AGBO CHINEDU'S RESEARCH WORK AS A REQUIREMENT FOR THE QUALIFICATION OF THE RESEARCH ASSISTANT JOB ROLE.

FINANCE

1. Write a 500-word explanation of Bitcoin stock-to-flow model and make an argument for why it is a bad model?

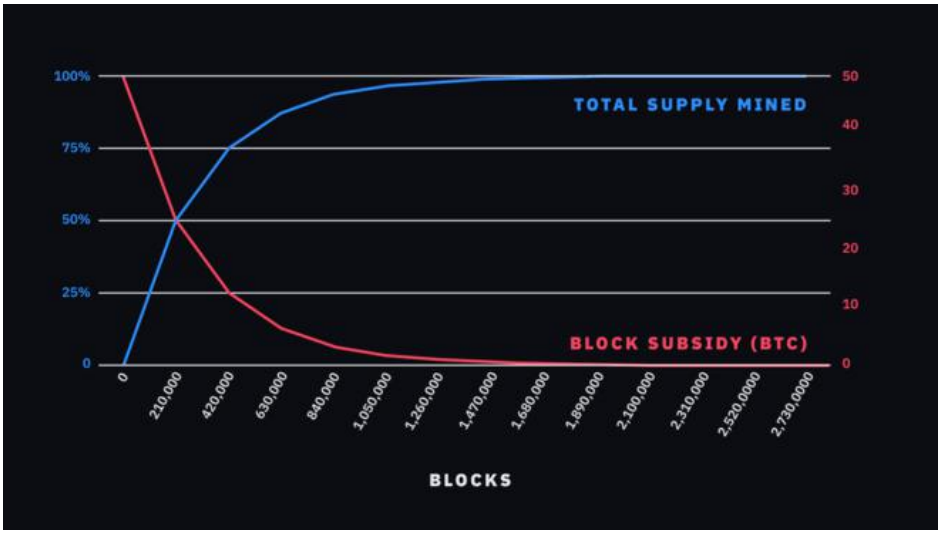
Bitcoin Stock to Flow Model

Stock to flow model is a way to measure the abudance of a particular resource. The Stock to Flow model measures the relationship between the currently available stock of a resource and its production rate.

The stock to flow ratio is the amount held in reserves divided by the amount produced annually.

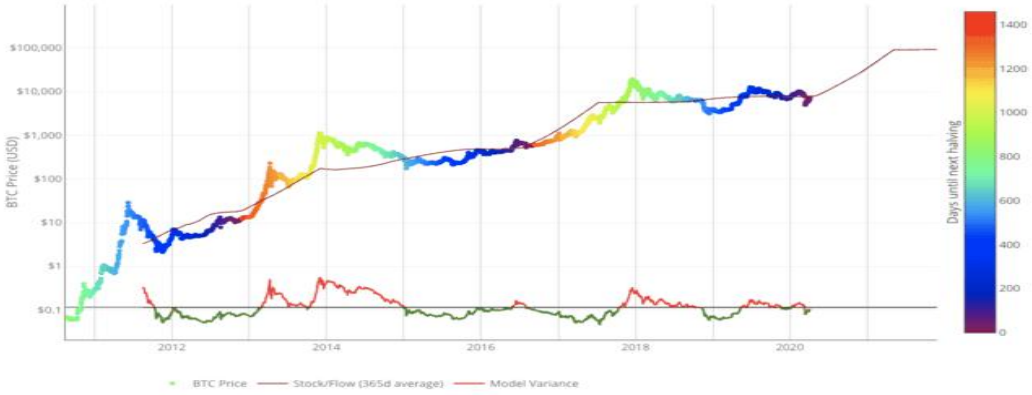
Bitcoin stock to flow model was published in march 2019. The model essentially treats bitcoins comparably to scarce commodities, like gold or silver.

According to the advocates of the Stock to Flow model, Bitcoin is a similar resource. It's scarce, relatively costly to produce, and its maximum supply is capped at 21 million coins. 210,000 blocks (roughly four years).



The current circulating supply of bitcoin is approximately 18 million bitcoins, while the new supply is approximately 0.7 million per year. Curently Bitcoin's Stock To Flow ratio is hovering at around 25.

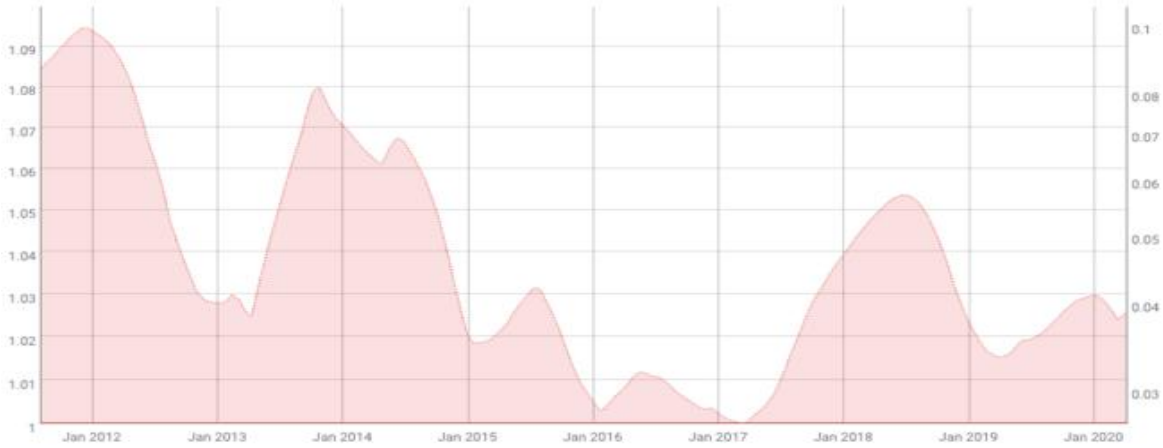
In the image below, you can see the historical relationship of the 365-day moving average of Bitcoin's Stock to Flow with its price. We've also indicated the dates of theBitcoin Halvings on the vertical axis.



Why Bitcoin Stock to Flow model is a bad model.

While Stock to Flow is an interesting model for measuring scarcity, it doesn't account for all parts of the picture. Models are only as strong as their assumptions. For one thing, Stock to Flow relies on the assumption that scarcity, as measured by the model, should drive value. According to critics of Stock to Flow, this model fails if Bitcoin doesn't have any other useful qualities other than supply scarcity.

According to this model, Bitcoin's volatility should also decrease over time. This is confirmed by historical data from Coinmetrics.



200-day Moving Average of 180-day Volatility of Bitcoin. Source: Coinmetrics.io

The valuation of an asset requires taking into account its volatility. If the volatility is predictable to some extent, the valuation model may be more reliable. However, Bitcoin is notorious for its large price moves.

While volatility might be decreasing on the macro level, Bitcoin has been priced in a free market from its inception. This means that the price is mostly self-regulated on 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 of volatility than other assets. So the model may not be able to account for this either.

Other external factors, such as economic Black Swan events, could also undermine this model. Though it's worth noting that the same applies to essentially any model that tries to predict the price of an asset based on historical data. A Black Swan event, by definition, has an element of surprise. Historical data can't account for unknown events

In Summary, Bitcoin has only been around for a little more than ten years. Some might argue that long-term valuation models like the Stock to Flow need a larger data set for more reliable accuracy.

B. (Please show your workings). Yara Inc is listed on the NYSE with a stock price of $40 - the company is not known to pay dividends. We need to price a call option with a strike of $45 maturing in 4 months. The continuously-compounded risk-free rate is 3%/year, the mean return on the stock is 7%/year, and the standard deviation of the stock return is 40%/year. What is the Black-Scholes call price?

SOLUTION:

C = $S_t N(d_1) - Ke^{-rt} N(d_2)$

Where:

$$d_1 = \frac{in\frac{s_t}{K} + (r + \sigma_v{}^2)t}{\sigma_s\sqrt{t}}$$

$$d_1 = \frac{in\frac{40}{45} + (0.03 + 0.4^2)\frac{1}{3}}{0.4\sqrt{\frac{1}{3}}}$$

$$d_1 = \frac{-0.11778 + 0.063}{0.2309}$$

$$d_1 = -0.235774$$

$$d_2 = d_1 - \sigma_s\sqrt{t}$$

$$d_2 = -0.235774 - 0.230940$$

$$d_2 = -0.466714$$

$$N = \frac{100\% - mean\%}{SD\%}$$

$$\frac{100 - 7}{40}$$

$$2.325\%$$

$$C = S_t N(d_1) - Ke^{-rt}N(d_2)$$

$$C = 40(0.02325)(0.235774) - 45\exp^{(-0.03(1/3))}(0.02325)(0.466714)$$

$$C = 0.21926982 - 0.4631654$$

$$C = 0.24389556$$

COMPUTER SCIENCE

2. Why is it a bad idea to use recursion method to find the fibonacci of a number?

Answer:

The reason why using recursion to find the fibonacci of a number is a bad idea is becaue if we use recursion , we will end up “overlapping” and thus, take additional time and computing power.

In other words, as your 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, making your program unable to finish. Thus with very

large inputs, your program will not finish (or at least in a reasonable time assuming you have places to be). Also the operation using recursion is way too expensive.

Furthermore Recursion used in long series buries the processor in a very long succession of “calls” with little or no processing in between them, which is probably the most inefficient thing a computer can do. In a long series, it can swamp the push-down list and cause a return to burning regular memory, and then cause swapping onto mass storage.

B.Write a function that takes in a Proth Number and uses Proth's theorem to determine if said number is prime? You can write this in any programming language but C/C++/Golang are preferred.

SOLUTION:

In mathematics, a Proth number is a positive integer of the form

$\mathbf{n = k * 2^n + 1}$

where k is an odd positive integer and n is a positive integer such that $2^n > k$.

The first few Proth numbers are –

3, 5, 9, 13, 17, 25, ……

```
Input: 25
Output: YES
    Taking k= 3 and n= 3,
    25 can be expressed in the form of
    (k.2^n + 1) as (3.2^3 + 1)
```

```
// CPP program to check Proth number

#include <bits/stdc++.h>
using namespace std;

// 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;
}

// Driver code
int main()
{

    // Get n
    int n = 25;

    // Check n for Proth Number
    if (isProthNumber(n - 1))
        cout << "YES";
    else
        cout << "NO";

    return 0;
}
```

MATHEMATICS

(Please show your workings). Over all real numbers, find the minimum value of a positive real number, y such that

y=sqrt((x+6)2+25)+sqrt((x−6)2+121)

SOLUTION:

$$y = sqrt((x+6)^2 + 25) + sqrt((x-6)^2 + 121)$$
$$y = sqrt((x+6)(x+6) + 25) + sqrt((x-6)(x-6) + 121)$$
$$y = ((x+6)+5) + ((x-6)+11)$$
$$y = (x+11) + (x+5)$$
$$x_1 = -5$$
$$x_2 = -11$$
$$y = -5 + 11$$
$$y_1 = 6$$
$$y = -11 + 6$$
$$y_2 = -5$$