Well-Behaved Solutions to Hofstadter-Like Recurrences

Nathan Fox

Rutgers University, Hill Center for the Mathematical Sciences, 110 Frelinghuysen Rd., Piscataway, NJ 08854-8019, USA, fox@math.rutgers.edu, http://www.rutgers.edu/~nhf12

March 5, 2016

Presentation at AMS Spring 2016 Southeastern Sectional Meeting, Special Session on Experimental Mathematics
Outline

1. The Hofstadter $Q$-Sequence
2. Well-Behaved Sequences
3. Our Method
4. Findings/Future Work
The Hofstadter Q-Sequence

Well-Behaved Sequences

Our Method

Findings/Future Work

References

1. The Hofstadter Q-Sequence
2. Well-Behaved Sequences
3. Our Method
4. Findings/Future Work
The Hofstadter Q-Sequence

- Formulated by Hofstadter in 1963
The Hofstadter Q-Sequence

- Formulated by Hofstadter in 1963
- Recurrence: \( Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2)) \)
The Hofstadter $Q$-Sequence

- Formulated by Hofstadter in 1963
- Recurrence: $Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2))$
- Initial Conditions: $Q(1) = Q(2) = 1$
The Hofstadter $Q$-Sequence

• Formulated by Hofstadter in 1963
• Recurrence: $Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2))$
• Initial Conditions: $Q(1) = Q(2) = 1$
• Sample Calculation:
The Hofstadter \( Q \)-Sequence

- Formulated by Hofstadter in 1963
- Recurrence: \( Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2)) \)
- Initial Conditions: \( Q(1) = Q(2) = 1 \)
- Sample Calculation:

\[
Q(3) = Q(3 - Q(2)) + Q(3 - Q(1))
\]
The Hofstadter $Q$-Sequence

- Formulated by Hofstadter in 1963
- Recurrence: $Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2))$
- Initial Conditions: $Q(1) = Q(2) = 1$
- Sample Calculation:

\[
Q(3) = Q(3 - Q(2)) + Q(3 - Q(1))
= Q(3 - 1) + Q(3 - 1)
\]
The Hofstadter $Q$-Sequence

- Formulated by Hofstadter in 1963
- Recurrence: $Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2))$
- Initial Conditions: $Q(1) = Q(2) = 1$
- Sample Calculation:

$$Q(3) = Q(3 - Q(2)) + Q(3 - Q(1))$$
$$= Q(3 - 1) + Q(3 - 1)$$
$$= Q(2) + Q(2)$$
The Hofstadter Q-Sequence

- Formulated by Hofstadter in 1963
- Recurrence: $Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2))$
- Initial Conditions: $Q(1) = Q(2) = 1$
- Sample Calculation:

$$Q(3) = Q(3 - Q(2)) + Q(3 - Q(1))$$
$$= Q(3 - 1) + Q(3 - 1)$$
$$= Q(2) + Q(2)$$
$$= 1 + 1 = 2$$

First few terms (OEIS A005185):
1, 1, 2, 3, 3, 4, 5, 5, 6, 6, 6, 8, 8, 8, 10, 9, 10, 11, 11, 12, 12, 12, 12, 16, 14, 14, 16, 16, 16, 16, 20, 17, 17, 20, 21, 19, 20
The Hofstadter \(Q\)-Sequence

- Formulated by Hofstadter in 1963
- Recurrence: \(Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2))\)
- Initial Conditions: \(Q(1) = Q(2) = 1\)
- Sample Calculation:

\[
Q(3) = Q(3 - Q(2)) + Q(3 - Q(1)) \\
= Q(3 - 1) + Q(3 - 1) \\
= Q(2) + Q(2) \\
= 1 + 1 = 2
\]

First few terms (OEIS A005185):
1, 1, 2, 3, 3, 4, 5, 5, 6, 6, 6, 8, 8, 8, 10, 9, 10, 11, 11, 12, 12, 12, 12, 16, 14, 14, 16, 16, 16, 16, 20, 17, 17, 20, 21, 19, 20
The Hofstadter Q-Sequence

Plot of First 10000 Terms (from OEIS)
The Hofstadter Q-Sequence

What is known?

- In general, very little
The Hofstadter $Q$-Sequence

What is known?

- In general, very little
- The pattern seen in the plot seems to continue
The Hofstadter $Q$-Sequence

What is known?

- In general, very little
- The pattern seen in the plot seems to continue
- If $\lim_{n \to \infty} \frac{Q(n)}{n}$ exists, it equals $\frac{1}{2}$
What is known?

- In general, very little
- The pattern seen in the plot seems to continue
- If $\lim_{n \to \infty} \frac{Q(n)}{n}$ exists, it equals $\frac{1}{2}$
- Well-defined for the first $10^{10}$ terms

Nathan Fox

Well-Behaved Solutions to Hofstadter-Like Recurrences
The Hofstadter Q-Sequence

What is known?
- In general, very little
- The pattern seen in the plot seems to continue
- If \( \lim_{n \to \infty} \frac{Q(n)}{n} \) exists, it equals \( \frac{1}{2} \)
- Well-defined for the first \( 10^{10} \) terms

How could \( Q(n) \) be undefined?
- What if \( Q(n - 1) \geq n? \)
The Hofstadter Q-Sequence

What is known?

- In general, very little
- The pattern seen in the plot seems to continue
- If $\lim_{n \to \infty} \frac{Q(n)}{n}$ exists, it equals $\frac{1}{2}$
- Well-defined for the first $10^{10}$ terms

How could $Q(n)$ be undefined?

- What if $Q(n - 1) \geq n$?
  
  Then, $Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2))$, but $Q(n - Q(n - 1))$ is $Q$ of a nonpositive number!
The Hofstadter $Q$-Sequence

What is known?

- In general, very little
- The pattern seen in the plot seems to continue
- If $\lim_{n \to \infty} \frac{Q(n)}{n}$ exists, it equals $\frac{1}{2}$
- Well-defined for the first $10^{10}$ terms

How could $Q(n)$ be undefined?

- What if $Q(n-1) \geq n$?
- Then, $Q(n) = Q(n - Q(n-1)) + Q(n - Q(n-2))$, but $Q(n - Q(n-1))$ is $Q$ of a nonpositive number!
- If this happens, we say the sequence dies at $n$. 
The Hofstadter Q-Sequence

What is known?
- In general, very little
- The pattern seen in the plot seems to continue
- If \( \lim_{n \to \infty} \frac{Q(n)}{n} \) exists, it equals \( \frac{1}{2} \)
- Well-defined for the first \( 10^{10} \) terms

How could \( Q(n) \) be undefined?
- What if \( Q(n - 1) \geq n \)?
- Then, \( Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2)) \), but \( Q(n - Q(n - 1)) \) is \( Q \) of a nonpositive number!
- If this happens, we say the sequence dies at \( n \).
- Open: Does \( (Q(n))_{n \geq 1} \) die?
Cheating Death

Convention: If $n \leq 0$, then $Q(n) = 0$
Cheating Death

- Convention: If $n \leq 0$, then $Q(n) = 0$
- Allows us to consider solutions that grow faster than $n$
Cheating Death

- Convention: If $n \leq 0$, then $Q(n) = 0$
- Allows us to consider solutions that grow faster than $n$
- Original Hofstadter question can still be asked as: Does $Q(n - 1)$ ever exceed $n$?
Cheating Death

- Convention: If $n \leq 0$, then $Q(n) = 0$
- Allows us to consider solutions that grow faster than $n$
- Original Hofstadter question can still be asked as: Does $Q(n-1)$ ever exceed $n$?
- Sequence can still die: If $Q(n-1) \leq 0$, then $Q(n)$ undefined.
1 The Hofstadter Q-Sequence

2 Well-Behaved Sequences

3 Our Method

4 Findings/Future Work
Golomb's Sequence

- Discovered by Golomb around 1990
Golomb’s Sequence

- Discovered by Golomb around 1990
- Same recurrence as Hofstadter:
  \[ Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2)) \]
Golomb’s Sequence

- Discovered by Golomb around 1990
- Same recurrence as Hofstadter:
  \[ Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2)) \]
- Initial Conditions: \( Q(1) = 3, \ Q(2) = 2, \ Q(3) = 1 \)

First few terms (A244477):
Golomb’s Sequence

- Discovered by Golomb around 1990
- Same recurrence as Hofstadter:
  \[ Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2)) \]
- Initial Conditions: \( Q(1) = 3, \ Q(2) = 2, \ Q(3) = 1 \)

First few terms (A244477):
Golomb’s Sequence

- Discovered by Golomb around 1990
- Same recurrence as Hofstadter:
  \[ Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2)) \]
- Initial Conditions: \( Q(1) = 3, \ Q(2) = 2, \ Q(3) = 1 \)

First few terms (A244477):

**Formula**

- \( Q(3n) = 3n - 2 \)
- \( Q(3n + 1) = 3 \)
- \( Q(3n + 2) = 3n + 2 \)
Ruskey’s Sequence

- Discovered by Ruskey in 2011
Ruskey’s Sequence

- Discovered by Ruskey in 2011
- Same recurrence as Hofstadter:
  \[ Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2)) \]
Ruskey’s Sequence

- Discovered by Ruskey in 2011
- Same recurrence as Hofstadter:
  \[ Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2)) \]
- Initial Conditions: \( Q(1) = 3, \ Q(2) = 6, \ Q(3) = 5, \ Q(4) = 3, \ Q(5) = 6, \ Q(6) = 8 \) (and \( Q(n) = 0 \) if \( n \leq 0 \))
Ruskey’s Sequence

- Discovered by Ruskey in 2011
- Same recurrence as Hofstadter:
  \[ Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2)) \]
- Initial Conditions: \( Q(1) = 3, \ Q(2) = 6, \ Q(3) = 5, \ Q(4) = 3, \ Q(5) = 6, \ Q(6) = 8 \) (and \( Q(n) = 0 \) if \( n \leq 0 \))

First few terms (A188670):
3, 6, 5, 3, 6, 8, 3, 6, 13, 3, 6, 21, 3, 6, 34, 3, 6, 55, 3, 6, 89, 3, 6, 144, 3, 6, 233, 3, 6, 377, 3, 6, 610, 3, 6, 987, 3, 6, 1597
Ruskey’s Sequence

- Discovered by Ruskey in 2011
- Same recurrence as Hofstadter:
  \[ Q(n) = Q(n - Q(n - 1)) + Q(n - Q(n - 2)) \]
- Initial Conditions: \( Q(1) = 3, Q(2) = 6, Q(3) = 5, Q(4) = 3, Q(5) = 6, Q(6) = 8 \) (and \( Q(n) = 0 \) if \( n \leq 0 \))

First few terms (A188670):
3, 6, 5, 3, 6, 8, 3, 6, 13, 3, 6, 21, 3, 6, 34, 3, 6, 55, 3, 6, 89, 3, 6, 144, 3, 6, 233, 3, 6, 377, 3, 6, 610, 3, 6, 987, 3, 6, 1597

**Formula**

- \( Q(3n) = F(n + 4) \)
- \( Q(3n + 1) = 3 \)
- \( Q(3n + 2) = 6 \)
What is a “Well-Behaved Sequence”? 

Linear-Recurrent Sequences

- Golomb’s and Ruskey’s solutions both eventually satisfy linear recurrences.
What is a “Well-Behaved Sequence”?

**Linear-Recurrent Sequences**

- Golomb’s and Ruskey’s solutions both eventually satisfy linear recurrences.
- Want to say that any linear-recurrent sequence is well-behaved.
What is a “Well-Behaved Sequence”?

**Linear-Recurrent Sequences**

- Golomb’s and Ruskey’s solutions both eventually satisfy linear recurrences.
- Want to say that any linear-recurrent sequence is well-behaved.
- Doesn’t seem entirely helpful for examining solutions to Hofstadter-like recurrences.
What is a “Well-Behaved Sequence”?

**Linear-Recurrent Sequences**

- Golomb’s and Ruskey’s solutions both eventually satisfy linear recurrences.
- Want to say that any linear-recurrent sequence is well-behaved.
- Doesn’t seem entirely helpful for examining solutions to Hofstadter-like recurrences.
- Restrict to a subset of linear-recurrent sequences that naturally includes Golomb’s and Ruskey’s sequences.
What is a “Well-Behaved Sequence”?

**Linear-Recurrent Sequences**

- Golomb’s and Ruskey’s solutions both eventually satisfy linear recurrences.
- Want to say that any linear-recurrent sequence is well-behaved.
- Doesn’t seem entirely helpful for examining solutions to Hofstadter-like recurrences.
- Restrict to a subset of linear-recurrent sequences that naturally includes Golomb’s and Ruskey’s sequences.

**Definition**

Given a sequence \((a_n)_{n \geq 1}\) and a positive integer \(m\), we say \((a_n)_{n \geq 1}\) is well-behaved with period \(m\) if the subsequences \((a_{mk+r})_{k \geq 1}\) are eventually nice for all integers \(0 \leq r < m\).
What is a “Nice Sequence”?

**Definition**

Given a sequence \((a_n)_{n \geq 1}\) and a positive integer \(m\), we say \((a_n)_{n \geq 1}\) is well-behaved with period \(m\) if the subsequences \((a_{mk+r})_{k \geq 1}\) are eventually **nice** for all integers \(0 \leq r < m\).

**Properties We Want**

- Polynomials should be nice
What is a “Nice Sequence”? 

Definition

Given a sequence \((a_n)_{n \geq 1}\) and a positive integer \(m\), we say \((a_n)_{n \geq 1}\) is well-behaved with period \(m\) if the subsequences \((a_{mk+r})_{k \geq 1}\) are eventually nice for all integers \(0 \leq r < m\).

Properties We Want

- Polynomials should be nice
- Sequences like the Fibonacci numbers should be nice
What is a “Nice Sequence”? 

**Definition**

Given a sequence \((a_n)_{n \geq 1}\) and a positive integer \(m\), we say \((a_n)_{n \geq 1}\) is well-behaved with period \(m\) if the subsequences \((a_{mk+r})_{k \geq 1}\) are eventually nice for all integers \(0 \leq r < m\).

**Properties We Want**

- Polynomials should be nice
- Sequences like the Fibonacci numbers should be nice

These conditions make Golomb’s and Ruskey’s sequences both well-behaved with period 3.
1. The Hofstadter Q-Sequence

2. Well-Behaved Sequences

3. Our Method

4. Findings/Future Work
General Setup

Goal

Find all well-behaved solutions for a fixed period \( m \).
General Setup

Goal

Find all well-behaved solutions for a fixed period \( m \).

- Recurrence of the form

\[
Q(n) = p(n) + \sum_{i=1}^{k} \alpha_i Q(\beta_i n - \gamma_i - \delta_i Q(n - \epsilon_i))
\]

- \( p(n) \) polynomial
- \( k \) positive integer
- \( \alpha_i, \beta_i, \gamma_i, \delta_i, \epsilon_i \) integers (with some restrictions)
General Setup

Goal

Find all well-behaved solutions for a fixed period $m$.

- Recurrence of the form

$$Q(n) = p(n) + \sum_{i=1}^{k} \alpha_i Q(\beta_in - \gamma_i - \delta_i Q(n - \epsilon_i))$$

- $p(n)$ polynomial
- $k$ positive integer
- $\alpha_i, \beta_i, \gamma_i, \delta_i, \epsilon_i$ integers (with some restrictions)

Running Example

$R(n) = R(n - R(n - 1)) + R(n - R(n - 2)) + R(n - R(n - 3))$ with period $m = 4$. 
Step 1: Guess Behaviors for the Subsequences

Relevant Subsequence Types

- \( Q(mk + r) = br \) for \( k \) sufficiently large (constant)
Step 1: Guess Behaviors for the Subsequences

Relevant Subsequence Types

- $Q(mk + r) = b_r$ for $k$ sufficiently large (constant)
- $Q(mk + r) = mk + b_r$ for $k$ sufficiently large (linear with slope 1)
Step 1: Guess Behaviors for the Subsequences

Relevant Subsequence Types

- \( Q(mk + r) = b_r \) for \( k \) sufficiently large (constant)
- \( Q(mk + r) = mk + b_r \) for \( k \) sufficiently large (linear with slope 1)
- \( \lim_{k \to \infty} \frac{Q(mk+r)}{mk} > 1 \) ("steep")
Step 1: Guess Behaviors for the Subsequences

**Relevant Subsequence Types**

- \( Q(mk + r) = b_r \) for \( k \) sufficiently large (constant)
- \( Q(mk + r) = mk + b_r \) for \( k \) sufficiently large (linear with slope 1)
- \( \lim_{k \to \infty} \frac{Q(mk+r)}{mk} > 1 \) ("steep")

Iterate through all \( 3^m \) possibilities.
Step 1: Guess Behaviors for the Subsequences

Relevant Subsequence Types

- $Q(mk + r) = b_r$ for $k$ sufficiently large (constant)
- $Q(mk + r) = mk + b_r$ for $k$ sufficiently large (linear with slope 1)
- $\lim_{k \to \infty} \frac{Q(mk+r)}{mk} > 1$ ("steep")

Iterate through all $3^m$ possibilities.

Running Example

Focus on case:
- $R(4k)$ steep
Step 1: Guess Behaviors for the Subsequences

Relevant Subsequence Types

- \( Q(mk + r) = b_r \) for \( k \) sufficiently large (constant)
- \( Q(mk + r) = mk + b_r \) for \( k \) sufficiently large (linear with slope 1)
- \( \lim_{k \to \infty} \frac{Q(mk+r)}{mk} > 1 \) ("steep")

Iterate through all \( 3^m \) possibilities.

Running Example

Focus on case:

- \( R(4k) \) steep
- \( R(4k + 1) \) linear with slope 1
Step 1: Guess Behaviors for the Subsequences

Relevant Subsequence Types

- $Q(mk + r) = b_r$ for $k$ sufficiently large (constant)
- $Q(mk + r) = mk + b_r$ for $k$ sufficiently large (linear with slope 1)
- $\lim_{k \to \infty} \frac{Q(mk+r)}{mk} > 1$ ("steep")

Iterate through all $3^m$ possibilities.

Running Example

Focus on case:

- $R(4k)$ steep
- $R(4k + 1)$ linear with slope 1
- $R(4k + 2)$ constant
Step 1: Guess Behaviors for the Subsequences

Relevant Subsequence Types

- \( Q(mk + r) = b_r \) for \( k \) sufficiently large (constant)
- \( Q(mk + r) = mk + b_r \) for \( k \) sufficiently large (linear with slope 1)
- \( \lim_{k \to \infty} \frac{Q(mk+r)}{mk} > 1 \) ("steep")

Iterate through all \( 3^m \) possibilities.

Running Example

Focus on case:

- \( R(4k) \) steep
- \( R(4k + 1) \) linear with slope 1
- \( R(4k + 2) \) constant
- \( R(4k + 3) \) constant
Step 2: Unpack the Recurrence

- Want to find a solution of the posited form and prove it by induction.
Step 2: Unpack the Recurrence

- Want to find a solution of the posited form and prove it by induction.
- No base case needed, since only need eventual solution.
Step 2: Unpack the Recurrence

- Want to find a solution of the posited form and prove it by induction.
- No base case needed, since only need eventual solution.
- Can substitute forms inductively into recurrences to remove nesting.

\[
R(4^k) = 4^k - b_3 - 1 + b_1 + R(4^k - b_2) + R(4 - b_1)
\]

\[
R(4^{k+1}) = 4^k + b_1
\]

\[
R(4^{k+2}) = R(2 - b_1) + b_3
\]

\[
R(4^{k+3}) = b_3 + R(3 - b_1)
\]
Step 2: Unpack the Recurrence

- Want to find a solution of the posited form and prove it by induction.
- No base case needed, since only need eventual solution.
- Can substitute forms inductively into recurrences to remove nesting.
- Find we need to fix congruence classes mod $m$ of some $b_r$’s. (Iterate through all possibilities.)
Step 2: Unpack the Recurrence

- Want to find a solution of the posited form and prove it by induction.
- No base case needed, since only need eventual solution.
- Can substitute forms inductively into recurrences to remove nesting.
- Find we need to fix congruence classes mod $m$ of some $b_r$’s. (Iterate through all possibilities.)

Running Example

Need to fix $b_2$ and $b_3$ mod 4. Focus on case $b_2 ≡ 0 \pmod{4}$ and $b_3 ≡ 3 \pmod{4}$:
Step 2: Unpack the Recurrence

- Want to find a solution of the posited form and prove it by induction.
- No base case needed, since only need eventual solution.
- Can substitute forms inductively into recurrences to remove nesting.
- Find we need to fix congruence classes mod $m$ of some $b_r$'s. (Iterate through all possibilities.)

Running Example

Need to fix $b_2$ and $b_3$ mod 4. Focus on case $b_2 \equiv 0 \pmod{4}$ and $b_3 \equiv 3 \pmod{4}$:

- $R(4k) = 4k - b_3 - 1 + b_1 + R(4k - b_2) + R(4 - b_1)$
Step 2: Unpack the Recurrence

- Want to find a solution of the posited form and prove it by induction.
- No base case needed, since only need eventual solution.
- Can substitute forms inductively into recurrences to remove nesting.
- Find we need to fix congruence classes mod $m$ of some $b_r$’s. (Iterate through all possibilities.)

Running Example

Need to fix $b_2$ and $b_3$ mod 4. Focus on case $b_2 \equiv 0 \pmod{4}$ and $b_3 \equiv 3 \pmod{4}$:

- $R(4k) = 4k - b_3 - 1 + b_1 + R(4k - b_2) + R(4 - b_1)$
- $R(4k + 1) = 4k + b_1$
Step 2: Unpack the Recurrence

- Want to find a solution of the posited form and prove it by induction.
- No base case needed, since only need eventual solution.
- Can substitute forms inductively into recurrences to remove nesting.
- Find we need to fix congruence classes mod $m$ of some $b_r$’s. (Iterate through all possibilities.)

Running Example

Need to fix $b_2$ and $b_3$ mod 4. Focus on case $b_2 \equiv 0 \pmod{4}$ and $b_3 \equiv 3 \pmod{4}$:

- $R(4k) = 4k - b_3 - 1 + b_1 + R(4k - b_2) + R(4 - b_1)$
- $R(4k + 1) = 4k + b_1$
- $R(4k + 2) = R(2 - b_1) + b_3$
Step 2: Unpack the Recurrence

- Want to find a solution of the posited form and prove it by induction.
- No base case needed, since only need eventual solution.
- Can substitute forms inductively into recurrences to remove nesting.
- Find we need to fix congruence classes mod $m$ of some $b_r$’s. (Iterate through all possibilities.)

Running Example

Need to fix $b_2$ and $b_3$ mod 4. Focus on case $b_2 \equiv 0 \pmod{4}$ and $b_3 \equiv 3 \pmod{4}$:

- $R(4k) = 4k - b_3 - 1 + b_1 + R(4k - b_2) + R(4 - b_1)$
- $R(4k + 1) = 4k + b_1$
- $R(4k + 2) = R(2 - b_1) + b_3$
- $R(4k + 3) = b_3 + R(3 - b_1)$
Step 2: Unpack the Recurrence

\[ b_2 \equiv 0 \pmod{4} \text{ and } b_3 \equiv 3 \pmod{4} \]

Running Example: Unpacking \( R(4k) \)

\[
R(4k) = R(4k - R(4k - 1)) + R(4k - R(4k - 2)) \\
+ R(4k - R(4k - 3))
\]
Step 2: Unpack the Recurrence

\[ b_2 \equiv 0 \pmod{4} \text{ and } b_3 \equiv 3 \pmod{4} \]

Running Example: Unpacking \( R(4k) \)

\[
R(4k) = R(4k - R(4k - 1)) + R(4k - R(4k - 2)) \\
+ R(4k - R(4k - 3)) \\
= R(4k - b_3) + R(4k - b_2) + R(4k - (4(k - 1) + b_1))
\]
Step 2: Unpack the Recurrence

\[ b_2 \equiv 0 \pmod{4} \quad \text{and} \quad b_3 \equiv 3 \pmod{4} \]

Running Example: Unpacking \( R(4k) \)

\[
R(4k) = R(4k - R(4k - 1)) + R(4k - R(4k - 2)) + R(4k - R(4k - 3))
\]
\[
= R(4k - b_3) + R(4k - b_2) + R(4k - (4(k - 1) + b_1))
\]
\[
= 4k - b_3 - 1 + b_1 + R(4k - b_2) + R(4 - b_1)
\]
Step 2: Unpack the Recurrence

\[ b_2 \equiv 0 \pmod{4} \text{ and } b_3 \equiv 3 \pmod{4} \]

Running Example: Unpacking \( R(4k) \)

\[
R(4k) = R(4k - R(4k - 1)) + R(4k - R(4k - 2)) + R(4k - R(4k - 3)) \\
= R(4k - b_3) + R(4k - b_2) + R(4k - (4(k - 1) + b_1)) \\
= 4k - b_3 - 1 + b_1 + R(4k - b_2) + R(4 - b_1)
\]

Running Example: Unpacking \( R(4k + 3) \)

\[
R(4k + 3) = R(4k + 3 - R(4k + 2)) + R(4k + 3 - R(4k + 1)) + R(4k + 3 - R(4k))
\]
Step 2: Unpack the Recurrence

\[ b_2 \equiv 0 \pmod{4} \text{ and } b_3 \equiv 3 \pmod{4} \]

Running Example: Unpacking \( R(4k) \)

\[
R(4k) = R(4k - R(4k - 1)) + R(4k - R(4k - 2)) + R(4k - R(4k - 3))
\]
\[
= R(4k - b_3) + R(4k - b_2) + R(4k - (4(k - 1) + b_1))
= 4k - b_3 - 1 + b_1 + R(4k - b_2) + R(4 - b_1)
\]

Running Example: Unpacking \( R(4k + 3) \)

\[
R(4k + 3) = R(4k + 3 - R(4k + 2)) + R(4k + 3 - R(4k + 1)) + R(4k + 3 - R(4k))
\]
\[
= R(4k + 3 - b_2) + R(4k + 3 - (4k + b_1)) + 0
\]
Step 2: Unpack the Recurrence

\[ b_2 \equiv 0 \pmod{4} \text{ and } b_3 \equiv 3 \pmod{4} \]

Running Example: Unpacking \( R(4k) \)

\[
R(4k) = R(4k - R(4k - 1)) + R(4k - R(4k - 2)) + R(4k - R(4k - 3))
= R(4k - b_3) + R(4k - b_2) + R(4k - (4(k - 1) + b_1))
= 4k - b_3 - 1 + b_1 + R(4k - b_2) + R(4 - b_1)
\]

Running Example: Unpacking \( R(4k + 3) \)

\[
R(4k + 3) = R(4k + 3 - R(4k + 2)) + R(4k + 3 - R(4k + 1)) + R(4k + 3 - R(4k))
= R(4k + 3 - b_2) + R(4k + 3 - (4k + b_1)) + 0
= b_3 + R(3 - b_1)
\]
Step 3: Structural Consistency

Needed Properties of $Q(mk + r)$ Expression

**Constant:** Expression should be constant.
Step 3: Structural Consistency

**Needed Properties of** $Q(mk + r)$ **Expression**

**Constant:** Expression should be constant.

**Linear with Slope 1:** Expression should be of the form $mk + c$. 

Running Example

$R(4k) = R(4k - b^2) + 4k - b^3 - 1 + b^1 + R(4 - b^1)$

Quadratic growth

$R(4k + 1) = 4k + b^1$  

$R(4k + 2) = R(2 - b^1) + b^3$  

$R(4k + 3) = b^3 + R(3 - b^1)$
Step 3: Structural Consistency

Needed Properties of \( Q(mk + r) \) Expression

- **Constant**: Expression should be constant.
- **Linear with Slope 1**: Expression should be of the form \( mk + c \).
- **Steep**: Expression should be neither of the above.
Step 3: Structural Consistency

Needed Properties of $Q(mk + r)$ Expression

- **Constant:** Expression should be constant.
- **Linear with Slope 1:** Expression should be of the form $mk + c$.
- **Steep:** Expression should be neither of the above.
  In this case, can determine order of growth.
Step 3: Structural Consistency

Needed Properties of $Q(mk + r)$ Expression

- **Constant:** Expression should be constant.
- **Linear with Slope 1:** Expression should be of the form $mk + c$.
- **Steep:** Expression should be neither of the above. In this case, can determine order of growth.

Running Example

- $R(4k) = R(4k - b_2) + 4k - b_3 - 1 + b_1 + R(4 - b_1)$
  - Quadratic growth
Step 3: Structural Consistency

Needed Properties of $Q(mk + r)$ Expression

**Constant:** Expression should be constant.

**Linear with Slope 1:** Expression should be of the form $mk + c$.

**Steep:** Expression should be neither of the above. In this case, can determine order of growth.

Running Example

- $R(4k) = R(4k - b_2) + 4k - b_3 - 1 + b_1 + R(4 - b_1)$
  - Quadratic growth
- $R(4k + 1) = 4k + b_1$
Step 3: Structural Consistency

Needed Properties of \( Q(mk + r) \) Expression

- **Constant:** Expression should be constant.
- **Linear with Slope 1:** Expression should be of the form \( mk + c \).
- **Steep:** Expression should be neither of the above. In this case, can determine order of growth.

Running Example

- \( R(4k) = R(4k - b_2) + 4k - b_3 - 1 + b_1 + R(4 - b_1) \)
  - Quadratic growth
- \( R(4k + 1) = 4k + b_1 \)
- \( R(4k + 2) = R(2 - b_1) + b_3 \)
Step 3: Structural Consistency

**Needed Properties of** $Q(mk + r)$ **Expression**

- **Constant**: Expression should be constant.
- **Linear with Slope 1**: Expression should be of the form $mk + c$.
- **Steep**: Expression should be neither of the above.
  In this case, can determine order of growth.

**Running Example**

- $R(4k) = R(4k - b_2) + 4k - b_3 - 1 + b_1 + R(4 - b_1)$
  - Quadratic growth
- $R(4k + 1) = 4k + b_1$
- $R(4k + 2) = R(2 - b_1) + b_3$
- $R(4k + 3) = b_3 + R(3 - b_1)$
Step 4: Formulating Constraints

Absolute Constraints

- If $Q(mk + r)$ constant, need $b_r > 0$. 
Step 4: Formulating Constraints

**Absolute Constraints**

- If $Q(mk + r)$ constant, need $b_r > 0$.
- If $Q(mk + r)$ constant, need its expression to equal $b_r$. 
Step 4: Formulating Constraints

**Absolute Constraints**

- If \( Q(mk + r) \) constant, need \( b_r > 0 \).
- If \( Q(mk + r) \) constant, need its expression to equal \( b_r \).
- If \( Q(mk + r) \) linear with slope 1, need \( b_r \) to equal the constant term in its expression.
Step 4: Formulating Constraints

### Absolute Constraints

- If $Q(mk + r)$ constant, need $b_r > 0$.
- If $Q(mk + r)$ constant, need its expression to equal $b_r$.
- If $Q(mk + r)$ linear with slope 1, need $b_r$ to equal the constant term in its expression.
- If $Q(mk + r)$ linear with slope greater than 1, need a steepness-enforcing constraint.
### Step 4: Formulating Constraints

#### Absolute Constraints

- If $Q(mk + r)$ constant, need $b_r > 0$.
- If $Q(mk + r)$ constant, need its expression to equal $b_r$.
- If $Q(mk + r)$ linear with slope 1, need $b_r$ to equal the constant term in its expression.
- If $Q(mk + r)$ linear with slope greater than 1, need a steepness-enforcing constraint.
- If $b_r$ was forced to have a certain congruence, need $b_r$ to have its assigned congruence.
Step 4: Formulating Constraints

Absolute Constraints

- If \( Q(mk + r) \) constant, need \( b_r > 0 \).
- If \( Q(mk + r) \) constant, need its expression to equal \( b_r \).
- If \( Q(mk + r) \) linear with slope 1, need \( b_r \) to equal the constant term in its expression.
- If \( Q(mk + r) \) linear with slope greater than 1, need a steepness-enforcing constraint.
- If \( b_r \) was forced to have a certain congruence, need \( b_r \) to have its assigned congruence.

Conditional Constraints

- \( Q(c) \) appears: \( c \leq 0 \Rightarrow Q(c) = 0 \).
Step 4: Formulating Constraints

**Absolute Constraints**

- If $Q(mk + r)$ constant, need $b_r > 0$.
- If $Q(mk + r)$ constant, need its expression to equal $b_r$.
- If $Q(mk + r)$ linear with slope 1, need $b_r$ to equal the constant term in its expression.
- If $Q(mk + r)$ linear with slope greater than 1, need a steepness-enforcing constraint.
- If $b_r$ was forced to have a certain congruence, need $b_r$ to have its assigned congruence.

**Conditional Constraints**

- $Q(c)$ appears: $c \leq 0 \Rightarrow Q(c) = 0$.
- $Q(c)$ and $Q(d)$ both appear: $c = d \Rightarrow Q(c) = Q(d)$.
Step 4: Formulating Constraints

Running Example: Absolute Constraints

- $R(4k) = 4k - b_3 - 1 + b_1 + R(4k - b_2) + R(4 - b_1)$
- No absolute constraints from here
Step 4: Formulating Constraints

Running Example: Absolute Constraints

- \( R(4k) = 4k - b_3 - 1 + b_1 + R(4k - b_2) + R(4 - b_1) \)
  - No absolute constraints from here
- \( R(4k + 1) = 4k + b_1 \)
  - \( b_1 = b_1 \)
Step 4: Formulating Constraints

Running Example: Absolute Constraints

- \( R(4k) = 4k - b_3 - 1 + b_1 + R(4k - b_2) + R(4 - b_1) \)
  - No absolute constraints from here
- \( R(4k + 1) = 4k + b_1 \)
  - \( b_1 = b_1 \)
- \( R(4k + 2) = R(2 - b_1) + b_3 \)
  - \( b_2 > 0 \)
  - \( b_2 = R(2 - b_1) + b_3 \)
  - \( b_2 \equiv 0 \pmod{4} \)
Step 4: Formulating Constraints

Running Example: Absolute Constraints

- \( R(4k) = 4k - b_3 - 1 + b_1 + R(4k - b_2) + R(4 - b_1) \)
  - No absolute constraints from here
- \( R(4k + 1) = 4k + b_1 \)
  - \( b_1 = b_1 \)
- \( R(4k + 2) = R(2 - b_1) + b_3 \)
  - \( b_2 > 0 \)
  - \( b_2 = R(2 - b_1) + b_3 \)
  - \( b_2 \equiv 0 \pmod{4} \)
- \( R(4k + 3) = b_3 + R(3 - b_1) \)
  - \( b_3 > 0 \)
  - \( b_3 = b_3 + R(3 - b_1) \)
  - \( b_3 \equiv 3 \pmod{4} \)
Step 4: Formulating Constraints

\[ R(2 - b_1), \ R(3 - b_1), \ \text{and} \ R(4 - b_1) \ \text{appear.} \]

**Running Example: Conditional Constraints**

- If \( 2 - b_1 \leq 0 \), then \( R(2 - b_1) = 0 \)
- If \( 3 - b_1 \leq 0 \), then \( R(3 - b_1) = 0 \)
- If \( 4 - b_1 \leq 0 \), then \( R(4 - b_1) = 0 \)
Step 4: Formulating Constraints

\[ R(2 - b_1), R(3 - b_1), \text{and } R(4 - b_1) \text{ appear.} \]

Running Example: Conditional Constraints

- If \( 2 - b_1 \leq 0 \), then \( R(2 - b_1) = 0 \)
- If \( 3 - b_1 \leq 0 \), then \( R(3 - b_1) = 0 \)
- If \( 4 - b_1 \leq 0 \), then \( R(4 - b_1) = 0 \)
- If \( 2 - b_1 = 3 - b_1 \), then \( R(2 - b_1) = R(3 - b_1) \)
- If \( 2 - b_1 = 4 - b_1 \), then \( R(2 - b_1) = R(4 - b_1) \)
- If \( 3 - b_1 = 4 - b_1 \), then \( R(3 - b_1) = R(4 - b_1) \)
Step 5: Satisfying Constraints

General Method

- Replace strict inequalities with loose ones, since variables are integers.
Step 5: Satisfying Constraints

**General Method**

- Replace strict inequalities with loose ones, since variables are integers.
- Add auxiliary variables to replace congruence constraints by equality constraints ($b_r \equiv s \pmod{m}$ becomes $b_r = mq_r + s$).
Step 5: Satisfying Constraints

General Method

- Replace strict inequalities with loose ones, since variables are integers.
- Add auxiliary variables to replace congruence constraints by equality constraints ($b_r \equiv s \pmod{m}$ becomes $b_r = mq_r + s$).
- Backtrack through conditional constraints.
Step 5: Satisfying Constraints

**General Method**

- Replace strict inequalities with loose ones, since variables are integers.
- Add auxiliary variables to replace congruence constraints by equality constraints ($b_r \equiv s \pmod{m}$ becomes $b_r = mq_r + s$).
- Backtrack through conditional constraints.
- Result is a linear integer program. Maple can solve these.
Step 5: Satisfying Constraints

General Method

- Replace strict inequalities with loose ones, since variables are integers.
- Add auxiliary variables to replace congruence constraints by equality constraints ($b_r \equiv s \pmod{m}$ becomes $b_r = mq_r + s$).
- Backtrack through conditional constraints.
- Result is a linear integer program. Maple can solve these.

Running Example: One Feasible Solution

- $b_1 = 0$
- $b_2 = 4$
- $b_3 = 3$
- $R(2) = 1 (= R(2 - b_1))$
- $R(3) = 0 (= R(3 - b_1))$
Step 6: Initial Condition

Constructing an Initial Condition

- Specific $Q$ values involved in constraints must satisfy those constraints.
Step 6: Initial Condition

Constructing an Initial Condition

- Specific $Q$ values involved in constraints must satisfy those constraints.
- What assumptions were made when unpacking?
  - Steep subsequences must be sufficiently large
  - Other subsequences must equal their eventual values
Step 6: Initial Condition

Constructing an Initial Condition

- Specific $Q$ values involved in constraints must satisfy those constraints.
- What assumptions were made when unpacking?
  - Steep subsequences must be sufficiently large
  - Other subsequences must equal their eventual values
- These give constraints that will be satisfied by sufficiently large $k$
  - Pick such a $k$ and populate the required values.
Step 6: Initial Condition

Running Example: One Initial Condition

**Specific values:** $R(2) = 1$ and $R(3) = 0$. 

$k = 2$ and $R(4) = 4$ satisfy the resulting constraints.

Resulting initial condition: 0, 1, 0, 4, 4, 4, 3.
Step 6: Initial Condition

### Running Example: One Initial Condition

- **Specific values:** $R(2) = 1$ and $R(3) = 0$.
- **$R(4k)$:** $R(4k - 1) = 3$, $R(4k - 2) = 4$, and $R(4k - 3) = 4k - 4$.
- **$R(4k + 1)$:** $R(4k) \geq 4k + 1$, $R(4k - 1) = 3$, and $R(4k - 2) = 4$.
- **$R(4k + 2)$:** $R(4k + 1) = 4k$, $R(4k) \geq 4k + 2$, and $R(4k - 1) = 3$.
- **$R(4k + 3)$:** $R(4k + 2) = 4$, $R(4k + 1) = 4k$, and $R(4k) \geq 4k + 3$. 

Need to use expression $R(4k) = R(4k - 3) + R(4k - 4) + R(4)$ to get useful info from inequalities.

$k = 2$ and $R(4) = 4$ satisfy the resulting constraints.

Resulting initial condition: 0, 1, 0, 4, 4, 4, 3.
Step 6: Initial Condition

Running Example: One Initial Condition

- **Specific values:** $R(2) = 1$ and $R(3) = 0$.
- $R(4k)$: $R(4k - 1) = 3$, $R(4k - 2) = 4$, and $R(4k - 3) = 4k - 4$.
- $R(4k + 1)$: $R(4k) \geq 4k + 1$, $R(4k - 1) = 3$, and $R(4k - 2) = 4$.
- $R(4k + 2)$: $R(4k + 1) = 4k$, $R(4k) \geq 4k + 2$, and $R(4k - 1) = 3$.
- $R(4k + 3)$: $R(4k + 2) = 4$, $R(4k + 1) = 4k$, and $R(4k) \geq 4k + 3$.

Need to use expression $R(4k) = R(4k - 3) + R(4k - 4) + R(4)$ to get useful info from inequalities.
Step 6: Initial Condition

Running Example: One Initial Condition

- **Specific values:** $R(2) = 1$ and $R(3) = 0$.
- $R(4k)$: $R(4k - 1) = 3$, $R(4k - 2) = 4$, and $R(4k - 3) = 4k - 4$.
- $R(4k + 1)$: $R(4k) \geq 4k + 1$, $R(4k - 1) = 3$, and $R(4k - 2) = 4$.
- $R(4k + 2)$: $R(4k + 1) = 4k$, $R(4k) \geq 4k + 2$, and $R(4k - 1) = 3$.
- $R(4k + 3)$: $R(4k + 2) = 4$, $R(4k + 1) = 4k$, and $R(4k) \geq 4k + 3$.
- Need to use expression $R(4k) = R(4k - 3) + R(4k - 4) + R(4)$ to get useful info from inequalities.
- $k = 2$ and $R(4) = 4$ satisfy the resulting constraints.
### Step 6: Initial Condition

**Running Example: One Initial Condition**

- **Specific values:** $R(2) = 1$ and $R(3) = 0$.
- **$R(4k)$:** $R(4k - 1) = 3$, $R(4k - 2) = 4$, and $R(4k - 3) = 4k - 4$.
- **$R(4k + 1)$:** $R(4k) \geq 4k + 1$, $R(4k - 1) = 3$, and $R(4k - 2) = 4$.
- **$R(4k + 2)$:** $R(4k + 1) = 4k$, $R(4k) \geq 4k + 2$, and $R(4k - 1) = 3$.
- **$R(4k + 3)$:** $R(4k + 2) = 4$, $R(4k + 1) = 4k$, and $R(4k) \geq 4k + 3$.

Need to use expression $R(4k) = R(4k - 3) + R(4k - 4) + R(4)$ to get useful info from inequalities.

- $k = 2$ and $R(4) = 4$ satisfy the resulting constraints.
- Resulting initial condition: $0, 1, 0, 4, 4, 4, 3$
Summary of Running Example

Recurrence:

\[ R(n) = R(n - R(n - 1)) + R(n - R(n - 2)) + R(n - R(n - 3)) \]
Summary of Running Example

- Recurrence:
  \[ R(n) = R(n - R(n - 1)) + R(n - R(n - 2)) + R(n - R(n - 3)) \]

- Initial Conditions: \[ R(1) = 0, \ R(2) = 1, \ R(3) = 0, \ R(4) = 4, \]
  \[ R(5) = 4, \ R(6) = 4, \ R(7) = 3 \] (and \( R(n) = 0 \) if \( n \leq 0 \))
Summary of Running Example

- Recurrence:
  \[ R(n) = R(n - R(n - 1)) + R(n - R(n - 2)) + R(n - R(n - 3)) \]

- Initial Conditions: \( R(1) = 0, R(2) = 1, R(3) = 0, R(4) = 4, R(5) = 4, R(6) = 4, R(7) = 3 \) (and \( R(n) = 0 \) if \( n \leq 0 \))

First few terms (A268368):
0, 1, 0, 4, 4, 4, 3, 12, 8, 4, 3, 24, 12, 4, 3, 40, 16, 4, 3, 60, 20, 4, 3, 84, 24, 4, 3, 112, 28, 4, 3, 144, 32, 4, 3, 180, 36, 4, 3, 220, 40, 4, 3
Summary of Running Example

- **Recurrence:**
  
  \[ R(n) = R(n - R(n - 1)) + R(n - R(n - 2)) + R(n - R(n - 3)) \]

- **Initial Conditions:** \( R(1) = 0, R(2) = 1, R(3) = 0, R(4) = 4, R(5) = 4, R(6) = 4, R(7) = 3 \) (and \( R(n) = 0 \) if \( n \leq 0 \))

First few terms (A268368):  
0, 1, 0, 4, 4, 4, 3, 12, 8, 4, 3, 24, 12, 4, 3, 40, 16, 4, 3, 60, 20, 4, 3, 84, 24, 4, 3, 112, 28, 4, 3, 144, 32, 4, 3, 180, 36, 4, 3, 220, 40, 4, 3

**Formula**

- \( R(4n) = 2n^2 + 2n \)
- \( R(4n + 1) = 4n \)
- \( R(4n + 2) = 4 \) (except \( R(2) = 1 \))
- \( R(4n + 3) = 3 \) (except \( R(3) = 0 \))
1. The Hofstadter Q-Sequence

2. Well-Behaved Sequences

3. Our Method

4. Findings/Future Work
Findings

Some Achievable Things

- Eventually quasi-quadratic solutions to Hofstadter’s $Q$-recurrence (e.g. A264757)
Findings

Some Achievable Things

- Eventually quasi-quadratic solutions to Hofstadter’s $Q$-recurrence (e.g. A264757)
- More generally, eventually quasipolynomial solutions to Hostadter’s $Q$-recurrence of all degrees (for cubic, see A264758)
Findings

Some Achievable Things

- Eventually quasi-quadratic solutions to Hofstadter’s $Q$-recurrence (e.g. A264757)
- More generally, eventually quasipolynomial solutions to Hostadter’s $Q$-recurrence of all degrees (for cubic, see A264758)
- Eventually quasilinear solutions to Hofstadter’s $Q$-recurrence where some subsequences have slope greater than 1 (e.g. A269328)
Findings

Some Achievable Things

- Eventually quasi-quadratic solutions to Hofstadter’s $Q$-recurrence (e.g. A264757)
- More generally, eventually quasipolynomial solutions to Hostadter’s $Q$-recurrence of all degrees (for cubic, see A264758)
- Eventually quasilinear solutions to Hofstadter’s $Q$-recurrence where some subsequences have slope greater than 1 (e.g. A269328)
  - Obtained via recurrences
Findings

Some Achievable Things

- Eventually quasi-quadratic solutions to Hofstadter’s $Q$-recurrence (e.g. A264757)
- More generally, eventually quasipolynomial solutions to Hostadter’s $Q$-recurrence of all degrees (for cubic, see A264758)
- Eventually quasilinear solutions to Hofstadter’s $Q$-recurrence where some subsequences have slope greater than 1 (e.g. A269328)
  - Obtained via recurrences
  - Obtained by adding linear subsequences with slope 1
Findings

Some Achievable Things

- Eventually quasi-quadratic solutions to Hofstadter’s $Q$-recurrence (e.g. A264757)
- More generally, eventually quasipolynomial solutions to Hostadter’s $Q$-recurrence of all degrees (for cubic, see A264758)
- Eventually quasilinear solutions to Hofstadter’s $Q$-recurrence where some subsequences have slope greater than 1 (e.g. A269328)
  - Obtained via recurrences
  - Obtained by adding linear subsequences with slope 1
- Solutions displaying combinations of the above components
Findings

Some Achievable Things

- Eventually quasi-quadratic solutions to Hofstadter’s $Q$-recurrence (e.g. A264757)
- More generally, eventually quasipolynomial solutions to Hostadter’s $Q$-recurrence of all degrees (for cubic, see A264758)
- Eventually quasilinear solutions to Hofstadter’s $Q$-recurrence where some subsequences have slope greater than 1 (e.g. A269328)
  - Obtained via recurrences
  - Obtained by adding linear subsequences with slope 1
- Solutions displaying combinations of the above components
- Ability to embed a large class of linear-recurrent sequences in solutions to Hofstadter-like recurrences
Findings

Some Achievable Things

- Eventually quasi-quadratic solutions to Hofstadter’s $Q$-recurrence (e.g. A264757)
- More generally, eventually quasipolynomial solutions to Hofstadter’s $Q$-recurrence of all degrees (for cubic, see A264758)
- Eventually quasilinear solutions to Hofstadter’s $Q$-recurrence where some subsequences have slope greater than 1 (e.g. A269328)
  - Obtained via recurrences
  - Obtained by adding linear subsequences with slope 1
- Solutions displaying combinations of the above components
- Ability to embed a large class of linear-recurrent sequences in solutions to Hofstadter-like recurrences
- Periodic solutions to some Hofstadter-like recurrences
Findings

Algorithm gives infinite families of solutions based on period, growth rates of subsequences, and congruence classes of constants
Findings

Algorithm gives infinite families of solutions based on period, growth rates of subsequences, and congruence classes of constants

Solution Families for Hofstadter’s Q-recurrence

- Period 2: 2 families (1 modulo shifting)
Findings

Algorithm gives infinite families of solutions based on period, growth rates of subsequences, and congruence classes of constants.

Solution Families for Hofstadter’s $Q$-recurrence

- Period 2: 2 families (1 modulo shifting)
- Period 3: 12 families (4 modulo shifting):
  - The Golomb Family
  - The Ruskey Family
  - Two families with two constant subsequences and one linear subsequence (including A264756)
- Period 4: 12 families (5 modulo shifting), all quasilinear
- Period 5: 35 families (7 modulo shifting), all quasilinear, one steep
- Period 6: 294 families (86 modulo shifting), diverse behaviors, including quadratics and mixing of exponentials with steep linears
- Period 7: 588 families (84 modulo shifting), diverse behaviors
Findings

Algorithm gives infinite families of solutions based on period, growth rates of subsequences, and congruence classes of constants

Solution Families for Hofstadter’s Q-recurrence

- Period 2: 2 families (1 modulo shifting)
- Period 3: 12 families (4 modulo shifting):
  - The Golomb Family
- Period 4: 12 families (5 modulo shifting), all quasilinear
- Period 5: 35 families (7 modulo shifting), all quasilinear, one steep
- Period 6: 294 families (86 modulo shifting), diverse behaviors, including quadratics and mixing of exponentials with steep linears
- Period 7: 588 families (84 modulo shifting), diverse behaviors
Algorithm gives infinite families of solutions based on period, growth rates of subsequences, and congruence classes of constants

**Solution Families for Hofstadter’s Q-recurrence**

- Period 2: 2 families (1 modulo shifting)
- Period 3: 12 families (4 modulo shifting):
  - The Golomb Family
  - The Ruskey Family
- Period 4: 12 families (5 modulo shifting), all quasilinear
- Period 5: 35 families (7 modulo shifting), all quasilinear, one steep
- Period 6: 294 families (86 modulo shifting), diverse behaviors, including quadratics and mixing of exponentials with steep linears
- Period 7: 588 families (84 modulo shifting), diverse behaviors
Findings

Algorithm gives infinite families of solutions based on period, growth rates of subsequences, and congruence classes of constants

<table>
<thead>
<tr>
<th>Period</th>
<th>Number of Families</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2 families</td>
<td>(1 modulo shifting)</td>
</tr>
<tr>
<td>3</td>
<td>12 families</td>
<td>(4 modulo shifting):</td>
</tr>
<tr>
<td></td>
<td>The Golomb Family</td>
<td></td>
</tr>
<tr>
<td></td>
<td>The Ruskey Family</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Two families with two constant subsequences and one linear subsequence (including A264756)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>12 families</td>
<td>(5 modulo shifting), all quasilinear</td>
</tr>
<tr>
<td>5</td>
<td>35 families</td>
<td>(7 modulo shifting), all quasilinear, one steep</td>
</tr>
<tr>
<td>6</td>
<td>294 families</td>
<td>(86 modulo shifting), diverse behaviors, including quadratics and mixing of exponentials with steep linears</td>
</tr>
<tr>
<td>7</td>
<td>588 families</td>
<td>(84 modulo shifting), diverse behaviors</td>
</tr>
</tbody>
</table>
Findings

Algorithm gives infinite families of solutions based on period, growth rates of subsequences, and congruence classes of constants

Solution Families for Hofstadter’s $Q$-recurrence

- Period 2: 2 families (1 modulo shifting)
- Period 3: 12 families (4 modulo shifting):
  - The Golomb Family
  - The Ruskey Family
  - Two families with two constant subsequences and one linear subsequence (including A264756)
- Period 4: 12 families (5 modulo shifting), all quasilinear
Algorithm gives infinite families of solutions based on period, growth rates of subsequences, and congruence classes of constants

**Solution Families for Hofstadter’s $Q$-recurrence**

- **Period 2:** 2 families (1 modulo shifting)
- **Period 3:** 12 families (4 modulo shifting):
  - The Golomb Family
  - The Ruskey Family
  - Two families with two constant subsequences and one linear subsequence (including A264756)
- **Period 4:** 12 families (5 modulo shifting), all quasilinear
- **Period 5:** 35 families (7 modulo shifting), all quasilinear, one steep
Findings

Algorithm gives infinite families of solutions based on period, growth rates of subsequences, and congruence classes of constants

**Solution Families for Hofstadter’s Q-recurrence**

- **Period 2**: 2 families (1 modulo shifting)
- **Period 3**: 12 families (4 modulo shifting):
  - The Golomb Family
  - The Ruskey Family
  - Two families with two constant subsequences and one linear subsequence (including A264756)
- **Period 4**: 12 families (5 modulo shifting), all quasilinear
- **Period 5**: 35 families (7 modulo shifting), all quasilinear, one steep
- **Period 6**: 294 families (86 modulo shifting), diverse behaviors, including quadratics and mixing of exponentials with steep linears
Findings

Algorithm gives infinite families of solutions based on period, growth rates of subsequences, and congruence classes of constants

Solution Families for Hofstadter’s Q-recurrence

- Period 2: 2 families (1 modulo shifting)
- Period 3: 12 families (4 modulo shifting):
  - The Golomb Family
  - The Ruskey Family
  - Two families with two constant subsequences and one linear subsequence (including A264756)
- Period 4: 12 families (5 modulo shifting), all quasilinear
- Period 5: 35 families (7 modulo shifting), all quasilinear, one steep
- Period 6: 294 families (86 modulo shifting), diverse behaviors, including quadratics and mixing of exponentials with steep linears
- Period 7: 588 families (84 modulo shifting), diverse behaviors
Future Work

- More complicated nested recurrences
Future Work

- More complicated nested recurrences
  - More levels of nesting
Future Work

- More complicated nested recurrences
  - More levels of nesting
  - Nonlinearities

Nathan Fox
Well-Behaved Solutions to Hofstadter-Like Recurrences
Future Work

- More complicated nested recurrences
  - More levels of nesting
  - Nonlinearities
  - Non-constant number of terms
Future Work

- More complicated nested recurrences
  - More levels of nesting
  - Nonlinearities
  - Non-constant number of terms

- Tanny frequently studies slow solutions
Future Work

- More complicated nested recurrences
  - More levels of nesting
  - Nonlinearities
  - Non-constant number of terms
- Tanny frequently studies slow solutions
  - Try to automatically find/prove these
Future Work

- More complicated nested recurrences
  - More levels of nesting
  - Nonlinearities
  - Non-constant number of terms
- Tanny frequently studies slow solutions
  - Try to automatically find/prove these
  - Interleave them with nice sequences
I would like to thank my Ph.D. advisor Dr. Doron Zeilberger for introducing me to this area and providing me with feedback throughout my work.

I would also like to thank the session organizers, Dr. Frank Garvan and Dr. Andrew Sills, for inviting me to speak.


Thank you!