I'm building a 'no loss ponzi' and struggling to come up with an appropriate formula for allocating interest.

The idea is simple: deposit Dai. Dai earns interest. Interest is given proportional to when you added your Dai to the pool. So if you are a later investor, some of the interest on your Dai is given to those that got in before you. You can always withdraw your original Dai- hence, no loss.

Assume the following two simplifications, which make the maths easier:

- users can only deposit 100 Dai at a time. Let's conceptualise this using NFTs. Instead of 'depositing' dai, you mint an NFT at the cost of 100 Dai.
- users can never withdraw only their interest, they must withdraw their principle at the same time (by destroying the NFT). They will then lose their spot in the ponzi pyramid.

I'm struggling to come up with a formula that will work out how much interest each NFT is owed when they try and sell/destroy their NFT that does does not have some fatal flaw.

Idea 1: rank all NFTs, payout according to rank.

The first NFT created has rank 1, the third rank 3, etc. We can then use a formula like this to allocate interest to each NFT:

$\text{Interest allocated to each NFT} = \text{Total interest} * (n + (n - 1)/2 - (r - 1)) / n^2$

Where n = the total number of NFTs and r = the NFT's rank. If we have three NFTs, this formula gives 44% of the interest to the first, 33% to the second, 22% to the third.

Problem 1:

I do not believe it is possible to implement this without using unbounded loops. Because, if someone 'sells' their NFT, then the rank needs to change for all NFTs of a higher rank. If someome sells an NFT with rank 5, then NFT with rank 6 will now have rank 5, all the way up to rank N. Is there a way to do this without unbounded loops?

Problem 2:

it ignores the fact that earlier investors need more interest anyway, before it is 'ponzid', simply because their Dai has been accruing interest longer. More specifically- someone could mint a fresh NFT and even though they would have the worst (highest) rank, they would still have some interest allocated to their NFT even though the Dai contributed has yet to accrue a single wei-Dai in interest, so the incentive would always be to just mint then instantly destroy the NFT.

Idea 2: payout according to timestamp.

If we were to payout with zero ponzi rules, simply on how long they have invested we can use this formula:

$\text{Interest allocated to each NFT} = \text{Total interest} * (Tc - Tp) / ((Tc - Ta) * N)$

Where Tc = current timestamp, Tp = timestamp NFT purchased, Ta = average (mean) purchase timestamp of all NFTs, N = total number of NFTs.

This is great, and solves both problem 1 and 2 above. It solves problem 1 because when an NFT is destroyed, you only need to update one variable- Ta. No loops needed. And of course it solves problem 2 because it is directly allocating interest based on time.

The problem, of course, is that there is no 'ponzi' element. There is zero incentive to get in early.

I thought I had a genius solution to this: when allocating interest- don't use the actual purchase time of each NFT, adjust it. For example:

$\text{adjusted purchase time} = Tp - (Ta - Tp) * W$

Where Tp = timestamp NFT minted, Ta = mean purchase timestamp of all NFTs, W = ponzi weighting factor (the higher, the more ponzi'd it is)

This will have the effect of decreasing the timestamp for NFTs that were minted before the mean purchase time, and increasing the timestamp for NFTs that were minted after the mean purchase time. We can then throw this adjusted Tp

into the previous formula, and those who got in before the average time will see an interest boost, those that got in after will see a drop. There is now an incentive to mint your NFT early- it is ponzi'd! I thought I was real clever here. But no:

Problem 3:

newly minted NFTs will end up with an adjusted purchase time that is in the future. Therefore, new NFTs will have a negative interest allocation, and the whole project will no longer be a 'no loss ponzi'.

Can anyone think of a way to solve either Problems 1&2 (so that Idea 1 can be implemented), or Problem 3 (so that Idea 2 can be implemented)? Or perhaps some entirely alternative approach that I am failing to see? It is not unlikely there is some very obvious solution that I can't see…

From a UX perspective, I think Idea 1 is the better approach, if it can be made to work, because I like the idea of the 'order' being important- just like a real ponzi, it SHOULD make a big difference if you get in a second before someone else. With Idea 2, it makes effectively no difference if you get in a second before someone else. Yet I fear Idea 1 is a non-starter.