Problem Set 3

Due: March 5

Reading: Notes for Week4

Problem 1. Let m, n be integers, not both zero. Define a set of integers, $L_{m,n}$, recursively as follows:

- **Base cases**: $m, n \in L_{m,n}$.
- **Constructor steps**: If $j, k \in L_{m,n}$, then
  1. $-j \in L_{m,n}$,
  2. $j + k \in L_{m,n}$.

Let L be an abbreviation for $L_{m,n}$ in the rest of this problem.

(a) Show *by structural induction* that

$$L \subseteq \{mx + ny \mid x, y \in \mathbb{Z}\}.$$

(b) Show that

$$\{mx + ny \mid x, y \in \mathbb{Z}\} \subseteq L.$$

(c) Conclude that any common divisor of m and n also divides every member of L.

(d) Show that if $j, k \in L$ and $k \neq 0$, then the remainder of j divided by k is in L.

(e) Show that there is a positive integer $g \in L$ which divides every member of L. *Hint:* The least positive integer in L.

(f) Conclude that $\gcd(m, n) = mx + ny$ for some $x, y \in \mathbb{Z}$.

Problem 2. We define full binary trees, FBT's, as a tagged recursive data type with tags from some set of "labels."

Definition. Base case: $\langle l, \texttt{leaf}\rangle$ is an FBT, where l is a label.

Constructor case: If B_1 and B_2 are FBT's, then $\langle l,\ B_1,\ B_2\rangle$ is an FBT, where l is a label.

The labels and leaf labels *appearing* in an FBT, B, are defined recursively in the obvious way:

Definition. Base case: If $B = \langle l, \texttt{leaf}\rangle$. Then l *appears* in B and is a *leaf label* of B.

Constructor case: If $B = \langle l, B_1,\ B_2\rangle$ is an FBT, then the labels that *appear* in B are the ones that appear in B_1, or in B_2; also, l *appears* in B. The *leaf labels* of B are the union of the leaf labels of B_1 and the leaf labels of B_2.

The FBT's with *unique* labels are also defined recursively:

Definition. Base case: If $B = \langle l, \texttt{leaf}\rangle$. Then B has *unique labels*.

Constructor case: If $B = \langle l, B_1,\ B_2\rangle$ is an FBT, then B has *unique labels* iff l does not appear in B_1 or B_2, and no other label appears in both B_1 and B_2.

If B is an FBT, let n_B be the number of labels appearing in B and f_B be the number of leaf labels of B. Use structural induction to prove that

$$f_B = \frac{n_B + 1}{2} \tag{1}$$

for all FBT's *with unique labels*. Also give a counterexample for an FBT that does not have unique labels. So your proof had better use uniqueness of labels at some point; be sure to indicate where.

Problem 3. *Search Trees*, ST, are a special case of the FBT's of Problem 2 with labels that are numbers. They are defined recursively as follows:

Base case: $\langle x, \texttt{leaf}\rangle$ is an ST for any real number, x.

Constructor case: If T_1 and T_2 are ST's, and x is a real number such that every label that appears in T_1 is smaller than x, and every label that appears in T_2 is larger than x, then

$$\langle x, T_1, T_2\rangle$$

is an ST.

(a) Draw all the ST's with labels 1, 2, 3, 4, 5.

(b) Define a function appears-in : $(\mathbb{R} \times \text{ST}) \to \{0, 1\}$ recursively on the definition of ST's as follows:

Base case:

$$\begin{aligned} &\text{appears-in}(x, \langle x, \texttt{leaf}\rangle) ::= 1; \\ &\text{appears-in}(x, \langle y, \texttt{leaf}\rangle) ::= 0, \qquad\qquad \text{if } x \neq y. \end{aligned}$$

Constructor case:

$$\text{appears-in}(x, \langle x, T_1, T_2\rangle) ::= 1; \tag{2}$$

$$\text{appears-in}(x, \langle y, T_1, T_2\rangle) ::= \text{appears-in}(x, T_1) \qquad \text{if } x < y; \tag{3}$$

$$\text{appears-in}(x, \langle y, T_1, T_2\rangle) ::= \text{appears-in}(x, T_2) \qquad \text{if } x > y. \tag{4}$$

Prove by structural induction on the definition of ST that

$$\text{appears-in}(x, T) = 1 \quad \text{iff} \quad x \text{ appears in } T. \tag{5}$$

Problem 4. The concatenation st of strings s and t is the string beginning with the symbols in s followed by the symbols in t. Notes 4 gave a recursive definition of concatenation based on the the definition of strings as a tagged recursive data type. As a way to ensure the somewhat technical recursive definition works as intended, we will verify that the recursive definition implies some of the basic properties we expect. So in what follows, you should *not* assume that the recursive definition of concatenation matches the "string followed by string" formulation, because that's what we're trying to check. But you do have a powerful principle for proving properties of recursive definitions, namely structural induction.

(a) By the base case of the recursive definition of concatenation in Notes 4, concatenating a string with the empty string, λ, *on the right* leaves the string unchanged, that is, $s\lambda = s$. From the "string followed by string" formulation, it's clear that concatenating *on the left* with the empty string also leaves a string unchanged. That is,

$$\lambda s = s \tag{6}$$

for all strings, s. Use structural induction to prove that the recursively defined concatenation operation satisfies (6).

(b) It's also clear from the "string followed by string" definition of concatenation that it is *associative*. That is,

$$(pq)s = p(qs) \tag{7}$$

for all $p, q, s \in A^*$. We should verify this for the recursive definition of concatenation as well. Do so using structural induction.

Problem 5. Define 2-person terminating *value* games of perfect information, VG's, just like 2-person terminating games of perfect information (2PTG's) in Notes 4, except that there is only one

Base case: $\langle \texttt{value}, r \rangle$ is a VG where r is any real number.

If the play of a VG ends at the game $\langle \texttt{value}, r \rangle$, then r is called the value of the play. Now the objective of one player (call him the *max*-player) is to have play end with as high a value as possible, and the other player (called the *min*-player) aims to have play end with as low a value as possible.

Given which of the players moves first, a strategy for the max-player is said to *ensure* the value, r, if play ends with a value of at least r, no matter what moves the min-player makes. Likewise, a strategy for the min-player is said to *hold down* the value to r, if play ends with a value of at most r, no matter what moves the max-player makes.

A VG is said to have *final max value*, r, if the max-player has a strategy that ensures r, and the min-player has a strategy that holds down the value to r, when the *max-player moves first*. Likewise, the VG has *final min value*, r, if the max-player has a strategy that ensures r, and the min-player has a strategy that holds down the value to r, when the *min-player moves first*.

A Fundamental Theorem for *finite* VG's is that every finite VG has a both a final max value and a final min value. (Note: the two values are usually different.)

(a) Prove this Fundamental Theorem for finite VG's by structural induction on the definition of VG's.

Note that this Fundamental Theorem may not hold exactly for infinite games. For example, suppose the max player moves first and his only moves are those that terminate with a real number less than 1. Then he can ensure a value as close to 1 as he wants, but he can't ensure 1.

(b) See if you can formulate a good generalization of the Fundamental Theorem that applies to all VG's. You need not prove it.

Problem 6. In the late 1960s, the military junta that ousted the government of the small republic of Nerdia outlawed multiplication and division by any number other than 3. Fortunately, a young dissident found a way to help the population multiply any two nonnegative integers without risking persecution by the junta. The procedure he taught people appears on the next page:

procedure *multiply*(x, y: nonnegative integers)
$r := x$;
$s := y$;
$a := 0$;
do until $s = 0$
    if $3 \mid s$ **then**
        $r := 3r$;
        $s := s/3$;
    else if $3 \mid (s-1)$ **then**
        $a := a + r$;
        $r := 3r$;
        $s := (s-1)/3$;
    else
        $a := a + 2r$;
        $r := 3r$;
        $s := (s-2)/3$;
return a;

We can model the algorithm as a state machine whose states are triples of nonnegative integers (r, s, a). The initial state is $(x, y, 0)$. The transitions are given by the rule that for $s > 0$:

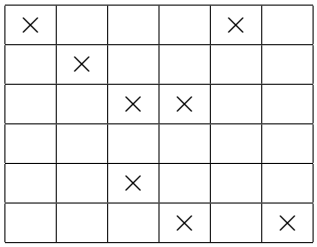
$$(r,s,a) \to \begin{cases} (3r, s/3, a) & \text{if } 3 \mid s \\ (3r, (s-1)/3, a+r) & \text{if } 3 \mid (s-1) \\ (3r, (s-2)/3, a+2r) & \text{otherwise.} \end{cases}$$

(a) List the sequence of steps that appears in the execution of the algorithm for inputs $x = 5$ and $y = 10$.

(b) Use the invariant method to prove that the algorithm is partially correct —that is, if $s = 0$, then $a = xy$.

(c) Prove that the algorithm terminates after at most $1 + \log_3 y$ executions of the body of the `do` statement.

Problem 7. In a MIT large conference room, n^2 students are sitting in a grid pattern of n rows of length n. A sudden outbreak of beaver flu (a rare variant of bird flu that lasts forever; symptoms include yearning for problem sets and loving mini-quizzes) causes some students to get infected. Here is an example where $n = 6$ and infected students are marked $\times$.

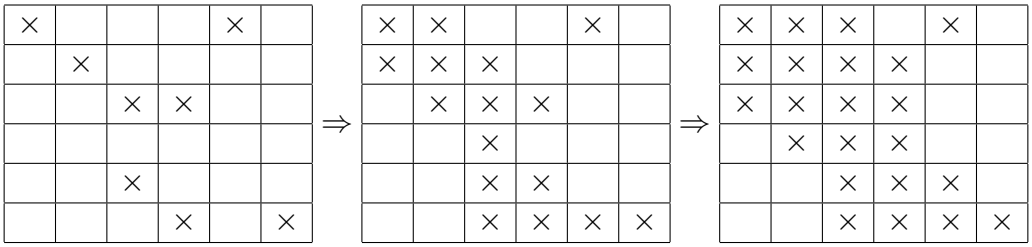


Two students are considered *adjacent* if they share an edge (that is, front, back, left or right, but NOT diagonal). So each student is adjacent to 2, 3 or 4 others, depending on whether the student is at an edge of the grid. An uninfected student can become infected iff

- the student is adjacent to *at least two* already-infected students.

Since beaver flu lasts forever, an infected student always remains infected.

In the example, the infection might spread as shown below.



Here we infected as many students as possible at each step, but the infection might also spread as slowly as one student at a time. In this example the infection can spread to all the students.

(a) It turns out that if fewer than n students in class are initially infected, the infection cannot spread to the whole class. Prove it!

Hint: There is a simple infection-process derived variable that is weakly decreasing, and the value of this variable when fewer than n students are infected is strictly smaller than its value when all students are infected. If you are stuck defining this variable, ask your recitation instructor for the one-word clue that makes it easy.

(b) What is the smallest number of infected students that could spread infection to the entire class?

(c) (optional, may be hard) What is the *largest* number of students who can be infected without the infection being able to spread to everyone? Briefly explain.

Student's Solutions to Problem Set 3

Your name:

Due date: March 5

Submission date:

Circle your TA/LA: Chiyoun Jay Jeffrey Jessica Tina

Collaboration statement: Circle one of the two choices and provide all pertinent info.

1. I worked alone and only with course materials.
2. I collaborated on this assignment with:

   got help from:[1]

   and referred to:[2]

DO NOT WRITE BELOW THIS LINE

Problem	Score
1	
2	
3	
4	
5	
6	
7	
Total	

[1]People other than course staff.
[2]Give citations to texts and material other than the Spring '07 course materials.