“Computational” & “Mathematical” Thinking:
Exploring Difference & Interdependence

Laura Broley, l_brole@live.concordia.ca

Introduction

Perhaps the simplest (albeit broadest) way to conceptualize computational and mathematical thinking is through the naturally corresponding professional communities of reference:

“Computational Thinking”
≈
“Thinking *like* a Computer Scientist”
(e.g., Wing, 2006)

”Mathematical Thinking”
≈
“Thinking *like* a Mathematician”
(e.g., Burton, 1984)

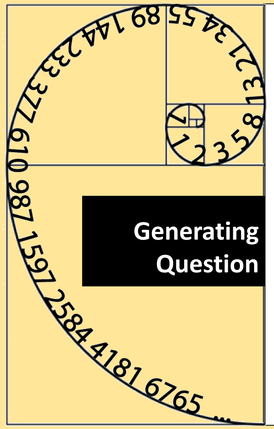
Such definitions may appear **impractically fluid**: computer scientists and mathematicians often think a lot alike.

However, the fluid boundaries also serve to highlight a significant point: mathematics and computer science are **mutually symbiotic**. Computer science could not exist without the past and current support of mathematics, just as much as mathematics must constantly renew itself in front of new problems proposed by computer science (Hodgson, 2010).

I had the enriching experience of exposure to localized examples of this symbiosis when completing assignments in a scientific computing course at Université de Montréal (MAT 6470, given by Dr. Anne Bourlioux). Reliving some of these specific examples has helped me to better understand not only the **potential difference**, but also the **resulting interdependence** of “Computational” and “Mathematical” Thinking.

It may also lead us to reflect on **(re)new(ed) ideas** and raise **(re)new(ed) questions** concerning the place of computational thinking in mathematics classrooms.

Reliving Assignment #1, MAT 6470, given by Dr. Anne Bourlioux (adapted from Goodman, 2002)



Generating Question

Fibonacci Numbers

$$f_0 = 1 = f_1$$
$$f_{k+1} = f_k + f_{k-1}$$
$$k = 1, 2, 3, 4, 5, \dots$$

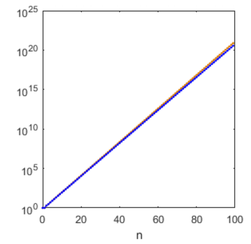
VS

Pibonacci Numbers

$$p_0 = 1 = p_1$$
$$p_{k+1} = cp_k + p_{k-1}$$
$$k = 1, 2, 3, 4, 5, \dots$$
$$c = 1 + \sqrt{3}/100$$

Suppose I hand you the 100th term in each sequence.

Could you reproduce the sequence in the reverse order, all the way back to $p_0 = f_0 = 1$?



The Pibonacci Numbers are just a small perturbation of the Fibonacci Numbers.

Matlab Code

fibo.m

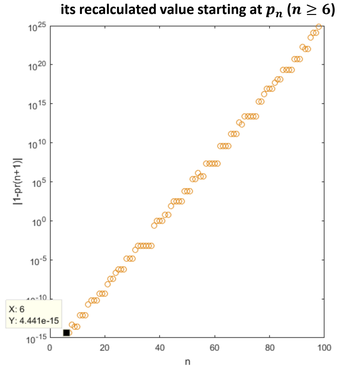
```
function
[fib,retour]=fibo(n)
% forward: calculate the
Fibonacci sequence (fib)
up to the nth term
fib(1)=1;
fib(2)=1;
for i=3:n;
fib(i)=fib(i-1)+fib(i-2);
end;
% backward: recalculate the
first term (retour) using
the recursion relation
starting at each term
retour(1)=1;
retour(2)=1;
for i=3:n;
fibr(i)=fib(i);fibr(i-1)=
fib(i-1);
for j=i-2:-1:1;
fibr(j)=fibr(j+2)-
fibr(j+1);
end;
retour(i)=fibr(1);
disp([i,fib(i),retour(i),
retour(i)-1]);
% display: at which term
the reversal began, the
recalculated value for the
first term, the difference
between this value and 1
end;
```

* the indices in Matlab start at 1 – not at 0. fib(1) represents f_0, fib(2) represents f_1, …
* type [f, fr] = fibo(100) to call the function for 100 terms and keep track of important data
* an analogue pibo function was also made

Explorations

Starting from…	*…the recalculated value of p_0 is…*
p_2	1
p_3	1
p_4	1
p_5	1
p_6	0.999999999999996
p_7	0.999999999999996
p_8	0.999999999999948
p_9	1.000000000000024
…	…

The absolute difference between $p_0 = 1$ and its recalculated value starting at p_n ($n \geq 6$)

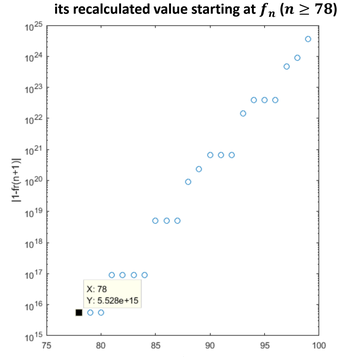


Observed behaviour is **not surprising:**

- do not return to exact p_0 as of starting at p_6
- small error ($\approx 4.441 \times 10^{-15}$) at first
- error grows gradually with n, reaching a large error ($\approx 1.623 \times 10^{25}$) when starting at p_{99}

Starting from…	*…the recalculated value of f_0 is…*
f_2	1
f_3	1
f_4	1
…	…
f_{77}	1
f_{78}	-5.528×10^{15}
f_{79}	-5.528×10^{15}
f_{80}	-5.528×10^{15}
…	…

The absolute difference between $f_0 = 1$ and its recalculated value starting at f_n ($n \geq 78$)



Observed behaviour is **surprising:**

- suddenly do not return to exact f_0 when starting at f_{78}
- error is immediately large ($\approx -5.528 \times 10^{15}$)
- error grows in steps, reaching the same magnitude as for the Pibonacci Numbers when starting at the 100th term

Explanations

Pibonacci Numbers: Any computer uses a finite number of bits to represent numbers. An irrational number like $c = 1 + \sqrt{3}/100$ would require an infinity of bits for an exact representation.

The rounding that occurs provokes a small error in the calculation of $p_3 = cp_2 + p_1$, which leads to a larger error in calculating $p_4 = cp_3 + p_2$, an even larger error in $p_5 = cp_4 + p_3$, and so on. Under the rounding rules used by Matlab, the error becomes large enough to have a visible impact on the return to p_0 when $n = 6$. Before this point, we do not observe the error, even though, in theory, there is one. After this point, the error accumulates, gradually growing in an exponential fashion.

Fibonacci Numbers: Matlab automatically uses a representation of 64 bits (double precision $\epsilon_{mach} = 2^{-52}$) for any number, including integers. Not all integers are exactly representable!

As of 2^{53}, Matlab will “miss-represent” certain integers (Overton , 1996):

Given a normalized positive machine number
$x = (1.b_1 b_2 \dots b_{52})_2 \times 2^E$,
the gap between x and the next largest normalized machine number is
$g = \epsilon_{mach} \times 2^E = 2^{-52} \times 2^E$.

If $0 \leq E \leq 52$, then $2^{-52} \leq g \leq 1$.
Thus, any whole number between 2^E and 2^{E+1},
$0 \leq E \leq 52$, has an exact representation in Matlab.

If $E \geq 53$, then $g \geq 2$.
For example, all integers of the form
$x = 2^{53} + (2k - 1), k \geq 1, k \in N$
are not representable in an exact way.

f_n is represented exactly if $n < 78$;
it may not be representable if $n \geq 78$:

Applying the theory of difference equations (Golub & Ortega, 1992, pp. 57-59),

$$f_n = \frac{(5+\sqrt{5})\lambda_1{}^n}{10} + \frac{(5-\sqrt{5})\lambda_2{}^n}{10} \quad (*)$$

$$\lambda_{1,2} = \frac{1 \pm \sqrt{5}}{2}$$

Since $|\lambda_2| < 1$ the second term in (*) tends to 0 as $n \to \infty$.
Thus, we can estimate that $f_n > 2^{53}$ as soon as

$$\frac{(5+\sqrt{5})\lambda_1{}^n}{10} > 2^{53}$$
$$\Leftrightarrow n > \frac{\ln(2^{53} \cdot 10/(5+\sqrt{5}))}{\ln(\lambda_1)} \approx 77.015$$

When $n = 77$ the first term in (*) is $\leq 2^{53}$ and the second term is negative (since λ_2 is negative). Thus, $f_n < 2^{53}$ when $n \leq 77$.
If $n = 78$, the first term in (*) $> 2^{53}$ and the second term is positive;
$f_n > 2^{53}$ if $n \geq 78$.

f_{78} is rounded (i.e., not represented exactly):

By definition, $f_{78} = f_{77} + f_{76}$.
By above, f_{77} and f_{76} have exact representations.
We can find them using a Matlab function dec2bin() and then add by hand:

```
                11 1111000011111000010000011001100110000011000000000
f76             (1.0011101000111010000111000010011011000000101000101)2 × 2^52
+ f77         + (1.1111110001101110000100010111001101001111001101000)2 × 2^52
-------------------------------------------------------------------------
f78           = (11.0011011010101000000101101100100111001001011111101)2 × 2^52
              = (1.1001101101010100000010110110010011100100101011111010)2 × 2^53
```

Notice that the fractional part of f_{78} requires 53 bits for an exact representation in double precision. Hence, to represent f_{78}, Matlab will round it to the nearest machine number; it will just drop the 1 after the vertical bar. The absolute error between f_{78} and its Matlab representation is 1. It may seem small, but it has huge consequences!

Conclusion/Questions

When I picture a student engaging in Computational Thinking, they are thinking (at least eventually) with computers, which have their own ways of “thinking”.

The computer is more powerful in that it enables students to calculate, visualize, simulate, and experiment like never before. They can calculate faster, with bigger numbers. They can visualize in highly organized ways. They can simulate phenomena too complex to be explored otherwise. And, at the touch of a button, they can try something and see what happens; then try something else!

But the computer is also constrained by its information-processing agent and operating environment, which can influence the way it “thinks” about mathematics (Balacheff, 1994). The student must not only tell the computer what to do; they must dissect what the computer does. The example on the left shows how this could lead to enriching mathematical and computational experiences.

It remains to discuss…

(1) Which part of the example uses computational thinking?
(2) Which part of the example uses mathematical thinking?
(3) Would an activity like this be appropriate for mathematics classrooms? If so, at what level?

References:

Balacheff, N. (1994). Didactique et intelligence artificielle. *Recherches en didactique des mathématiques*, *14*(1/2), 9-42.

Burton, L. (1984). Mathematical thinking: The struggle for meaning. *Journal for Research in Mathematics Education*, *15*(1), 35-49.

Golub, G.H., & Ortega, J.M. (1992). *Scientific computing and differential equations: An introduction to numerical methods.* San Diego, CA: Academic Press, Inc.

Goodman, J. (2002). *Assignment 1.* Retrieved from https://www.math.nyu.edu/faculty/goodman/teaching/SciComp2002/assignments/assignment_1.pdf

Hodgson, B. (2010). Conférence d’ouverture: Des petits cailloux aux éposutouflantes puces: chassés-croisés mathématiques/informatiques au fil des âges. *Bulletin AMQ*, *1*(3), 7-36.

Overton, M.L. (1996). *Floating point representation* [.pdf Document].

Wing, J. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.