

CSC236 Problem Set 1

Xuanqi Wei 1009353209

September 19, 2023

Contents

1	Question 1	1
2	Question 2	3
3	Question 3	6

1 Question 1

(a) According to the definition of P:

$$\forall g_1 \in G_1, \exists t_1 \in T_1, t_1 \text{ tiles } g_1 \implies \forall g_2 \in G_2, \exists t_2 \in T_2, t_2 \text{ tiles } g_2$$

(b) Firstly, assume

$$\forall g_1 \in G_1, \exists t_1 \in T_1, t_1 \text{ tiles } g, \text{ which is the antecedent.}$$

Secondly, I will do the consequent part, which:

Let g_2 be an arbitrary element from G_2

Then, I want to prove that

$$\exists t_2 \in T_2, t_2 \text{ tiles } g_2$$

by selecting a satisfying element t_2 from T_2 and prove the element t_2 satisfies $t_2 \text{ tiles } g_2$.

(c) The diagram above illustrates one instance of G_2 grids, which being tiled by triominoes.

Firstly, we already know that for P(1), the statement $\forall g_1 \in G_1, \exists t_1 \in T_1, t_1 \text{ tiles } g$ is true which is the antecedent of this direct proof.

Secondly, the above diagram is an element of the set of all $2^2 \times 2^2$ grid with one square removed, which is an element of G_2 . By visualising those colorful triominoes, we see a combination triominoes, t_2 , which is an element of the set of all tilings of elements of G_2 using triominoes, belonging to T_2 , exists and tiles g_2 .

Therefore, the diagram above illustrates an instance of that direct proof.

(d) Given the statement to prove: $\forall n \in \mathbb{N}, P(n)$, which for each natural n you can tile any $2^n \times 2^n$ grid with one cell missing using only triominoes.

Proof: We prove this by Simple Induction on n.

Base Case: Let $0 \leq n \leq 1$.

Since G_0 is the set of $2^0 \times 2^0$ grid with one cell removed, which G_0 does not contain any grid, meaning there does not exist $g_0 \in G_0$, which the statement $\forall g_0 \in G_0, \exists t_0 \in T_0, t_0 \text{ tiles } g_0$ is vacuously True.

Since G_1 is the set of all $2^1 \times 2^1$ grids with one cell removed, which by definition is a single triominoe.

Therefore, $\forall g_1 \in G_1, \exists t_1 \in T_1, t_1 \text{ tiles } g_1$ is true, which P(1) is true.

Induction Step: Let $k \in \mathbb{N}$.

Induction Hypothesis: Assume that $P(k)$ is true.

By Induction Hypothesis, we know that $P(k)$ is true, which $\forall g_k \in G_k, \exists t_k \in T_k, t_k \text{ tiles } g_k$ is true. I will take 3 different g_k s, the first with right bottom corner square missing, the second with right top corner square missing, and the third with left top corner square missing. I will make the missing corners in these 3 g_k s face inwards and add a triomino

which will result in getting a ‘L’ shape. The remaining $\frac{1}{4}$ place is missing a cell to form a g_{k+1} , which can actually be an arbitrary element from G_k . By Induction Hypothesis, since $\forall g_k \in G_k, \exists t_k \in T_k, t_k \text{ tiles } g_k$ is true, the remaining G_k place can be covered by trimonoos, proving the $P(k+1)$ is true.

Therefore, we’ve proved $\forall n \in \mathbb{N}, P(n)$ is true.



2 Question 2

```

1  from typing import Any
2
3
4  3 usages    ▸ Henry-wxq +1
5  def q_2(n: int, x: Any) -> Any:
6      """Implement a Python function with parameters x and n that (ignoring floating-point issues) returns c_n.
7
8      Precondition:
9      1. x represents a non-zero real number.
10     2. n is a natural number
11     """
12     # Since c_1 is used in both n = 1 and recursion, I will put it at the front.
13     c_1 = x + 1 / x
14
15     if n == 1:
16         # From the definition of c_n, when n is 1, return the corresponding c_1
17         return c_1
18     elif n == 2:
19         # From the definition of c_n, when n is 2, return the corresponding c_2
20         c_2 = x * x + (1 / x) * (1 / x)
21         return c_2
22     else:
23         """This is the recursion part. According to the discovery from hint which will be stated below, I come up with a
24         general function for c_n.
25         """
26         # Aim at returning the recursive value of c for n minus 1 after reaching the case when n equals to 1.
27         c_minus1 = q_2(n-1, x)
28         # Aim at returning the recursive value of c for n minus 2. Since we don't know whether n is an even number or
29         # an odd number, we need to add both n equals to 1 and n equals to 2 to our base case.
30         c_minus2 = q_2(n-2, x)
31         # Calculate the c_n based on the discovery.
32         c_n = c_1 * c_minus1 - c_minus2
33         return c_n

```

Figure 1: Python function for Q2-a

- (a) The above code is the Python function with parameter x and n that (ignoring floating-point issues) returns c_n , the comments are both in the code above and below.

Firstly, I clearly stated the pre-conditions on x and n in a header comment, which $x \in \mathbb{R}/\{0\}$ and $n \in \mathbb{N}$.

Secondly, at line 12, I write the calculation of c_1 because it will be used in both 'if' statement at line 14 and 'else' statement at line 21, avoiding redundancy.

Thirdly, I implemented the based case when n equals to 0 and n equals to 1 according to the definition of c_n .

Fourthly, I implemented the recursion based on the discovery from hint.

$$\begin{aligned}
 \left(x + \frac{1}{x}\right) \cdot \left(x^n + \frac{1}{x^n}\right) &= x^{n+1} + \frac{1}{x^{n-1}} + x^{n-1} + \frac{1}{x^{n+1}} \\
 &= \left(x^{n+1} + \frac{1}{x^{n+1}}\right) + \left(x^{n-1} + \frac{1}{x^{n-1}}\right)
 \end{aligned}$$

According to the definition of c_n , gives:

$$\begin{aligned} c_1 \cdot c_n &= c_{n+1} + c_{n-1} \\ \implies c_{n+1} &= c_1 \cdot c_n - c_{n-1} \end{aligned}$$

Thus, we generalize the above equation into: $c_n = c_1 \cdot c_{n-1} - c_{n-2}$, which is the core of our recursive part, at line 31.

Fifthly, aiming at returning the recursive value of c_{n-1} after reaching the case when n equals to 1, I write the code line 26. Aiming at return the recursive value of c_{n-2} , I write the code at line 29. Since we don't know whether n is an even number or an odd number, we need to add both n equals to 1 and n equals to 2 to our base case at line 16 and at line 19.

Finally, we can obtain the c_n using the recursive function without use any loops, or any helper functions, nor call any exponentiation functions.

- (b) To state a recurrence for the sequence c , I will start from $n = 0$, which $c_0 = x^0 + \frac{1}{x^0} = 2$. Then I will go to $n = 1$, which $c_1 = x + \frac{1}{x}$.

Moreover, for $n \geq 2$, from the hint, we have:

$$\begin{aligned} \left(x + \frac{1}{x}\right) \cdot \left(x^n + \frac{1}{x^n}\right) &= x^{n+1} + \frac{1}{x^{n-1}} + x^{n-1} + \frac{1}{x^{n+1}} \\ &= \left(x^{n+1} + \frac{1}{x^{n+1}}\right) + \left(x^{n-1} + \frac{1}{x^{n-1}}\right) \end{aligned}$$

According to the definition of c_n , gives:

$$\begin{aligned} c_1 \cdot c_n &= c_{n+1} + c_{n-1} \\ \implies c_{n+1} &= c_1 \cdot c_n - c_{n-1} \end{aligned}$$

Thus, we generalize the above equation into: $c_n = c_1 \cdot c_{n-1} - c_{n-2}$.

Therefore, we have:

$$c_n = \begin{cases} 2 & \text{for } n = 0 \\ x + \frac{1}{x} & \text{for } n = 1 \\ c_1 \cdot c_{n-1} - c_{n-2} & \text{for } n \geq 2 \end{cases}$$

- (c) If $x + \frac{1}{x}$ is an integer, then $x^n + \frac{1}{x^n}$ is an integer for each $n \in \mathbb{N}$

Assume $x + \frac{1}{x}$ is an integer.

Given statement to prove: $\forall n \in \mathbb{N}$, $P(n)$, which $P(n)$: $x^n + \frac{1}{x^n}$ is an integer where $x \in \mathbb{R}/\{0\}$

Proof: We prove this by complete induction on n .

Base Case: Let $0 \leq n \leq 1$

For $n = 0$, we have $x^0 + \frac{1}{x^0} = 2$ is an integer, which $P(0)$ is True.

By assumption, $x + \frac{1}{x}$ is an integer, which $P(1)$ is True.

We've proved that $P(0)$ & $P(1)$ is true.

Induction Step: Let $n > 1$

Induction Hypothesis: Assume $\forall k, 1 \leq k < n, P(k)$

WTS: $P(n)$

From induction hypothesis, when $k_1 = 1, 1 \leq k_1 < n, P(1)$ is true, which $x + \frac{1}{x}$ is an integer.

From induction hypothesis, when $k_{n-1} = n - 1, 1 \leq k_{n-1} < n, P(n - 1)$ is true, which $x^{n-1} + \frac{1}{x^{n-1}}$ is an integer.

From induction hypothesis, when $k_{n-2} = n - 2, 1 \leq k_{n-2} < n, P(n - 2)$ is true, which $x^{n-2} + \frac{1}{x^{n-2}}$ is an integer.

Thus, we obtain that

$$\begin{aligned} & \left(x + \frac{1}{x}\right) \cdot \left(x^{n-1} + \frac{1}{x^{n-1}}\right) - \left(x^{n-2} + \frac{1}{x^{n-2}}\right) \text{ is an integer.} \\ &= x^n + \frac{1}{x^{n-2}} + x^{n-2} + \frac{1}{x^n} - x^{n-2} - \frac{1}{x^{n-2}} \\ &= x^n + \frac{1}{x^n} \text{ is an integer} \end{aligned}$$

I've proved that $P(n)$ is true.

To conclude, I've proved that if $x + \frac{1}{x}$ is an integer, then $x^n + \frac{1}{x^n}$ is an integer for each $n \in \mathbb{N}$.

■

3 Question 3

(a)

```

4 def q3_a_func(n: int) -> int:
5     """Implement a Python function that takes a positive natural number n and returns a_n.
6
7     Precondition: n is a positive natural number
8     """
9     if n == 1:
10         # From the definition of a_n, when n is 1, return a_n equals to 1.
11         return 1
12     else:
13         """ This is the recursion. Aim at returning the recursive value of a_n after reaching the base case when
14         a_n equals to 1.
15         """
16         return q3_a_func(math.floor(math.sqrt(n))) * q3_a_func(math.floor(math.sqrt(n))) \
17             + 2 * q3_a_func(math.floor(math.sqrt(n)))

```

Figure 2: Python function for Q3-a

(b)

```

20 def q3_b_func(n: int) -> int:
21     """Implement a Python function that takes a positive natural number n and raises an exception if n is 1, otherwise
22     it returns a_n.
23
24     Precondition: n is a positive natural number
25     """
26     if n == 1:
27         # By question requirement, when n is 1, raises an Exception.
28         raise Exception("Sorry, n must be greater than 1")
29     elif n == 2 or n == 3:
30         """Since when n equals to 2 or n equals to 3, the floor of square root of n is 1, and, in this function, we
31         don't have the value of a_n when n equals 1. Thus, we need to manually add the value of a_n when n equals to 2
32         and n equals to 3 to prevent the error when calling the recursive.
33         """
34         return 3
35     else:
36         """This is the recursion. Aim at returning the recursive value of a_n after reaching the case when a_n equals
37         to 2 or a_n equals to 3.
38         """
39         return q3_b_func(math.floor(math.sqrt(n))) * q3_b_func(math.floor(math.sqrt(n))) \
40             + 2 * q3_b_func(math.floor(math.sqrt(n)))

```

Figure 3: Python function for Q3-b

- (c) When $n_0 = 2$, n_0 is the smallest natural n_0 so that a_n is a multiple of 3 for each natural $n \geq n_0$.

Given statement to prove: $\forall n \in \mathbb{N}, n \geq n_0, P(n)$, which $P(n) : a_n$ is a multiple of 3.

Let $n \in \mathbb{N}$.

Proof: We prove this by complete induction on n .

Base Case: Let $2 \leq n < 4$.

$$\begin{aligned}
 P(2) : a_2 &= (a_{\lfloor \sqrt{2} \rfloor})^2 + 2 \cdot a_{\lfloor \sqrt{2} \rfloor} \\
 &= a_1^2 + 2 \cdot a_1 = 1^2 + 2 = 3 \text{ is a multiple of 3.}
 \end{aligned}$$

$$\begin{aligned}
 P(3) : a_3 &= (a_{\lfloor \sqrt{3} \rfloor})^2 + 2 \cdot a_{\lfloor \sqrt{3} \rfloor} \\
 &= a_1^2 + 2 \cdot a_1 = 1^2 + 2 = 3 \text{ is a multiple of 3.}
 \end{aligned}$$

Thus, I've proved the base case is true.

Induction Step: Let $n \geq 4$.

Induction Hypothesis: Assume $\forall k, 2 \leq k < n, P(k)$

Since $n \geq 4$, gives $\lfloor \sqrt{n} \rfloor < n$.

Since $\lfloor \sqrt{n} \rfloor < n$ and $4 \leq n$, gives $2 \leq \lfloor \sqrt{n} \rfloor$ as 2 is the smallest value of $\lfloor \sqrt{n} \rfloor$, which gives,

$$2 \leq \lfloor \sqrt{n} \rfloor < n$$

Since $\lfloor \sqrt{n} \rfloor$ is an integer which $\lfloor \sqrt{n} \rfloor \geq 2$, from induction hypothesis, we can always find $k' = \lfloor \sqrt{n} \rfloor$, which $P(k')$ is true and $a_{k'} = 3p, p \in \mathbb{N}$.

Thus gives,

$$\begin{aligned}
 a_n &= (\lfloor \sqrt{n} \rfloor)^2 + 2 \cdot a_{\lfloor \sqrt{n} \rfloor} \\
 &= (a_{k'})^2 + 2 \cdot a_{k'} \\
 &= (3p)^2 + 2 \cdot (3 \cdot p) \\
 &= 9 \cdot p^2 + 6 \cdot p \\
 &= 3 \cdot (3 \cdot p^2 + 2p)
 \end{aligned}$$

Let $q = 3 \cdot p^2 + 2 \cdot p$. Since $p \in \mathbb{N}$, gives $q \in \mathbb{N}$, which

$$a_n = 3q, q \in \mathbb{N}, \text{ where } a_n \text{ is a multiple of 3.}$$

I've proved that $P(n)$ is true.

To conclude, I've proved when $n_0 = 2$, n_0 is the smallest natural n_0 so that a_n is a multiple of 3 for each natural $n \geq n_0$.

■