Assignment 1: Loop Invariants

CSDS 310: Algorithms

1

The greatest common divisor (GCD) of two integers a and b is defined as the largest integer that can divide both a and b without a remainder. For example, the GCD of 30 and 54 is 6, whereas the GCD of 7 and 5 is 1. The following procedure was developed by Euclid to compute the greatest common divisor of two positive integers a and b. In this exercise, we will prove the correctness of this algorithm.

```
procedure EUCLIDEAN(a,b)
        x ← a
        y ← b
        while x ≠ y do
                if x > y then
                        x ← x - y
                else
                        y ← y - x
        return x
```

a

State the loop invariant for the while loop in this procedure.

✓ **Answer**

For any integers a and b,
Both pairs of integers $(a-b, b)$ and $(a, b-a)$ will have the same GCD as (a, b)

b

Prove the loop invariant.

✓ **Answer**

$\forall a, b \in \mathbb{Z}$,
$g = \mathrm{GCD}(a, b)$

By definition of GCD, $g|a$ and $g|b$
$g|a \implies \exists n_a \in \mathbb{Z} : gn_a = a$
$g|b \implies \exists n_b \in \mathbb{Z} : gn_b = b$
$a - b = g(n_a - n_b) \implies g|a - b$
$b - a = g(n_b - n_a) \implies g|b - a$

By definition of GCD, $\mathrm{GCD}(n_a, n_b) := 1$
If not, $g \neq \mathrm{GCD}(a, b)$

$\mathrm{GCD}(n_a - n_b, n_b) = 1 \implies \mathrm{GCD}(a - b, b) = g$
If not, $\mathrm{GCD}(n_a, n_b) \neq 1$

$\mathrm{GCD}(n_a, n_b - n_a) = 1 \implies \mathrm{GCD}(a, b - a) = g$
If not, $\mathrm{GCD}(n_a, n_b) \neq 1$

$\therefore \mathrm{GCD}(a, b) = \mathrm{GCD}(a - b, b) = \mathrm{GCD}(a, b - a)$

c

Prove that procedure $\mathrm{EUCLIDIAN}$ always terminates provided that a and b are positive integers.

✓ **Answer**

$\forall a, b \in \mathbb{N}_+$
If $a = b$
The procedure terminates

If $a > b$
$0 < a - b < a$
$a - b$ is a valid new a
$0 < \mathrm{MAX}(a - b, b) < \mathrm{MAX}(a, b)$

If $b > a$
$0 < b - a < b$
$b - a$ is a valid new b
$0 < \mathrm{MAX}(a, b - a) < \mathrm{MAX}(a, b)$

After each iteration, the MAX of the new variables is guaranteed to decrease while remaining above 0.
This indicates the maximum number of loops remaining is equivalent to the maximum of the two values, as the maximum decreases of a minimum of 1 each iteration upon our valid set of inputs.

In the ultimate worst case scenario, both a and b will end at 1, as 0 will never be achieved, thus terminating our loop with $a = b = 1$

d

Using the termination property of your loop invariant, prove that procedure Euclidean computes and returns the greatest common divisor of a and b.

✓ **Answer**

Our loop invariant proves that whatever values a and b are updated to through this algorithm, their GCD is equivalent to the GCD of the original inputs.

We also have proved that our procedure will approach $a = b$ in finite time through our termination proof.

Since our updated values of a and b will share the GCD as our original inputs, a and b will approach $a = b$ and $a = b; \mathrm{GCD}(a, b) = a = b$
Our loop terminates with the GCD of the original inputs in both a, b

2

Let A and B be two arrays, each consisting of n numbers sorted in increasing order. We are also given a number x and would like to find and return i and j such that $A[i] + B[j] = x$. If no such pair exists, we would like to return FALSE. We would like to develop an algorithm to solve this problem in linear time.

a

Using pseudo-code, describe an algorithm that correctly solves this problem in linear time.

✓ **Answer**

```
procedure findSum(A[n],B[n],x)
        i ← 1
        j ← n
        s ← A[i] + B[j]
        while s ≠ x do
                if s > x then
                        j ← j - 1
                        if j = 0
                                return false
                else
```

```
                    i ← i + 1
                    if i = n + 1
                            return false
            s ← A[i] + B[j]
    return (i, j)
```

b

Using loop invariants, prove that your algorithm is correct.

✓ **Answer**

Loop Invariant:
With i, j being the solution
$i_{current} \leq i; j_{current} \geq j$

Initialization:
We initialise i to be the smallest valid solution for i : 1
We initialise j to be the largest valid solution for j : n

Maintenance:
If our sum is larger than our target, then this implies that our index of j is too large, as the only remaining possibilities for i will increase our sum.
This shows that there are no valid combinations with $B[j]$ as they will all be too large for all remaining valid i, thus we decrement j

If our sum is smaller than our target, then this implies that our index of i is too small, as the only remaining possibilities for j will decrease our sum.
This show that there are no valid combinations with $A[i]$ as they will all be too small for all remaining valid j, thus we decrement i

Termination:
The procedure defined above is guaranteed to remove all non-solutions from the pool of valid solutions $(i_{current} \leq i; j_{current} \geq j)$. If no solutions exist, all possible combinations will be removed from the pool and false will be returned $(i > n \text{ or } j < 1)$. If a solution exists, we are guaranteed to not eliminate it, and thus return it.

3

Two species are sharing planet Kepler-442b: Pisidians and Lydians. Members of either species roam individually. They can run into each other only in pairs (randomly) and whenever any two individuals run into each other, they fight. Since Pisidians are stronger and Lydians have magical powers, the outcome of a fight is always the same:

- If two Lydians fight, one of the Lydians dies and the other survives.
- If two Pisidians fight, both Pisidians die and a new Lydian comes to existence.
- If a Pisidian and a Lydian fight, the Lydian dies and the Pisidian survives.

At the beginning of time, there were m Lydians and n Pisidians on Kepler-442b. Observing that there will be only one individual left on Kepler-442b at eternity, identify the species of the eternal individual as a function of m and/or n. (Hint: Describe the process using a loop by writing a pseudocode, define a loop invariant, and prove the loop invariant)

✓ **Answer**

```
procedure LydianWins(m, n)
        total ← m + n
        pisidians ← n
        for each _ ← total-1..1
                total ← total - 1
                // If both selected people are Pisidians
                if RANDOM() < pisidians / total && RANDOM() <
(pisidians - 1) / total
                        pisidians ← pisidians - 2
                if pisidians = 1 || total = pisidians
                        return false
                else if pisidians = 0
                        return true
        return ERROR
```

Loop Invariant:
The number of Pisidians $\mod 2$ remains the same across all loops, as it is only ever modified in pairs, as p and $p-2$ have the same $\mod 2$

Initialization:
We initialise our total to be the sum of both populations, and our Pisidian counter to the initial amount of Pisidians

Maintenance:
The only two cases possible in our loop will either not effect the number of Pisidians or decrease their populations by two

$p \mod 2 = p \mod 2$
$p-2 \mod 2 = p \mod 2$

Thus our Loop Invariant holds true no matter the case within our loop.

Termination:
Since under all cases, our total is always decreasing by 1, we are guaranteed to end

with one survivor after $t-1$ loops, where Pisidians will be 1 or 0. The number of each population will never be below 0 as $t > p$ and $p = 1$ or $p = 0$ will cause the loop to exit as well, as the simulation need not continue any longer.

If $p = 1$, it is impossible for a Lydian to win, as Pisidians only die in pairs, thus making a sole Pisidian immortal.

If $p = 0$, it is impossible for a Pisidian to win, as they are all dead.

Loop Invariant Proof:
Regardless of case, our total will always decrease by 1, this is a known fact of this scenario.

If both selected people are Pisidians, then their population will decrease by 2, however, we know that $p-2$ and p will equal under $\mod 2$, thus holding our loop invariant, and proving that this situation need not a simulation, as Pisidians will always win if they begin with an odd number, and always lose if they begin with an even number.