Poll: Weak vs. Strong Induction

What is the difference between weak induction and strong induction?

A) Weak induction is less complex, but also less powerful: some claims can be proven using strong induction that cannot be proven using weak induction.

B) Weak induction can be used to prove statements parameterized by an integer (e.g., $\forall n \in \mathbb{Z}^{\geq 0} P(n)$), whereas strong induction can be used to prove statements about non-integer things like data structures (lists, trees, etc.).

C) In the inductive step of the proof, weak induction requires making more assumptions (hence it is weaker).

D) None of the above

COSC 290 Discrete Structures

Lecture 14: Strong induction

Prof. Michael Hay

Wednesday, Feb. 28, 2018

Colgate University

Plan for today

1. Weak induction example: analyzing algorithms
2. Strong induction
3. Examples of strong induction

Weak induction example: analyzing algorithms

An algorithm for making change

Input: An integer $n \geq 8$
Output: Integers k, ℓ such that $n = 5k + 3\ell$

```
1:  Let m = 8 and k = 1 and ℓ = 1                        ▷ Thus, m = 5k + 3ℓ
2:  while m < n do
3:                                   ▷ Invariant: before loop body m = 5k + 3ℓ
4:      m = m + 1
5:      if k ≥ 1 then
6:          k = k − 1 and ℓ = ℓ + 2
7:      else
8:          k = k + 2 and ℓ = ℓ − 3
9:                                   ▷ Invariant: after loop body m = 5k + 3ℓ
10: return k, ℓ
```

Our proof implies that $k \geq 0$ and $\ell \geq 0$ throughout this algorithm. This is not obvious looking at the code.

Showing algorithm correctness

Define predicate $P(t)$ as follows: at the start of the t^{th} iteration (i.e., when the while condition is checked for the t^{th} time), we have $m = 5k + 3\ell$.

Claim: $\forall t \in \mathbb{Z}^{\geq 1} : P(t)$.

How does this help prove algorithm correctness?

Observe that m gets incremented by 1 each time through the loop and the algorithm breaks out of the loop when $m = n$.

Thus, if the above claim holds, then when this algorithm terminates, $m = 5k + 3\ell$ and $m = n$ so the algorithm returns the correct answer.

Inductive proof showing algorithm correctness

- **Base case:** Show the claim holds for $t = 1$ (i.e., line 2 is being executed for the first time)
  - $k = 1$ and $\ell = 1$ and $m = 8$, so $m = 5k + 3\ell$ holds.
- **Inductive case:** Let t be any integer in $\mathbb{Z}^{\geq 1}$
  - **Assume:** Assume $P(t)$.
  - **Want to show:** $P(t+1)$.
  - $P(t)$ being true means at start of t^{th} iteration, $m = 5k + 3\ell$.
  - On line 4, m increases by 1.
  - Cases:
    - $k \geq 1$: Then k decreases by 1 and ℓ increases by 2, leading to a net change of $-(5 \times 1) + (3 \times 2) = +1$.
    - $k = 0$: Then k increases by 2 and ℓ decreases by 3, leading to a net change of $(5 \times 2) - (3 \times 3) = +1$
  - Either way, lines 5-8 adjust k and ℓ so that $5k + 3\ell$ increases by 1, matching increase in m.
  - Thus, $m = 5k + 3\ell$ at *end* of t^{th} iteration (and therefore also at the start of the $(t+1)^{th}$ iteration).

Minor details

Technically, our claim

$$\forall t \in \mathbb{Z}^{\geq 1} : P(t)$$

is a little too simple. Why? Eventually the algorithm *terminates*, so $P(t)$ does not hold for *all* t.

We can modify $P(t)$ to be "if the algorithm executes for a t^{th} iteration, then …" and adjust the proof accordingly.

Exercise

Prove that the algorithm uses at most two nickels. In other words, for any input n, the algorithm never has $k > 2$ or $k < 0$ at any point during the execution of the algorithm. (Hint: do induction on the number of times thru the loop.)

Input: An integer $n \geq 8$

Output: Integers k, ℓ such that $n = 5k + 3\ell$

```
Let m = 8 and k = 1 and ℓ = 1          ▷ Thus, m = 5k + 3ℓ
while m < n do
    m = m + 1
    if k ≥ 1 then
        k = k − 1 and ℓ = ℓ + 2
    else
        k = k + 2 and ℓ = ℓ − 3
return k, ℓ
```


Proof of claim that $k \in \{0, 1, 2\}$

Proof by induction: We will use proof by induction on the number of times through the loop (i.e., the number of times while condition is checked).

- **Base case (first iteration):**
  - Before the loop, $k = 1 \in \{0, 1, 2\}$.
- **Inductive case (t iterations):**
  - **Assume:** We will assume that at start of t^{th} iteration (while condition checked for the t^{th} time), we have $k \in \{0, 1, 2\}$.
  - **Want to show:** We will show $k \in \{0, 1, 2\}$ at the start of the $(t+1)^{th}$ iteration.
  - Cases:
    1. $k \in \{1, 2\}$, then $k = k - 1$, so now $k \in \{0, 1\}$.
    2. $k = 0$, then $k = k + 2$, so k is now 2.
  - Therefore at the end of the t^{th} iteration, $k \in \{0, 1, 2\}$. This means at the start of $(t+1)^{th}$ iteration, $k \in \{0, 1, 2\}$.
- **Conclusion:** By induction, the claim has been shown.

Strong induction

Weak vs. Strong Induction

Claim: $\forall n \in \mathbb{Z}^{\geq 0} : P(n)$.

Proof by induction:

Base case: prove $P(0)$ is true.

Inductive case:

- Weak induction:

$$\forall n \in \mathbb{Z}^{\geq 1} : P(n-1) \implies P(n)$$

- Strong induction:

$$\forall n \in \mathbb{Z}^{\geq 1} : (P(0) \wedge P(1) \wedge \cdots \wedge P(n-1)) \implies P(n)$$

Weak vs. strong

- Weak induction:

$$\forall n \in \mathbb{Z}^{\geq 1} : P(n-1) \implies P(n)$$

- Strong induction:

$$\forall n \in \mathbb{Z}^{\geq 1} : (P(0) \land P(1) \land \cdots \land P(n-1)) \implies P(n)$$

With strong, you get to assume more. Assume $P(0)$ is true, $P(1)$ is true, ..., and $P(n-1)$ is true.

Be careful! Sometimes requires writing *more* base cases. Why? For example, if your inductive case references $P(n-12)$, then it doesn't apply for proving $P(n)$ when $n < 12$!

Weak and strong are formally equivalent: anything you can prove with weak you can prove with strong and vice versa.

(Flawed) Example: three-cent coins redux

Claim: For any price $n \geq 8$, the price n can be paid using only 5-cent coins and 3-cent coins.

Proof by strong induction:

- **Base case:** For $n = 8$, we can pay with one three-cent coin and one five-cent coin.
- **Inductive case:** Assume claim is true for any m such that $8 \leq m \leq n-1$, show it is true for n.
  Since it's true for $P(n-3)$, we can simply add one more three-cent coin to pay price n.

Wait, what?

Poll: error in three-cent coins proof

- **Base case:** For $n = 8$, we can pay with one three-cent coin and one five-cent coin.
- **Inductive case:** Assume claim is true for any m such that $8 \leq m \leq n-1$, show it is true for n.
  Since it's true for $P(n-3)$, we can simply add one more three-cent coin to pay price n.

Where does this proof go wrong? (Be able to explain your answer)

A) Base case is incorrect.
B) Inductive case is incorrect.
C) This claim can be proven with (weak) induction but not strong induction.
D) There's nothing wrong with this proof.
E) None / More than one of above

Proof for three-cent coins

Claim: For any price $n \geq 8$, the price n can be paid using only 5-cent coins and 3-cent coins.

Proof by strong induction:

- **Base cases:**
  - For $n = 8$, we can pay with 1 three-cent coin and 1 five-cent coin.
  - For $n = 9$, we can pay with 3 three-cent coin and 0 five-cent coins.
  - For $n = 10$, we can pay with 0 three-cent coins and 2 five-cent coins.
- **Inductive case:** Assume claim is true for any m such that $11 \leq m \leq n-1$, show it is true for n.
  Since it's true for $P(n-3)$, we can simply add one more three-cent coin to pay price n.

Strong induction

With strong induction, how many base cases should you have?

It depends on the problem, but...

What **too few** looks like: Check the argument for the inductive case for the *smallest* n that is *not* a base case. If the proof doesn't quite work for this n, then you may have too few base cases.

What **too many** looks like: Review your proof. Does the argument for a base case resemble the argument you make in the inductive case? Avoid being repetitive.

Examples of strong induction

Jacobsthal numbers & Tilings

Jacobsthal numbers are defined as follows:

- $J_0 := 0$
- $J_1 := 1$
- $J_n := J_{n-1} + 2J_{n-2}$ for $n \geq 2$

We have two separate claims about Jacobsthal numbers.

1. **Claim**: for any $n \geq 0$, given $n \times 2$ grid, the number of tilings using either 1×2 dominoes or 2×2 squares is J_{n+1}.
2. **Claim**: $J_n = \frac{2^n - (-1)^n}{3}$

Working in small groups, prove each of these using strong induction.

(Feel free to draw pictures as part of your proof for 1.)