Project 1: Factoring

by

Mårten Pålsson 8802030539 <mpals@kth.se>
Henrik Sohlberg 8805080531 <hsoh@kth.se>

KTH Computer Science and Communication
DD2440 - avalg12

2012-10-24

Abstract

asdasd

Contents

1 Introduction

1.1 Introduction

The purpose of this project was to investigate different methods and algorithms for prime factorization. Prime factorization, or Prime decomposition as it is also called, is the practice of splitting an integer into a series of prime factors that when multiplied together form the original number [1]. There are many algorithms for factoring integers into primes but the one we have used and will discuss in the greatest detail in this report is Pollard Rho's algorithm for factoring numbers.

1.2 Problem Statement

As stated above the problem in this case consists of factoring numbers into their constituent primes. The number of integers to be factorized are 100 and the time limit of doing so is 15 seconds. There is also a limit of 32 MB of memory that can be allocated to the program.

2 Background

This section briefly discuss some algorithms used to factorize numbers.

2.1 Trial Division

Trial Division is a naive approach where an integer is divided by a series of precomputed prime numbers to discover small factors quickly. If the rest of such a division is 0 then we know that the prime in question is a factor of the given integer. The resulting quotient is then sent back to Trial Division to try to break it down even further. This is a quick way of finding small prime factors and reducing the magnitude of the given Integer.

2.2 Perfect Powers

Perfect power refers to integers that can be written on the form $N = m^k$ where $m > 1$ and $k \geq 2$ [2]. If N is found to be a perfect power it is only necessary to continue the factorization using m as the input and store any found factors k times before printing. This is because any factors found in m would have been found k times in N since m is the k^{th} root of N. This further lessens the magnitude of N and reduces thus the computation time.

2.3 Fermat's Factorization Method

Pierre de Fermat's method to factorize a number is to represent it as the difference between two squares.

$N = a^2 - b^2 = (a + b)(a - b)$

To factorize N, start with a value $\lfloor\sqrt{N}\rfloor + 1 = a$. Then $b = a^2 - N$, if b is a perfect square, you have found two factors $(a + b)$ and $(a - b)$ to N.

2.4 Cycle Detection

Cycle detection refers to the problem of finding a cycle in a sequence of iterated function values. [3]

2.4.1 Floyd's Cycle Finding Algorithm

Also known as "tortoise and the hare" algorithm [3] which finds a cycle in a sequence. It uses two pointer, where one pointer (the hare) moves with double the speed to the other pointer (the tortoise) through the elements of the sequence. The algorithm detects the occurrence when the hare has sprinted exactly one whole cycle further than the tortoise, and uses this information to calculate the length of the found cycle. If the hare reaches the end of the sequence first, there is no cycle.

2.5 Pollard's Rho Algorithm

Pollard's rho algorithm is especially effective at finding small prime factors and is based on the concept of the existence of an subgroup to the group N [4][5]. The algorithm

first makes the assumption that for an integer N there exists a factor d. Secondly, it uses a pseudo-random function to generate "random" numbers together with an implementation of the Floyd's cycle finding algorithm [2.4.1]. An iteration consist of generating two random numbers, x_k and $x_{k/2}$, and performing a greatest common divisor operation between the absolute value $(x_k - x_{k/2})$ and the number to be factorized, N. Pollard's rho algorithm succeeds when $\gcd(|x_k - x_{k/2}|, N) = d$ where

1. $d > 1$, and $d \neq N$

(*Behind the scene*), we know that N is a multiple of d. When $x_k = x_{k/2} \mod d \rightarrow x_k - x_{k/2}$ is a multiple of d. If $\gcd(|x_k - x_{k/2}|, N) > 1$ and $\neq N$ we find a factor.

The time complexity of Pollard's rho algorithm using a random function $f \mod p$ to find the factor p is, according to Hastad [6], $O(\sqrt{p}) = O(\sqrt{N})$. Since p is unknown, this function is not possible to use, but experience shows that the "random" function $x_{i+1} \equiv {x_i}^2 + 1$ also behaves this way.

3 Approach

3.1 Design Approaches

3.2 Our Approach

3.3 Pseudo code

```
Find the winning map
currentState ← startMap
while newStates do
   currentState ← prioQueue.remove;
   for all moves in currentState do
      nextMap ← newMap(move)
      if nextMap is winning state then
         return nextMap
      else if nextMap has moves & !visited then
         prioQueue ← nextMap
      end if
   end for
end while
return
```

4 Results

5 Discussion

6 Conclusion

References

[1] Prime Factorization [Homepage on the Internet]. [cited 10/10/23]. Available from: `http://mathworld.wolfram.com/PrimeFactorization.html`

[2] Perfect Power [Homepage on the Internet]. [cited 10/10/23]. Available from: `http://mathworld.wolfram.com/PerfectPower.html`

[3] Cycle Detection [Homepage on the Internet]. [cited 10/10/23]. Available from: `http://en.wikipedia.org/wiki/Cycle_detection`

[4] Pollard's rho algorithm [Homepage on the Internet]. [cited 10/10/23]. Available from: `http://en.wikipedia.org/wiki/Pollard\%27s_rho_algorithm`

[5] Pollard's Rho Method [Homepage on the Internet]. [cited 10/10/23]. Available from: `http://www.csh.rit.edu/~pat/math/quickies/rho/`

[6] Notes for the course advanced algorithms [Homepage on the Internet]. [cited 10/10/23]. Available from: `http://www.csc.kth.se/utbildning/kth/kurser/DD2440/avalg07/algnotes.pdf`