COMS10014 Solutions 6: Induction

1. Induction over the Integers

1. Let us write $L(n) = \sum_{i=1}^{n}(2i-1)$ and $R(n) = n^2$, then $P(n) \equiv L(n) = R(n)$. We prove $P(n)$ by induction.

   For the base case, take $n = 1$, then the claim is $L(1) = R(1)$ and indeed $L(1) = 2 \times 1 - 1 = 1$ and $R(1) = 1^2 = 1$, so $L(1) = R(1)$.

   For the induction step, we prove $P(n) \vdash P(n+1)$. Assume $P(n)$ for some integer n. Then $L(n+1) = \sum_{i=1}^{n+1}(2i-1) = \sum_{i=1}^{n}(2i-1) + 2\big((n+1)-1\big) = L(n) + 2n + 1$. Using the induction hypothesis, we can replace $L(n)$ by $R(n)$, so $L(n+1) = R(n) + 2n + 1 = n^2 + 2n + 1 = (n+1)^2 = R(n+1)$.

   Since we have shown $P(1)$ and $P(n) \vdash P(n+1)$, this proves $P(n)$ for all $n \geq 1$ by induction.

2. With the same idea as above, let $L(n) = \sum_{i=0}^{n} 2^i$ and $R(n) = 2^{n+1} - 1$. We prove $L(n) = R(n)$ by induction.

   For the base case, take $n = 0$, then $L(0) = 2^0 = 1$ and $R(0) = 2^1 - 1 = 1$ too.

   For the induction step, we show $(L(n) = R(n)) \vdash (L(n+1) = R(n+1))$. Assume that $L(n) = R(n)$ and calculate

$$\begin{aligned} & L(n+1) \\ = & \sum_{i=0}^{n+1} 2^i \\ = & \sum_{i=0}^{n} 2^i + 2^{n+1} \\ = & L(n) + 2^{n+1} \\ \text{(induction step)} & \\ = & R(n) + 2^{n+1} \\ = & 2^{n+1} - 1 + 2^{n+1} \\ = & 2(2^{n+1}) - 1 \\ = & 2^{n+2} - 1 \\ = & R(n+1) \end{aligned}$$

   Since we have shown $P(0)$ and $P(n) \vdash P(n+1)$, we conclude $P(n)$ for all n by induction.

3. The base case is $P(0)$ which is

$$\sum_{i=0}^{0} r^i = r^0 = 1 = \frac{1-r}{1-r} = \frac{1-r^{0+1}}{1-r}$$

   For the induction step, assume $P(n)$ and calculate

$$\sum_{i=0}^{n+1} r^i = \left(\sum_{i=0}^{n} r^i\right) + r^{n+1} = \frac{1-r^{n+1}}{1-r} + r^{n+1} = \frac{1-r^{n+1}}{1-r} + \frac{(1-r)r^{n+1}}{1-r}$$
$$= \frac{1-r^{n+1}+r^{n+1}-r^{n+2}}{1-r} = \frac{1-r^{(n+1)+1}}{1-r}$$

   Which is the statement $P(n+1)$.

   Since we have shown $P(0)$ and $P(n) \vdash P(n+1)$, we conclude $P(n)$ for all n by induction.

4. $P(n)$ here is the statement $3 \mid n^3 - n$ (read: "3 divides $n^3 - n$"), which means that there is some integer k such that $n^3 - n = 3k$.

   For the base case, consider $P(0)$, and $0^3 - 0 = 0$ which is divisible by 3 as $0 = 3 \times 0$.

   To show $P(n) \vdash P(n+1)$, assume $P(n)$ for some n. This means that that there is some integer k such that $n^3 - n = 3k$. Then $(n+1)^3 - (n+1) = n^3 + 3n^2 + 3n + 1 - n - 1 = (n^3 - n) + 3(n^2 + n)$. Using the induction assumption, we can rewrite this as $3k + 3(n^2 + n)$ which is clearly a multiple of 3. To be really formal, there is a k' such that $(n+1)^3 - (n+1) = 3k'$, namely $k' = k + (n^2 + n)$. This proves $P(n+1)$.

   Since we have shown $P(0)$ and $P(n) \vdash P(n+1)$, this proves $P(n)$ for all n by induction.

2. Sums of Squares

a. The general pattern is that the square of each n is the square of the previous one, plus the next odd number counting up. As a formula, $n^2 = (n-1)^2 + (2n-1)$ since the expression $2n - 1$ goes through the values 1, 3 ,5, 7, …

b. We can just multiply out the brackets on the right hand side to get
$$(n-1)^2 + (2n-1) = n^2 - 2n + 1 + 2n - 1 = n^2$$

c. Let $L(n) = n^2$ and $R(n) = (n-1)^2 + (2n-1)$. We prove $L(n) = R(n)$ for $n \geq 1$ by induction.

   The base case is $n = 1$ where $L(1) = 1$ and $R(1) = 0 + 1 = 1$, therefore $L(1) = R(1)$.

   For the induction step, assume $L(n) = R(n)$ for some n and show $L(n+1) = R(n+1)$. Starting with $L(n+1) = (n+1)^2$, we can multiply this out to get $n^2 + 2n + 1$ which is $L(n) + 2n + 1$. Using the induction hypothesis, we rewrite this as $R(n) + 2n + 1$ which is $(n-1)^2 + (2n-1) + 2n + 1 = n^2 - 2n + 1 + 2n - 1 + 2n + 1 = n^2 + 2n + 1$. Remember, we are trying to show this is equal to $R(n+1) = n^2 + 2(n+1) - 1$ so we rewrite $n^2 + 2n + 1 = n^2 + 2n + 1 + (1 - 1) = n^2 + 2n + 2 - 1 = n^2 + 2(n+1) - 1$ which is $R(n+1)$.

   Therefore, we have shown $L(1) = R(1)$ and that if $L(n) = R(n)$ then $L(n+1) = R(n+1)$ which proves the statement for $n \geq 1$ by induction.

3. Prime Decomposition

a. Suppose the statement is false, that is there is at least one integer $n \geq 2$ that does not have a prime decomposition. Pick the smallest such n, so that all numbers from 2 to $n - 1$ do have a prime decomposition (the well-ordering principle).

   The proof is by case distinction: if n is a prime, then we have a contradiction, as n would be its own prime decomposition. If n is not a prime, then that means we can write $n = ab$ for some integers $a, b > 1$, since if we could not do this then n would meet the definition of a prime. But now, since $a > 1$ then $b < n$, and since $b > 1$ then $a < n$. Therefore, since n is the *smallest* integer with no prime decomposition, both a and b do have prime decompositions.

   This means we can write $a = p_1 \times p_2 \times \ldots$ where the p_i are all prime, and $b = q_1 \times q_2 \times \ldots$ where the q_j are all prime. But $n = ab$ so $n = (p_1 \times \ldots) \times (q_1 \times \ldots)$, which after removing

the brackets is a prime decomposition of n. This contradicts the assumption that n does not have a prime decomposition, and therefore the original assumption that the statement to prove is false, is itself false. The statement is therefore proven true.

b. Let $P(n)$ be the statement that n has a prime decomposition. We prove $P(n)$ by strong induction.

The base case is $n = 2$, and $P(2)$ is the statement that 2 has a prime decomposition. 2 is prime, so this is true.

For the strong induction step, we assume $P(1)$ up to $P(n)$ are all proven, and we show $P(n+1)$. If $n+1$ is prime, then clearly $P(n+1)$ holds. If $n+1$ is not prime, then we can write $n+1 = ab$ with $a, b > 1$ and therefore $a, b < n$. By the strong induction hypothesis, this means that $P(a)$ and $P(b)$ hold and there are decompositions $p_1 \times \ldots$ for a and $q_1 \times \ldots$ for b. Since $n+1 = ab$, we have $n+1 = p_1 \times \ldots \times q_1 \times \ldots$ which is a prime decomposition of $n+1$. Therefore, $P(n+1)$ holds, and the claim is proven by strong induction.

The mathematically astute reader might be wondering whether the amount of "..." in the above proof is a concern, as these are usually an abbreviation for "I could do this properly, but I won't". Here is the fully formal version. A prime decomposition for an integer n is a map $d\colon \mathbb{N}_+ \to \mathbb{N}$ (that is, 0 is not in the domain but is in the codomain) where $d(i)$ is the number of times the i −th prime appears in the decomposition. For example, for $n = 126 = 2 \times 3^2 \times 7$, the first prime 2 appears once, the second prime 3 appears twice, the 3rd prime 5 appears 0 times, and the 4th prime 7 appears once, so the map here is $d(1) = 1, d(2) = 2, d(3) = 0, d(4) = 1$ and $d(i) = 0$ for $i > 4$. We could of course write this as $d = (1,2,0,1,0,0\ldots)$ if we allow some "dots", or as $d = (1,2,0,1)$ with the convention that we only write the list until the point where all remaining components are 0. With this convention, if d_n is the decomposition for the integer n, then if p_i is the i-th prime we have

$$n = \prod_{i \in \mathbb{N}_+} p_i^{d(i)}$$

The key step in the proof (whether with or without induction) is then that for $n = ab$ we can find decompositions d_a and d_b, and then the decomposition of n without any "dots" is

$$n = \prod_{i \in \mathbb{N}_+} p_i^{d_a(i)+d_b(i)}$$

which we could, if we wanted to, define as $d_a \oplus d_b$ as a sum operation on decompositions.

4. Structural Induction

This problem is a paraphrase of *Hofstadter's MU*, from the Artificial Intelligence researcher Douglas Hofstadter's book Gödel, Escher, Bach.

The invariant is that the number of air symbols in a valid string can never be a multiple of 3 (including 0). We prove this with structural induction.

Invariant. The number of air symbols in a valid string is always of the form $3k+1$ or $3k+2$ for some integer k. (That is, the remainder r of the number of air symbols modulo 3 satisfies $r \neq 0$.)

1. The starting string has one air symbol, which is of the form $3 \times 0 + 1$.
2. Rain rule: this does not change the number of air symbols. If you apply the rain rule to a string meeting the invariant, then the result meets the invariant too.
3. Sun rule: suppose you apply the sun rule to a string with $3k + r$ air symbols, which by the induction assumption has $r = 1$ or $r = 2$. We do a case distinction:
    a. $r = 1$: the sun rule doubles everything except the initial fire symbol, so it doubles the number of air symbols. The new string has $2(3k + 1) = 3(2k) + 2$ air symbols, which meets the invariant again.
    b. $r = 2$: here $2(3k + 2) = 3(2k + 1) + 1$ which meets the invariant again (that is, the sun rule changes a remainder of 1 into 2 and vice versa, but if the string before met the invariant, then so does the string afterwards).
4. Wind rule: this removes three air symbols, so if the number before was $3k + r$ for $k > 0$ then the number afterwards is $3(k - 1) + r$. Again, if the string before met the invariant $r \neq 0$, then so does the string afterwards.

Therefore,

- The only rule that produces a valid string "from nothing", the starting string, produces a string that meets the invariant.
- All rules that transform strings have the property that if the input string meets the invariant, then so does the output string.

By structural induction, we conclude that all valid strings under this set of rules meet the invariant. The string (fire, water) has zero air symbols, which does not meet the invariant, therefore it is not possible to create this string using the rules given.

5. Invariant Induction: Square-and-Multiply

The claim is that at the end of the algorithm, $y = g^x \pmod p$. The invariant here is

$$y = \prod_{j=0}^{i} g^{(2^j)x_j} \pmod p \quad \wedge \quad a = g^{(2^{i+1})} \pmod p$$

where i is the number of loop iterations that have run so far. Using $g^a g^b = g^{ab}$ (which still holds modulo p and, by induction, for any finite number of factors), we can rewrite the first one as

$$y = g^s \pmod p \quad \text{for} \quad s = \sum_{j=0}^{i} (2^j)x_j$$

the point being that

$$x = \sum_{j=0}^{k} (2^j)x_j$$

since that is the binary expansion of x. Therefore, once we reach $i = k$, we have $y = g^x \pmod p$.

For our base case, before the loop runs for the first time, we have $y = 1$ (the neutral element of multiplication) and $a = g = g^1 \pmod p$. By convention, an empty product is 1. After the loop has run for the first time, we have $y = 1$ if $x_0 = 0$ and $y = g \pmod p$ if $x_0 = 1$. In both cases, $y = g^{x_0} \pmod p$ since g^0 is 1 (we have defined that $g > 0$) and $a = g^2 \pmod p$.

For the induction step, assume that at the start of the pass through the loop with counter value i, we have (using the convention that capital letters refer to values of variables before the loop)

$$Y = \prod_{j=0}^{i-1} g^{(2^j)x_j} \pmod{p} \quad \wedge \quad A = g^{(2^i)} \pmod{p}$$

if $x_i = 0$, then y gets multiplied by a (and reduced modulo p). So we can say that

$$y = Y \times A^{x_i} \pmod{p}$$

since for $x_i = 0$ this does nothing (it multiplies y with $A^0 = 1$) whereas for $x_i = 1$ it multiplies y wiith $A^1 = A$. But $A = g^{(2^i)} \pmod{p}$. Therefore,

$$y = g^{(2^i)x_i} \times \prod_{j=0}^{i-1} g^{(2^j)x_j} \pmod{p}$$

which is of course

$$y = \prod_{j=0}^{i} g^{(2^j)x_j} \pmod{p}$$

with the index increased by 1. As for the other variable, we know $A = g^{(2^i)} \pmod{p}$ and $a = A \times A$ giving $a = \left(g^{(2^i)}\right)^2 = g^{2\times(2^i)} = g^{(2^{i+1})} \pmod{p}$ as required.

We have shown that, if the invariant holds at the start of a loop pass, then it still holds at the end, and therefore when the pass with $i = k$ finishes, we have computed $y = g^x \pmod{p}$ as required.

Note that we could have added $y < p$ and $a < p$ to the invariant, as any computation that updates these variables finishes with a modulo-p operation.