# Week 5 Part B - Tuesday lecture highlights

- Distinguish between and use as appropriate each of structural induction, mathematical induction, and strong induction
- Prove correctness of iterative and recursive algorithms using induction

**Recall: Proof by Strong Induction** (Rosen 5.2 p337, zybooks 8.1)
To prove that a universal quantification over the set of all integers greater than or equal to some base integer $b$ holds, pick a fixed non-negative integer $j$ and then:

| | |
|---|---|
| Basis Step: | Show the statement holds for $b$, $b+1$, $\ldots$, $b+j$. |
| Recursive Step: | Consider an arbitrary integer $n$ greater than or equal to $b+j$, assume (as the **strong induction hypothesis**) that the property holds for **each of** $b$, $b+1$, $\ldots$, $n$, and use this and other facts to prove that the property holds for $n+1$. |

For which non-negative integers $n$ can we make change for $n$ with coins of value 5 cents and 3 cents?

Restating: We can make change for __________, we cannot make change for _______________, and

______________________________________ $\star$

**Proof of $\star$ by mathematical induction** ($b = 8$)

**Basis step**: WTS property is true about 8

**Inductive step**: Consider an arbitrary $n \geq 8$. Assume (as the IH) that there are nonnegative integers $x, y$ such that $n = 5x + 3y$. WTS that there are nonnegative integers $x', y'$ such that $n + 1 = 5x' + 3y'$. We consider two cases, depending on whether any 5 cent coins are used for $n$.

*Case 1*: Assume .
Define $x' =$
and $y' =$
(both in $\mathbb{N}$ by case assumption).
Calculating:

$$5x' + 3y' \overset{\text{by def}}{=}$$
$$\overset{\text{rearranging}}{=}$$
$$\overset{\text{IH}}{=}$$

*Case 2*: Assume .
Therefore $n = 3y$ and $n \geq 8$, by case assumption. Therefore, $y \geq 3$ Define $x' = 2$ and $y' = y - 3$ (both in $\mathbb{N}$ by case assumption). Calculating:

$$\begin{aligned} 5x' + 3y' &\overset{\text{by def}}{=} 5(2) + 3(y-3) = 10 + 3y - 9 \\ &\overset{\text{rearranging}}{=} 3y + 10 - 9 \\ &\overset{\text{IH and case}}{=} n + 10 - 9 = n + 1 \end{aligned}$$

**Proof of $\star$ by strong induction** ($b = 8$ and $j = 2$)

**Basis step**: WTS property is true about $8, 9, 10$

**Inductive step**: Consider an arbitrary $n \geq 10$. Assume (as the IH) that the property is true about each of $8, 9, 10, \ldots, n$. WTS that there are nonnegative integers $x', y'$ such that $n + 1 = 5x' + 3y'$.

## Algorithms for making change

Change making (greedy) algorithm in pseudocode

```
procedure change(c_1, c_2, ..., c_r : values of denominations of coins, where c_1 > c_2 > ... > c_r ; n : a positive integer)

for i := 1 to r
  d_i := 0 {d_i counts the number of coin of denomination c_i used}
  while n ≥ c_i
    d_i := d_i + 1 {Add a coin of denomination c_i}
    n := n − c_i

return d_1, d_2, ..., d_r {d_i the number of coins of denomination c_i in the change for i=1, 2, ..., r}
```

The greedy approach doesn't work with 5¢ and 3¢ coins even for large values of n. However, we can write two new algorithms inspired by the proofs that we completed using mathematical induction and strong induction.

## Recursive algorithms for making change

One recursive algo for making change using 5¢ and 3¢ coins

```
procedure change1(n : a positive integer)

















return (d_1, d_2) {d_1 , d_2 are the number of 5¢ and 3¢ coins respectively }
```

Another recursive algo for making change using 5¢ and 3¢ coins

```
procedure change2(n : a positive integer)















return (d_1, d_2) {d_1 , d_2 are the number of 5¢ and 3¢ coins respectively }
```

**Proving correctness of algorithms** What does it take to show that some algorithm is correct?

## Example 1: Prove that the algorithm findMax is correct

$findMax$ takes an input a sequence of numbers and returns the maximum number in the sequence

```
procedure findMax(a_1, a_2, ..., a_n: a sequence of n integers)













return
```

**Prove that the algorithm findMax is correct (contd)**

## Example 2: Prove the correctness of the Division Algorithm

Division Algorithm

```
procedure DivisionAlgo(n: positive integer; d: positive integer)












return
```

**Example 2: Prove the correctness of the Division Algorithm (contd)**