Cosc 30: Discrete Mathematics

Prishita Dharampal

Problem 1(a). **Faulty induction.**
Induction proofs can break for many reasons. Careless summoning of the genie or misuse of the induction boilerplate may result in fake "proofs" for statements that are just plain false. Point out the most significant error from the following arguments and justify.
Let B be a set of blocks, arranged into stack(s). The blocks in each stack are in a linear order from bottom to top, with every block touching exactly one block from below and one from above (except for the top- and bottom-most blocks); two blocks from two separate stacks do not touch one another.
Consider the "proof" of the following statement: If every block in B is touching at least one other block in B, then all blocks in B are in a single stack.

> We prove the theorem by induction on the number of blocks in B. Let $P(B)$ be the statement "if every block in B is touching at least one other block in B, then all blocks in B are in a single stack."
> Consider an arbitrary set of blocks B arranged in stacks with exactly n blocks, where every block in B touches at least one other block in B. Assume as the induction hypothesis that $P(B')$ holds for any set of blocks B' with fewer than n blocks.
>
> - When B is empty, the statement "every block in B is touching at least one other block in B" is vacuously true, and "all blocks in B are in a single stack" is also vacuously true.
>
> - Otherwise, B is non-empty. Remove one block b from B to obtain a set B' containing $n-1$ blocks. By the induction hypothesis on B', all the blocks in B' are in a single stack. Because in $P(B)$ every block in B is assumed to touch at least one other block in B, block b must touch another block in B (which must be in B'). This means b is in the same stack as all blocks in B'. Therefore, all blocks in B are in a single stack.
>
> In both cases $P(B)$ holds, so the theorem holds.

Problem 1(b).
Recall the definition of big-O notation: the proposition $f(n) = O(g(n))$ means

$$\exists C > 0\ \exists N\ \forall n \geq N, \quad f(n) \leq C \cdot g(n).$$

Let $\texttt{MS}(n)$ be defined by the following pseudocode:

```
MS(n):
    if n = 1: return 1
    else: return 2 * MS(floor(n/2)) + n
```

Consider the following "proof" claiming that $\texttt{MS}(n) = O(n)$.
We prove the statement by induction on n. Let n be an arbitrary positive integer, and assume that for every positive integer $m < n$, $\texttt{MS}(m) = O(m)$.

- If $n = 1$, then $\texttt{MS}(1) = 1 \leq C \cdot 1$ for any $C \geq 1$.
- If $n > 1$, then
$$\texttt{MS}(n) = 2 \cdot \texttt{MS}(\lfloor n/2 \rfloor) + n.$$
By the induction hypothesis, there exist constants $C' > 0$ and N' such that
$$\texttt{MS}(\lfloor n/2 \rfloor) \leq C' \cdot \lfloor n/2 \rfloor \quad \text{for all } n \geq N'.$$
Thus,
$$\texttt{MS}(n) \leq 2C' \cdot \lfloor n/2 \rfloor + n \leq C'n + n = (C' + 1)n.$$
Choosing $C \geq C' + 1$ gives $\texttt{MS}(n) \leq Cn$.

Therefore, $\texttt{MS}(n) = O(n)$.

Problem 2. Balanced parentheses.
A balanced parentheses string is a string over the symbols `[` and `]`, defined recursively as one of the following:

- the empty string ϵ;
- a string $\texttt{[}w\texttt{]}$ for some balanced parentheses string w;
- a string xy for some nonempty balanced parentheses strings x and y.

For example, `[[[][]]][][[[]]][]` is a balanced parentheses string of length 18.

1. Prove by induction that removing any pair of consecutive symbols `[]` (if one exists) from a balanced parentheses string results in another balanced parentheses string.

⋆ (b) Prove by induction that removing any pair of consecutive symbols `][` (if one exists) from a balanced parentheses string results in another balanced parentheses string.

Problem 3. **Fibonacci representations.** The Fibonacci numbers are defined by

$$F_n = \begin{cases} 0 & \text{if } n = 0, \\ 1 & \text{if } n = 1, \\ F_{n-1} + F_{n-2} & \text{otherwise.} \end{cases}$$

Prove that every non-negative integer can be written as a sum of distinct, non-consecutive Fibonacci numbers using the following algorithm:

```
TOFIBO(n):
    if n = 0: return <>
    else:
        F_l <- largest Fibonacci number <= n
        <i_k, ..., i_1> <- TOFIBO(n - F_l)
        return <l, i_k, ..., i_1>
```

In other words, no two consecutive Fibonacci numbers F_i and F_{i+1} appear in the sum. (Assume $F_0 = 0$ never appears.)
For example,

$$30 = F_8 + F_6 + F_2.$$

Problem 4⋆. Ludicrous tiling.
Let Q be the first quadrant of the real plane $\mathbb{R}^2$. You have an unlimited supply of L-shaped tiles, each consisting of a 2×2 square with one unit square removed.
Prove that you can tile the entire quadrant Q using only L-shapes and backward-L-shapes, with no rotations allowed.

Problem 5⋆. Rooted binary tree.
A rooted binary tree is defined recursively as either:

- the empty tree, denoted `null`; or
- a root node $T.x$, a left subtree T_ℓ, and a right subtree T_r, where T_ℓ and T_r are rooted binary trees.

In C, such a structure can be defined as:

```
struct binaryTree {
    int root;
    struct binaryTree *leftSubtree;
    struct binaryTree *rightSubtree;
};
```

Let T be a rooted binary tree with n vertices. Deleting any node v splits T into at most three subtrees: one containing the left child of v, one containing the right child of v, and one containing the parent of v (if they exist).
We call v a *central node* if each resulting subtree has at most $n/2$ nodes. Prove that every rooted binary tree has at least one central node.