CSE 390Z: Mathematics of Computing

Week 7 Workshop

0. Strong Induction: Stamp Collection

A store sells 3 cent and 5 cent stamps. Use strong induction to prove that you can make exactly n cents worth of stamps for all $n \geq 10$.

Hint: you'll need multiple base cases for this - think about how many steps back you need to go for your inductive step.

1. Strong Induction: Functions

Consider the function $f(n)$ defined for integers $n \geq 1$ as follows:
$f(1) = 1$ for $n = 1$
$f(2) = 4$ for $n = 2$
$f(3) = 9$ for $n = 3$
$f(n) = f(n-1) - f(n-2) + f(n-3) + 2(2n-3)$ for $n \geq 4$

Prove by strong induction that for all $n \geq 1$, $f(n) = n^2$.

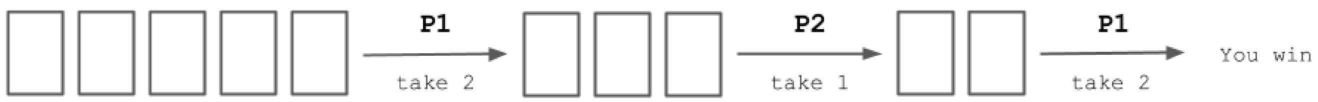
2. Strong Induction: Collecting Candy

A store sells candy in packs of 4 and packs of 7. Let P(n) be defined as "You are able to buy n packs of candy". For example, $P(3)$ is not true, because you cannot buy exactly 3 packs of candy from the store. However, it turns out that P(n) is true for any $n \geq 18$. Use strong induction on n to prove this.

Hint: you'll need multiple base cases for this - think about how many steps back you need to go for your inductive step.

3. Strong Induction: Cards on the Table

I've come up with a new card game that is played between 2 players as follows. We start with some integer $n \geq 1$ cards on the table. The two players then take turns removing cards from the table; in a single turn, a player can choose to remove either 1 or 2 cards from the table. A player wins by taking the last card. For example:



The person I've been playing with has been *very* careful about dealing the cards, and keeps winning; I think they know something I don't about this game. I want to use induction to prove that if $3|n$, the second player (P2) can guarantee a win, and if n is not divisible by 3, the first player (P1) can guarantee a win.

(a) How many base cases does this proof need? What should they be?

(b) Use strong induction to prove that if $3|n$, P2 can guarantee a win, and if n is not divisible by 3, P1 can guarantee a win.

4. How does mod work again?

In 311, you learned the mathematical definition of mod. Here is one recursive implementation of mod:

```
// returns the result of a mod m
public static int modr(int a, int m) {
    if (a < m) {
        return a;
    } else {
        return modr(a - m, m);
    }
}
```

Use strong induction to prove that given an arbitrary positive integer m, my `modr` method correctly returns the result of a mod m for any non-negative integer a (i.e. $a \in \mathbb{Z}$, $a \geq 0$).

5. Recursively Defined Sets

Write a recursive definition of a set of integers satisfying the given properties.

(a) All integers x satisfying $x \equiv 2 \pmod{5}$.

(b) The set of coordinates where the x-value is an index, and the y-value is the Fibonacci number at that index. I.e. the coordinates $(0,0) \in S$, $(1,1) \in S$, $(2,1) \in S$, $(3,2) \in S$, etc.

6. Recursively Defined Sets Counterexamples

Find a counterexample for each of the following claims.

(a) Every recursively defined set of integers has infinitely many elements.

(b) If R is a recursively defined set of integers with infinitely many elements, then there is some integer m such that every integer greater than m is in R.