"Fibonacci via Recursion and Iteration"

Provided by TryEngineering - www.tryengineering.org

Lesson Focus

This lesson introduces how to calculate an arithmetic series, specifically Fibonacci. In the first of two hour-long sessions, using a spreadsheet (e.g. Microsoft Excel or Google Drive Sheets), students are shown how to calculate a series based on two prior values (the iterative solution), and by using a user-defined function (the recursive solution). With a large enough domain, most computers will exhibit real delays in calculating the recursion for values greater than 30. In the second session, they will explore why the iterative solution is faster, and why the recursive solution significantly slows down for large values. This lesson assumes that the teacher is well versed in using spreadsheets, including copy-down formulas.

Age Levels

Recommended for ages14 – 18

Objectives

Introduce students to:

- How series occur in nature
- Recursive algorithms for arithmetic series, not just Fibonacci
- Iterative solutions that rely on stored data to make recursive solutions more efficient
- Informal ideas about time complexity

Anticipated Learner Outcomes

Students will be able to describe how to solve a class of problems like Fibonacci:

- With recursion
- With iteration that exploits data
- And articulate that when an imbedded recursive solution can be recast as an iterative one, effort (e.g. time) can be significantly reduced

Alignment to Curriculum Frameworks

See attached curriculum alignment sheet.

Internet Connections

- Doodling in Math: Spirals, Fibonacci, and Being a Plant [1 of 3]
- Doodling in Math Class: Spirals, Fibonacci, and Being a Plant [2 of 3]
- Doodling in Math: Spirals, Fibonacci, and Being a Plant [Part 3 of 3]
- Free Spreadsheet alternatives
- http://www.regentsprep.org/regents/math/algtrig/atp2/sequencewordpractice.htm
- http://www.algebralab.org/lessons/lesson.aspx?file=Algebra_SeqSeriesApps.xml

Recommended Reading

- The Golden Section: Nature's Greatest Secret, Scott Olson, Wooden Books. 2006
- Growing Patterns: Fibonacci Numbers in Nature, Sara C. Campbell, 2010

Optional Writing Activity

This activity introduces the idea of how to efficiently calculate an arithmetic series, such as Fibonacci. There are other interesting series out there, so research one on the Internet and explain how it compares to Fibonacci. Is the recursive solution just as efficient? Is there a way to describe it using data storage?

Fibonacci via Recursion and Iteration

For Teachers:
Teacher Resources

◆ Lesson Objectives

- Introduce recursion via the Fibonacci numbers.
- Demonstrate how mathematical expressions can be calculated using computers, and spreadsheets in particular.
- Illustrate what happens when a purely recursive solution is implemented as code.
- Introduce user-defined functions (macros) in a spreadsheet.
- Demonstrate how stored data can significantly simplify some expensive calculations.
- Provide an opportunity for discussing Internet safety and responsibility.

◆ Materials

- Access to videos online. Download and cache ahead of time if necessary.
- Have sufficient copies of the worksheets below for all students to participate at least twice, in a recursive calculation.
- A spreadsheet with user-defined functions available (Microsoft Excel, or Google Drive Sheets suggested).
- A stopwatch (a cell phone will do).

◆ Procedure

It is assumed that you are familiar with doing simple calculations using spreadsheets. It will be helpful if your students have had some familiarity with spreadsheets, but it is not necessary. Session 1 gives you guidance in helping your students create a very simple sheet that calculates Fibonacci using existing calculations. The solutions up to n=40 are calculated instantaneously. You can do this exercise on any spreadsheet, including Mac Numbers on an iPad. You will need to know how to 'drag copy' on your particular sheet. (in Excel and Google Drive Sheets it is in the lower right corner, in Mac Numbers it is in the middle).

The second exercise involves creating a user-defined function in the spreadsheet. This version actually executes the solution recursively, and for large n (around 35 – 40) it can take observable time to complete the solution.

The student worksheet contains both the Visual Basic (Excel:mac 2011) and Google App Script (Drive Sheet) recursive definitions. If you don't have access to either, or another spreadsheet that has user-defined functions, you can have the students trace the solution out, or, if you know some programming, create a demo for them. An example of the trace for Fibonacci(7) appears at the end of this resource.

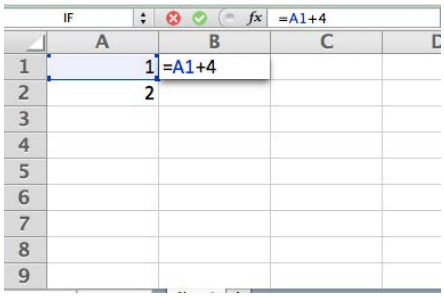
Since Excel macros are notorious as a virus injection technique, please talk to your students about not passing macro-included Excel spreadsheets from computer to computer. Your students are novices and can't guarantee that the macros are safe. Please note that you may have to tinker with the Visual Basic for other versions of Excel. Before teaching this lesson, please make sure you are very comfortable with the process for your environment. Finally, novices tend to like to try big numbers - entering Fib(1000) in an Excel sheet may cause the sheet to lock up. Make sure you know how to stop a process on your computer without entirely shutting down the system. Google claims to

cut off a calculation on a spreadsheet if it takes more than 30 seconds. Discuss responsible computer use with your students; experimentation that is not carefully thought out can impact other users.

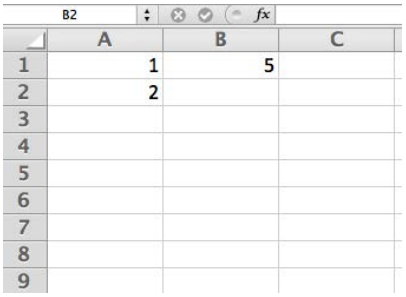
Session 1:
Have your students view the video Doodling in Math: Spirals, Fibonacci, and Being a Plant [1 of 3] and talk about how series occur in the natural world.

Explain how the Fibonacci series is calculated - that each result is the sum of the two previous results. Have them use Worksheet 1 to calculate it by hand.

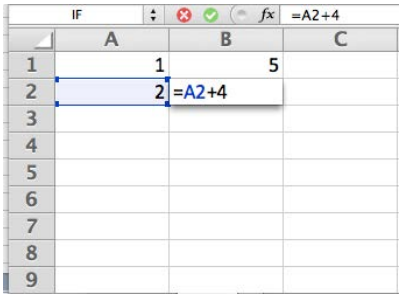
Using your spreadsheet of choice, and depending on the experience your students have had with them, demonstrate how they can calculate a simple result. Three examples:



Calculate a result from a cell



The result appears



The result of a copy drag

Guide them through creating a spreadsheet that calculates Fibonacci. The starting sample appears on Worksheet 1.

Recursive Macros
If there is time, introduce them to creating user-defined functions on their spreadsheet.

- In Excel, they will need to select the 'Developer' tab, and then select 'Macro' as described in Worksheet 2
- In the Google Drive Sheet, they will need to select 'Tools', then 'Script Editor'. The syntax for the code differs in subtle ways as shown on the Worksheet.

Have them test it with numbers below 10, and make sure they save their work. In Excel they will need to save a macro enabled file. Talk a bit about viruses and how they can be introduced, because they are adding code that could conceivably do anything!

Have them try out a few values and see what happens. Be prepared to force quit for very large numbers, but for the next session, leave off the explanation as to why there is a delay.

Session 2
Depending on what was accomplished in the first session, review the phenomenon of the first spreadsheet calculation. Review that each cell calculated its result from two previously stored values.

If you had time in the prior session, ask students to open the spreadsheet with the recursive macro. Otherwise do that activity now. Using a stopwatch, have students time how long it takes to calculate n > 25. Once everyone has had a chance to explore this phenomenon, have a discussion about what is happening. Based on the following

diagram, you may want to trace out the function calls on a white board, or have the students do it themselves.

```
Fibonacci(6) =
        Fibonacci(5) =
                Fibonacci(4) =
                        Fibonacci(3) =
                                Fibonacci(2)  = 1
                                +
                                Fibonacci(1)  =  1
                        +
                        Fibonacci(2) =       1
                +
                Fibonacci(3) =
                        Fibonacci(2)  =      1
                        +
                        Fibonacci(1)  =      1
        +
        Fibonacci(4) =
                Fibonacci(3) =
                        Fibonacci(2)  =      1
                        +
                        Fibonacci(1)  =      1
                +
                Fibonacci(2) =               1
```

Fibonacci trace for n = 6

Review the difference:

- In the data stored solution, each calculation relies on the fact that the previous two calculations have been recorded
- In the recursive solution, each call to Fibonacci has to calculate ALL of the sub-problems. Although the recursive solution is an elegant mathematical expression, it is not very efficient, in fact it is extremely inefficient

To conclude, explain that computer scientists not only look for solutions to problems, but they look for elegant and efficient solutions that exploit creative ways of thinking about data. Even in the Cloud, there are trade-offs between speed and space. This is the study of 'complexity'. It raises questions such as how fast you can calculate something, how much space you need, what other problems (which classes of problems) have the same complexity.

If there is time, you may want to ask them exactly how much data you have to keep. (The answer is only the two previous values.) If your students have some programming experience, challenge them to write the data store version as code. Challenge them to find other series like Fibonacci (hint: explore the other Vi Hart videos).

Time Needed

- 2 sessions, at most 1 hour each. This can be done in a single two-hour session. Shorter class time is possible if the first video and at least one of the Internet Connections are assigned as homework before the session.

Fibonacci via Recursion and Iteration

Student Resource:

The Vi Hart videos on the Fibonacci numbers demonstrate how these numbers are found in nature. But only in passing is it mentioned that a Fibonacci number is calculated by adding the previous two Fibonacci numbers. Every Fibonacci number, except the first two, is found by adding the previous two Fibonacci numbers. This rule doesn't make sense for the first two Fibonacci numbers and so these are defined as being equal to '1.' A mathematical series like Fibonacci is *recursive,* which just means that the current value is dependent upon previous values, except for a *base case,* when there are no previous values. It makes sense that things that grow in nature would be dependent upon something that happened previously.

Mathematicians like to keep things precise and succinct, and they define the Fibonacci numbers as follows:

> For any 0 < n <= 2 (basically when n = 1 or n = 2) Fibonacci(n) = 1
> For any n > 2 Fibonacci(n) = Fibonacci(n-1) + Fibonacci(n-2)

Computer scientists write *algorithms* using very inventive short-cuts that exploit the ability of computers to repeat tedious tasks. An algorithm is a process or set of rules that describes a solution to a problem. Computers follow the instructions described in an algorithm. To create inventive short-cuts often requires thinking about a problem from a particular point of view. One very powerful point of view is *recursion.* A nice way to think about recursion was invented by a computer scientist named Seymour Papert. The idea is that if you have a problem that appears long and boring, think about how you can off-load the problem onto a 'smaller sibling'. This is not quite the Tom Sawyer approach to painting a fence, but it is close. No matter what, the idea is to minimize the amount of work you have to do to get to a solution. So if you are calculating the 7^{th} Fibonacci number, you rely on the fact that someone else calculated both the 6^{th} and 5^{th} number for you.

You will learn how to calculate a series like Fibonacci by using existing data, and you will see how this differs from using a pure recursive function. Computer scientists are always looking for the most efficient solution, both in terms of the time it takes and the amount of data to be stored. You will have a chance to see how the description that mathematicians like isn't the most efficient when it's implemented as computer code. Computer scientists spend a fair amount of time looking for (sneaky) ways of making math expressions faster to calculate.

This lesson also introduces you to using a spreadsheet to do the tedious work for you. Although spreadsheets are not thought of as programming languages, you can actually do some pretty sophisticated things with them if you know how to write a function as a program.

Your teacher will select a spreadsheet application for you to use. The following worksheets provide some explanations for using Microsoft Excel and Google Drive Sheets. Other applications work pretty much the same way for simple calculations: however, for the second activity you will need an application that allows you to define 'macros' (like Excel) or 'add-ons' (like Google Drive Sheets). If you don't have one, you can always just do the calculation by hand.

Fibonacci via Recursion and Iteration

Student Worksheet 1:

Here is a definition of the Fibonacci series.

For any 0 < n <= 2 (basically when n = 1 or n =2) Fibonacci(n) = 1
For any n > 2 Fibonacci(n) = Fibonacci(n-1) + Fibonacci(n-2)

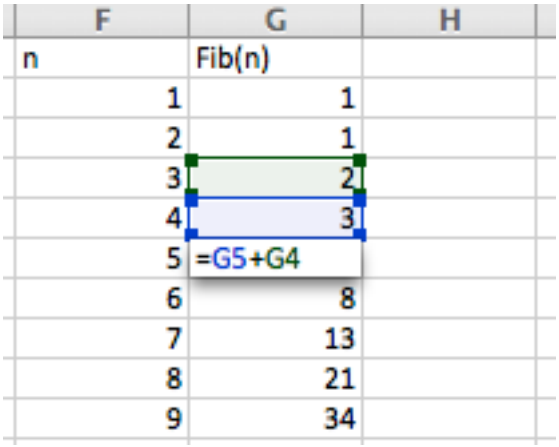
Part 1: Calculate Fibonacci(7) by hand:

Please fill in the rest of the numbers in the table:

n is	1	2	3	4	5	6	7
Fibonacci(n)							

After your teacher instructs you on how to get a spreadsheet application, open up a new spreadsheet, label a column 'n' (for 'number') and enter integers in a column starting with '1'. Go as high as 40. (Hint: there are ways to do this automatically, but it may be faster to just type them in.)

In the column to the right, label the column 'Fib(n)'. Enter the number '1' in the next two cells in that column. The next entry is a calculation (it starts with '='). Select the previous two cells and add them together as illustrated below. Each cell in this column with the exception of the first two, should have this formula. Be careful to use 'copy down' rather than 'copy'. For example, in this spreadsheet cell G6 sums G5 and G4, G7 sums G6 and G5.



F	G	H
n	Fib(n)	
1	1	
2	1	
3	2	
4	3	
5	=G5+G4	
6	8	
7	13	
8	21	
9	34	

How long did it take the spreadsheet to calculate the Fibonacci numbers?

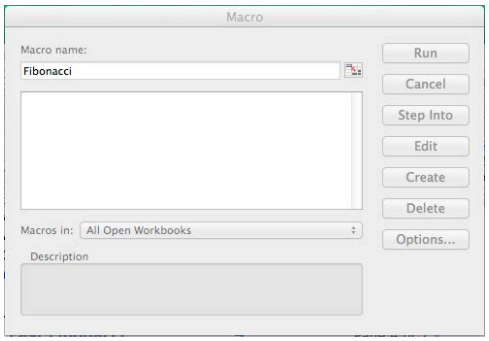
Can you time it?

Fibonacci via Recursion and Iteration

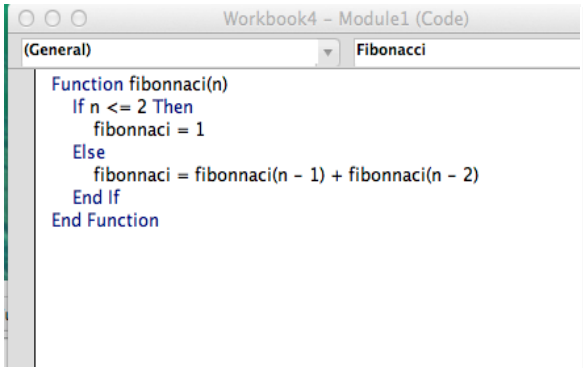
Student Worksheet 2:
User Define Functions

It is a well-kept secret that you can write programs for a spreadsheet. This exercise shows you how to create 'user-defined' functions, which are called 'macros' in Microsoft Excel, and 'Add-ons' in Google Drive. Your teacher will instruct you how to create a user-defined function in the application available to you. Illustrations in Excel and Google Drive Sheets appear here as a guide.

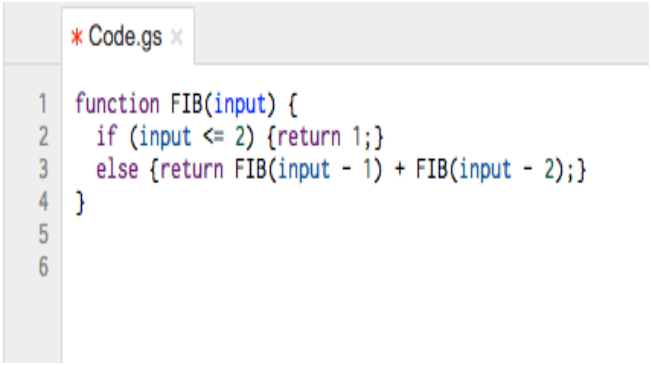
1. Open a spreadsheet application. Find your way to the tabs or pull down menus for user-defined functions. In Microsoft Excel you select the 'Developer' tab and then select 'Macro'. A pop-up window will appear. Name your macro 'Fibonacci'. In Google Drive open the 'Tools' menu and select 'Script Editor'.



2. Write the code shown here to define your function. Depending on the versions you have, your teacher may provide you with a slightly different version for Excel.

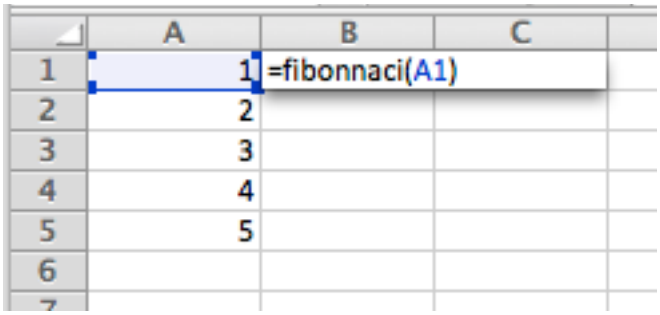


Excel Macro written in Visual Basic



Google Sheet written in a JavaScript

3. Enter your function in a cell in your spreadsheet.



4. Experiment with values for 'n'. Get a stopwatch ready. Numbers between 30 and 40 should produce noticeable delays. Make a chart of results on a separate sheet of paper. Take care not to use numbers that are too large (above 40), and make sure you know how to force quit on your application. Also make sure you save your user defined function.

5. Unless you are working on a very fast computer, there should be a difference in how fast the calculations on Worksheet 1 were generated compared to the user defined function. Why? What's going on here?

"Fibonacci via Recursion and Iteration"

For Teachers: Alignment to Curriculum Frameworks

Note: All lesson plans in this series are aligned to the Computer Science Teachers Association K-12 Computer Science Standards, and if applicable also the U.S. Common Core State Standards for Mathematics, the U.S. National Council of Teachers of Mathematics' Principles and Standards for School Mathematics, the International Technology Education Association's Standards for Technological Literacy, the U.S. National Science Education Standards and the U.S. Next Generation Science Standards.

◆National Science Education Standards Grades 9-12 (ages 14-18)

CONTENT STANDARD E: Science and Technology

As a result of activities, all students should develop

- ✦ Understandings about science and technology

◆Next Generation Science Standards & Practices Gr.9-12 (ages 14-18)

Practice 5: Using Mathematics and Computational Thinking

- ✦ Create and/or revise a computational model or simulation of a phenomenon, designed device, process, or system.
- ✦ Use mathematical, computational, and/or algorithmic representations of phenomena or design solutions to describe and/or support claims and/or explanations

Practice 6: Constructing Explanations and Designing Solutions

- ✦ Apply scientific ideas, principles, and/or evidence to provide an explanation of phenomena and solve design problems, taking into account possible unanticipated effects.

◆ Principles and Standards for School Mathematics (ages 14 - 18)

Number and Operations Standards

- Compute fluently and make reasonable estimates
  - ✦ develop fluency in operations with real numbers, vectors, and matrices, using mental computation or paper-and-pencil calculations for simple cases and technology for more-complicated cases

Algebra Standards

- Represent and analyze mathematical situations and structures using algebraic symbols
  - ✦ use symbolic algebra to represent and explain mathematical relationships
  - ✦ use a variety of symbolic representations, including recursive and parametric equations for functions and relations

◆ Principles and Standards for School Mathematics (all ages)

Problem Solving Standards

- ✦ Solve problems that arise in mathematics and other contexts

Connections

- ✦ Recognize and apply mathematics in contexts outside of mathematics

"Fibonacci via Recursion and Iteration"

For Teachers: Alignment to Curriculum Frameworks

◆Common Core State Standards for School Mathematics Gr. 9-12 (ages 14-18)

Mathematics | High School—Functions

f-If - Interpreting functions

- Understand the concept of a function and use function notation
  - ✦ CCSS.Math.Content.HSF-IF.A.3 Recognize that sequences are functions, sometimes defined recursively, whose domain is a subset of the integers. For example, the Fibonacci sequence is defined recursively by f(0) = f(1) = 1, f(n+1) = f(n) + f(n-1) for n ≥ 1.

◆Common Core State Practices & Standards for School Mathematics (all ages)

- ✦ CCSS.MATH.PRACTICE.MP1 Make sense of problems and persevere in solving them.
- ✦ CCSS.MATH.PRACTICE.MP4 Model with mathematics.
- ✦ CCSS.MATH.PRACTICE.MP5 Use appropriate tools strategically.

◆ Standards for Technological Literacy - All Ages

Nature of Technology

- ✦ Standard 2: Students will develop an understanding of the core concepts of technology

The Designed World

- ✦ Standard 17: Students will develop an understanding of and be able to select and use information and communication technologies

◆CSTA K-12 Computer Science Standards Grades 9-12 (ages 14-18)

5.3 Level 3: Applying Concepts and Creating Real-World Solutions (L3)

5.3.A *Computer Science in the Modern World (MW)*

- ✦ Computational Thinking (CT)
  3. Explain how sequence, selection, iteration, and recursion are building blocks of algorithms.