Logistics

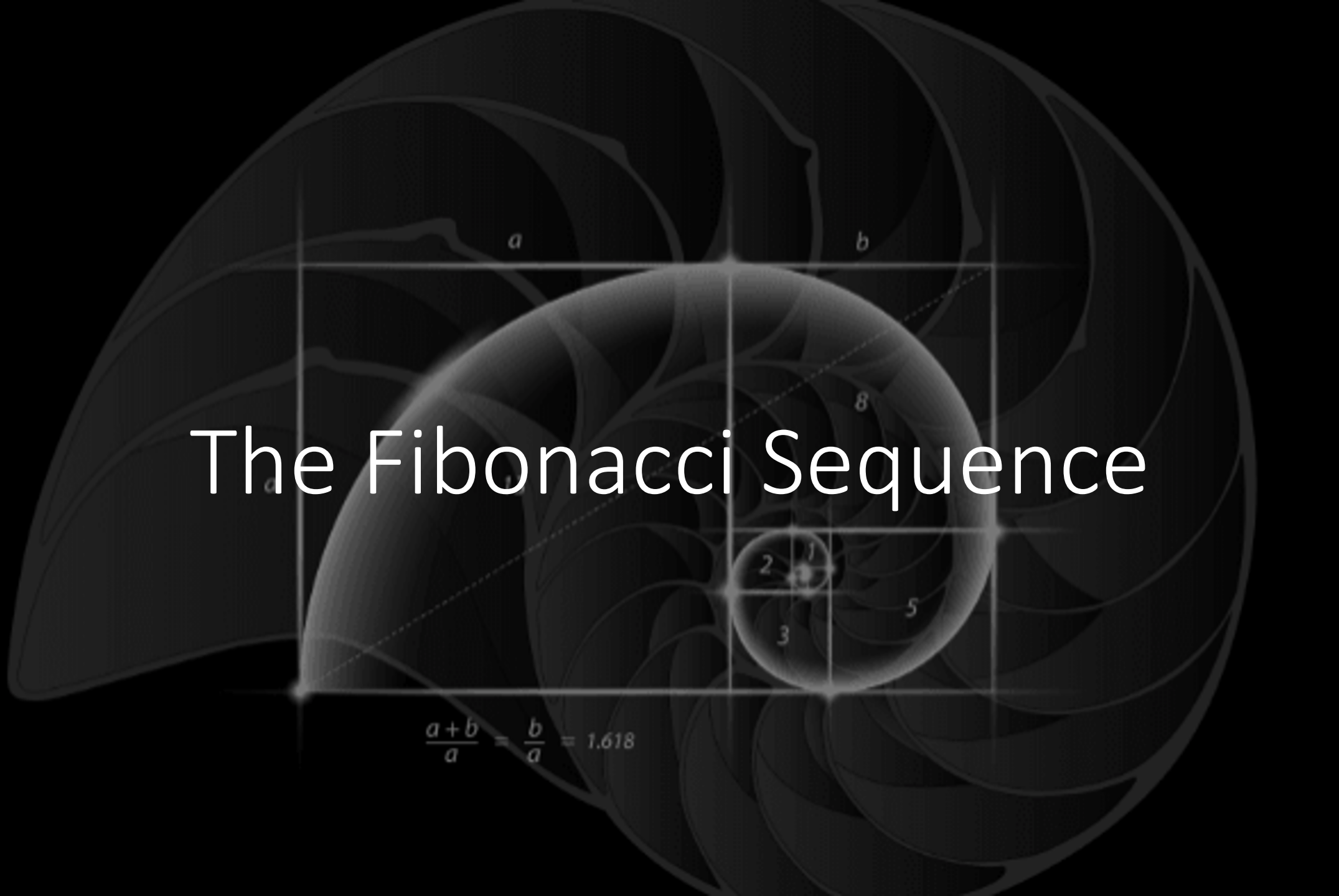
Look for the email Mathworks sent you inviting you to CS151 so you can complete this week's homework.

There is no reading assignment this week.

MATLAB: Arrays (Vectors)

Prof Matthew Fricke

Version 1.0. Send corrections to mfricke@unm.edu.



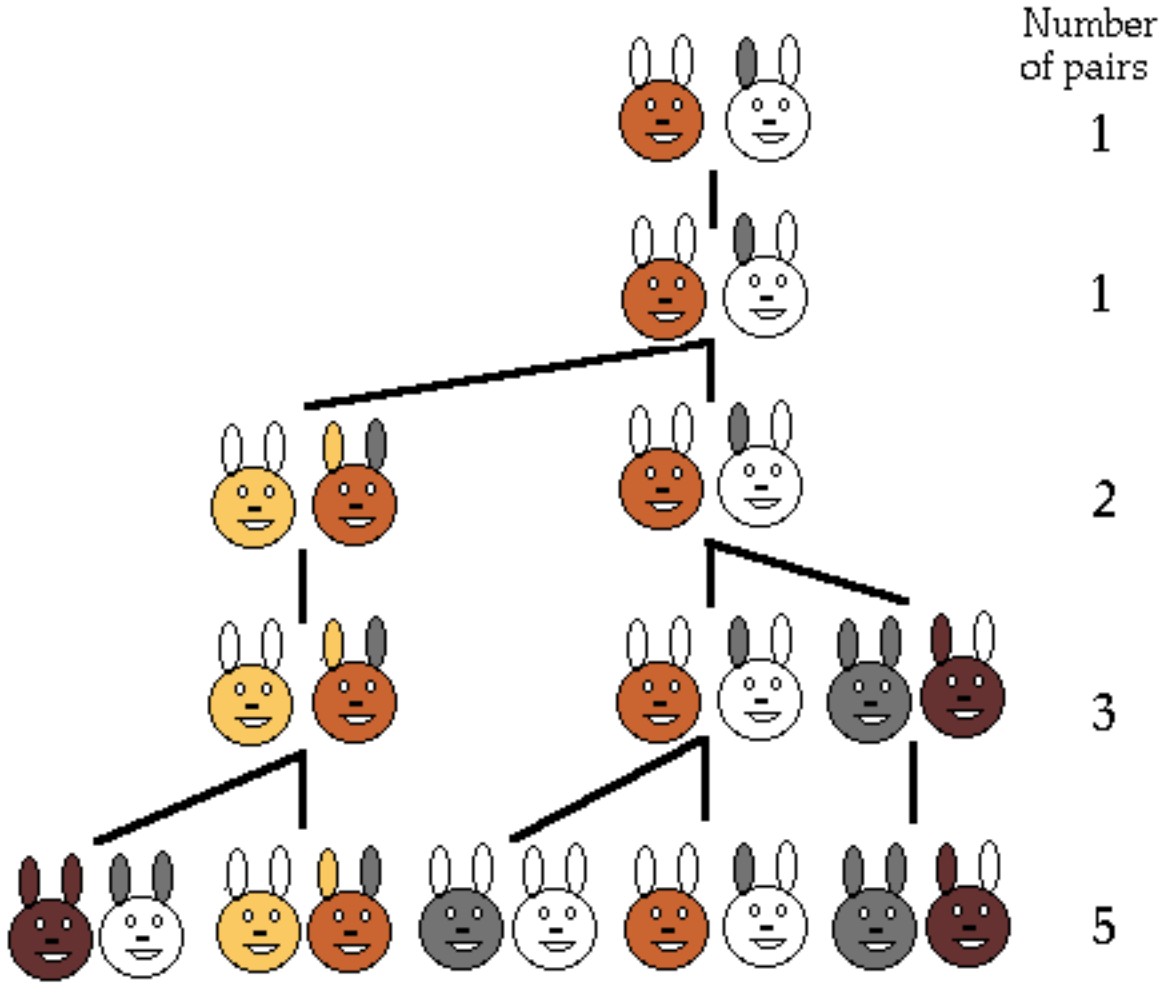
The Fibonacci Sequence

People see Fibonacci Numbers Everywhere

- Fibonacci's sequence answers the question:

- If you have a breeding male and female rabbit at time 0, how many pairs of rabbits will you have at time t?

https://bungalowofbunnies.weebly.com/blog/archives/05-2017

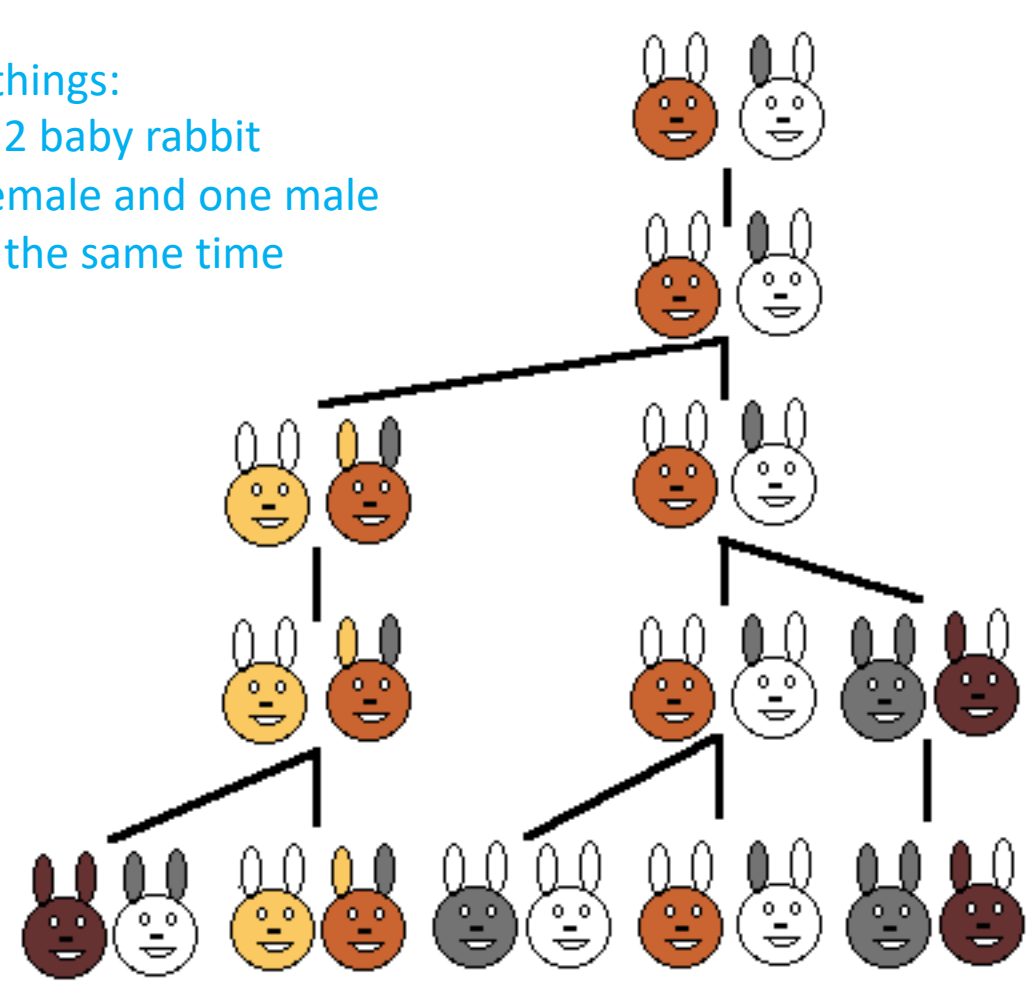


The number of pairs of rabbits in the field at the start of each month is 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

http://www.maths.surrey.ac.uk/hosted-sites/R.Knott/Fibonacci/fibnat.html

Assumes lots of untrue things:

1) Rabbits always have 2 baby rabbit
2) One baby rabbit is female and one male
3) The all reproduce at the same time

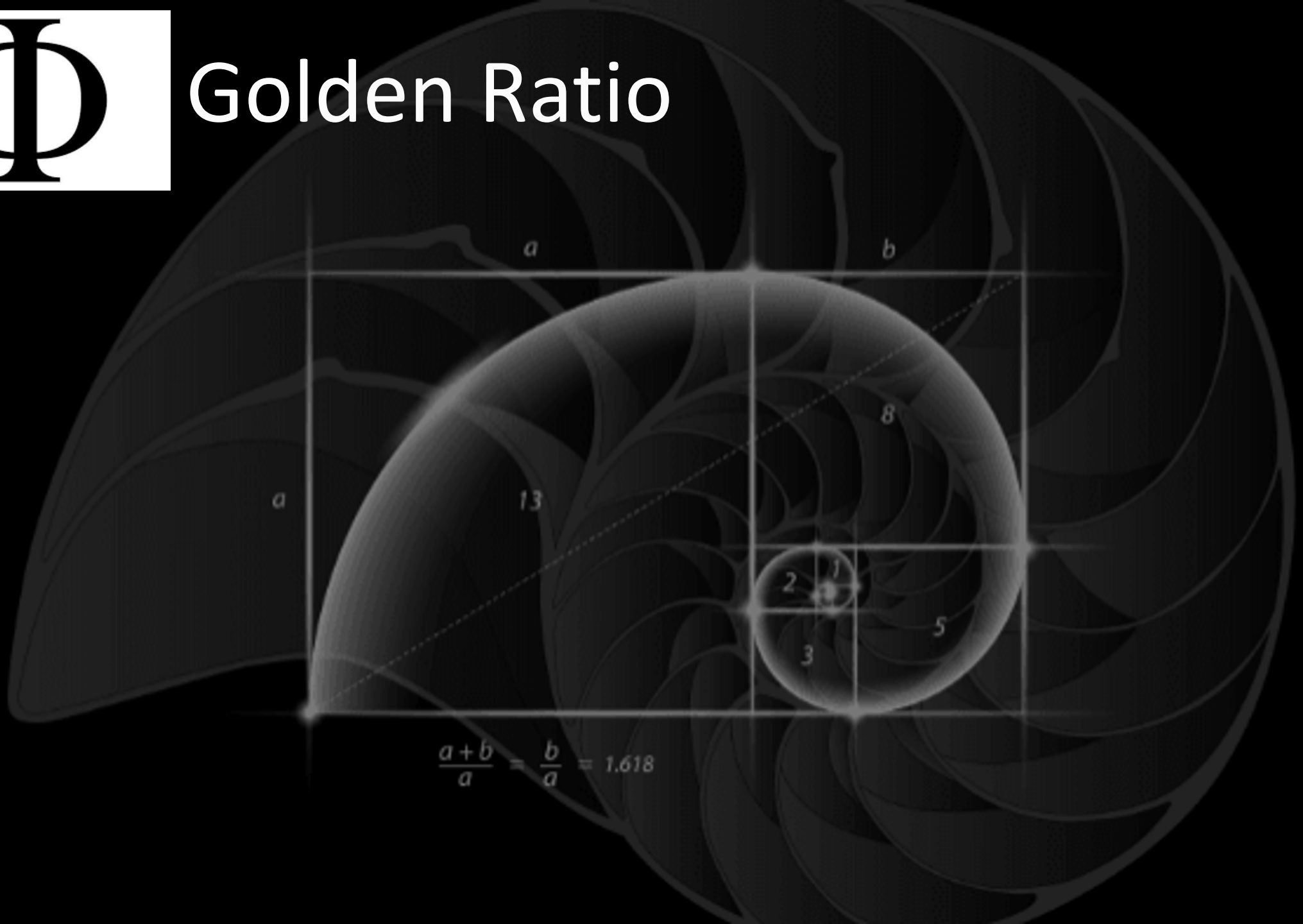


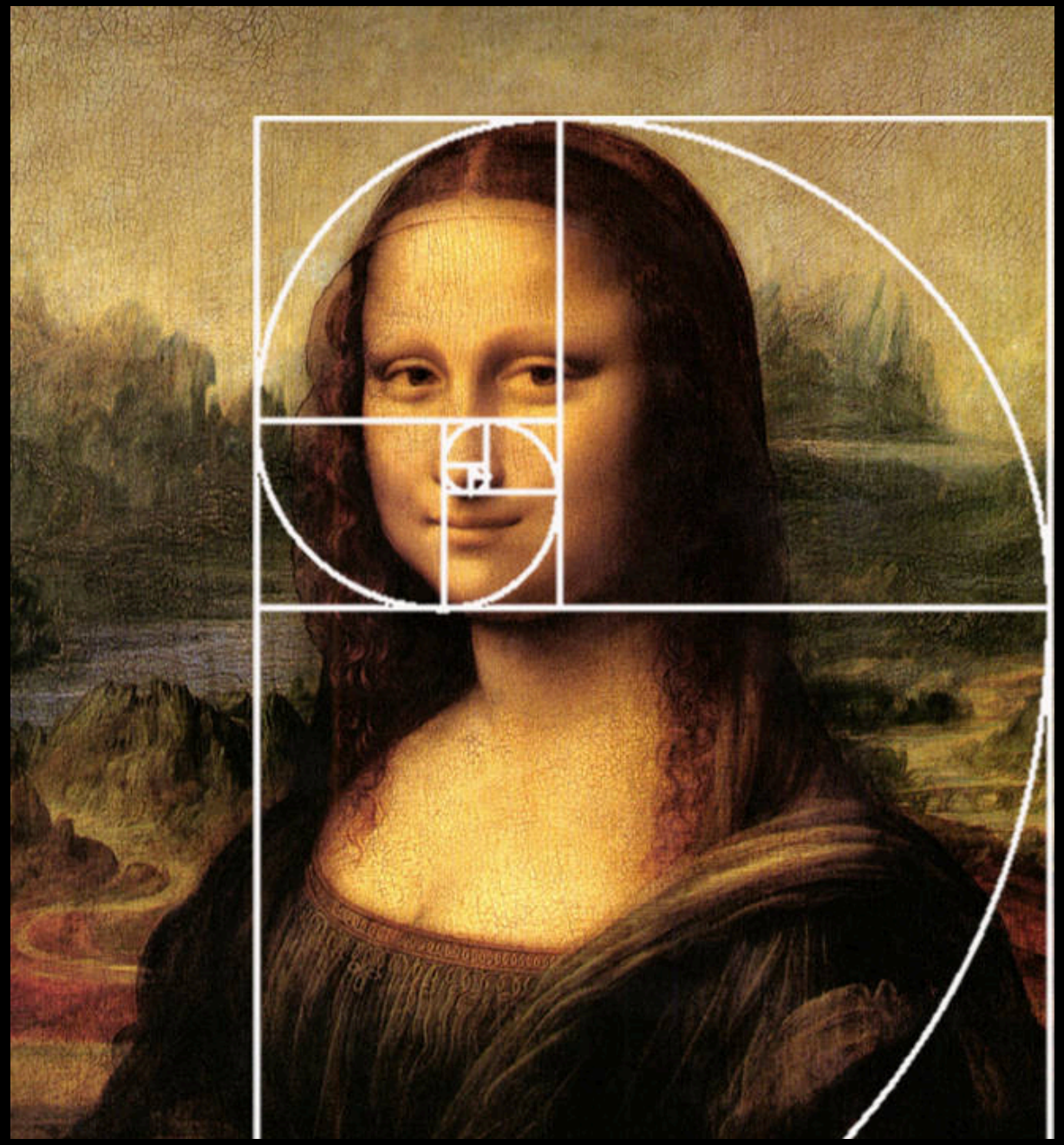
Number of pairs	Month
1	1
1	2
2	3
3	4
5	5

The number of pairs of rabbits in the field at the start of each month is 1, 1, 2, 3, 5, 8, 13, 21, 34, ...

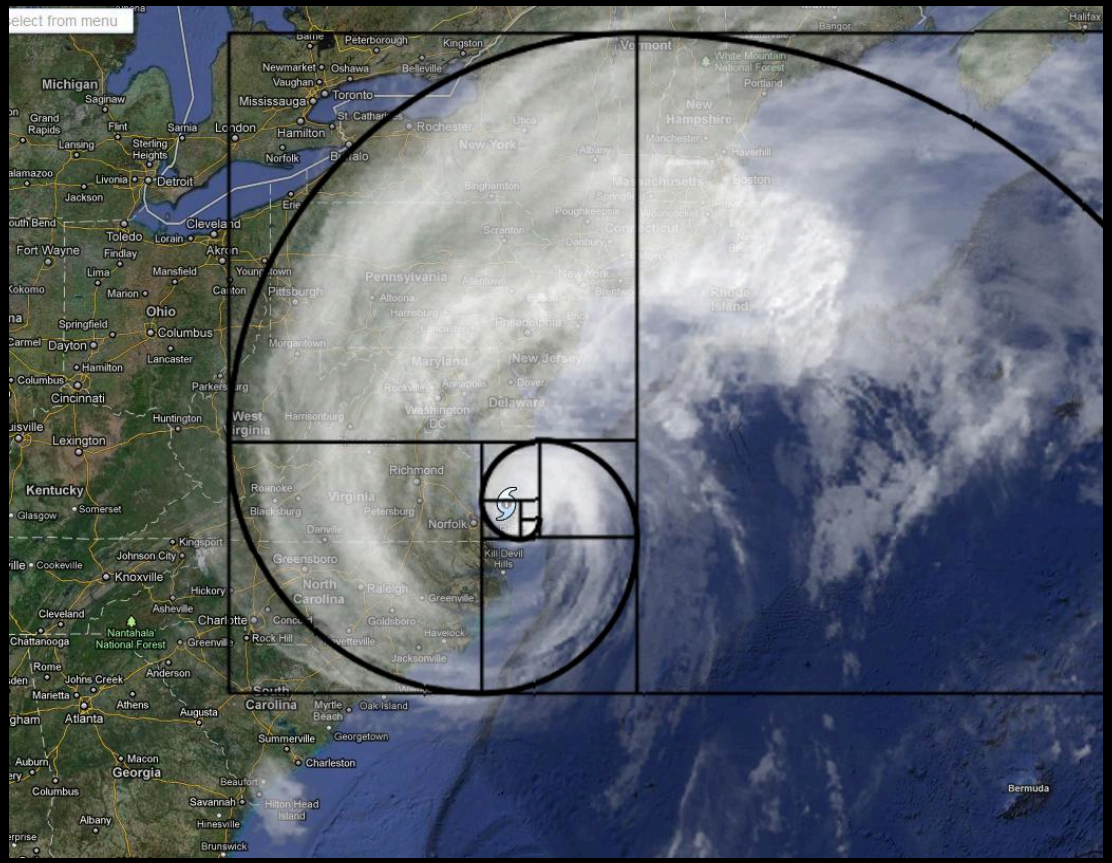
http://www.maths.surrey.ac.uk/hosted-sites/R.Knott/Fibonacci/fibnat.html

Golden Ratio



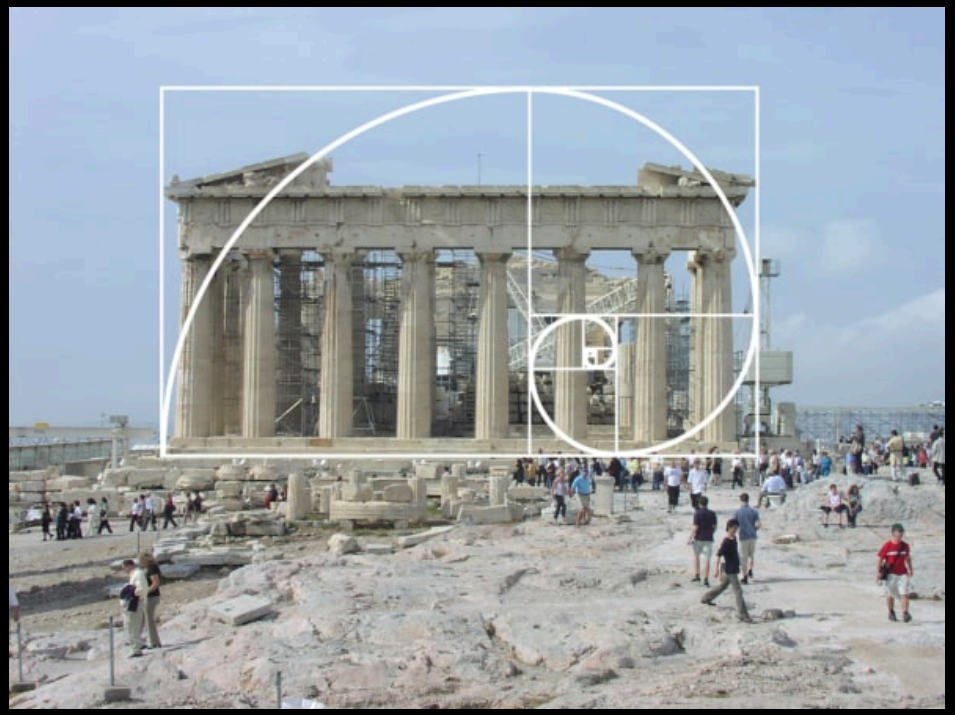


People see Fibonacci Numbers Everywhere



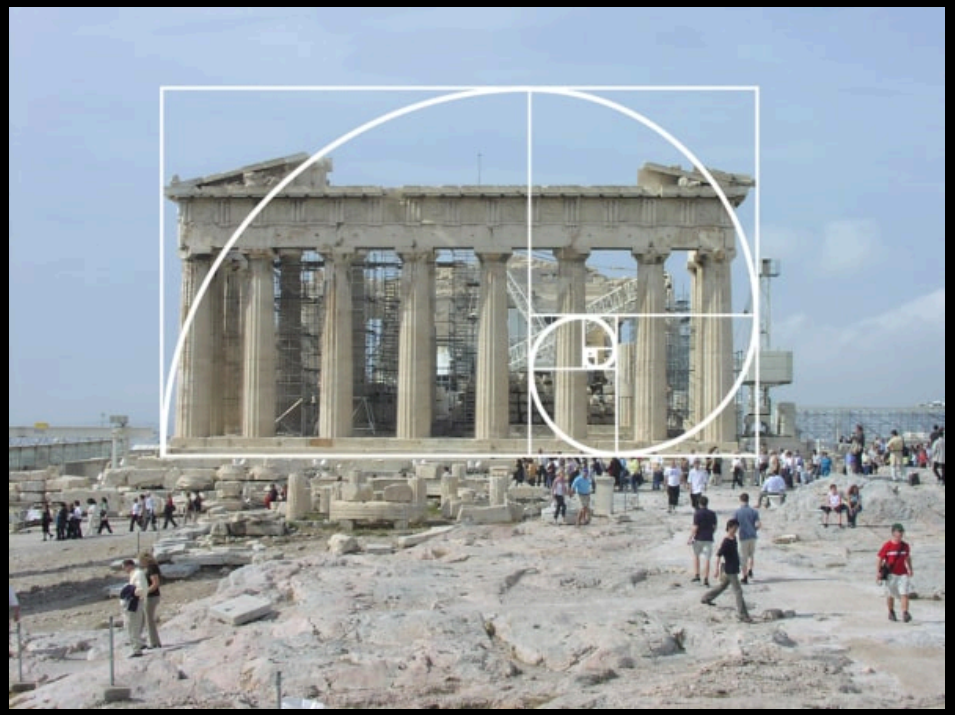
While Fibonacci numbers are undoubtedly important, sometimes people go overboard.

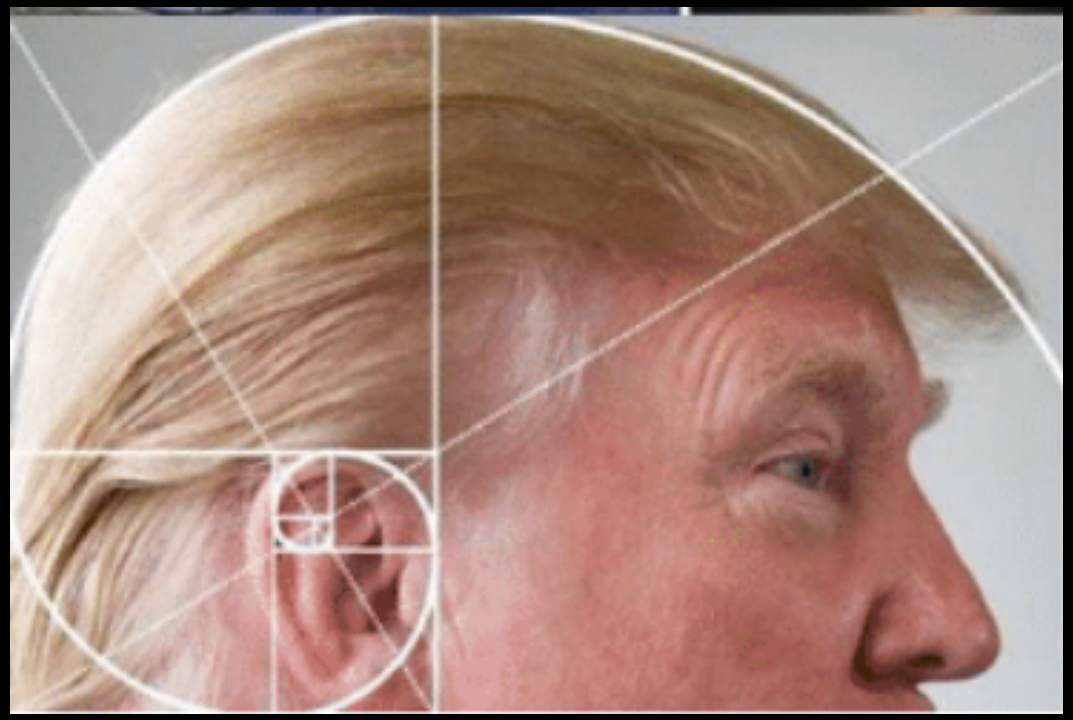
People see Fibonacci Numbers Everywhere – Too many places



While Fibonacci numbers are undoubtedly important, sometimes people go overboard.

People see Fibonacci Numbers Everywhere – Too many places





While Fibonacci numbers are undoubtedly important, sometimes people go overboard.

Fibonacci Sequence

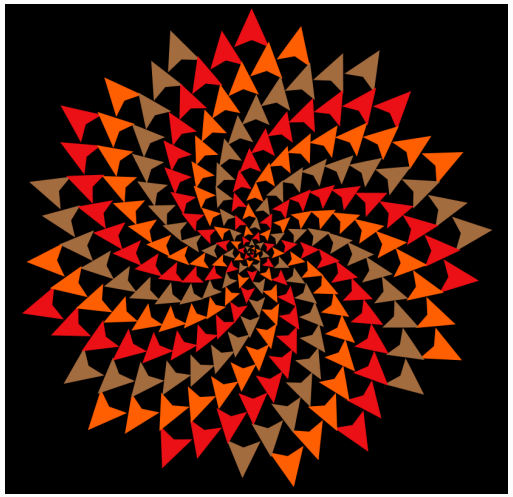
Calculating the sequence is popular in intro programming because it is a really good example of building something complex from very simple rules.

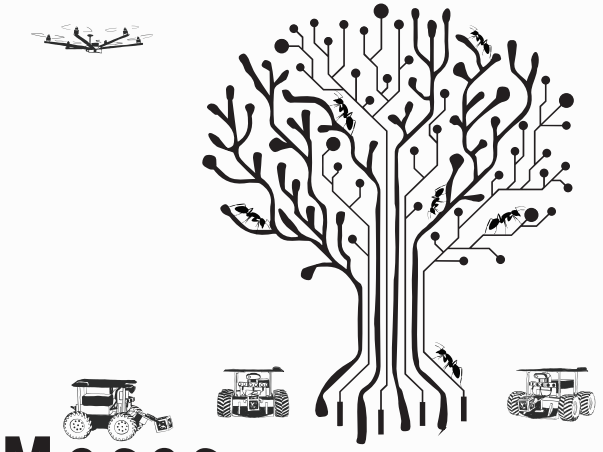
All you need to know is the previous two values in the sequence.

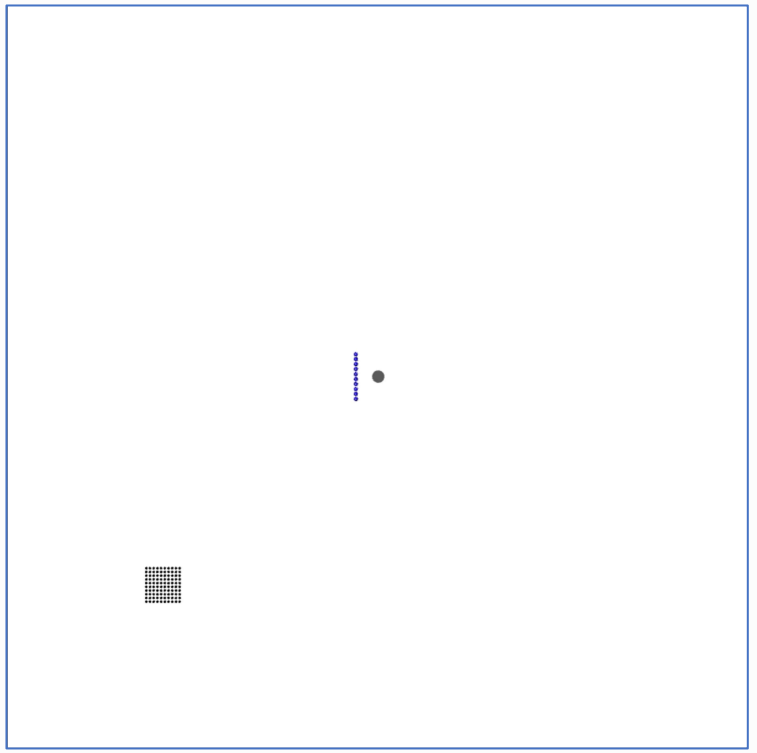
The Fibonacci sequence and the golden ratio show up over and over in computer science and mathematics.

http://www.algomation.com/algorithm/towers-hanoi-recursive-visualization

Golden Foraging Algorithm







A Most Irrational Foraging Algorithm

Abhinav Aggarwal
Department of Computer Science
University of New Mexico
Albuquerque, USA
abhiag@unm.edu

William F. Vining
Moses Biological Computation Lab
University of New Mexico
Albuquerque, USA
wfvining@cs.unm.edu

Diksha Gupta
Department of Computer Science
University of New Mexico
Albuquerque, USA
dgupta@unm.edu

Jared Saia
Department of Computer Science
University of New Mexico
Albuquerque, USA
saia@cs.unm.edu

Melanie E. Moses
Moses Biological Computation Lab
University of New Mexico
Albuquerque, USA
melaniem@unm.edu

ABSTRACT

We present a foraging algorithm, GoldenFA, in which search direction is chosen based on the Golden Ratio. We show both theoretically and empirically that GoldenFA is more efficient for a single

the flower [17]. The Golden ratio and Fibonacci numbers have been used in computer science for various applications like obtaining optimal schedules for security games [10], Fibonacci hashing [14], bandwidth sharing [8], data structures [4] and game theoretic mod-

Homework Problem 4

The Fibonacci sequence defined by

$$F = 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, \ldots$$

where the k^{th} term is given by

$$F_k = F_{k-1} + F_{k-2}$$

https://grader.mathworks.com/courses/11316-cs151-computer-science-fundamentals/assignments/31585-matlab-homework-1-expressions-flow-control-and-functions/problems/139176-iterative-fibonacci-vector-output-integer-datatype

Fibonacci Sequence

MATLAB Syntax in Yellow

In file Fibonacci.m

```
function fib = fibonacci(N)
fib = zeros(1,N, 'uint32')
fib(1) = 1
fib(2) = 1
for i = 3:N
fib(i) = fib(i-1) + fib(i-2)
End
```

In file MATLAB interpreter

```
fib = fibonacci(5)
fib = 1×5 uint32 row vector

      1 1 2 3 5
```

Python Syntax in Green

In file Fibonacci.py

```
def fibonacci( N ):
    fib = [0] * N
    fib[0] = 1
    fib[1] = 1
    for i in range(2,N):
        fib[i] = fib[i-1] + fib[i-2]

    return fib
```

In file Python3 interpreter

```
>>> import Fibonacci
>>> Fibonacci.fibonacci(5)
[1, 1, 2, 3, 5]
>>>
```

The Fibonacci Sequence is an example of a one dimensional array.

Two dimensional arrays (matrices) are extremely important in all areas of math, science, and engineering.

Next we will see how to define a matrix in Python and write a program to multiply two matrices together.

The Fibonacci Sequence is an example of a one dimensional array.

Two dimensional arrays (matrices) are extremely important in all areas of math, science, and engineering.

Next we will see how to define a matrix in Python and write a program to multiply two matrices together.

Matrix multiplication is a common computer task. It is important in everything from modelling pandemics, Google's web page search algorithm, and any video game graphics.

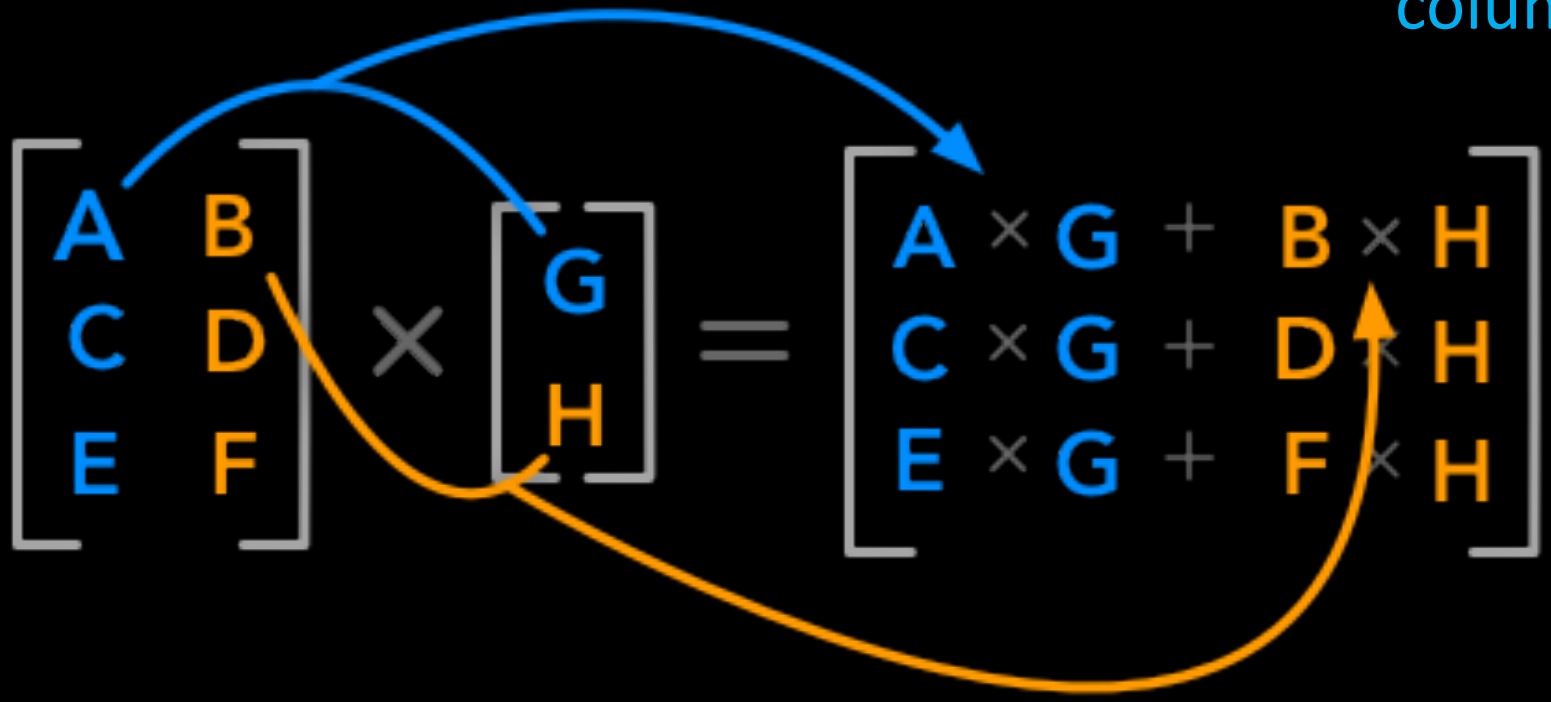
2 dimensional arrays in Python (Matrix)

```
>>> A = [1,2,3]
>>> A
[1, 2, 3]
>>> A = [[1,2,3],[4,5,6],[7,8,9]]
>>> A[1]
[4, 5, 6]
>>> A[1][2]
6
```

$$(\mathbf{AB})_{ij} = \sum_{k=1}^{m} A_{ik} B_{kj}$$

Where **AB** is the resulting matrix from our multiplication.

We multiply the elements in the rows of matrix B by the columns of the matrix A.



From geeksforgeeks.com

A B R

R = AB

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} =$$

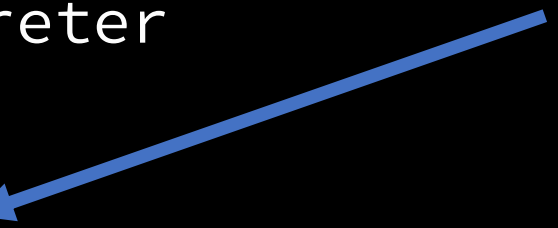
Matrix Multiplication

We want to write a function that takes two Matrices and returns their product (i.e. A x B).

```
>>> x = matrix_multiply(A,B)
```

Matrix Multiplication

```
In file Python3 interpreter
>>> import matmul
>>> A = [[1]*500]*500
>>> B = [[1]*500]*500
>>> x = matmul.matrix_multiply(A,B)
```



Convenient way to generate large matrices in a short command (all 1s)

Do you see how it works?

Matrix Multiplication: First we figure out how big the resulting matrix has to be.

In file matmul.py

```
# Program to multiply to matrices together. Arguments are matrices A and B.
# The result R = A x B
def matrix_multiply( A, B ):

# Determine the number of rows and columns in matrix A and matrix B
    A_num_rows = len(A)
    B_num_rows = len(B)
    A_num_cols = len(A[0])
    B_num_cols = len(B[0])

# Result will have the same number of rows as A and the number of cols of B
    R_num_rows = A_num_rows
    R_num_cols = B_num_cols
    R = [[0]*R_num_cols]*R_num_rows
```

Matrix Multiplication

In file matmul.py continued from previous slide…

```
    for i in range(len(A)):

        # iterating by column by B
        for j in range( B_num_cols ):

            # iterating by rows of B
            for k in range( B_num_rows ):
                R[i][j] += A[i][k] * B[k][j]


    return R
```

Timing how fast our code is…

In file matmul.py continued from previous slide…

```
import time

# Begin timing
tic = time.perf_counter()

    Do something cool that we want to time…

# The cool thing is done so record the end time
toc = time.perf_counter()

# pretty print
print(f"Completed multiplication in {toc - tic:0.4f} seconds")
```

Matrix Multiplication

In file matmul.py

```
import time

# Program to multiply to matrices together. Arguments are matrices A and B.
# The result R = A x B
def matrix_multiply( A, B ):

    # Begin timing
    tic = time.perf_counter()

    # Determine the number of rows and columns in matrix A and matrix B
    A_num_rows = len(A)
    B_num_rows = len(B)
    A_num_cols = len(A[0])
    B_num_cols = len(B[0])

    # Result will have the same number of rows as A and the number of cols of B
    R_num_rows = A_num_rows
    R_num_cols = B_num_cols
    R = [[0]*R_num_cols]*R_num_rows

    for i in range(len(A)):

        # iterating by column by B
        for j in range( B_num_cols ):

            # iterating by rows of B
            for k in range( B_num_rows ):
                R[i][j] += A[i][k] * B[k][j]

    # End timing and pretty print
    toc = time.perf_counter()
    print(f"Completed multiplication in {toc - tic:0.4f} seconds")

    return R
```

In file Python3 interpreter

```
>>> import importlib
>>> importlib.reload(matmul)
>>> A = [[1]*500]*500
>>> B = [[1]*500]*500
>>> x = matmul.matrix_multiply(A,B)
Completed multiplication in 27.7860 seconds
```

Matrix Multiplication

In file matmul.py

```
import time

# Program to multiply to matrices together. Arguments are matrices A and B.
# The result R = A x B
def matrix_multiply( A, B ):

    # Begin timing
    tic = time.perf_counter()

    # Determine the number of rows and columns in matrix A and matrix B
    A_num_rows = len(A)
    B_num_rows = len(B)
    A_num_cols = len(A[0])
    B_num_cols = len(B[0])

    # Result will have the same number of rows as A and the number of cols of B
    R_num_rows = A_num_rows
    R_num_cols = B_num_cols
    R = [[0]*R_num_cols]*R_num_rows

    for i in range(len(A)):

        # iterating by column by B
        for j in range( B_num_cols ):

            # iterating by rows of B
            for k in range( B_num_rows ):
                R[i][j] += A[i][k] * B[k][j]

    # End timing and pretty print
    toc = time.perf_counter()
    print(f"Completed multiplication in {toc - tic:0.4f} seconds")

    return R
```

In file Python3 interpreter

```
>>> import matmul
>>> A = [[1]*500]*500
>>> B = [[1]*500]*500
>>> x = matmul.matrix_multiply(A,B)
Completed multiplication in 27.7860
seconds
```

Here is an equivalent MATLAB program:

```
>> B = ones(500);
>> A = ones(500);
>> tic;
R=A*B;
toc;
Elapsed time is 0.011255
seconds.
```