**Algorytmy sprawdzające, czy dana liczba jest prawdopodobnie pierwsza.**

1. Test Millera-Rabina

Probabilistyczny test, w którym prawdopodobieństwo błędnego stwierdzenia złożoności lub pierwszości liczby zależy od ilości wykonanych iteracji pętli algorytmu. Prawdopodobieństwo to wynosi (1/4)2. Schemat wykonania dla liczby nieparzystej n i k prób, wygląda następująco:

Wylicz s - maksymalną potęgę dwójki dzielącą n-1.  
Podstaw d = n/2s  
powtórz k razy:  
 wylosuj a takie, że 1<a<n  
 sprawdź czy ad mod n różne od 1  
 jeżeli tak to sprawdz czy ad\*2rmod n rozne od n-1 dla wszystkich r, takich że: 0≤r≤s-1  
 jeżeli tak to to przerwij test - liczba nie jest pierwsza  
jeżeli nie przerwano testu dla żadnej z prób to liczba prawdopodobnie jest pierwsza

Algorytm ten można bardzo łatwo rozproszyć. Każda iteracja jest zdarzeniem niezależnym i rozłącznym względem innych iteracji. Wobec tego w dowolny sposób można je rozdzielać pomiędzy dowolną ilość klastrów obliczeniowych. Najtrudniejszą częścią tego algorytmu jest wyliczenie wyrażenia ad\*2rmod n, można w tym miejscu skorzystać z algorytmu szybkiego potęgowania modularnego.

Opis algorytmu:

<http://www.algorytm.org/algorytmy-arytmetyczne/test-pierwszosci-test-millera-rabina.html>

<http://edu.i-lo.tarnow.pl/inf/alg/001_search/0019.php>

<http://pl.wikipedia.org/wiki/Test_Millera-Rabina>

1. Test pierwszości Fermata (wykorzystujący Małe TW Fermata)

Test pierwszości Fermata jest testem probabilistycznym opartym na Małym TW Fermata. Jest to jeden z prostszych testów stwierdzających z dużym prawdopodobieństwem czy liczba jest pierwsza. Jego schemat dla liczby nieparzystej p i k prób wygląda następująco:

powtórz k razy:  
 wylosuj a takie, że 1<a<p  
 sprawdź czy ap-1 mod p = 1  
 jeżeli nie to przerwij test - liczba nie jest pierwsza  
jeżeli równość była spełniona dla wszystkich prób, liczba prawdopodobnie jest pierwsza

Podobnie jak test Millera-Rabina, test Fermata można bardzo łatwo rozpraszać. Każde powtórzenie algorytmu można rozwiązywać niezależnie od innych powtórzeń. Główną wadą tego algorytmu jest to, że daje pozytywny wynik nie tylko dla liczb pierwszych, ale również dla liczb Carmichela, które są złożone. W praktyce nie jest to jednak duży problem, ponieważ liczby Carmichela są dużo rzadsze od liczb pierwszych (mniej więcej jedna na milion liczb pierwszych).

Opis algorytmu:

<http://www.algorytm.org/algorytmy-arytmetyczne/test-pierwszosci-test-fermata.html>

<http://edu.i-lo.tarnow.pl/inf/alg/001_search/0018.php>

[http://pl.wikipedia.org/wiki/Test\_pierwszo%C5%9Bci\_Fermata](http://pl.wikipedia.org/wiki/Test_pierwszości_Fermata)

1. Chiński Test pierwszości

Chiński test pierwszości opiera się na twierdzeniu, które mówi, że jeżeli 2*p* mod *p* = 2, to liczba p jest złożona. Problemem wydaje się być obliczenie liczby 2*p*, która dla dużych p może być astronomicznie duża. Możemy jednak skorzystać z zasad arytmetyki modularnej, a w szczególności z zależności: *ax*+*y* mod *n* = (*ax* mod *n*) × (*ay* mod *n*) mod *n.* Korzystając z tej zależności możemy rekurencyjne wyliczyć reszty modulo kolejnych potęg liczby 2 (każdą liczbę można rozbić na sumę potęg dwójki). Przykładowo dla liczby 387:

387 = 256 + 128 + 2 + 1

2387 mod 13 = (2256 mod 13) × (2128 mod 13) × (22 mod 13) × (21 mod 13) mod 13

21 mod 13 = 2 mod 13 = 2  
22 mod 13 = (21 mod 13) × (21 mod 13) mod 13 = (2  × 2) mod 13 = 4 mod 13 = 4  
24 mod 13 = (22 mod 13) × (22 mod 13) mod 13 = (4  × 4) mod 13 = 16 mod 13 = 3  
28 mod 13 = (24 mod 13) × (24 mod 13) mod 13 = (3  × 3) mod 13 = 9 mod 13 = 9  
216 mod 13 = (28 mod 13) × (28 mod 13) mod 13 = (9  × 9) mod 13 = 81 mod 13 = 3  
232 mod 13 = (216 mod 13) × (216 mod 13) mod 13 = (3  × 3) mod 13 = 9 mod 13 = 9  
264 mod 13 = (232 mod 13) × (232 mod 13) mod 13 = (9  × 9) mod 13 = 81 mod 13 = 3  
2128 mod 13 = (264 mod 13) × (264 mod 13) mod 13 = (3  × 3) mod 13 = 9 mod 13 = 9  
2256 mod 13 = (2128 mod 13) × (2128 mod 13) mod 13 = (9  × 9) mod 13 = 81 mod 13 = 3

Mając reszty potęg (zaznaczone na czerwono) wymnażamy je modulo 13:

2387 mod 13 = (2256 mod 13) × (2128 mod 13) × (22 mod 13) × (21 mod 13) mod 13  
2387 mod 13 = 3 × 9 × 4 × 2 mod 13 = 216 mod 13 = **8**

Chiński Test pierwszości zdaje się być algorytmem dużo trudniejszym do rozproszenia niż dwa powyższe testy (najbardziej obliczenio-chłonna część algorytmu wykonywana jest rekurencyjnie). Jego wadą jest również to, że daje również pozytywny wynik dla liczb pseudopierwszych przy podstawie 2 (są to liczby złożone). Na szczęście są one dużo rzadsze niż liczby pierwsze.

Opis algorytmu:

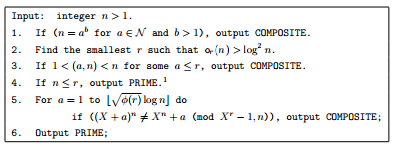
<http://edu.i-lo.tarnow.pl/inf/alg/001_search/0017.php>

http://en.wikipedia.org/wiki/Chinese\_remainder\_theorem

1. Test pierwszości ARS (Test pierwszości Agrawal-Kayal-Saxena)

Jest to test derministyczny! Czas działania algorytmu jest wielomianowy, zależny od liczby cyfr badanej liczby. Zawsze zwraca poprawną odpowiedź. Opiera się na równości: (x - a)^{n} \equiv (x^{n} - a) \pmod{n}, która jest prawdziwa, gdy n jest liczbą pierwszą. Algorytm składa się z dwóch części, wpierw należy znaleźć liczbę r, spełniającą zależność: or(n) > log2n. Następnie należy sprawdzić równość dwóch wielomianów: (x - a)^{n} \equiv (x^{n} - a) \pmod{n, x^{r} - 1}, dla wszystkich a, które spełniają następujący warunek:  a \le 2 \sqrt{r} \log_{2}(n),

Algorytm test ARS:



Największą zaletą tego algorytmu jest to, że zawsze zwraca poprawną odpowiedź, która nie jest obarczona jakimkolwiek błędem. Zdaje się, że algorytm dość łatwo rozdzielić na wiele klastrów obliczeniowych, mimo iż jest dość skomplikowany. Sprawdzanie równości par wielomianów z drugiego kroku może być realizowane niezależnie dla każdej pary.

Pełny opis algorytmu:

http://www.cse.iitk.ac.in/users/manindra/algebra/primality\_v6.pdf