Introduction to Stock – to – Flow Model

The Stock to Flow model (SF or S2F) is a way to measure the abundance of a given resource. The Stock / Flow ratio is the quantity of a resource held in reserves divided by the quantity that is produced annually. The Stock to Flow model is generally applied to natural resources. Take the example of gold. Although estimates may vary, the World Gold Council estimates that some 190,000 tonnes of gold have already been mined. This value (that is, the total stock) is what we can call stock. In the meantime, there are about 2,500-3,200 tons of gold mined each year. This value is what we can call a flow. We can calculate the ratio of inventory to flow using these two metrics.

So, what this really means is; the higher the Stock / Flow ratio, the less new supply enters the market in relation to the total supply. As such, an asset with a higher stock-to-flow ratio should, in theory, retain its value in the long run. In contrast, consumables and industrial commodities will normally have a low Inventory / Flow ratio.

Stock ratio to Bitcoin flow?

The current supply of Bitcoin in circulation is approximately 18 million bitcoins, while the new supply is approximately 0.7 million per year. At the time of this writing, the ratio of stock to Bitcoin flow is hovering around 25. After the next halving in May 2020, the ratio will increase to 50 degrees.

Limitations of the Stock to Flow model

Although stock for flow is an interesting model for measuring scarcity, it does not take all parts of the picture into account. The models are as solid as their assumptions. On the one hand, Stock to Flow is based on the assumption that scarcity, as measured by the model, must generate value. According to critics of Stock to Flow, this model fails if Bitcoin lacks other useful qualities besides the scarcity of supply. The scarcity, the predictable flow and the global liquidity of gold have made it a relatively stable reserve of value compared to fiat currencies, which are subject to devaluation.

According to this model, Bitcoin's volatility is also expected to decrease over time. The valuation of an asset requires taking into account its volatility. If volatility is predictable to some extent, the valuation model can be more reliable. However, Bitcoin is known for its large price movements. Although volatility may be decreasing at the macro level, Bitcoin has been priced in a free market since its inception. This means that the price is mainly self-regulated in the open market by users, traders and speculators. Combine that with relatively low liquidity, and Bitcoin is likely to be more exposed to sudden spikes in volatility than other assets. Therefore, the model may also not be able to cope with this.

Other external factors, such as the Black Swan's economic events, may also undermine this model. A Black Swan event has an element of surprise. Historical data cannot take into account unknown events.

Question 2a

WHY IT IS A BAD IDEA TO USE RECURSION METHOD TO FIND THE FIBONACCI OF A NUMBER.

It is way too expensive. if we use recursion to find the Fibonacci, we will end up “overlapping” and thus, take additional time and computing power.

Where the inputs are very small values, recursion could be used to do a few unnecessary computations, and still end up with a correct answer. As the input gets larger, and it doesn’t even need to be that large, maybe only 50 or 100 is large enough to the point where all those extra “overlapping” computations will add up, and the program will not finish. With very large inputs, it is guaranteed that the program will not finish(or at least in a reasonable time). That is why it is definitely a bad idea to use recursion.

Question 2b

Using C++, a function that takes proth numbers and checks for proth primes.

C++ implementation of the above approach

#include <bits/stdc++.h>

using namespace std;

int prime[1000000];

// Calculate all primes upto n.

void SieveOfEratosthenes(int n)

{

// Initialize all entries it as true.

// A value in prime[i] will finally

// false if i is Not a prime, else true.

for (int i = 1; i <= n + 1; i++)

prime[i] = true;

prime[1] = false;

for (int p = 2; p \* p <= n; p++) {

// If prime[p] is not changed,

// then it is a prime

if (prime[p] == true) {

// Update all multiples of p

// greater than or equal to

// the square of it numbers

// which are multiple of p and are

// less than p^2 are already been marked.

for (int i = p \* p; i <= n; i += p)

prime[i] = false;

}

}

}

// Utility function to check power of two

bool isPowerOfTwo(int n)

{

return (n && !(n & (n - 1)));

}

// Function to check if the Given

// number is Proth number or not

bool isProthNumber(int n)

{

int k = 1;

while (k < (n / k)) {

// check if k divides n or not

if (n % k == 0) {

// Check if n/k is power of 2 or not

if (isPowerOfTwo(n / k))

return true;

}

// update k to next odd number

k = k + 2;

}

// If we reach here means there

// exists no value of K such

// that k is odd number and n/k

// is a power of 2 greater than k

return false;

}

// Function to check whether the given

// number is Proth Prime or Not.

bool isProthPrime(int n)

{

// Check n for Proth Number

if (isProthNumber(n - 1)) {

// if number is prime, return true

if (prime[n])

return true;

else

return false;

}

else

return false;

}

// Driver Code

int main()

{

int n = 41;

// if number is proth number,

// calculate primes upto n

SieveOfEratosthenes(n);

for (int i = 1; i <= n; i++)

// Check n for Proth Prime

if (isProthPrime(i))

cout << i << endl;

return 0;

}

Maths – Problem

y=√(〖(x+6)〗^2+25)+ √(〖(x-6)〗^2+121)

y=〖[√(〖(x+6)〗^2+25)]〗^(1/2)+ 〖[√(〖(x-6)〗^2+121)]〗^(1/2)

dy/dx= 1/2 . 2(x+6)/[(x+6)^2+25]^(1⁄2) + 1/2 . 2(x-6)/[(x-6)^2+121]^(1⁄2)

dy/dx= (x+6)/√(〖(x+6)〗^2+25)+ (x-6)/√(〖(x-6)〗^2+121)

At Critical point dy/dx=0

(x+6)/√((x+6)^2+25)+ (x-6)/√((x-6)^2+121)=0

-6 < x < 6

(x+6) √((x-6)^2+121)+(x-6) √(〖(x+6)〗^2+25)=0

(x+6) √((x-6)^2+121)=-(x-6) √(〖(x+6)〗^2+25)

Squaring both sides

(x+6)^2 [(x-6)^2+121]=(x-6)[(x+6)^2+25]

(x^2+12x+36)(x^2-12x+36+121)=(x^2-12x+36)(x^2-12x+36+25)

(x^2+12x+36)(x^2-12x+157)=(x^2-12x+36)(x^2-12x+61)

x^4-12x^3+157x^2+12x^3-144x^2+1884x+36x^2-432x+5652=x^4+121x^3+61x^2-12x^3-144x^2-732x+36x^2+432x+2196

Collecting like terms

〖13x〗^2+1452x+5652=-〖83x〗^2-300x+2196

〖96x〗^2+1752x+3456=0

Dividing through by 8

〖12x〗^2+219x+432=0

x=(-b±√(b^2-4ac))/2a Where b=219 ,a=12 ,c=432

x=(-219±√((219)^2-4(12)(432) ))/2(12)

x=(-219±√27225)/24

x=(-219±165)/24

x=-2.25 or-16

Only x=-2.25 Satisfies-6<x<6,therefore y will have only one value for x which can be assumed to be minimum But let check

For minimum Value,

(d^2 y)/(d^2 x)>0

dy/dx=(x+6)^2 [(x+6)^2+25]^((-1)⁄2)=(x-6) [(x-6)^2+121]^((-1)⁄2)

(d^2 y)/(d^2 x)= (-1)/2 (x+6). 2(x+6)/√([(x+6)^2+25]^3 )+ 1/√((x+6)^2+25)-(-1)/2 (x-6). 2(x-6)/√([(x-6)^2+121]^3 )+1/√((x-6)^2+121)

(d^2 y)/(d^2 x)= (-〖(x+6)〗^2)/√([(x+6)^2+25]^3 )+ 1/√((x+6)^2+25)- 〖(x+6)〗^2/√([(x-6)^2+121]^3 )+1/√((x-6)^2+121)

Put x=-2.25

(d^2 y)/(d^2 x)= (-14.0625)/244.14+1/6.25-68.0625/2599+1/13.75

(d^2 y)/(d^2 x)=0.149 hence confirming that x=-2.25 is a minimum

y=√(〖(x+6)〗^2+25)+ √(〖(x-6)〗^2+121)

y=√(〖(-2.25+6)〗^2+25)+ √(〖(-2.25-6)〗^2+121)

y=√39.0625+ √189.0625

y=6.25+ 13.75

y=20 Ans.