Problem Set 6 Solutions

Required reading: F97 Lecture Notes 7 (Section 1), F97 Lecture Notes 8 (excluding Section 2.4).

Additional references: F97 Lecture Notes 7 (Section 2), F97 Lecture Notes 8 (Section 2.4), S99 Handout 18.

Remember to include a **collaboration statement** on the front page of your solutions (see Handout 2, *Course Information*).

Problem 1. Well-orderings

Prove that for any finite set S, $(P(S), \subseteq)$ is a partially-ordered set and moreover, every nonempty subset of $P(S)$ has a minimal element. (Recall that $P(S)$ stands for the power set of S. Also, from the notes on well-founded posets (S99, H18): A *minimal* element, m, in a partial order has the property that no element is strictly below it. That is, if $s \preceq m$, then $s = m$, for all s in the partialy ordered set. For example, the set of three sets

$$\{\{0,1\}, \{0,2,3\}, \{0,1,2,3\}\}$$

partially ordered by containment has two minimal elements, namely, the first two.)

Solution:
We need to show that $\subseteq$ is reflexive, transitive, and antisymmetric over the elements of 2^S, and that every nonempty subset of $P(S)$ has a minimal element.

Reflexive: $B \subseteq B$ since $B = B$.

Transitive: Assume $A \subseteq B$ and $B \subseteq C$. Then $x \in A \Rightarrow x \in B \Rightarrow x \in C$, so $A \subseteq C$.

Antisymmetric: If $A \subseteq B$ and $B \subseteq A$, then clearly they have the same elements, so $A = B$.

Let $\emptyset \neq T \subseteq P(S)$. Define $A = \{|B| : B \in T\}$. Now by the well ordering principle, A has a least element, say a (since A is a subset of natural numbers.) Pick $C \in T$ such that $|C| = a$. Then C is a minimal element in T.

Problem 2. Terminating games

Consider the following game for two players. The players alternate moves. A move consists of a pair (x, y) of natural numbers, subject to the constraint that no move may be $\geq$ to any previous move in both coordinates. The player who makes the $(0,0)$ move is the loser.

(a) Identify a winning strategy for the first player, and argue its correctness.

Solution:
The first player has two winning strategies:
Strategy 1: If the first player starts with $(1, 1)$ then whatever the second player picks, either $(0, n)$ or $(0, n)$ for some n, the first player will respond with, respectively, $(n, 0)$ or $(0, n)$. By symmetry, the first player will always have a move whenever the second player had a move, except when the second player picks $(0, 0)$ and loses.
Strategy 2: The second winning strategy is slightly more tricky: The first player starts by picking $(0, 2)$,[1] and then:

- If the 2nd player picks $(0, n)$ for any $n \geq 1$, 1st player responds with $(1, n - 1)$.
- If the 2nd player picks $(1, n)$ for any $n \geq 0$, 1st player responds with $(0, n + 1)$.

And the above procedure should be repeated for all consecutive moves. Notice that if player 2 had a valid move, so does player 1, unless player 2 picked $(0, 0)$ and lost.
Furthermore, there are no other winning strategies for player 1, because if she starts with any other legal move, player 2 will be able to adopt either of the two winning strategies and win for sure:

- If player 1 picks some (m, n) where $m, n \geq 1$ and $(m, n) \neq (1, 1)$, then player 2 can pick $(1, 1)$ next, and then use the first strategy to win against player 1.
- If player 1 picks some $(0, n)$ or $(n, 0)$ where $n \geq 3$, then player 2 can respectively pick $(0, 2)$ or $(2, 0)$ next, and then use the second strategy to win against player 1.
- If player 1 picks $(1, 0)$ or $(0, 1)$ then player 2 picks the other one and wins.

At any point in the game, a valid move (a, b) is said to be *bounded* iff both a is bounded above by the x-coordinate of some previous move and b is bounded above by the y-coordinate of some previous move. For example, if the previous moves have been $(1, 3), (3, 1), (2, 2)$, then there are ten possible bounded moves: $\{(1, 2), (2, 1), (1, 1), (0, 3), (0, 2), (0, 1), (0, 0), (1, 0), (2, 0), (3, 0)\}$. There is also an infinite number of possible unbounded moves at this point, namely, any move with one coordinate equal to 0 and the other larger than 3.

For any state S of the game after the first move, let $X(S)$ be the minimum of the x-coordinates of the previous moves, $Y(S)$ be the minimum of the y-coordinates, and $B(S)$ be the number of possible bounded moves. Define the *measure*, $M(S)$, of the state to be the triple $(X(S), Y(S), B(S))$.

Define the following ordering on triples of natural numbers:

$$(x, y, k) < (x', y', k') \iff [(x < x') \vee (x = x' \wedge y < y') \vee (x = x' \wedge y = y' \wedge k < k')$$

(b) Prove that any nonempty subset of $\mathbb{N}^3$ has a least element under the above ordering.

Solution:
Let S be some nonempty subset of $\mathbb{N}^3$. Let $S_x = \{x \mid \exists_{y,z}\ (x, y, z) \in S\}$. Then $S_x \subset \mathbb{N}$, and

[1] Of course, picking $(2, 0)$ would be equivalent, and the following argument will hold for it as well, with x and y coordinates replaced

so by the well-ordering principle, there is a smallest element x_{min} in S_x. Now take a set $S_y = \{y \mid \exists_z\ (x_{min}, y, z) \in S\}$. Since $S_y \subseteq \mathbb{N}$, and it is not empty, it has a smallest element y_{min}. Similarly, the set $S_z = \{z \mid (x_{min}, y_{min}, z) \in S\}$ has a smallest element z_{min}.

Clearly, $(x_{min}, y_{min}, z_{min})$ is the smallest element in S. If there was some smaller element (x', y', z') in S, then:

- If $x' < x_{min}$ then $x' \neq x_{min}$ would be the smallest element in S_x. Impossible.
- If $x' = x_{min}$ and $y' < y_{min}$ then $y' \neq y_{min}$ would be the smallest element in S_y. Impossible.
- If $x' = x_{min}$, $y' = y_{min}$ and $z' < z_{min}$ then $z' \neq z_{min}$ would be the smallest element in S_z. Impossible.

(c) Conclude that the game always terminates, regardless of the players' strategy.

Solution:
We will show below that the measure $M(S) = (X(S), Y(S), B(S))$ decreases with every move according to the ordering in $\mathbb{N}^3$ defined above.

Assume by contradiction that for some move leading from state $S = (x, y)$ to state $S' = (x', y')$, we have that $M(S') \geq M(S)$. There are three cases in which $M(S) \leq M(S')$ could be true:

1. $X(S) < X(S')$, which would mean that the minimum of the x-coordinates has increased, which is impossible (the minimum of a set to which we are adding some new element cannot increase).
2. $X(S) = X(S') \wedge Y(S) < Y(S')$, which would mean that the minimum of the y-coordinates has increased, which is impossible for the same reason as above.
3. $X(S) = X(S') \wedge Y(S) = Y(S') \wedge B(S) \leq B(S')$. Let $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ be the points that were picked up to move $S \to S'$ (i.e., the points that constitute state S), and let (x', y') be the new point. If $X(S) = X(S')$ and $Y(S) = Y(S')$, then $x' \geq \min_i\{x_i\} \wedge y' \geq \min_i\{y_i\}$.
   We first show that (x', y') must be a bounded point in state S. Assume otherwise. That would mean that either x' or y' were not bounded in state S. Without loss of generality, assume x' was not bounded, i.e., that $x' > x_i$ for all i. On the other hand, we know that $y' \geq y_j = \min_i\{y_i\}$. But then $y_j \leq (y')$ and $x_j < x'$ which would make (x', y') an illegal move. Hence (x', y') must be a bounded point of state S.
   Now we show that since (x', y') is bounded in S, state $S' = S \cup \{(x', y')\}$ must have fewer bounded points than S, i.e., that $B(S') < B(S)$. This will finish the proof that $X(S) = X(S') \wedge Y(S) = Y(S') \wedge B(S) \leq B(S')$ cannot be true.
   Notice that a point (x, y) is bounded in state $S = \{x_i, y_i\}$ iff it is a valid move and $x \leq \max_S\{x_i\}$ and $y \leq \max_S\{y_i\}$. Since, if (x', y') is bounded in S then the maximal x and y coordinates do not change from S to S'. Consequently, if $x \leq \max_{S'}\{x_i\} \wedge y \leq \max_{S'}\{y_i\}$ then $x \leq \max_S\{x_i\} \wedge y \leq \max_S\{y_i\}$. Also, if a point (x, y) is valid in S', it is also valid in S (it clearly follows from the fact that $S \subset S'$). From these two facts it follows that if a point is bounded in S', it is also bounded in S. In other words, the set

$\mathcal{B}'$ of bounded points of S' is a subset of $\mathcal{B}$, the set of bounded points of S. Since the latter is a finite set[2] and $\mathcal{B}' \subset \{\mathcal{B} \setminus (x', y')\}$, it follows that $B(S') = |\mathcal{B}'| < |\mathcal{B}| = B(S)$.

This proves that measure $M(S)$ always decreases and hence finishes the proof that the game always terminates.

Problem 3. Summations

Recall that H_n is the nth harmonic number, i.e.,

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \cdots + \frac{1}{n}$$

(a) Rosen §1.7 exercise 21 (Do not quote the answer from the back, and instead explicitly show the steps involved in calculating that answer.)

Solution:
(i)

$$\begin{aligned}
\sum_{k=1}^{n} 2k-1 &= \sum_{k=1}^{n} k^2-(k-1)^2 \\
&= n^2-(n-1)^2+(n-1)^2-(n-2)^2+(n-2)^2-(n-3)^2 \\
&\quad +\cdots+1^2-1^2+0^2 \\
&= n^2
\end{aligned}$$

(ii) $\sum_{k=1}^{n} 2k-1 = -n + 2\sum_{k=1}^{n} k$. This implies that

$$\begin{aligned}
\sum_{k=1}^{n} k &= \frac{1}{2}\left[n + \sum_{k=1}^{n} 2k-1\right] \\
&= \frac{1}{2}(n+n^2) \\
&= \frac{1}{2}n(1+n)
\end{aligned}$$

□

Find a closed form for each of the following:

(b) $\sum_{k=0}^{\infty} i^k/2^{k/4}$, where i is $\sqrt{-1}$.

Solution:
Note that we can separate this sum into several pieces:

[2] After the first move, the set of bounded points is always finite, because for whatever values x_{max}, y_{max}, there are finite number of pairs $(x, y) \in \mathbb{N}^2$ s.t. $x \leq x_{max}$ and $y \leq y_{max}$.

$$\begin{array}{ll}
(1) & 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \cdots + \\
(2) & \frac{i}{2^{1/4}} + \frac{i}{2^{5/4}} + \frac{i}{2^{9/4}} + \frac{i}{2^{13/4}} + \cdots + \\
(3) & \frac{-1}{2^{2/4}} + \frac{-1}{2^{6/4}} + \frac{-1}{2^{10/4}} + \frac{-1}{2^{14/4}} + \cdots + \\
(4) & \frac{-i}{2^{3/4}} + \frac{-i}{2^{7/4}} + \frac{-i}{2^{11/4}} + \frac{-i}{2^{15/4}} + \cdots
\end{array}$$

You should recognize the first summation: it converges to 2. The second summation is therefore simply $\frac{i}{2^{1/4}}$ times the first summation, leaving $2^{3/4}i$. Similarly, the third and fourth summations are $-\sqrt{2}$ and $-2^{1/4}i$, respectively. The total is then $(2-\sqrt{2}) + i(2^{3/4} - 2^{1/4})$, or $(1 + 2^{-1/4}i)(2-\sqrt{2})$. □

(c) $\sum_{k=1}^{\infty} 1/k(k+n)$ (Your solution may involve the harmonic number function.)

Solution:
Note by partial fractions that

$$\frac{n}{k(k+n)} = \frac{1}{k} - \frac{1}{k+n}$$

Let $K > n$, then

$$\sum_{k=1}^{K} \left(\frac{1}{k} - \frac{1}{k+n} \right) = \sum_{k=1}^{n} \frac{1}{k} - \sum_{k=K+1}^{K+n} \frac{1}{k}$$

But

$$\sum_{k=K+1}^{K+n} \frac{1}{k} \leq \frac{n}{K+1}$$

Therefore

$$\lim_{K\to\infty} \sum_{k=K+1}^{K+n} \frac{1}{k} = 0$$

Hence

$$\lim_{K\to\infty} \sum_{k=1}^{K} \left(\frac{1}{k} - \frac{1}{k+n} \right) = H_n$$

i.e.

$$\sum_{k=1}^{\infty} \frac{1}{k(k+n)} = \frac{H_n}{n}$$

(d) $\sum_{i=1}^{\infty}\sum_{j=1}^{\infty} ix^iy^{i+2j}$

Solution:

$$\begin{aligned}
\sum_{i=1}^{\infty}\sum_{j=1}^{\infty} ix^iy^{i+2j} &= \sum_{i=1}^{\infty}\left[i(xy)^i\left(\sum_{j=1}^{\infty} y^{2j}\right)\right] \\
&= \sum_{i=1}^{\infty}\left[i(xy)^i\left(\sum_{j=1}^{\infty} (y^2)^j\right)\right] \\
&= \sum_{i=1}^{\infty}\left[i(xy)^i\left(\frac{y^2}{1-y^2}\right)\right] \\
&= \frac{y^2}{1-y^2}\sum_{i=1}^{\infty}\left[i(xy)^i\right] \\
&= \frac{y^2}{1-y^2}\cdot\frac{xy}{(1-xy)^2} \\
&= \frac{xy^3}{(1-y^2)(1-xy)^2}
\end{aligned}$$

Find asymptotically tight bounds[3] for the following:

(e) $\sum_{k=1}^{n} k^4$

Solution:

$$\begin{aligned}
\sum_{k=1}^{n+1} k^4 &\le \int_1^{n+2} x^4\ dx \\
&= \frac{(n+2)^5-1}{5} \\
&\sim \frac{n^5}{5}
\end{aligned}$$

$$\begin{aligned}
\sum_{k=1}^{n+1} k^4 &\ge \int_0^{n+1} x^4\ dx \\
&= \frac{(n+1)^5}{5} \\
&\sim \frac{n^5}{5}
\end{aligned}$$

[3] We say that function f provides an asymptotically tight bound for a function g if $lim_{n\to\infty} f(n)/g(n) = 1$. That is, $f \sim g$ using the notation from Lecture Notes 8, §2.4.

□

(f) $\sum_{k=1}^{n} e^{\sqrt{k}}/\sqrt{k}$

Solution:
Note that with $u = \sqrt{x}$, $du = \frac{1}{2\sqrt{x}}\, dx$. So,

$$\begin{aligned}\sum_{k=1}^{n} \frac{e^{\sqrt{k}}}{\sqrt{k}} &\leq \int_1^{n+1} \frac{e^{\sqrt{x}}}{\sqrt{x}}\, dx \\ &= 2(e^{\sqrt{n+1}} - e) \\ &\sim 2e^{\sqrt{n}}\end{aligned}$$

To see why the last line holds, notice that $\sqrt{n+1} - \sqrt{n} = \frac{1}{\sqrt{n+1}+\sqrt{n}}$, so $\sqrt{n+1} - \sqrt{n} \to 0$ as $n \to \infty$. Thus $\lim_{n\to\infty} \frac{e^{\sqrt{n+1}}}{e^{\sqrt{n}}} = 1$.
Finally,

$$\begin{aligned}\sum_{k=1}^{n} \frac{e^{\sqrt{k}}}{\sqrt{k}} &\geq \int_0^{n} \frac{e^{\sqrt{x}}}{\sqrt{x}}\, dx \\ &= 2e^{\sqrt{n}} - 2 \\ &\sim 2e^{\sqrt{n}}.\end{aligned}$$

□

Problem 4. Linked lists

You are given a one-way linked list with starting link l_0. For any link, l, we let $S(l)$ denote the next link in the list after l. If $S(l) =$ NULL, then link l is the last link on the list. But the list may link back to itself (that is, it may be that $S(l_1) = S(l_2)$ for some links $l_1 \neq l_2$), in which case there is no last link because the list ends in a cycle. To determine whether or not there is a cycle, you could follow the links down the list recording the links encountered while checking to see if a link is ever revisited, but this could take a lot of memory and time. So you use the following procedure to detect cycles:

```
a := l_0;
b := l_0;
while ((S(a) != NULL) and (S(b) != NULL) and (S(S(b)) != NULL))
{
    a := S(a);
    b := S(S(b));
    if a = b
        return "CYCLE"
};
return "NO CYCLE"
```

Prove that this procedure terminates and produces the correct answer.

Solution:
If the list has no cycles, then within $n/2$ steps, where n is the length of the list, either `S(b)` or `S(S(b))` will be `NULL`, so the process will break from the while loop and return `NO CYCLE`.

If the list does contain a cycle, then eventually both markers `a` and `b` will be stuck in the cycle. Let d_j denote the distance measured clockwise from `b` to `a` at the j^{th} execution of the while loop. (Assume the markers are moving around the cycle in the clockwise direction.) If $d_j = 0$, then `a` = `b`, so the process terminates and outputs `CYCLE`. Otherwise, note that at each execution of the while loop, this distance decreases by exactly one, so after d_j steps, the process outputs `CYCLE` and terminates.

Problem 5. Mystery Problem

Given a list of numbers $A[1], \ldots, A[n]$, sorted in increasing order, and a number a, consider the following procedure which takes as arguments natural numbers $i, j \in \{1, \ldots, n\}$ such that $i \leq j$:

```
Proc(i, j):
    start  := i;
    end    := j;
    middle := start + quotient((end - start)/2);
    if A[middle] = a
        return middle;
    if (start = middle)
        return "NOT FOUND";
    if A[middle] > a
        Proc(start, middle)
    else
        Proc(middle+1, end)
```

What does `Proc`(i, j) do? Prove your answer. Note that `start`, `middle`, and `end` are always integers.

Solution:

If a is in the list, say $a = A[k]$, then `Proc`$(1, n)$ returns k. If a is not in the list, then `Proc`$(1, n)$ returns `NOT FOUND`.

To prove this, we prove more generally that for any $1 \leq i \leq j \leq n$, if a is in the list $A[i], \ldots, A[j]$—say $a = A[k]$—then `Proc`(i, j) returns k. And if a is not in the list $A[i], \ldots, A[j]$, then `Proc`(i, j) returns `NOT FOUND`. The proof is by induction on the value of $j - i$.

If $j - i = 0$, that is $j = i$, then in the call `Proc`(i, j), the initial value of `middle` will be i, and the call will return i or `NOT FOUND` according to whether $a = A[i]$, that is, whether a is in the list $A[i]$. So in the base case, the call behaves according to thte hypothesis.

Now suppose $j - i > 0$. In the call `Proc`(i, j), let k be the initial value of `middle`. From the assignment which defines the initial value of `middle`, we conclude that $i \leq k < j$. If $a = A[k]$, then the call will return k; but in this case, a is indeed in the list $A[i], \ldots, A[j]$, so this is the return behavior specified by the induction hypothesis for $j - i$. Next, if $A[k] > a$, then the call of `Proc`(i, j) will return the same result as the call `Proc`(i, k). But now $k - i < j - i$ so by induction, this call returns the index of a if a is in the list $A[i], \ldots, A[k]$, and returns `NOT FOUND` otherwise. But since $A[k] > a$, it follows that a is in the list $A[i], \ldots, A[j]$ iff it is in the list $A[i], \ldots, A[k]$, so the answer returned by the call `Proc`(i, k) is the one required by the induction hypothesis for $j - i$. A similar argument verifies the induction hypothesis for $j - i$ in the case that $A[k] < a$.