Programowanie dla fizyków - projekt „Algorytm rozkładu liczby naturalnej na czynniki pierwsze metodą Fermata”

Weronika Smagór, Aleksandra Motor, Patryk Polczyk
Fizyka Techniczna II rok
Wydział Inżynierii Materiałowej i Fizyki Politechniki Krakowskiej

Wrzesień 2020
Kraków

Spis treści

1 Wstęp

Pierre de Fermat był francuskim prawnikiem i lingwistą, a ponadto matematykiem samoukiem. Nie publikował swoich odkryć, przez co pozostawały nieznane. Niektóre z nich zostały następnie niezależnie odkryte przez Kartezjusza, co wywołało spór o pierwszeństwo. Większość jego prac matematycznych została opublikowana dopiero po śmierci przez jego syna, Samuela. Pierre de Fermat dokonał wielu odkryć w teorii liczb, m.in. sformułował słynne Wielkie Twierdzenie Fermata, które zostało wspomniane w serialu "Star Trek: Następne pokolenie" czy w powieści dla młodzieży Kornela Makuszyńskiego "Szatan z siódmej klasy". Także w wielu innych książkach bohaterowie usiłowali znaleźć dowód na prawdziwość tego twierdzenia oraz dojść do odtworzenia oryginalnego dowodu, o którym Fermat napisał na marginesie. Tajemniczość twórczości Fermata zachęciła nas do zgłębinia jej wpływu na współczesną kryptografię.

Rozkład na czynniki pierwsze liczb całkowitych mimo pozornej prostoty do dziś pozostaje jednym z ważniejszych zagadnień obliczeniowych. Ma wieloraki wpływ na postać dzisiejszej informatyki, poczynając od czysto teoretycznych zagadnień związanych z teorią obliczeń i złożoności obliczeniowej, a kończąc na całkowicie praktycznym podejściu związanym z nowoczesnymi systemami kryptograficznymi. Algorytm Fermata może działać wolniej niż naiwne szukanie dzielników n, dlatego nie ma on praktycznego zastosowania. Zasada jego działania stanowi jednak podstawę znacznie efektywniejszych algorytmów faktoryzacji, takich jak sito kwadratowe i GNFS. Celem tej pracy jest przedstawienie algorytmu Fermata oraz jego implementacja w języku Python.

2 Algorytm rozkładu liczby naturalnej na czynniki pierwsze metodą Fermata

Algorytm Fermata jest to metoda faktoryzacji, czyli rozkładu liczby na czynniki pierwsze. Metoda ta szybko znajduje rozkład p jeśli jego dzielniki są bliskie pierwiastkowi kwadratowemu z p. Z powodu istnienia tej metody, tworząc klucze kryptograficzne oparte na iloczynach liczb pierwszych (RSA), unika się iloczynów niewiele różniących się liczb.

Działanie algorytmu polega na szukaniu pary liczb x i y takich że:

$$p = x^2 - y^2$$

Problem: Znaleźć rozkład liczby $p > 1$ na iloczyn czynników pierwszych.

Jest to prosty sposób znajdowania czynników (liczb dzielących p bez reszty). Opiera się on na spostrzeżeniu, iż jeśli potrafimy znaleźć dwie liczby naturalne x i y, takie że:

$$p = x^2 - y^2 \Longrightarrow p = (x + y)x(x - y)$$

To czynnikami liczby p są:

$$m = x + y \quad oraz \quad n = x - y$$

Znalezione czynniki m i n nie muszą być liczbami pierwszymi, zatem metodę Fermata stosujemy również do ich rozkładu. Jest to możliwe, ponieważ czynniki liczby nieparzystej są również nieparzyste. Czynniki 2 można wyeliminować z p przed zastosowaniem metody Fermata, zatem nie jest to żadne ograniczenie.

Metoda Fermata jest szczególnie efektywna w przypadku czynników leżących w pobliżu pierwiastka z p W przeciwnym razie jej efektywność jest taka sama lub nawet gorsza jak dla metody próbnych dzieleń. W praktyce metoda Fermata nie jest używana, jednak jest ważna z powodu jej wagi dla innych, bardziej zaawansowanych metod poszukiwań rozkładu liczby na czynniki pierwsze.

3 Sposób rozwiązania

3.1 Rozwiązanie – wersja 1

Algorytm wykorzystuje bezpośrednio własność $p = x^2 - y^2$ do znalezienia liczb x i y. Poszukiwania rozpoczynamy od x równego pierwiastkowi z p zaokrąglonemu w górę do najbliższej liczby całkowitej. Obliczamy:

$$y^2 = x^2 - p$$

Następnie sprawdzamy, czy y jest liczbą całkowitą. Jeśli tak, to znaleźliśmy odpowiednie liczby x i y. Ponieważ liczba p z założenia jest nieparzysta, to wszystkie jej dzielniki są nieparzyste. Obliczamy dzielniki m i n i jeśli są różne od 1, to wywołujemy rekurencyjnie procedurę Fermata do znalezienia ich rozkładu. Jeśli jeden z czynników jest równy 1, to drugi musi być równy p i p jest liczbą pierwszą – wypisujemy p i kończymy.
Ponieważ czynnik musi być mniejszy lub równy p, to:

$$m = x + y \leqslant p$$

Przy wzroście x rośnie również y lub pozostaje takie samo. Zatem gdy suma x + y przekroczy p, to nie znajdziemy już żadnego dzielnika liczby p i algorytm można zakończyć przyjmując, iż p jest pierwsze.

Algorytm rozkładu liczby naturalnej na czynniki pierwsze metodą Fermata

Wejście:
p – liczba rozkładana na czynniki pierwsze, $p \in N$, p jest nieparzyste.

Wyjście:
Czynniki pierwsze liczby p.

Zmienne pomocnicze:
x, y, z – zmienne do rozkładu p. $x, y, z \in N$.
m, n – czynniki $p, m, n \in N$.

Lista kroków dla procedury rekurencyjnej Fermat (p)

K01:	x $\longleftarrow ceil(sqr(p))$	obliczamy wartość początkową dla x
K02:	$z \longleftarrow x^2 - p$	$z = y^2$
K03:	$y \longleftarrow ceil(sqr(z))$	sprawdzamy, czy z jest kwadratem liczby naturalnej
K04:	Jeśli $z \neq y^2$, to idź do kroku K11	
K05:	$m \longleftarrow x + y$	obliczamy czynniki
K06:	$n \longleftarrow x - y$	
K07:	Jeśli n = 1, to idź do kroku K13	przerywamy przy n = 1, gdyż p jest pierwsze
K08:	Fermat (m)	rozkładamy rekurencyjnie czynniki m i n
K09:	Fermat (n)	
K10:	Zakończ	
K11:	$x \longleftarrow x + 1$	następne x
K12:	Jeśli $x + y < p$, to idź do kroku K02	kontynuujemy pętlę
K13:	Pisz p	p jest pierwsze
K14:	Zakończ	

Program w pierwszym wierszu czyta dowolną liczbę naturalną $p > 1$ i w następnym wierszu wypisuje kolejne czynniki pierwsze tej liczby. Czynniki mogą nie być wypisywane w kolejności rosnącej. Zwróć uwagę, iż do poprawnego działania zmienna x musi być zadeklarowana jako 64-bitowa – w przeciwnym razie dla dużych liczb jej kwadrat wyjdzie poza zakres liczb 32-bitowych. W efekcie program może dokonywać rozkładu na czynniki pierwsze liczb 32-bitowych.

C++

```cpp
// Metoda Fermata
// Data   : 27.03.2008
// (C)2020 mgr Jerzy Wałaszek
//----------------------------

#include <iostream>
#include <cmath>

using namespace std;

void Fermat ( unsigned int p )
{
  unsigned long long x, y, z, m, n;

  x = ( unsigned long long )ceil ( sqrt ( p ) );
  do
  {
    z = x * x - p;
    y = ( unsigned long long )floor ( sqrt ( z ) );
    if( z == y * y )
    {
      m = x + y;
      n = x - y;
      if( n == 1 ) break;
      Fermat ( m );
      Fermat ( n );
      return;
    }
    x++;
  } while( x + y < p );
  cout << p << " ";
}

int main( )
{
  unsigned int p;

  cin >> p;
  while( p % 2 == 0 )
  {
    p /= 2;
    cout << 2 << " ";
  }
  if( p > 1 ) Fermat ( p );
  cout << endl;
  return 0;
}
```

Rysunek 1: Algorytm rozkładu liczby naturalnej na czynniki pierwsze metodą Fermata w języku C++

```
855855
11 7 13 3 3 5 19
```

Rysunek 2: Wynik otrzymany przez program.

Podstawową wadą pierwszego rozwiązania jest konieczność wyznaczania wartości pierwiastka kwadratowego. Jest to operacja dosyć czasochłonna.

3.2 Rozwiązanie – wersja 2

Algorytm Feramata można zapisać w inny sposób, tak aby nie było konieczności wyznaczania pierwiastków kwadratowych w pętli poszukującej dzielników. Dokonajmy najpierw kilku prostych spostrzeżeń.

Jeśli mamy daną wartość kwadratu liczby naturalnej, np. x^2, to kwadrat następnej liczby x + 1 można wyznaczyć następująco:

$$(x+1)^2 = x^2 + 2x + 1 = x^2 + dx$$

gdzie:

$$dx = 2x + 1$$

Ponieważ x wzrosło o 1, to nowy przyrost dx' wyniesie:

$$dx' = 2(x+1) + 1 = 2x + 2 + 1 = (2x+1) + 2 = dx + 2$$

Wynika stąd, iż mając wartość początkową kwadratu x^2 oraz dx, kolejne kwadraty obliczamy w pętli wg reguły:

$$x'^2 = x^2 + dx, x = (dx-1)/2, \quad dx' = dx + 2$$

Poniższy program demonstruje tę zasadę wyznaczania kwadratów kolejnych liczb naturalnych od 0 do 15.
kx – kwadrat x, początkowo 0
dx – przyrost kwadratu, początkowo dx = 1, gdyż x = 0
x – kolejna liczba od 0 do 15. Liczbę tę wyznaczamy bezpośrednio z przyrostu dx.

C++

```
// Wyznaczanie kwadratów
// kolejnych liczb naturalnych
// Data   : 2.04.2008
// (C)2020 mgr Jerzy Wałaszek
//----------------------------

#include <iostream>
#include <iomanip>

using namespace std;

int main( )
{
  int kx, dx, x;

  kx = 0; dx = 1;
  do
  {
    x = ( dx - 1 ) >> 1;
    cout << setw ( 5 ) << x
         << setw ( 5 ) << kx
         << endl;
    kx += dx; dx += 2;
  } while( x < 15 );
  cout << endl;
  return 0;
}
```

Wynik:

```
    0    0
    1    1
    2    4
    3    9
    4   16
    5   25
    6   36
    7   49
    8   64
    9   81
   10  100
   11  121
   12  144
   13  169
   14  196
   15  225
```

Rysunek 3: Program wyznaczający kwadraty kolejnych liczb naturalnych w języku C++ oraz otrzymany wynik.

Zastosujmy następującą strategię. Mamy daną liczbę naturalną p, $p > 1$ i p jest nieparzyste. Obliczamy:

$$x = ceil(sqr(p)), \quad y = 0$$
$$e = x^2 - y^2 - p$$

Liczba e jest wartością błędu pomiędzy $x^2 - y^2$ a liczbą p. Wyznaczamy przyrosty dla x^2 oraz dla y^2:

$$dx = 2x + 1$$
$$dy = 2y + 1 = 1, \ gdyż\ początkowo\ y = 0$$

Teraz w pętli badamy błąd e. Jeśli jest równy 0, to

$$p = x^2 - y^2$$
$$x = (dx - 1)/2$$
$$y = (dy - 1)/2$$
$$m = x + y = (dx - 1)/2 + (dy - 1)/2 = (dx + dy - 2)/2 = (dx + dy)/2 - 1$$
$$n = x - y = (dx - 1)/2 - (dy - 1)/2 = (dx - dy)/2$$

Jeśli n jest różne od 1, to m i n rozkładamy dalej na czynniki tym samym algorytmem. W przeciwnym razie liczba p jest pierwsza, zwracamy ją i kończymy.

Jeśli e jest różne od zera, to nie znaleźliśmy jeszcze x i y spełniających równanie Fermata. Badamy znak e.

Jeśli $e > 0$, to przeważa x^2, zatem zwiększamy y^2 i odejmujemy od e przyrost dy. Po tej operacji dy jest zwiększane o 2, aby w kolejnym obiegu otrzymać kwadrat następnego y–a właściwie jego przyrost w stosunku do poprzedniego kwadratu. Zwiększanie y kontynuujemy do momentu, aż $e \leqslant 0$, czyli aż osiągniemy równowagę lub przeważy y 2.

Jeśli $e < 0$, to przeważa y^2, zatem zwiększamy x^2 i dodajemy do e przyrost dx, który następnie zwiększamy o 2. Operację kontynuujemy do momentu, aż $e \geqslant 0$.

Wracamy na początek pętli, aż do uzyskania e = 0. W tym miejscu algorytm można zoptymalizować, sprawdzając, czy $x + y \geqslant p$, czyli $(dx + dy)/2 > p$. Jeśli tak, to liczba p jest pierwsza i nie znajdziemy rozkładu na inne czynniki poza p i 1. Zatem przerywamy pętlę zwracając p.

W tej wersji algorytmu Fermata nie wykonujemy w pętli operacji pierwiastkowania, a jedynie proste dodawania. Dzięki temu algorytm staje się dużo szybszy od algorytmu podanego w rozwiązaniu 1.

Algorytm rozkładu liczby naturalnej na czynniki pierwsze metodą Fermata

Wejście:
p – liczba rozkładana na czynniki pierwsze, $p \in N$, p jest nieparzyste.

Wyjście:
Czynniki pierwsze liczby p.

Zmienne pomocnicze:
e – wartość błędu. $e \in C$.
m, n – czynniki p, m, $n \in N$.
dx, dy – przyrosty kwadratów x i y. $dx, dy \in N$.

Lista kroków dla procedury rekurencyjnej Fermat (p)

K01:	$m \longleftarrow ceil(sqr(p))$	wyznaczamy wartość błędu e oraz przyrosty dx i dy
K02:	$e \longleftarrow m^2 - p$	$e = x^2 - p$, ponieważ początkowo y = 0
K03:	$dx \longleftarrow 2m + +1$	dx – przyrost x^2
K04:	$dy \longleftarrow 1$	dy – przyrost y^2
K05:	Dopóki $e \neq 0$, wykonuj kroki K06...K16	w pętli staramy się zrównać $x^2 - y^2$ z p - wtedy e = 0
K06:	Dopóki $e > 0$, wykonuj kroki K07...K08	przeważa x^2, zwiększamy y^2
K07:	$e \longleftarrow e - dy$	modyfikujemy błąd o przyrost y^2
K08:	$dy \longleftarrow dy + 2$	przyrost y^2 dla zwiększonego o 1 y
K09:	Dopóki e ¡ 0, wykonuj kroki K10...K11	przeważa y^2, zwiększamy x^2
K10:	$e \longleftarrow e + dx$	modyfikujemy błąd o przyrost x^2
K11:	$dx \longleftarrow dx + 2$	przyrost x^2 dla zwiększonego x
K12:	Jeśli $(dx + dy)/2 \leqslant p$, to następny obieg pętli K05	sprawdzamy drugi warunek zakończenia pętli
K13:	$dx \longleftarrow 2$	jeśli jest spełniony, to p jest liczbą pierwszą
K14:	$dy \longleftarrow 0$	dx i dy ustawiamy tak, aby otrzymać n = 1
K15:	Idź do kroku K16	przerywamy pętlę K05
K16:	$m \longleftarrow (dx + dy)/2 - 1$	czynnik większy
K17:	$n \longleftarrow (dx - dy)/2$	czynnik mniejszy
K18:	Jeśli n = 1, to idź do kroku K21	sprawdzamy, czy p jest pierwsze
K19:	Fermat (m); Fermat (n)	jeśli nie, to czynniki m i n rozkładamy dalej
K20:	Zakończ	
K21:	Pisz p	p jest pierwsze. Wyprowadzamy je i kończymy
K22:	Zakończ	

Program w pierwszym wierszu czyta liczbę p i w następnym wierszu wypisuje kolejne czynniki pierwsze tej liczby. Liczba p nie musi być nieparzysta. Przed przekazaniem jej do procedury Fermat program usuwa z p wszystkie czynniki równe 2.

C++

```cpp
// Metoda Fermata
// Data   : 2.04.2008
// (C)2020 mgr Jerzy Wałaszek
//----------------------------

#include <cstdlib>
#include <iostream>
#include <cmath>

using namespace std;

void Fermat ( long long p )
{
  long long e, dx, dy, m, n;
  m  = ( long long )sqrt ( p );
  dx = ( m << 1 ) + 1;
  e  = m * m - p;
  dy = 1;
  while( e != 0 )
  {
    while( e > 0 )
    {
      e -= dy; dy += 2;
    }
    while( e < 0 )
    {
      e += dx; dx += 2;
    }
    if( ( ( dx + dy ) >> 1 ) > p )
    {
      dx = 2; dy = 0; break;
    }
  }
  m = ( ( dx + dy ) >> 1 ) - 1;
  n = ( dx - dy ) >> 1;
  if( n != 1 )
  {
    Fermat ( m ); Fermat ( n );
  }
  else cout << p << " ";
}

int main( )
{
  long long p;
  cin >> p;
  while( p % 2 == 0 )
  {
    cout << 2 << " ";
    p >>= 1;
 }
 if( p > 1 ) Fermat ( p );
 cout << endl;
 return 0;
}
```

Wynik:

```
254821743888
2 2 2 2 230816797 23 3
```

Rysunek 4: Algorytm rozkładu liczby naturalnej na czynniki pierwsze metodą Fermata w języku C++ oraz otrzymany wynik.

4 Wyniki

W celu sprawdzenia poprawności napisanych przez nas programów, wykonaliśmy test na liczbie użytej w przykładzie. Pierwszy program, działający na podstawie pierwszej wersji rozwiązania nazwaliśmy fermat1.py (rys.5), natomiast

wykorzystujący drugą wersję - fermat2.py (rys.6).

```
Podaj nieparzystą liczbę całkowitą, którą chcesz rozłożyć na czynniki pierwsze:855855
[11, 7, 13, 3, 3, 5, 19]
```

Rysunek 5: Wynik obliczeń programu fermat1.py.

```
Program rozkłada dodatnie liczby nieparzaste na iloczyn liczb pierwszych.
Podaj liczbe do sprawdzenia: 855855
[11.0, 7.0, 13.0, 3.0, 3.0, 5.0, 19.0]
```

Rysunek 6: Wynik obliczeń programu fermat2.py.

Następnie zmierzyliśmy czas pracy każdego z programów.

Rysunek 7: Wyniki pomiarów

5 Podsumowanie

Na podstawie przedstawionego algorytmu napisaliśmy dwa programy rozkładajace liczby naturalne na czynniki pierwsze metodą Fermata w języku Python oraz porównalismy ich czas pracy.
Zaimplementowanie pierwszej wersji rozwiązania było dużo łatwiejsze, jednak czas pracy programu fermat1.py był w każdym przypadku znacznie dłuższy. Zgodnie z przewidywaniami zastąpienie operacji pierwiastkowania w pętli, prostym dodawaniem w programie fermat2.py okazało się dużo szybszym rozwiązaniem od algorytmu podanego w wersji pierwszej.

Literatura

[1] https://pl.wikipedia.org/wiki/Algorytm_Fermata

[2] https://pl.wikipedia.org/wiki/Rozklad_na_czynniki

[3] https://pl.wikipedia.org/wiki/Kryptologia

[4] https://eduinf.waw.pl/

[5] https://pl.wikipedia.org/wiki/Pierre_de_Fermat

[6] https://www.ki.agh.edu.pl/sites/default/files/usefiles/172/theses/mateusz.niezabitowski.algorytmy.faktoryzacji.w.zastosowaniach.kryptograficznych.v1.0-final.pdf