1.FINANCE:

•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 (S2F) by plan B, is a popular model that is used to predict the price and rise of the bitcoin in future (based on its scarcity).**

  The stock to flow model is not an entirely accurate model to me because of several reasons I'll be stating below:

Not everyone agrees on what is gold's stock-to-flow ratio

Although most quants agree that gold's stock-to-flow ratio is in the 60s, a few believe it's much higher. For example, Philip Barton, a gold analyst, argues that gold's stock to flow is between 400 and 800!

Admittedly, Barton is an outlier. The consensus is that gold's stock-to-flow ranges between 50 and 70. Still, it's worth noting that some believe there is far more stock out there than we realize. How could gold's stock-to-flow ratio be 800?

The main reasoning is that when humans first started collecting gold, there was a lot of low-hanging fruit. Enormous gold nuggets were easy to grab in the streams and other sources. It's reasonable to assume that for billions of years, gold nuggets simply sat in the streams since no animal valued them.

When humans began collecting gold, the global gold stock must have soared exponentially. Its curve must have looked like the first decade of bitcoin's supply curve.

That's why Barton and others believe that gold's stock-to-flow ratio is 10 times higher than what most people think it is.

If that's true, then bitcoin's stock-to-flow model is somewhat inaccurate since it predicts that bitcoin will exceed gold's stock-to-flow ratio by 2025.

This problem is the least of all the problems with bitcoin's stock-to-flow narrative. It gets worse.

Gold's stock-to-flow isn't fixed

Plan B presents timelines that show bitcoin's ever-rising stock-to-flow ratio. In those charts, he depicts gold (and silver) as a single data point, as if their stock-to-flow ratios are constant.

Bitcoin's stock-to-flow model gives you the impression that gold's stock-to-flow is nearly constant.

The reality is that gold's stock-to-flow ratio is constantly fluctuating.

Gold's stock-to-flow ratio has gone as low as 45 in 1940 and as high as 90 in 1920. Gold has bounced around those extremes.

Gold's stock-to-flow does not drive its price

Plan B observed that because bitcoin's mining flow gets cut in half every four years, bitcoin's stock-to-flow ratio leaps every 4 years. These halving events boost Bitcoin's stock-to-flow ratio dramatically. It's reasonable to assume that the contracting supply is fueling the BTC's dramatic price rise.

Intuitively, the stock-to-flow model makes sense once you conduct a simple thought experiment. Imagine if the flow of gold suddenly slowed to a trickle. Instead of 3,000 tons of gold being mined annually, we only managed to dig 3 tons of gold each year.

What do you predict would happen to gold's price?

It would skyrocket, right?

That's because investors, national banks, jewelry makers, phone makers (0.034 grams of gold in each phone), and other industries that use gold would have to fight over a (nearly) fixed supply.

Still, if the stock to flow ratio were such a great price predictor, then we should see gold analysts use it all the time.

But they don't. Gold bugs only cite gold's stock-to-flow to help explain why gold has monetary value. But they don't use it to predict gold's price.

That's because, gold's stock-to-flow ratio is sometimes uncorrelated to gold's price.

In short, gold's price is uncorrelated to its stock-to-flow ratio.

I haven't read a single gold expert argue that stock to flow drives the price of gold over the years. As proof, at the bottom of this article, you can watch my interview with Jan Nieuwenhuijs, a famous gold researcher and analyst.

However, according to the S2F model, the alleged driver of bitcoin's price rise is its ever-increasing stock-to-flow ratio. As bitcoin's stock-to-flow ratio goes up, bitcoin's price follows. That is the thesis.

That implies that if bitcoin's stock-to-flow ever stabilizes, then its price should stabilize too.

Gold's stock-to-flow and price history tells a different story. Over the last 120 years, gold's STF ratio has cycled up and down while gold's inflation-adjusted price has gone all over the place.

The number of Bitcoin transactions right from time has always gone up, so as the price, presently the transactions are barely rising, because it can't do anymore transactions. The bitcoin community (the people who really matter) do not have the to increase the block size of the bitcoin.

•(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?

Answer

NB:

S_0 => Stock Price

X => Exercise Price (Strike Price)

R => Risk Free Increase Rate

T => Time

σ => Standard Deviation If a Log Returns (Volatility)

Recall that the Black-Scholes Formula = $C_0 = S_0 N(d_1) - X e^{-rT} N(d_2)$

Where $d_1 = \frac{Ln (S_0/x) + (r+(\sigma^2/2))T}{\sigma\sqrt{T}}$

$d_2 = \frac{Ln (S_0/x) + (r-(\sigma^2/2))T}{\sigma\sqrt{T}}$

Stock Price(S_0)= $40
Strike Price(X)= $45

Time to mature in days = 4months -> 4/12-> 1/3-> 121.67days

Risk Free Increase rate(r) = 3% = 0.03

Standard Deviation of Log return (σ)=40%=0.40

Calculating for d_1

$d_1 = \frac{Ln (S_0/x) + (r+(\sigma^2/2))T}{\sigma\sqrt{T}}$

$d_1 = \frac{Ln(40/45) + (0.03+(0.40^2/2)\ 121.67}{0.40\sqrt{121.67}}$

$d_1 = \frac{0.118 + 13.384}{4.412}$

$d_1 = 3.06$

$d_1 = 3.06 \approx 0.9987$

using the standard Normal table

$d_2 = \frac{Ln (S_0/x) + (r-(\sigma^2/2))T}{\sigma\sqrt{T}}$

$d_2 = \frac{Ln(40/45) + (0.03+(0.40^2/2)\ 121.67}{0.40\sqrt{121.67}}$

$d_2 = \frac{0.118 + (-6.084)}{4.412}$

$d_2 = -1.35$

$d_2 = -1.35 \approx 0.0901$
using the Standard Normal Table

Therefore using Black Scholes Formula
$C_0 = S_0 N(d_1) – X e^{-rT} N(d_2)$
$C_0 = 40(0.9987) – 45ℓ^{-(0.03)(121.67)}$ x (0.0901)
$C_0 = 39.948 – 156.011$
$C_0 = - 116.063$

2.COMPUTER SCIENCE

- ***Why is it a bad idea to use recursion method to find the fibonacci of a number?***
  - ⇨ **It is a bad idea to use recursion method to find the Fibonacci of a number because it uses too much memory which could result in a stack overflow.**
  - ⇨ **and also because of its time complexities as the number increases.**

•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

Answer

Using C++

```
// C++ implementation that takes in a proth
number and determine if it's prime
#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;
}
```

3.MATHS

•(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

$$Y = \sqrt{[(x + 6)^2 + 25]} + \sqrt{[(x - 6)^2 + 121]}$$

To find the maximum value of a positive real number
we say
dy/dx = 0 at Critical point

$$dy/dx = 0 \Rightarrow \tfrac{1}{2}(x(x+6)(x+6)^2 + 25)^{1/2} + \tfrac{1}{2}\,(2(x-6)(x-6)^2+121)^{-1/2} = 0$$

$$\Rightarrow \frac{x+6}{\sqrt{[(x + 6)^2 + 25]}} + \frac{x - 6}{\sqrt{[(x - 6)^2 + 121]}} = 0$$

$$(x + 6)\sqrt{[(x - 6)^2 + 121]} + (x - 6)\sqrt{[(x + 6)^2 + 25]} = 0$$

$$(x + 6)\sqrt{[(x - 6)^2 + 121]} = -(x - 6)\sqrt{[(x + 6)^2 + 25]}$$

Square both sides

$$(x + 6)^2((x - 6)^2 + 121) = (x - 6)^2((x + 6)^2 + 25)$$

$$(x + 6)^2(x - 6)^2 + 121(x + 6)^2 = (x - 6)^2(x + 6)^2 + 25(x - 6)^2$$

=> $121(x + 6)^2 = 25(x – 6)^2$

$121(x + 6)^2 – 25(x – 6)^2 = 0$

$(11(x + 6))^2 – (5(x – 6))^2 = 0$

difference of two squares

$(11(x + 6) – 5(x – 6))\ (11(x +6) + 5(x – 6)) = 0$

$\therefore 11(x + 6) – 5(x – 6) = 0$ or $11(x + 6) + 5(x – 6) = 0$ {66-30=36}

such that

$11x + 66 – 5x + 30 = 0$ or $11x + 66 + 5x – 30 = 0$

$6x/6 = -96/6$ $16x/16 = -36/16$

$x = -16$ $x = -2¼$

at x = -16 at $x = -2¼ = -9/4$

$Y = \sqrt{125} + \sqrt{605} = 35.77$ $\therefore Y \text{ min} = 20$