DOI: https://doi.org/10.53052/9788367652001.14

Radosław WALUŚ[1]

Opiekun naukowy: Sławomir HERMA[2], Dawid KOTRYS[3]

LICZBY NADPIERWSZE

Streszczenie: Liczby nadpierwsze to liczby pierwsze, których cyframi są liczby pierwsze. Suma tych cyfr również jest liczbą pierwszą. W artykule zostanie opisany przykładowy algorytm generowania takich liczb oraz ich zastosowanie.

Słowa kluczowe: Liczby pierwsze, algorytm, zastosowanie liczb pierwszych

SUPER-PRIMES

Summary: Super primes are prime numbers whose digits are also prime numbers. The sum of these digits must be a prime number. The article will describe an example of an algorithm for generating such numbers and their application.

Keywords: Prime numbers, algorithm, application of super-primes

1. Liczby nadpierwsze oraz ich właściwości

Liczba nadpierwsza to liczba naturalna, pierwsza, której cyfry są liczbami pierwszymi. Suma tych licz również musi być liczbą pierwszą. Składają się one zawsze z cyfr 2, 3, 5 i 7. Właściwości takich liczb są takie same jak właściwości liczb pierwszych. Odrzucamy tylko te liczby pierwsze, których suma cyfr nie jest liczbą pierwszą. Jako, że zbiór liczb nadpierwszych jest podzbiorem zbioru liczb pierwszych, to zbiór liczb nadpierwszych jest zbiorem nieskończonym (według Dowodu Euklidesa na nieskończoność zbioru liczb pierwszych). W przypadku liczb nadpierwszych nie możemy mówić o liczbach bliźniaczych (za wyjątkiem liczb 3 i 5 oraz 5 i 7). Różnica między dwiema następnymi liczbami zaczynając od liczby 7 jest liczbą 4 bądź jej wielokrotnością, co zostanie uwzględnione w algorytmie generującym te liczby.

2. Algorytm generacji liczb nadpierwszych

Algorytm służący do generowania liczb nadpierwszych złożony jest z trzech głównych funkcji. Sprawdzającej czy dana liczba jest pierwszą, zliczającej poszczególne cyfry liczby oraz funkcji generującej liczby nadpierwsze. Poszczególne

[1] V Liceum Ogólnokształcące w Bielsku-Białej, walusr04c@lo5.bielsko.pl
[2] dr inż., Katedra Inżynierii Produkcji, Akademia Techniczno-Humanistyczna w Bielsku-Białej, Wydział Budowy Maszyn i Informatyki, sherma@ath.bielsko.pl
[3] dr, Katedra Matematyki, Akademia Techniczno-Humanistyczna w Bielsku-Białej, Wydział Budowy Maszyn i Informatyki, dkotrys@ath.bielsko.pl

części tego algorytmu zostaną opisane poniżej. Na potrzeby artykułu został użyty najbardziej prosty sposób sprawdzania i generowania liczb pierwszych.

2.1. Funkcja sprawdzająca pierwszość liczb

Poniżej znajduje się fragment kodu źródłowego badający pierwszość liczb:

```
bool Prime(unsigned long long int n){
    if(n < 2){
        return false;
    }

    for(int i = 2; i*i <= n; i++){
        if(n % i == 0){
            return false;
        }
    }

    return true;
}
```

Jak widać powyżej najpierw sprawdzamy czy dana liczba jest mniejsza od 2, jeżeli tak to nie jest ona liczbą pierwszą i zostaje odrzucona. W przypadku kiedy liczba jest większa od 2 sprawdzamy jej pierwszość, jeżeli nie ma ona żadnych dzielników wykluczając siebie samą to następuje sprawdzenie czy jej cyfry oraz suma tych cyfr również są liczbą pierwszą.

2.2. Funkcja sprawdzająca pierwszość liczb

Na początku funkcja tworzy zmienną, w której zliczana będzie suma cyfr danej liczby, która zostaje przekazana do funkcji, aby została poddana zbadaniu.

```
Bool Sum(unsigned long long int n){
    int addition = 0;

    while(n > 0){
        if(Prime(n % 10) == false){
            return false;
        }

        addition = addition + n % 10;
        n = n / 10;
    }

    if(Prime(addition) == false){
        return false;
    }

    return true;
}
```

Zmienna ta została nazwana “addition”. Po stworzeniu zmiennej i przypisaniu jej wartości 0 zostaje uruchomiona pętla, w której sprawdzana jest pierwszość

poszczególnych cyfr liczby. Wykorzystywana jest do tego funkcja z punktu 2.1. Jeżeli liczba nie spełnia wymogów, pętla zostaje zatrzymana a funkcja zwraca wartość fałsz. Natomiast gdy cyfra jest liczbą pierwszą zostaje ona dodana do naszej zmiennej, która sumuje poszczególne cyfry. Po sprawdzeniu wszystkich cyfr danej liczby następuje sprawdzenie sumy tych cyfr. Jeżeli nie jest ona liczbą pierwszą to funkcja zwraca wartość fałsz, a w przeciwnym wypadku wartość prawda.

2.3. Funkcja generująca liczby nadpierwsze

Funkcja generująca liczby nadpierwsze to główna funkcja opisywanego algorytmu. Przyjmuje ona następującą postać:

```
int main(){
    unsigned long long int max;
    cout << "Enter maximum number: "; cin >> max;

    unsigned long long int i;
    int increment = 1;
    i = 2;

    while(i <= max){
        if(i > 2){
            increment = 2;
        }
        else if(i > 7){
            increment = 4;
        }

        if(Prime(i) == true && Sum(i) == true){
            cout << i << "\n";
        }

        i = i + increment ;
    }

    return 0;
}
```

W pierwszym kroku użytkownik proszony jest o podanie odgórnej granicy, po której algorytm przestaje poszukiwać kolejnych cyfr. Krok ten nie jest potrzebny, najlepszym rozwiązaniem jest poszukiwanie liczb nadpierwszych bez przestawania. W następnym kroku zostaje utworzona zmienna “increment”. Jej celem jest przyspieszenie działania programu przy wykorzystaniu właściwości z punktu 1., właściwością tą jest różnica poszczególnych liczb nadpierwszych o 4 jej wielokrotności. Dla liczb mniejszych bądź równych 7 wartość tej zmiennej wynosi 2. Natomiast od liczby 7 zwiększona zostaje ona do liczby 4 ze względu na to, że różnica pomiędzy kolejnymi liczbami nadpierwszymi jest co najmniej równa 4. W kolejnym kroku sprawdzamy czy nasza liczba spełnia warunki bycia liczbą nadpierwszą tj. sprawdzenie pierwszości jej cyfr oraz sumy tych cyfr. Jeżeli spełnia ona te warunki zostaje wypisana.

2.4. Możliwości rozwoju algorytmu

Algorytm ten używa najprostszego sposobu sprawdzania pierwszości liczby bądź generowania liczb pierwszych. Istnieje wiele innych, bardziej złożonych algorytmów pozwalających przyspieszyć proces generowania liczb nadpierwszych. Przykładowym sposobem wydajniejszego sprawdzania pierwszości jest sposób przedstawiany w algorytmie Miller'a-Rabina'a bądź algorytm Fermat'a. Możliwe jest również przyspieszenie sposobu generowania liczb nadpierwszych. Wszystkie liczby pierwsze oprócz 2 i 3 mają postać:

$$p = 6k \pm 1, k \in N.$$

Pozostałe liczby są postaci:

$$p = 6k = 2 \cdot 3 \cdot k$$
$$p = 6k \pm 2 = 2(3k \pm 1)$$
$$p = 6k \pm 2 = 3(2k \pm 1)$$

Są one podzielne przez 2 lub 3. Są one podzielne przez 2 lub 3 więc nie mogą być liczbami pierwszymi. Wygenerowane w ten sposób liczby sprawdzamy podzielnikami nieparzystymi zaczynając od 5 do pierwiastka z *p*. Zmniejszamy w ten sposób do 1/3 liczbę testowanych liczb.

Sam program możemy również przyspieszyć wiedząc, że liczby posiadające w swoich cyfrach liczby 0, 1, 4, 6, 8 oraz 9 nie są liczbami nadpierwszymi. Znając tą właściwość możliwe jest zmienienie sposobu generowania liczb pomijając liczby zaczynające się od powyższych cyfr tj. 10, 2000 i tym podobne. Pomijając je jesteśmy w stanie drastycznie przyspieszyć prędkość z jaką generowane są liczby nadpierwsze.

3. Zastosowanie liczb nadpierwszych

Podobnie jak z liczbami pierwszymi liczby nadpierwsze można wykorzystać w kryptologii. Szczególną zaletą generowania liczb nadpierwszych po przyspieszeniu algorytmu jest możliwość szybkiego generowanie bardzo dużych liczb nadpierwszych.
Jednym z minusów tego algorytmu jest to, że pomijane są ogromne ilości liczb pierwszych w trakcie generowania liczb nadpierwszych związane z ich właściwościami.

LITERATURA

1. MILLER G. L.: Riemann's Hypothesis and Tests for Primality, Journal of Computer and System Sciences (1976).
2. CALDWELL C. K. XIONG YENG: What is the smallest prime? Journal of Integer Sequences, 15(2012) 12.9.7.
3. DU SAUTOY M.: The Music of the Primes: Searching to Solve the Greatest Mystery in Mathematics, Harper Perennial 2003.