbb{N}\).
Proof by induction

- 3 steps:
  - Prove $P(1)$. [the basis]
  - Assume $P(n)$ [the induction hypothesis]
  - Using $P(n)$ prove $P(n + 1)$ [the inductive step]
Example

- Show that any postage of $\geq 8\$ can be obtained using 3\$ and 5\$ stamps.
- First check for a few values:
  - 8\$ = 3\$ + 5\$
  - 9\$ = 3\$ + 3\$ + 3\$
  - 10\$ = 5\$ + 5\$
  - 11\$ = 5\$ + 3\$ + 3\$
  - 12\$ = 3\$ + 3\$ + 3\$ + 3\$
- How to generalize this?
Example

- Let $P(n)$ be the statement “$n$ cents postage can be obtained using 3¢ and 5¢ stamps”.

- Want to show that “$P(k)$ is true” implies “$P(k+1)$ is true” for all $k \geq 8$.

- 2 cases:
  1) $P(k)$ is true and the $k$ cents contain at least one 5¢.
  2) $P(k)$ is true and the $k$ cents do not contain any 5¢.
**Example**

**Case 1:** k cents contain at least one 5¢ stamp.

Replace 5¢ stamp by two 3¢ stamps.

**Case 2:** k cents do not contain any 5¢ stamp.

Then there are at least three 3¢ stamp.

Replace three 3¢ stamps by two 5¢ stamps.
Examples

- Show that $1 + 2 + 2^2 + \ldots + 2^n = 2^{n+1} - 1$
- Show that for $n \geq 4$ $2^n < n!$
- Show that $n^3 - n$ is divisible by 3 for every positive $n$
- Show that the sum of the first $n$ odds equals $n^2$
  or $1 + 3 + 5 + \ldots + (2n-1) = n^2$

Let’s do some of the proofs
1. Let $P(n)$ be the statement that $2^0 + 2^1 + \ldots + 2^n = 2^{n+1} - 1$ for positive integers $n$.

a) What is the statement $P(1)$?

$$P(1) \text{ is } 2^0 = 2^1 - 1$$

b) Show that $P(1)$ is true, completing the base of the induction.

$$2^0 = 2^1 - 1 = 1$$

c) What is the inductive hypothesis?

$$P(n): \quad 2^0 + 2^1 + \ldots + 2^n = 2^{n+1} - 1$$

d) What do you need to prove in the inductive step?

$$P(n) \rightarrow P(n+1) \text{ or}$$

$$2^0 + 2^1 + \ldots + 2^n = 2^{n+1} - 1 \implies 2^0 + 2^1 + \ldots + 2^n + 2^{n+1} = 2^{n+2} - 1$$
e) Complete the inductive step.

\[ 2^0 + 2^1 + \ldots + 2^n + 2^{n+1} = \]

\[ 2^0 + 2^1 + \ldots + 2^n + 2^{n+1} = \]

\[ 2^{n+1} - 1 + 2^{n+1} = \]

\[ 2^{n+2} - 1 \]
Show that \( n^3 - n \) is divisible by 3 for every positive \( n \)

Base: \( 1 - 1 = 0 \) \( 0 \% 3 = 0 \)

Step: assume \( n^3 - n \) \% 3 = 0

then

\[
(n+1)^3 - (n+1) =
\]

\[
(n + 1) (n^2 + 2n + 1) - (n+1) =
\]

\[
n^3 + 2n^2 + n + (n^2 + 2n + 1) - (n+1) =
\]

\[
n^3 - n + 3n^2 + 3n
\]

\( n^3 - n \% 3 = 0 \) and \( (3n^2 + 3n)\%3 = 0 \)

therefore \( (n^3 - n + 3n^2 + 3n) \% 3 = 0 \)
Show that $1 + 3 + \ldots + (2n-1) = n^2$
A Geometrical interpretation
All horses have the same color

- **Base case:** If there is only one horse, there is only one color.

- **Induction step:** Assume as induction hypothesis that within any set of \( n \) horses, there is only one color. Now look at any set of \( n + 1 \) horses. Number them: 1, 2, 3, ..., \( n \), \( n + 1 \). Consider the sets \{1, 2, 3, ..., \( n \}\} and \{2, 3, 4, ..., \( n + 1 \}\}. Each is a set of only \( n \) horses, therefore within each there is only one color. But the two sets overlap, so there must be only one color among all \( n + 1 \) horses.

- This is clearly wrong, but can you find the flaw?
All horses have the same color

- The inductive step requires that $k \geq 3$, otherwise there is no intersection! So $P(2)$ should be the base case, which is obviously incorrect.

- In the book there is a similar example.
More induction examples

Let $n$ be a positive integer. Show that every $2^n \times 2^n$ chessboard with one square removed can be tiled using right triominoes, each covering three squares at a time.
Strong induction

- Induction:
  - $P(1)$ is true.
  - $\forall n \in \mathbb{N}, P(n) \rightarrow P(n + 1)$.
  - Implies $\forall n \in \mathbb{N}, P(n)$

- Strong induction:
  - $P(1)$ is true.
  - $\forall n \in \mathbb{N}, (P(1) \land P(2) \land \ldots \land P(n)) \rightarrow P(n + 1)$.
  - Implies $\forall n \in \mathbb{N}, P(n)$
Example

- Prove that all natural numbers $\geq 2$ can be represented as a product of primes.

- **Basis:** 2: 2 is a prime.

- **Assume** that 1, 2, ..., $n$ can be represented as a product of primes.
Example

- **Show** that n+1 can be represented as a product of primes.
  - If n+1 is a prime: It can be represented as a product of 1 prime, itself.
  - If n+1 is composite: Then, \( n + 1 = ab \), for some \( a, b < n + 1 \).
    - Therefore, \( a = p_1 p_2 \ldots p_k \) & \( b = q_1 q_2 \ldots q_l \), where the \( p_i \)'s & \( q_i \)'s are primes.
    - Represent \( n+1 = p_1 p_2 \ldots p_k q_1 q_2 \ldots q_l \).
Induction and Recursion

- Induction is useful for proving correctness/design of recursive algorithms

Example

```java
// Returns base ^ exponent.
// Precondition: exponent >= 0
public int pow(int x, int n) {
    if (n == 0) {
        // base case; any number to 0th power is 1
        return 1;
    } else {
        // recursive case: x^n = x * x^(n-1)
        return x * pow(x, n-1);
    }
}
```
public int pow(int x, int n) {
    if (n == 0){
        return 1;
    } else {
        return x * pow(x, n-1);
    }
}

Claim: the algorithm correctly computes $x^n$.
Proof: By induction on $n$
Base case: $n = 0$: it correctly returns 1
Inductive step: assume that for $n$ the algorithm correctly returns $x^n$.
Then for $n+1$ it returns $x \cdot x^n = x^{n+1}$.
Induction and Recursion

- n! of some integer n can be characterized as:
  \[ n! = 1 \text{ for } n = 0; \text{ otherwise} \]
  \[ n! = n \times (n - 1) \times (n - 2) \times \ldots \times 2 \times 1 \]
- Want to write a recursive method for computing it. By definition: \( n! = n \times (n - 1)! \)
- This is all we need to put together the method:

```java
public int factorial(int n) {
    if (n == 0) {
        return 1;
    } else {
        return n * factorial(n-1);
    }
}
```
Induction in CS

- Induction is a powerful tool for showing algorithm correctness – not just for recursive algorithms (CS320)

- Loop invariant thinking is inductive thinking
  base case: invariant true before loop
  step: invariant kept true by loop body
  conclusion: invariant true after loop

  combine that with: test is false after loop
Induction vs recursion

- Let’s do some of the examples…