Problem Set 4 Solutions

Problem 1. [**40 points**]

In this problem you are asked to design a protocol for college admissions. The protocol must have the following properties:

There are N students $S_1, \ldots, S_N$ and M universities $U_1, \ldots, U_M$. Each student provides a preference ranking of all M universities and each university provides a preference ranking of all N students. For each i university U_i has n_i slots to fill and sum of all slots is equal to the number of students. That is, $\sum_{i=1}^{M} n_i = N$. The protocol should assign every student to a university so that

1. For each i University U_i gets n_i students,
2. There does not exist students S_i, S_j and universities U_k, U_l where S_i is assigned to U_k and S_j is assigned to U_l, but S_j prefers U_k to U_l, and U_k prefers S_j to S_i.
3. The assignment algorithm should be student optimal in the sense that every student gets their top choice among all assignments that satisfy conditions 1 and 2 above.

(a) [**8 points**] Model the problem as a state machine. State clearly what the states and state transitions are. What is the start state?

We define the following protocol.

1. Each student applies to their favorite university that he/she has not yet ruled out.
2. University i says "maybe" to the n_i applicants with the better ranking. (They say "maybe" to everyone if the have less that n_i applicants). They say "no way" to the rest.
3. Any students who is turned down crosses the university of his/her list.

The state is given by a vector of N arrays containing the crossed out universities for each student. The preference list for each student can be included as part of the state or can be considered as universal constants. The state transitions are described by the universities that are crossed off in any time steps. There is only one initial state with no names crossed off any list.

Termination condition: When no university has more than n_i applicants, each university accepts the remaining applicants.

(b) **[8 points]** Use derived variables to prove termination. How long does the protocol take to terminate?

Theorem 0.1 *The algorithm terminates in at most NM steps.*

Proof. Define the derived variable $f : Q \to \mathbb{N}$ given by

$f(q)$ = number of universities over all students' lists that are not crossed off.

f is decreasing because at every step before termination at least one name is crossed of the lists. Note that Initially no university is crossed off, so there are a total NM universities in the lists.

$$\text{execution time} \leq f(q_0) \leq NM.$$

■

(c) **[8 points]** Prove that every student is assigned and every university gets the right number of students.

Define the derived variables

$f_i(q)$ = number of universities crossed out of the list of student i.

$g_i(q)$ = number of students currently on-hold in university U_i.

$r_i(q)$ = rank of worst student to whom U_i says maybe.

Facts about f_i

1) Nondecreasing
2) non-skipping
3) $\forall i, f_i(q_0) = 0$
4) $f_i(q) = M$ if and only if student S_i applied to every university and did not get accepted anywhere

Facts about g_i

5) g_i is nondecreasing
6) U_i says no if and only if $g_i(q) > n_i$
7) If for some t, $g_i(q_t) \geq n_i$ then $g_i(q_s) \geq n_i$ for every $s > t$.

Facts about r_i

8) r_i is non-increasing.

Theorem 0.2 *Every student is assigned and every university gets the right number of students.*

Proof. By contradiction. Suppose that student S_i did not get assigned to any university.

Then

$$\begin{aligned} S_i \text{ applied to every university (fact 4)} &\rightarrow \\ \text{Every university rejected } S_i. &\rightarrow \\ \text{Every university received at least } n_i \text{ applications (fact 6)} &\rightarrow \\ \text{Every university accepted } n_i \text{ students (fact 7)} &\rightarrow \\ \text{Every student got accepted} & \end{aligned}$$

■

(d) **[8 points]** Prove that there are no assignments violating condition 2.

Theorem 0.3 *There does not exist students S_i, S_j and universities U_k, U_l where S_i is assigned to U_k and S_j is assigned to U_l, but S_j prefers U_k to U_l, and U_k prefers S_j to S_i.*

Proof. By contradiction. Suppose there is a matching (S_i, U_k) and (S_j, U_l) but S_j prefers U_k to U_l, and U_k prefers S_j to S_i.

$$\begin{aligned} (S_j, U_l) &\rightarrow \\ S_j \text{ applied to } U_k \text{ first (fact 1)} &\rightarrow \\ U_k \text{ rejected } S_j &\rightarrow \\ \text{Every applicant accepted in } U_k \text{ has better rank that } S_j \text{ (fact 8)} &\rightarrow \\ S_i \text{ has better rank than } S_j. & \end{aligned}$$

The last sentence contradicts the fact that U_k prefers S_j to S_i.

■

(e) **[8 points]** Prove that condition 3 holds.

Theorem 0.4 *The assignment is student optimal.*

Proof. Let M be the set of all stable pairings (pairings that don't violate condition 2).

By contradiction. Assume some students doesn't get his/her optimal university from the realm of possibility.

Let S_i be the first student that gets rejected by his/her optimal university U_k, and let $C = \{S_{j_1}, \ldots, S_{j_{n_k}}\}$ be the set of students currently getting a "maybe" in U_k.

By fact 8, U_k prefers every student in C to S_i. Since S_i is the first student that gets rejected by his/her optimal university, no student in C has been rejected by their optimal university yet, so every student in C likes U_k at least as much as their optimal.

Let L be a pairing in M where U_k accepts S_i, and some student in C (say S_j) is accepted in U_l (there must be such a U_l since $|C \cup S_i| = n_k + 1$ and U_k can only accept n_k students).

Then (S_j, U_k) violates condition 2.

■

Problem 2. [20 points]

Consider the following multiplication algorithm ,known as the "Russian peasant's multiplication"

x, y, are arbitrary positive integers whose values are to be multiplied. a, is the accumulator for the answer.
r, s, are temporary variables.

Initially, $r = x$, $s = y$, $a = 0$. We denote the next value of r by r', etc...

Do until $s = 0$

If s is even and $s > 0$,
$r' := 2r$
$s' := \frac{s}{2}$

If s is odd,
$a' := a + r$
$r' := 2r$
$s' := \frac{s-1}{2}$

The answer is the value in the accumulator a when $s = 0$.

Here is an example of the execution of the algorithm.

```
$x$   $y$   $r$   $s$   $a$
 5     9     5     9     0
```

10	4	5
20	2	5
40	1	5
80	0	45

Model the computation as a state machine. Prove it terminates using a derived variable. Prove that it is correct using an invariant.

The state is given by the value in (r, s, a). x and y can be included as part of the state or viewed as universal constants.

There are two types of state transitions (even or odd) as described in the algorithm. The set of initial states consists of the single state $q_0 = (x, y, 0)$.

Theorem 0.5 *The algorithm terminates.*

Proof. Define the derived variable $f(r, s, a) = \lceil \log_2 s \rceil$. Notice that since at each step s' becomes $\frac{s}{2}$ if s is even and $\frac{s-1}{2}$ if s is odd, f is decreasing.

$$\text{execution time} \leq f(q_0) = f(x, y, 0) \leq \lceil \log_2 y \rceil.$$

So the algorithm terminates in at most $\lceil \log_2 y \rceil$ steps. ■

We prove the following invariant.

$P(r, s, a) =$ "$xy = rs + a$".

Base case: For any initial state we have $P(x, y, 0)$ which is true.

Inductive step: Consider a transition from (r, s, a) to (r', s', a'). We need to prove that $P(r, s, a) \rightarrow P(r', s', a')$.

There are two cases based on the type of action, *even* or *odd*.

even:
Doubling r while halving s doesn't change the product rs, and a doesn't change. So $r's' + a' = rs + a = xy$.

odd:
$r' = 2r$, $s' = \frac{s-1}{2}$, and $a' = a + r$. So $r's' + a' = 2r\frac{s-1}{2} + a + r$. Reorganizing terms yields $r's' + a' = rs + a = xy$.

The invariant is proved.

Corollary: When $s = 0$, $a = xy$.

Proof. By our invariant, we have $P(r, 0, a) =$ "$r \cdot 0 + a = xy$", so $a = xy$. ■

Problem 3. [10 points]

Evaluate $\gcd(F_n, F_{n-1})$ for any n, where F_n is the nth Fibonacci number.

Lemma 0.6 *If* $x \geq y$, $\gcd(x, y) = \gcd(x - y, y)$

Proof.

Let $g = \gcd(x, y)$. Then $g|x$ and $g|y$, so $g|(x - y)$. To show that $g = \gcd(x - y, y)$ we argue by contradiction. Suppose that there exists $g' > g$ such that $g'|(x - y)$ and $g'|y$. But then it follows that $g'|x$, $g'|y$, and $g' > \gcd(x, y)$. This is a contradiction. ■

From the lemma it follows that $\gcd(F_n, F_{n-1}) = \gcd(F_{n-2}, F_{n-1})$.

Theorem 0.7 $\forall n \geq 1, \gcd(F_n, F_{n-1}) = 1$

Proof.

By induction.

Let $P(n) =$ "$\gcd(F_n, F_{n-1}) = 1$".

Base case: $\gcd(F_1, F_0) = \gcd(1, 0) = 1$.

Inductive step We assume $P(n)$ is true in order to conclude that $P(n + 1)$ is true.

$\gcd(F_{n+1}, F_n) = \gcd(F_{n-1}, F_n) = \gcd(F_n, F_{n-1}) = 1$

■

Problem 4. [10 points]

Derive an example of preferences for the unisex version of the mating problem (all boys or all girls) which show that it is not possible to always pair all the players so that the marriages satisfy the stability criterion.

(Hint: 4 people is enough)

Consider the amoebas Ambrosius (A), Bacchus (B), Cassius (C), and Dominicus (D) with the following preference lists.

A: (D,B,C)
B: (A,D,C)
C: (A,B,D)
D: (B,A,C)

There are three possible assignments

1) (A,B) and (C,D) is unstable because (A,D) is rogue.
2) (A,C) and (B,D) is unstable because (A,B) is rogue.
3) (A,D) and (B,C) is unstable because (B,D) is rogue.

Problem 5. **[20 points]**

Luke Skywalker is using a training device (like the one in Starwars I) to develop his light-saber fencing skills. The training device is a contraption that in every time step makes a move in a three dimensional grid and emits a non-lethal laser beam. This particular training device can only move as follows: At every step it can move 1 unit north or south AND 1 unit east or west AND 1 unit up or down. Luke is using the Force to guide his feelings but also common sense to guard against possible attack points.

If the device starts at position $(0, 0, 0)$, should Luke ever guard against an attack coming from position $(1, 1, 2)$? Prove your answer using state machines and invariants.

(Note: There are many invariants that will work for this problem.)

The state is the position of the device in the 3-dimensional grid described by the vector (x, y, z). State transitions occur by adding or subtracting 1 from each component of the vector. The start state is $(0, 0, 0)$.

For the purposes of the problem it will suffice to prove the following invariant.

$P(x, y, x) =$ "$y + z$ is even".

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

Inductive step: Assume $P(x, y, z)$ is true in order to show that $P(x', y', z')$ is true for any next state (x', y', z').

By the inductive hypothesis, $y + z$ is even. There are four cases.

1) $y' = y + 1$, $z' = z + 1$
$y' + z' = (y + 1) + (z + 1) = y + z + 2$ is even.
2) $y' = y + 1$, $z' = z - 1$
$y' + z' = (y + 1) + (z - 1) = y + z$ is even.
3) $y' = y - 1$, $z' = z + 1$
$y' + z' = (y - 1) + (z + 1) = y + z$ is even.
4) $y' = y - 1$, $z' = z - 1$
$y' + z' = (y - 1) + (z - 1) = y + z - 2$ is even.

As a corollary, $(1, 1, 2)$ is unreachable because $1 + 2$ is odd.