COSC 290 Discrete Structures

Lecture 16: Proof by strong induction

Prof. Michael Hay
Wednesday, Oct. 4, 2017
Colgate University

Plan for today

1. Binary Search (wrap up exercise)
2. Strong induction
3. Exercises

Binary Search (wrap up exercise)

Binary Search

procedure BinarySearch2(A, n, x) ▷ Find x in *sorted* array A
  Set l to 0 and u to $n-1$.
  while True **do**
    if $l > u$ **then**
      Set i to -1 and break.
    Set m to $\lfloor (l+u)/2 \rfloor$. ▷ integer division
    if $x < A[m]$ **then**
      Set u to $m-1$.
    else if $x > A[m]$ **then**
      Set l to $m+1$.
    else ▷ It must be that $A[m] = x$
      Set i to m and $l = u = m$ and break.
  return i

Claims

- Claim 1: If the algorithm terminates, it returns the correct answer.
- Claim 2: The algorithm eventually terminates.

Supporting lemma

Lemma

The following invariant holds throughout the execution of the algorithm: If x is in array A, then the set $\{l, \ldots, u\}$ contains the index where x is located. (If $l > u$, this is an empty set and thus the set does not contain any index.)

Prove using induction on t, number of iterations through the while loop.

Proof

- **Proof by induction**: Induction on t, number of iterations through the while loop.
  - **Base case** ($t = 0$)**:** Initially, $l = 0$ and $u = n - 1$ and by definition MustBe(l, u) is true.
  - **Inductive case** ($t \geq 0$)**:** Assume it's true after iteration t.
    - **Assume:** By inductive hypothesis, MustBe(l, u) is true at start of loop in iteration $t + 1$.
    - **Want to show:** MustBe(l, u) is true at end of while loop.
    - Shown using proof by cases (next slide)

Cases

Given: MustBe(l, u) is true at start of iteration.

- $l > u$: Then algorithm breaks out of the loop without changing l or u. Since inductive hypothesis tells us MustBe(l, u) is true at the start of this iteration, it is true after as well.
- $x < A[m]$: Then x cannot be in $\{m \ldots u\}$ because it's less than $A[m]$ and the array is sorted. Since MustBe(l, u) is true and it cannot be in $\{m \ldots u\}$, then MustBe$(l, m - 1)$. By setting $u = m - 1$, MustBe(l, u) still holds.
- $x > A[m]$: similar reasoning as previous case.
- $A[m] = x$: then x is in the array at index m and thus, MustBe(m, m). By setting $l = u = m$, MustBe(l, u) is still true.

Revisit Claims

- Claim 1: If the algorithm terminates, it returns the correct answer.
- Claim 2: The algorithm eventually terminates.

Proof sketch of Claim 1

Claim 1: If the algorithm terminates, it returns the correct answer.

Proof sketch: Assume algorithm terminates. What does this tell us? The only way to terminate is to reach a break statement.

Since Lemma 1 is true, then the correct value is returned in both cases:

- it returns -1 when set is empty.
- it returns m when the set contains a single element $\{m\}$.

Proof sketch of Claim 2

Claim 2: The algorithm eventually terminates.

Proof sketch: Using lemma, we know that MustBe(l, u) is true in each iteration.

Use induction to show that the range $\{l, \ldots, u\}$ shrinks by at least 1 in each iteration and when its size is 0 or 1, it terminates.

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:

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

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.

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: three-cent coins redux

- **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.

Exercises

Tilings

Jacobsthal numbers: $J_0 = 0$, $J_1 = 1$ and $J_n = J_{n-1} + 2J_{n-2}$ for $n \geq 2$.

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: $J_n = \frac{2^n - (-1)^n}{3}$