https://home.cs.colorado.edu/~srirams/courses/csci2824-spr14/pollardsRho.html

Pollard's Rho algorithm is for factoring numbers in time: $O(\sqrt{p})$ where p is the smallest prime factor of the number being factored

Let's say we are trying to factor a number $N = pq$

- If we try to guess the factors, we have a $2/(N-1)$ chance each time to find the factor

Applying the Birthday Paradox:

- Instead of picking just one number, we pick k numbers
- We ask if $x_i - x_j$ divides N for some i, j
- For $k \approx \sqrt{n}$, we have around a $1/2$ chance to pick a factor
  - However, this by itself doesn't save any effort since we need to do k^2 pairwise comparisons / divisions
- Instead we look to see if there if $\backslash\mathrm{t}gcd(x_i - x_j, N) > 1$
  - Instead of looking for just p and q, we look for $p, 2p, 3p, \ldots, (q-1)p, q, \ldots$
  - We have $p + q - 2$ numbers we can look for
- Now we need a k that is on the order of $N^{1/4}$ and do pairwise comparisons
  - But if N is very large, we would have to store this many numbers in memory

We use a pseudo random number generator $f(x) = x^2 + a \pmod{n}$ for a randomly chosen a

- We successively apply $x_{n+1} = f(x_n)$
- We then check consecutive numbers and see if their difference shares factors with N
- However, we will eventually cycle
  - We could store all numbers in memory, but this goes back to the original problem
  - Instead, we use Floyd's cycle detection algorithm
- This has a good probability of finding a factor if it exists, otherwise after a few tries we can give up