An Analysis of the Complexity of the Pollard Rho Factorization Method

Brandon Luders*
Georgia Institute of Technology

January 6, 2005

1 Objective

The objective of this paper is to computationally determine the mean and standard deviation of cycle lengths used in the Pollard rho factorization method. Both values will be shown to be $O(\sqrt{p})$, where p is a prime divisor of the number being factored. These values appear to be larger than previous estimates of the cycles. While the mean running time is known well, the results for the standard deviation appear to be less familiar. In fact, for p sufficiently large ($p > 300,000$), the mean and standard deviation are approximated quite well by an intuitive heuristic. Consequently, the runtime of the factorization method is $O(\sqrt{p})$, with deviations from the mean runtime of $O(\sqrt{p})$, as well.

2 The Pollard Rho Factorization Method

Given a relatively small known composite n, the Pollard rho algorithm seeks a non-trivial factor of n. The method finds these factors by implicitly traversing Z_p, where p is a prime factor of n, while traversing Z_n. If a random map $f\colon \mathrm{Z}_n \to \mathrm{Z}_n$ is used to iterate from a random seed s, creating the sequence

*This research is supported by a NSF VIGRE grant to the School of Mathematics, Georgia Insitute of Technology. Professor Michael Lacey served as the advisor for this project.

$\{s, f(s), f(f(s)), f^3(s), \dots\}$, on the average a cycle will form after $\sqrt{p}$ steps. When a cycle in p has been found, we have $f^j(s) \equiv f^k(s) \bmod p, j \neq k$. If these two iterates, which do not differ mod p, do differ mod n, then this difference is a multiple of p. In this case the greatest common divisor of $f^j(s) - f^k(s)$ and n yields a nontrivial divisor.

"Bad seeds" that do not result in a factor of n rarely occur in this algorithm, making the cycle finding described above the limiting factor of the complexity. It can be shown that iterative sequences under a random map have short 'tails': if a cycle of length L_n is found after n iterations, then $L_n - n << L_n$. Since the algorithm depends on finding a cycle in Z_p, the complexity is ultimately proportional to the mean cycle length of Z_p under the map f.

3 Random Maps

To minimize the complexity, a map is selected which minimizes the mean cycle length while exhibiting 'random' behavior, so that factors can be found. Furthermore, the map needs to be unrelated to the arithmetic structure of n or p, so it can be used generally. A simple map which meets these constraints is the random map $f(i) = k_i, k_i \in \mathrm{Z}_p$. A new random map is selected for each seed. It can be shown by the "birthday paradox" phenomenon that the mean cycle length for a prime p under a random map is proportional to $\sqrt{p}$.

A cycle-finding algorithm which generates a random map for each trial is shown in Algorithm 1, termed the random computational model. The values given by this algorithm verify the square-root relationship between p and the mean cycle length (Figure 1).

4 Quadratic Maps

In practice, a random map requires too many computations to be used. A quadratic map used in place of a random map proves to be "random enough" to support the factorization method, while minimizing computations. We examine this assumption with the quadratic map $f(i) = i^2 + a \bmod p, a \in \mathrm{Z}_p$. The parameter a is termed the Pollard rho parameter and can have a significant effect on the mean cycle length. The choice $a = 0$ covers only the quadratic residues and is not "random enough" to be used. Other specific

choices of a can be shown to generate non-ideal mean cycle lengths, such as $a = p - 2$. However, the effects of these choices are not easily quantified, and thus we do not attempt to systematically identify non-ideal values in the algorithms below.

A cycle-finding algorithm which uses a quadratic map with random choices of a is shown in Algorithm 2, termed the quadratic computational model. A plot of the outputs of this algorithm is given in Figure 2.

For $p < 50,000$, the random computational model and quadratic computational model yield nearly identical results (Figure 3). Thus for relatively low values of p the quadratic map is "as random" as the random map and can be used without concern. However, for $p > 50,000$ the quadratic computational model deviates from the random computational model, generating consistently higher mean cycle lengths. We notice an underlying oscillatory trend of the quadratic map for p between approximately 50,000 and 300,000 (Figures 4–5). For $p > 300,000$, the oscillatory effect is too damped to significantly affect the mean cycle length.

In summary, for relatively small ($p < 50,000$) and relatively large ($p > 300,000$) values of p, the quadratic map is as random as the random map and can be used in the Pollard rho factorization method with minimal complexity. In these ranges, the complexity of the Pollard rho factorization method is indeed $O(\sqrt{p})$. However, for $50,000 < p < 300,000$, the behavior of the quadratic computational model diverges from that of the random computational model in curious ways.

5 The Random Heuristic Model

We can explicitly compute the mean and standard deviation of the random model through some recursive formulas. However, it appears that a comprehensive theory of the cycle behavior of a random map is beyond our current knowledge.

In the heuristic used we establish an expectation value based on the probability of each cycle length occurring. We also assume the map is purely random, and begin with an arbitrary seed s in Z_p.

Consider random $f(s)$. There is a $1/p$ chance it maps to itself, resulting in a cycle of length 1, and a $(p-1)/p$ chance it maps to a new number. Suppose it does map to a new number. The next map has a $1/p$ chance of returning to the original s, resulting in a cycle of length 2, a $1/p$ chance of

mapping to iself, resulting in a cycle of length 1, and a $(p-2)/p$ chance of mapping to a new number. (See Figure 6.) This pattern can be used to develop the expectation

$$L_p = \sum_{n=1}^{1} \frac{n}{p} + \frac{p-1}{p}\left(\sum_{n=1}^{2} \frac{n}{p} + \frac{p-2}{p}\left(\sum_{n=1}^{3} \frac{n}{p} + \frac{p-3}{p}\left(\cdots\left(\sum_{n=1}^{p-1} \frac{n}{p} + \frac{1}{p}\left(\sum_{n=1}^{p} \frac{n}{p}\right)\right)\cdots\right)\right)\right)$$

$$\Rightarrow L_p = s_1 + \frac{p-1}{p}\left(s_2 + \frac{p-2}{p}\left(s_3 + \frac{p-3}{p}\left(\cdots\left(s_{p-1} + \frac{1}{p}\left(s_p\right)\right)\cdots\right)\right)\right)$$

where $s_i = \sum_{n=1}^{i} \frac{n}{p}$ and L_p is the mean cycle length in Z_p, as defined previously.

This explicit formula can be modeled and implemented using Algorithm 3. The output from this model is shown in Figure 7. We note that the random computational model follows the random heuristic model exactly for all values of p (Figure 8). Thus the random heuristic model provides further evidence of a complexity of $O(\sqrt{p})$ for the Pollard rho method.

Some simple manipulations of the above arguement can be made to derive a formula for the variance of the cycle lengths. An interesting extension would be to develop asymptotics for the mean and variance of the cycle lengths of random maps.

6 Variance and Standard Deviation

With minor modifications to the running totals taken at the end of Algorithms 1 and 2, the random computational model and quadratic computational model can both be used to show the variance and standard deviation of the cycle lengths of p. Plotting of such data shows a linear trend for the variance of both models ($O(p)$) and a square-root trend for the standard deviation of both models ($O(\sqrt{p})$). However, as one would expect, the standard deviation experiences a second-order effect, displaying oscillations similar to those in the mean data but toa larger magnitude. These oscillations are further magnified in the variance data.

7 Conclusions

The Pollard rho factorization method has a complexity and standard deviation of $O(\sqrt{p})$ and a variance of $O(p)$ when a random map is used. The same results arise when a quadratic map is used, excluding values of p between approximately 50,000 and 300,000, when unusual oscillatory behavior resulting in larger, non-ideal values appears in the data for all three variables.

8 Further Research

The oscillations that occur in the quadratic computational models are both precise and curious, and deserve to be further researched. The exact cause of these "jumps" from ideal behavior is not known, as well as why it takes plce only in the particular range of values of p specified. It is possible such behavior reciprocates for higher values of p not considered here. Computational error is also a possibility.

A better heuristic model for the variance of cycle lengths under a random map also needs to be developed to verify information presented by the random computational model.

Information about the standard deviation is interesting, as it offers guidelines on how long the Pollard rho method should be run. The results suggest that one needs to add on a fixed multiple of $\sqrt{p}$. Better information about this multiple would have to be derived from the asymptotic distribution of the number of cycles in the random model. This project is also left to the future.

9 Algorithms

```
public static double randMean(int p, int trials, long seed) {
      Random rand = new Random(seed);
      int u,v;
      long sn = 0l;
      int[] f = new int[p];
      for (int i=0; i<trials; i++) {
            for (int j=0; j<p; j++) {
                  f[j] = rand.nextInt(p);
            }
            u = rand.nextInt(p);
            v = f[u];
            while (u!=v) {
                  u = f[u];
                  v = f[f[v]];
            }
            u = cycleLength(u,f);
            sn += u;
      }
      return (double)sn/trials;
}

private static int cycleLength(int x, int[] f) {
      int counter = 1;
      int cur = f[x];
      while (cur!=x) {
            cur = f[cur];
            counter++;
      }
      return counter;
}
```

Algorithm 1. The random computational model. Takes in the prime p being evaluated, the number of trials to be run, and a random seed to use. Generates a new random map for each trial, then uses the Floyd cycle-finding method until a cycle is found and its length measured. The mean cycle length is returned. (Note: All algorithms are written in the Java language.)

```
public static double findMean(int p, int trials, long seed) {
      Random rand = new Random(seed);
      int a,u,v;
      long sn = 0l;
      for (int i=0; i<trials; i++) {
            a = rand.nextInt(p-1)+1;
            u = rand.nextInt(p);
            v = iterate(u,a,p);
            while (u!=v) {
                  u = iterate(u,a,p);
                  v = iterate(iterate(v,a,p),a,p);
            }
            u = cycleLength(u,a,p);
            sn += u;
      }
      return (double)sn/trials;
}

private static int iterate(int x, int a, int p) {
      return (x*x+a)\%p;
}

private static int cycleLength(int x, int a, int p) {
      int counter = 1;
      int cur = iterate(x,a,p);
      while (cur!=x) {
            cur = iterate(cur,a,p);
            counter++;
      }
      return counter;
}
```

Algorithm 2. The quadratic computational model. Takes in the prime p being evaluated, the number of trials to be run, and a random seed to use. Generates a new quadratic map for each trial by selecting random values of a, the uses the Floyd cycle-finding method until a cycle is found and its length measured. The mean cycle length is returned.

```
public static double meanHeuristic(int p0) {
      double ans = 0;
      double p = (double)p0;
      for (int i=1; i<=p; i++) {
            ans += ((p-i+1)*(p-i+2))/(2*p);
            ans *= i/p;
      }
      return ans;
}
```

Algorithm 3. The random heuristic model. Takes in the prime *p0* being evaluated and returns its theoretical mean cycle length according to the heuristic.

10 Figures

Note that all figures were generated using Microsoft Excel. (The first figure is on the next page.)

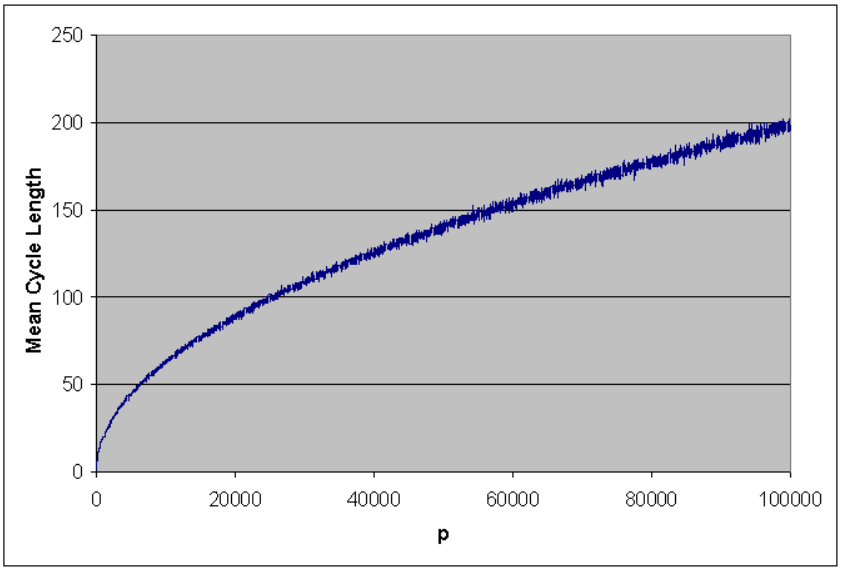

Figure 1: The values of the random computational model for primes p below 100,000, with 5,000 trials and a random seed of 0.

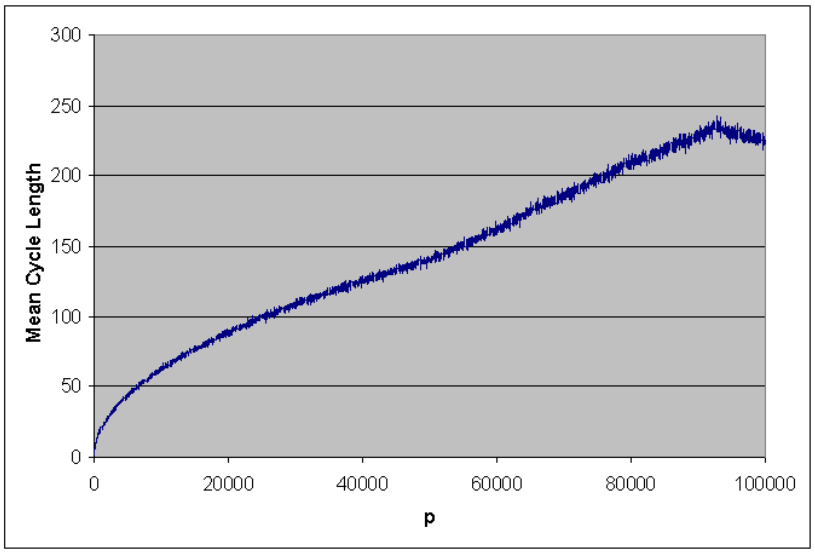

Figure 2: The values of the quadratic computational model for primes p below 100,000, with 5,000 trials and a random seed of 0.

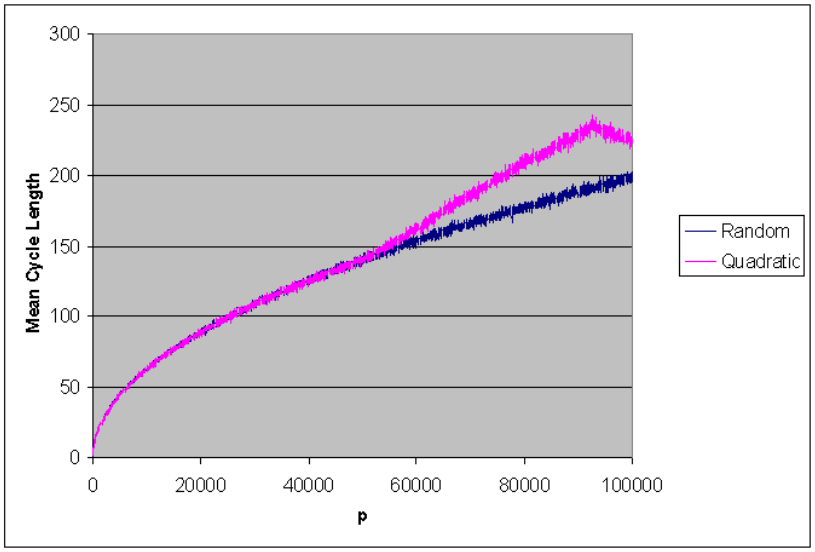

Figure 3: A superposition of Figure 1 with Figure 2.

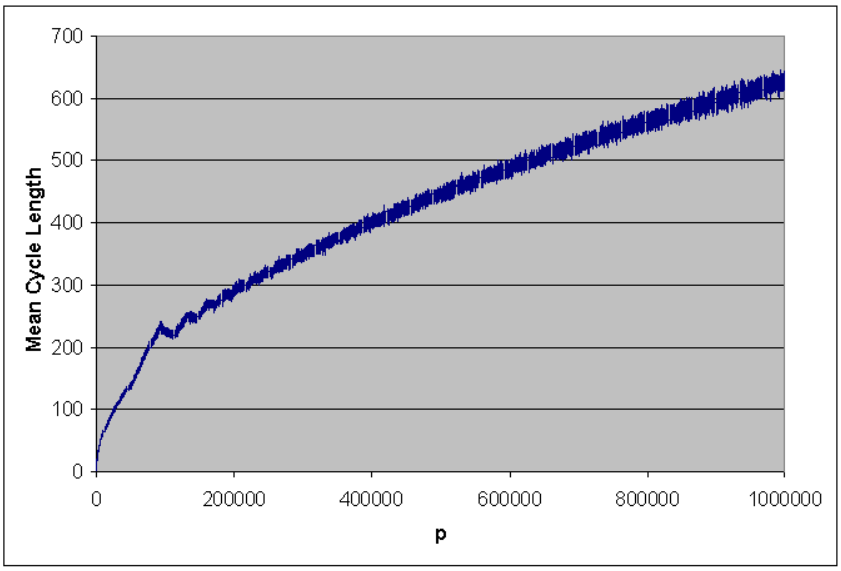

Figure 4: The values of the quadratic computational model for primes p below 1,000,000, with 5,000 trials and a random seed of 0.

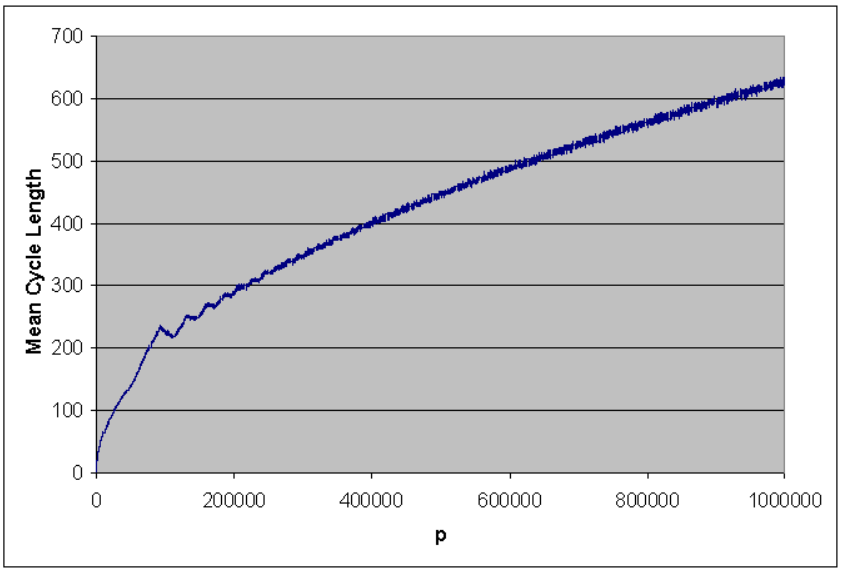

Figure 5: Figure 4, smoothed out with a weighted average of each value and its six neighbors.

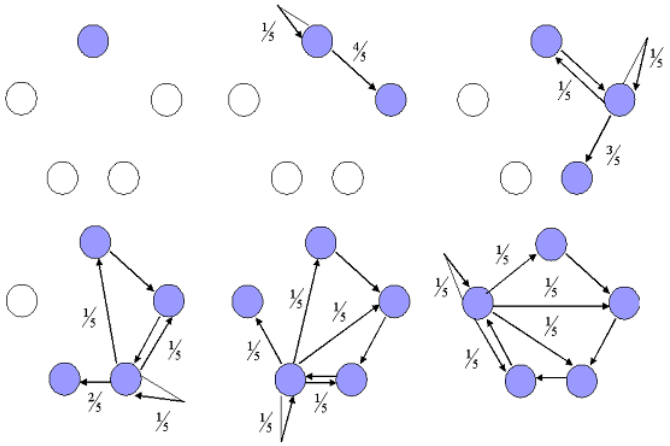

Figure 6: An example of the random heuristic method for $p = 5$.

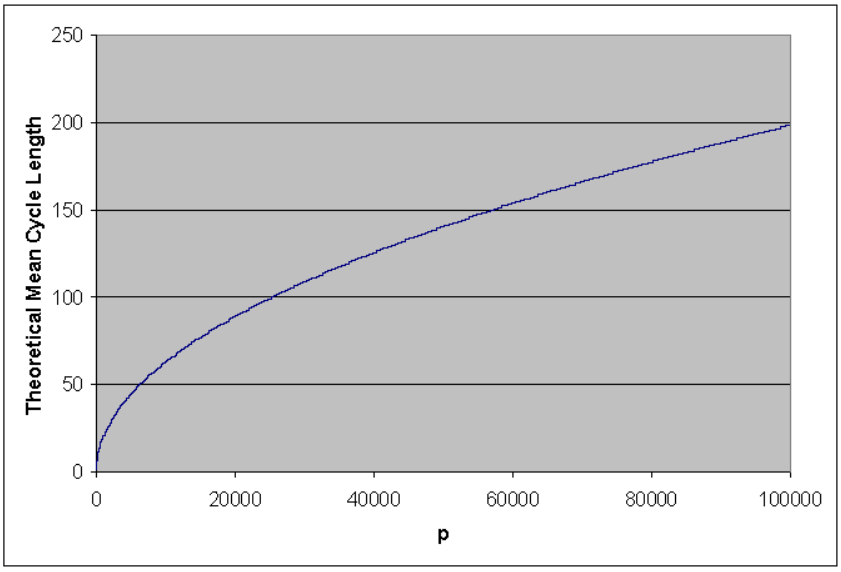

Figure 7: The values of the random heuristic model for primes p below 100,000.

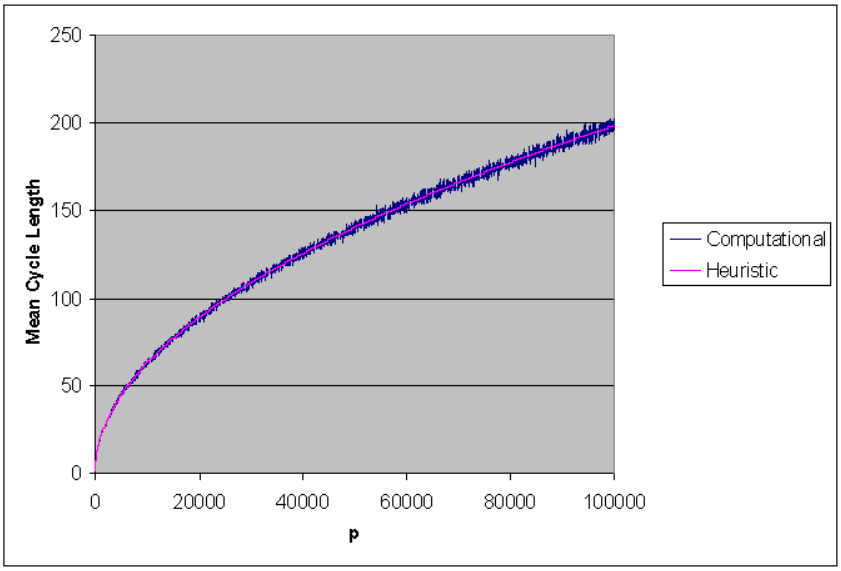

Figure 8: A superposition of Figure 1 with Figure 7.