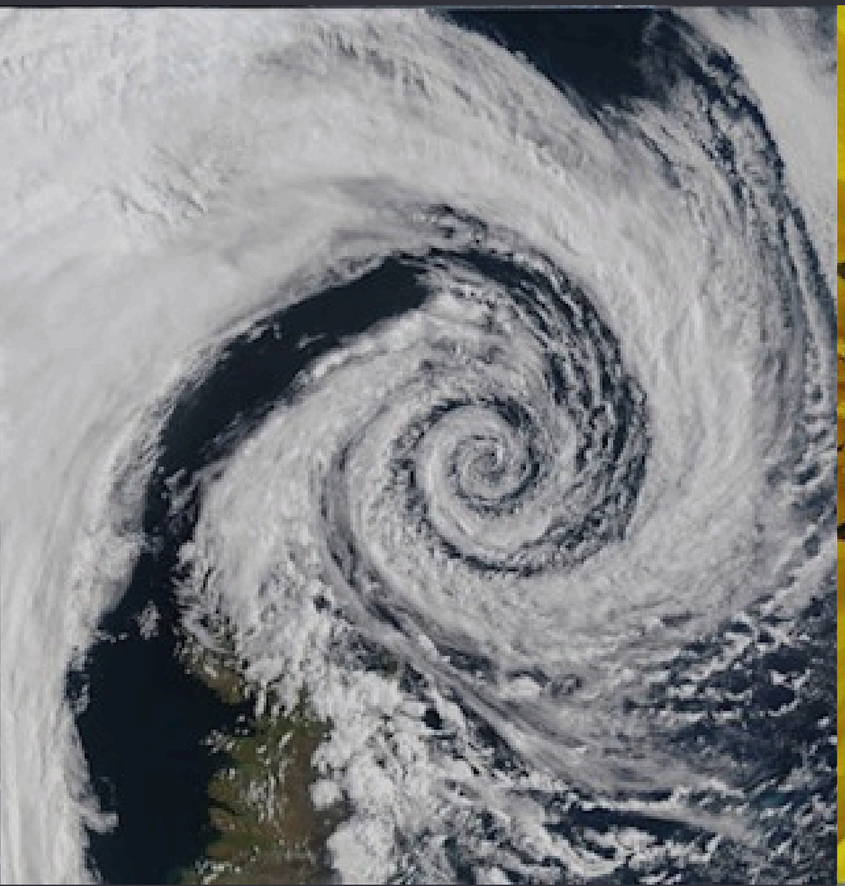
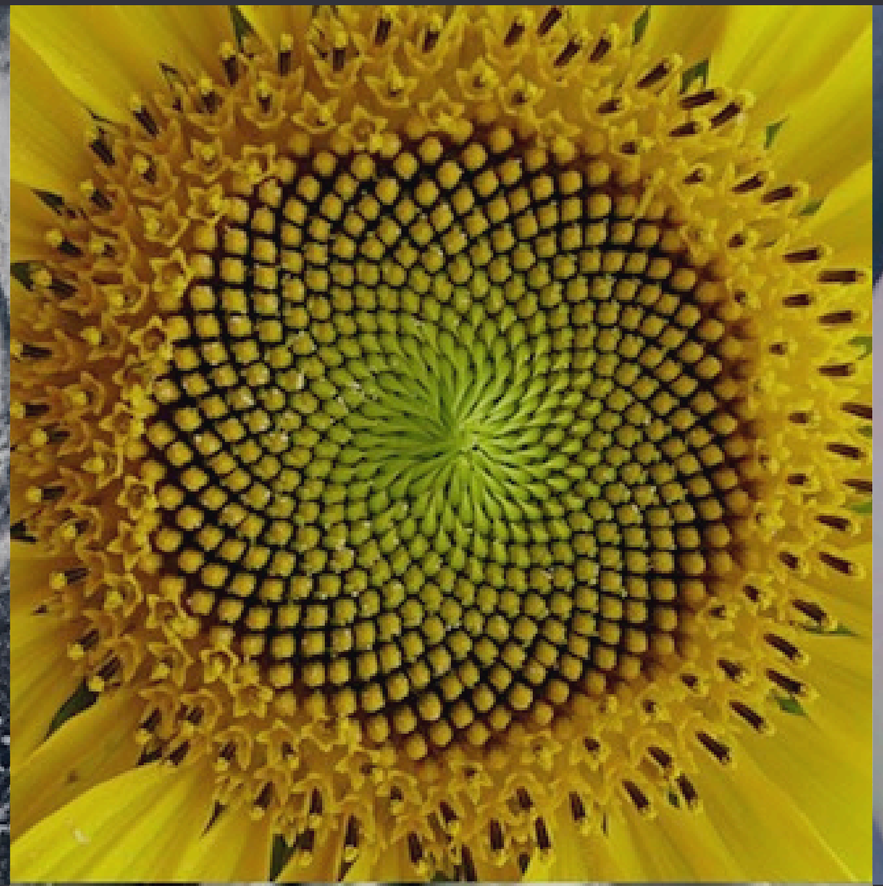
A Fibonacci-sorozat matematikája és megjelenése a természetben

„Miért van az, hogy a természetben a spirálok ‘rendben’ vannak?”

- A napraforgóban a magok nem össze-vissza nőnek, és tökéletesen kitöltik a teret.



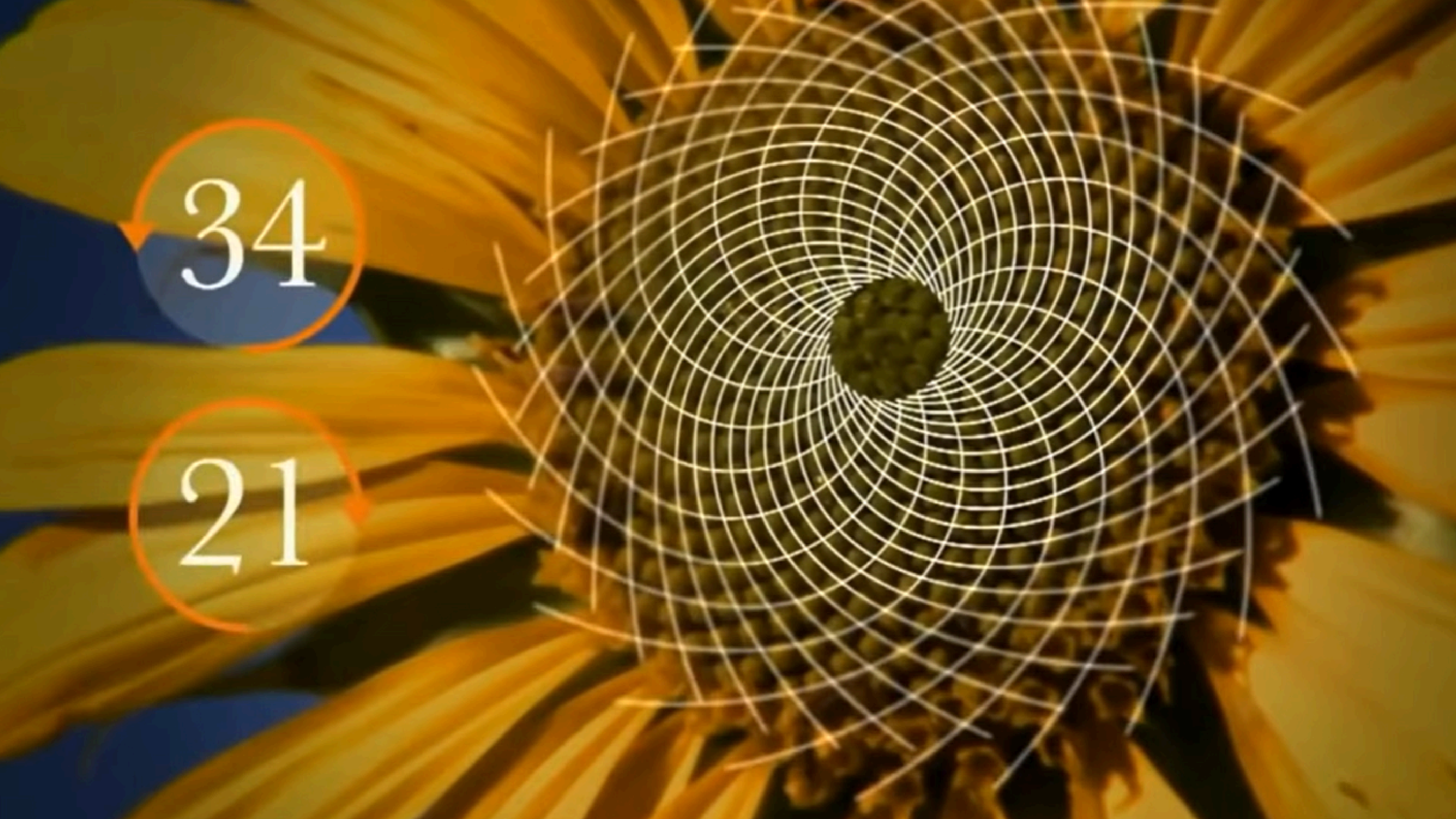


- **kagylókban,**
- **rózsafejekben,**
- **rovarok pajzsán is.**

A fenyőtoboz pikkelyei nem véletlenszerű irányban sorakoznak, hanem két, egymással ellentétes spirálrendszerben — és ezek száma gyakran két egymást követő szám: 8 és13.

Erre egy látványos példa:

Erre egy látványos példa:


34
21

A matematikában ezt a rendet **Leonardo Bonacci**, közismertebb nevén Fibonacci fogalmazta meg egy egyszerű kérdéssel a **13. század**ban:

„Hogyan szaporodik egy nyúlpopuláció, ha minden pár havonta egy új párt hoz létre?”

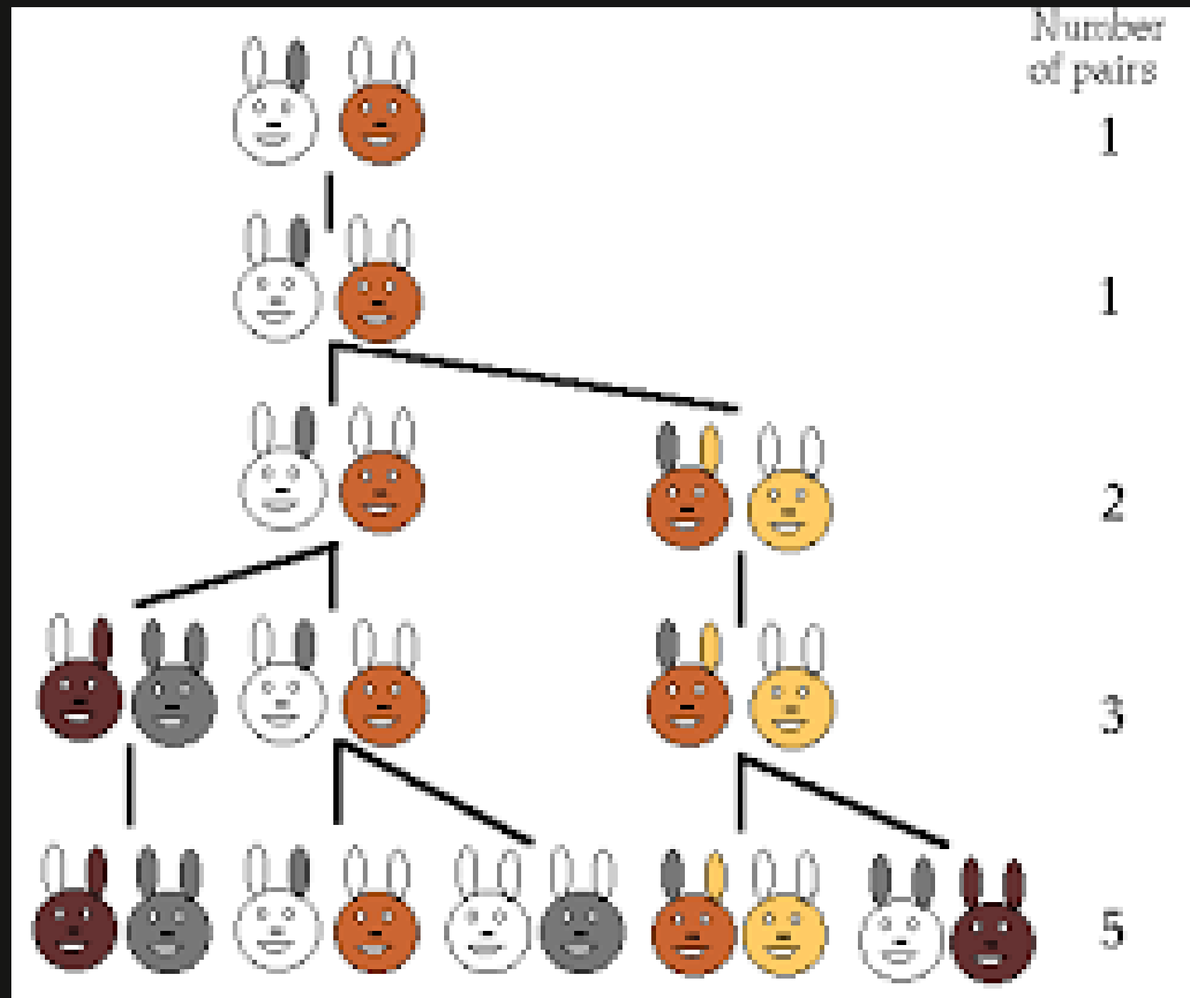
Leonardo Bonacci, közismert nevén **Fibonacci**, a Pisai Köztársaságból származó olasz matematikus volt, akit a „középkor legtehetségesebb nyugati matematikusának” tartanak.
Született: Pisa, Olaszország

„Hogyan szaporodik egy nyúlpopuláció, ha minden pár havonta egy új párt hoz létre?”

Alapfelételezések:

- Kezdetben **1 pár** nyúl van.
- A nyulak **1 hónap után** válnak termékennyé. (Tehát az új párok csak a következő hónapban kezdenek szaporodni.)
- Minden termékeny **pár havonta 1 új párt** hoz létre.
- Egyetlen nyúl sem pusztul el.

Ez a 4 feltétel biztosítja a sorozat szabályos növekedését.



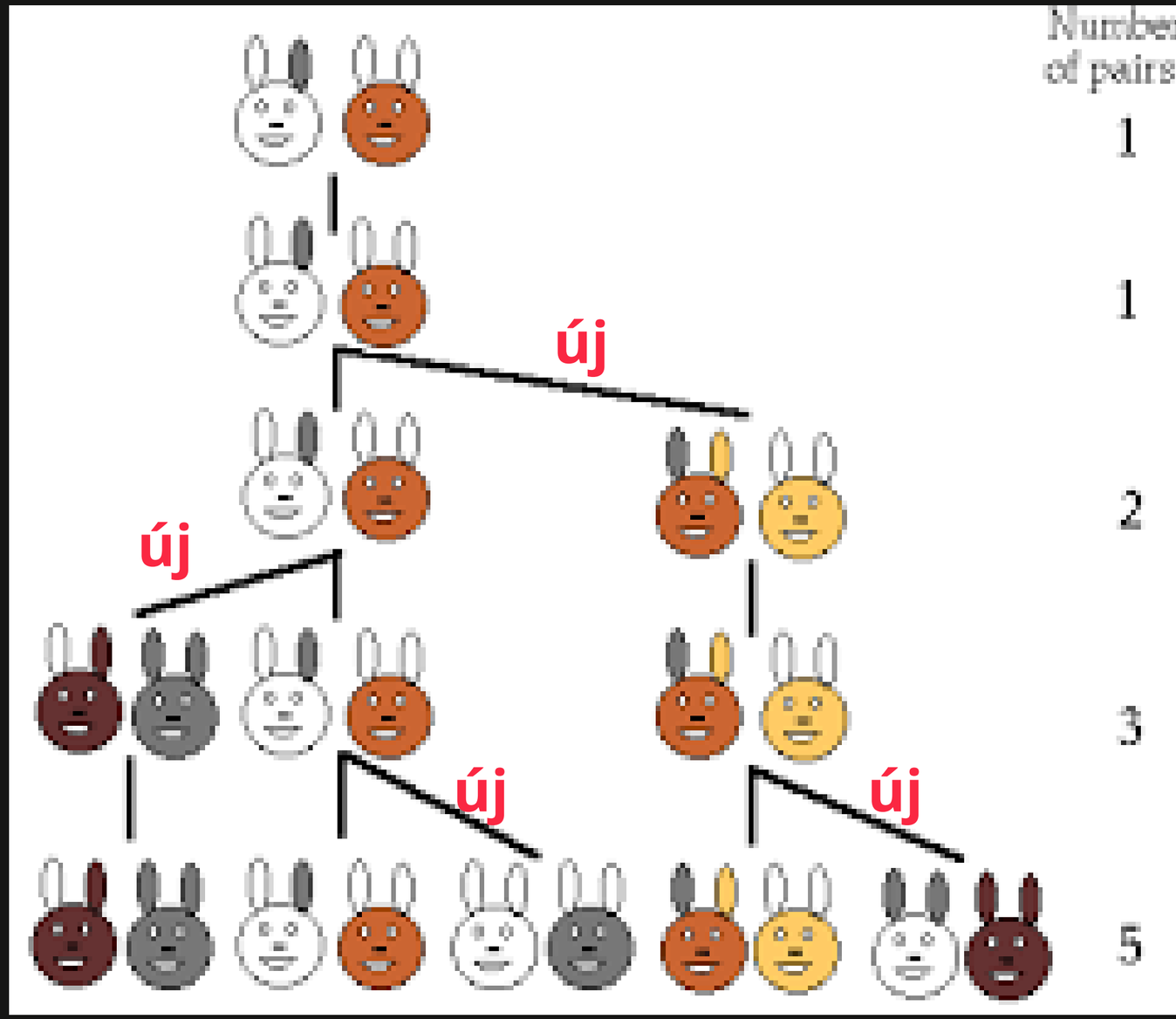
Hónapról hónapra lebontva, a logika levezetése

0.hónap
Van 1 pár nyúl.
Ez még nem szaporodik, mert fiatal.
-Összesen: 1 pár

1.hónap
A nyúl most már felnőtt → termékeny.
De csak most lett termékeny, ezért még nem hoz létre újat ebben a hónapban (Fibonacci így definiálta).
-Összesen: 1 pár

2.hónap
A „régi” pár (1 pár) most már szaporodik.
Létrejön 1 új pár.
Összesen: 2 pár (1 öreg + 1 fiatal)

3.hónap
Az 1 „öreg pár” újabb párt hoz létre.
A múlt hónapban született új pár még fiatal → nem szaporodik.
Tehát:
Öreg párok: 1, Fiatal párok (újak): 1
Újonnan születő pár: 1
Összesen: 3 pár

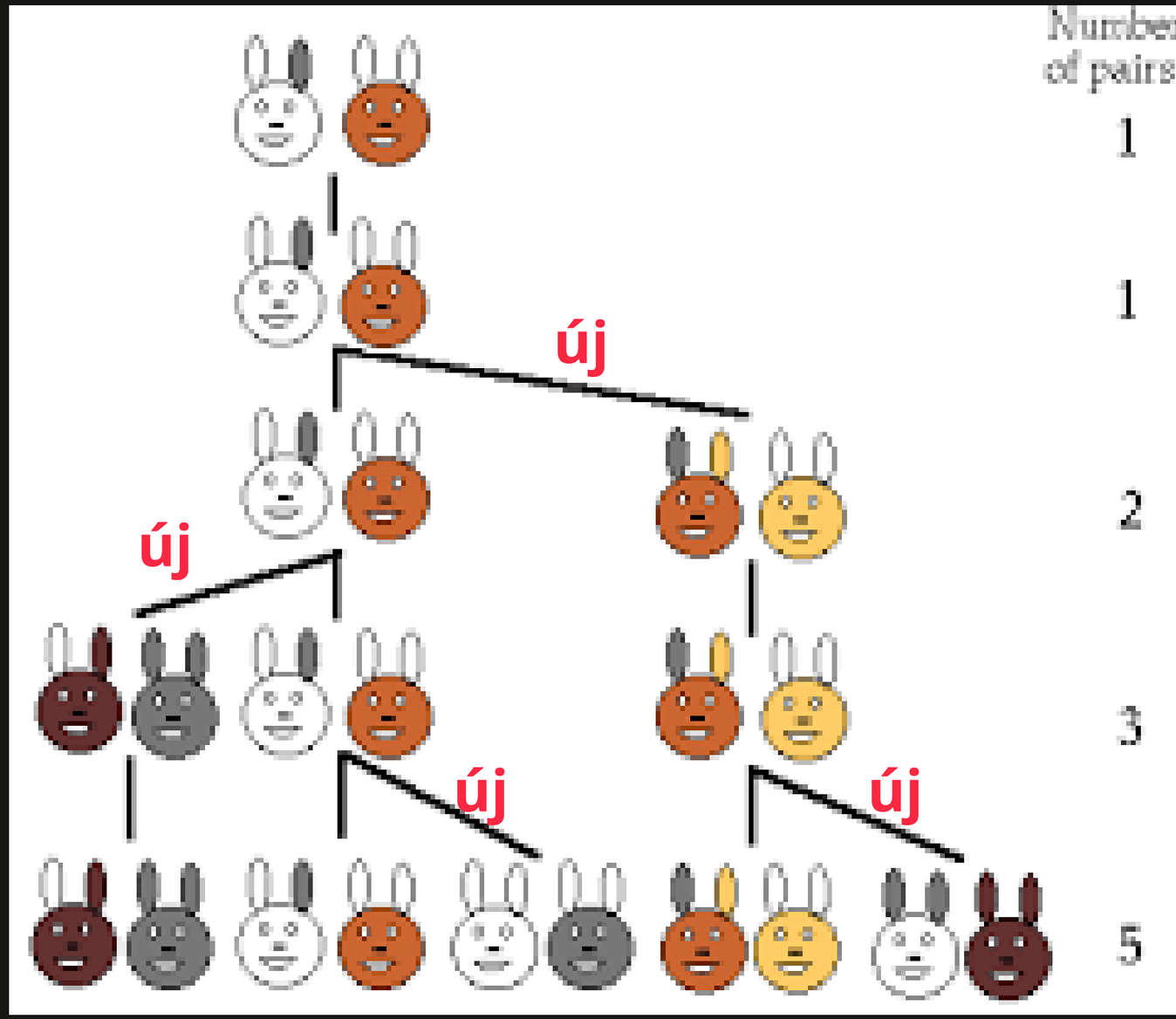


Hónapról hónapra lebontva, a logika levezetése

3.hónap
Az 1 „öreg pár” újabb párt hoz létre.
A múlt hónapban született új pár még fiatal → nem szaporodik.
Tehát:
Öreg párok: 1, Fiatal párok (újak): 1
Újonnan születő pár: 1
Összesen: 3 pár

4.hónap
Most már az előző havi fiatal pár is termékeny!
Öreg párok: 2
Mindkét öreg 1–1 új párt hoz létre → 2 új pár
Fiatal párok: 0 (mert mind termékennyé vált)
Összesen: 5 pár

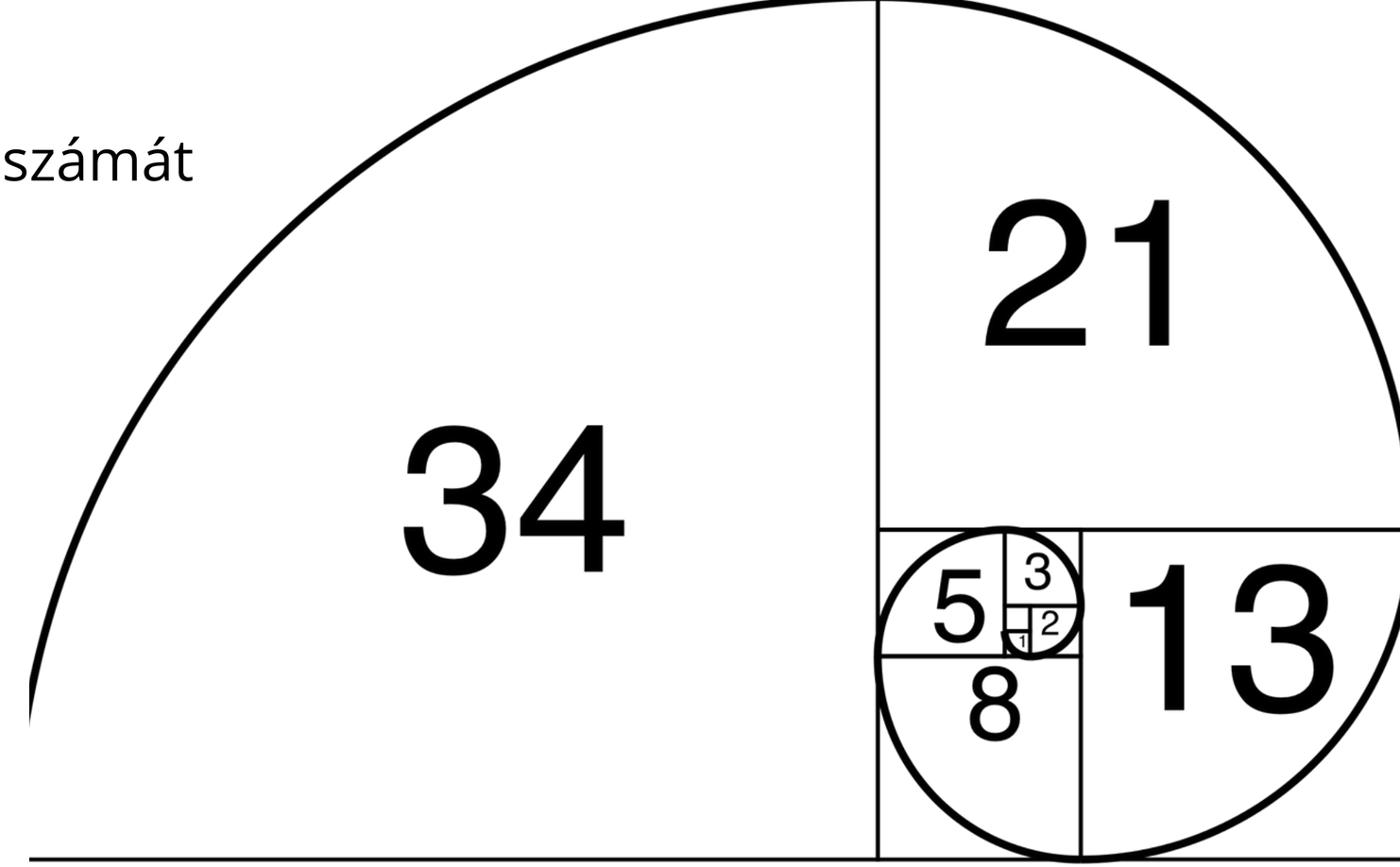
5.hónap
Az előző hónapban született 2 pár most még fiatal → ők nem szaporodnak.
Az összes régebbi pár (3 pár) újakat hoz létre.
Tehát:
Öreg párok: 3
Új fiatal párok: 3
Összesen: 8 pár



Ha csak a hónap végi összes párok számát nézzük:

Hónap Pár

1. 1
2. 1
3. 2
4. 3
5. 5
6. 8

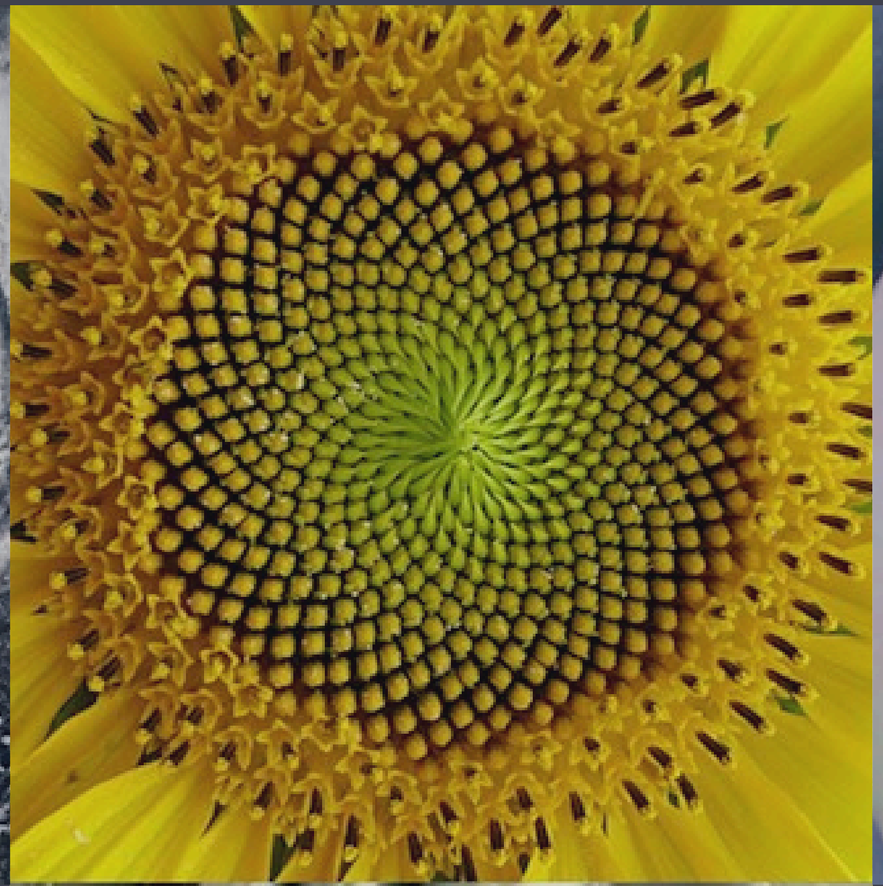


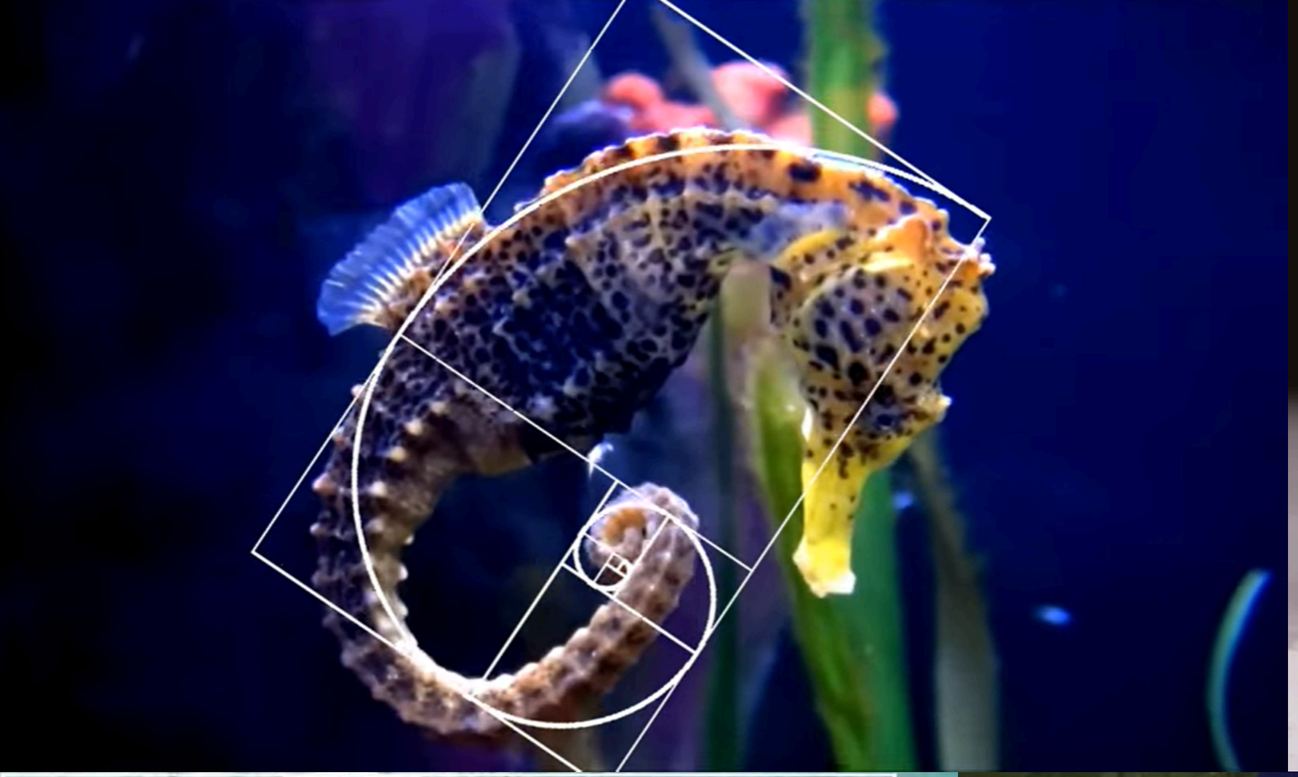
Ez pontosan a Fibonacci-sorozat első tagjai:

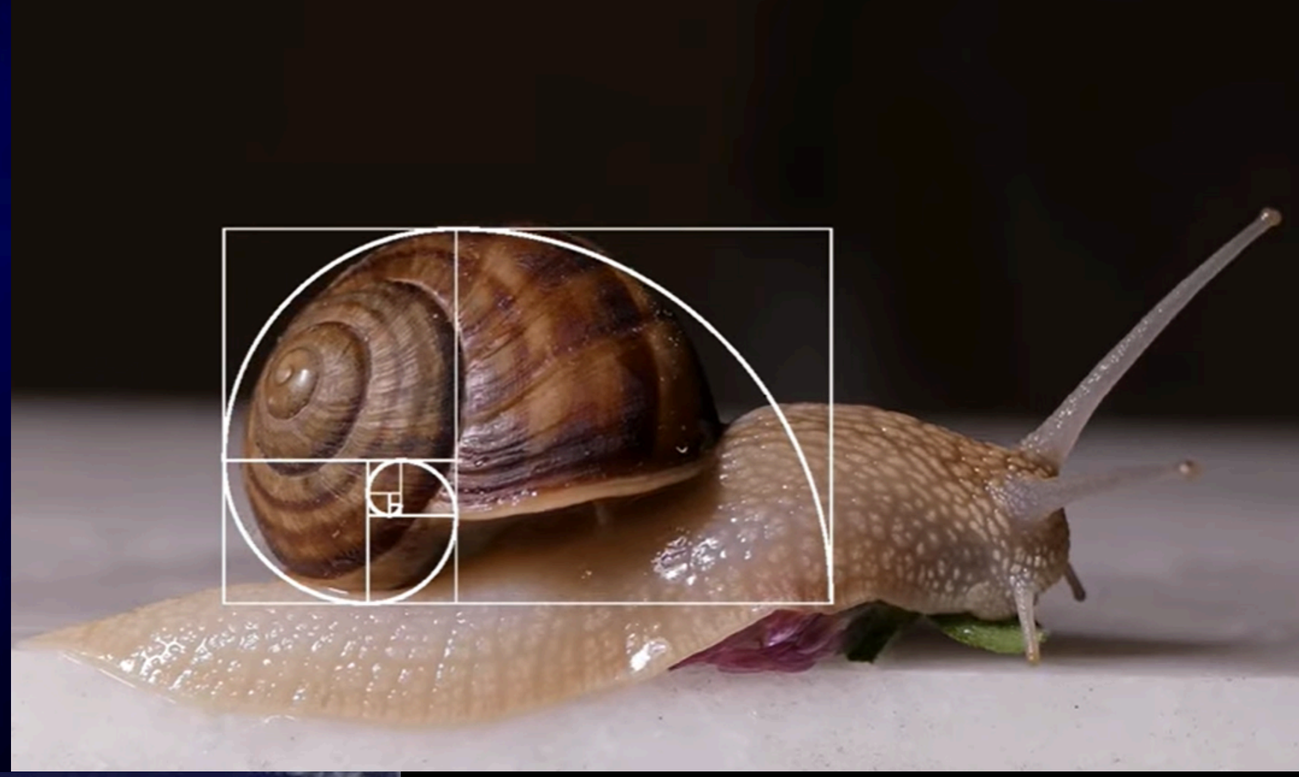
1, 1, 2, 3, 5, 8, 13, 21,34...

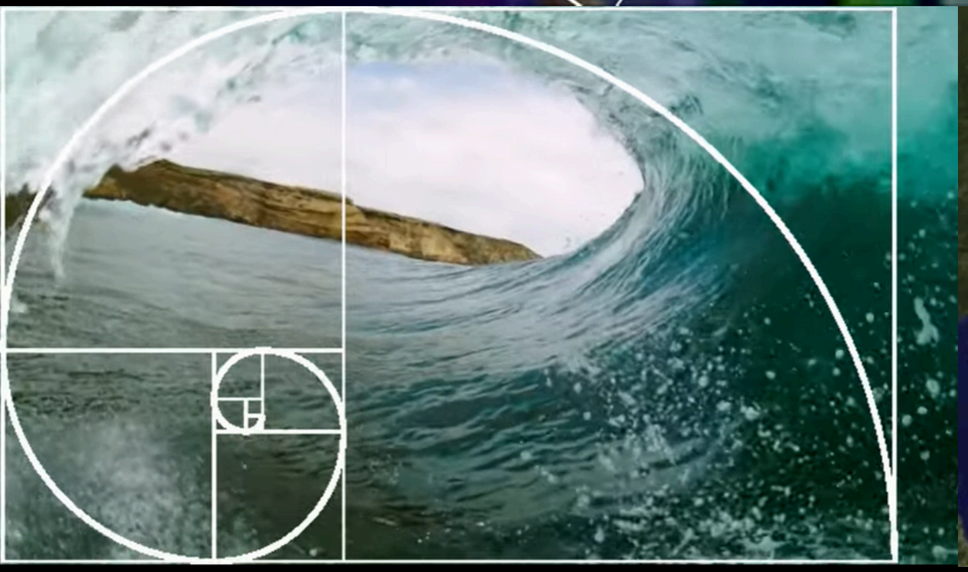
A természet mintha ugyanazt a matematikát használná, amit Fibonacci leírt — csak éppen mérnöki pontossággal és hibátlan gazdaságossággal.

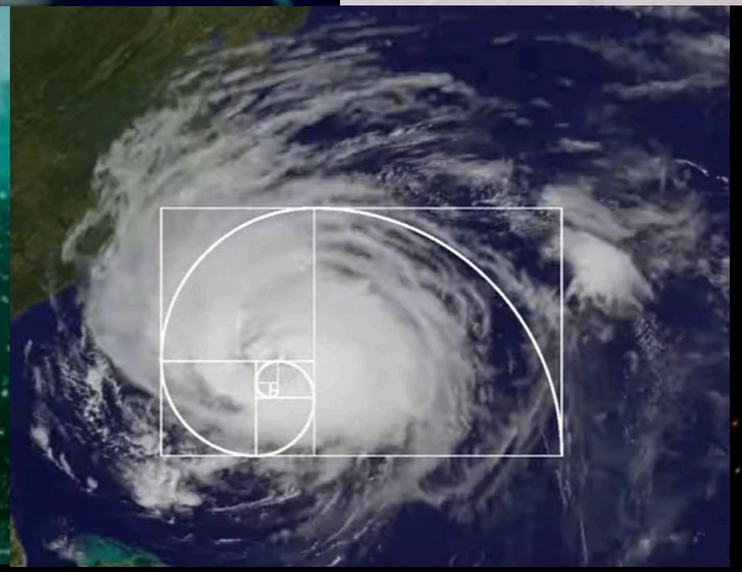
A következőkben azt vizsgáljuk, hogyan épül fel ez a sorozat matematikailag, és pár példát, miként jelenik meg a természetben.











A matematikai logika

1) Az **összes új** pár = az **előző havi öregek** száma = **F(n−1)**
2) Az **összes öreg** pár = az **előző havi összes** pár = F**(n)**
3) Az **új összes pár** = **öregek + újak** = **F(n) + F(n−1)** = F(n+1)

Így kapjuk:

F(n+1) = F(n) + F(n−1) * F(n+1) = F(n) + F(n-1)* F(n+1) = F(n) + F(n−1)

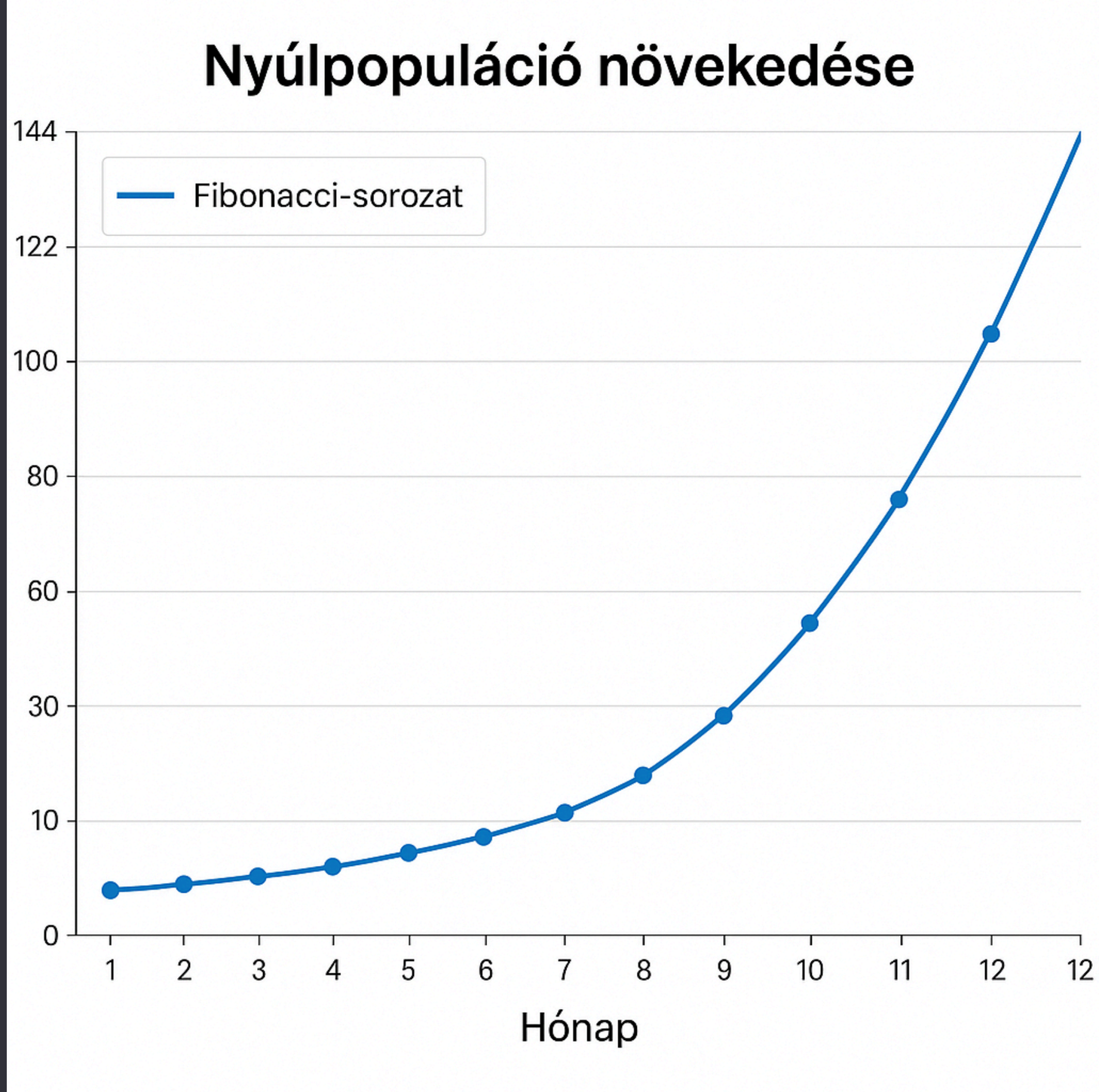
A Fibonacci-sorozat azért jön ki a nyúlproblémából, mert a következő havi nyúlpárok száma mindig az előző havi összes és az azt megelőző havi összes összegéből áll össze — pontosan úgy, ahogy a sorozat definíciója megkívánja.

A sorozat definíciója

A Fibonacci-sorozat egyik legegyszerűbb megfogalmazása a **rekurzió**, ahol egy **függvény önmagára hivatkozik**, és **minden értéket az előző kettőből** számít ki. Matematikai szempontból ez rendkívül elegáns, mert pontosan leképezi a definíciót: **„az n-edik elem az előző kettő összege.”** A programozásban azonban a rekurzió lassú lehet, mert minden hívás újra kiszámítja az előző értékeket, így az időigény exponenciálisan növekedhet. Ennek alternatívája az **iteratív** módszer, ahol egymást követve tároljuk az értékeket és végigszámoljuk a sorozatot, így a függvény nem hívja önmagát. Az iteráció nem olyan „szép”, mint a rekurzió, de sokkal hatékonyabb, **időben lineáris és memóriahasználat szempontjából is gazdaságosabb**. Ez jól példázza, hogy a matematikai elegancia és a gyakorlati hatékonyság nem mindig esik egybe, és a választás attól függ, melyiket tartjuk elsődlegesnek.

Tehát: a programozásban nem a leghatékonyabb módszer, mert

- lassú lehet,
- az időigény exponenciálisan és folytonosan növekedhet.



készítette: Nagy Virág (2025)

Köszönöm a figyelmet!