Temat: Arytmetyka modularna

Drugi raport z postępu prac

Projekt został napisany w języku C++. W drugim etapie dodane zostały nowe algorytmy napisane w języku asembler (Chiński Test Pierwszości, Małe Twierdzenie Fermata).

Pierwszym algorytmem, który został zaimplementowany jest Chiński test pierwszości. Główna zasada działania algorytmu jest następująca: jeśli liczba *a*, jest liczbą pierwszą to wyrażenie $2^a - 2$ jest podzielne przez liczbę *a*. Na tej podstawie można wyciągnąć wniosek: jeśli $2^a mod\ a\ \neq 2$ to liczba a nie jest liczbą pierwszą. Zważywszy na fakt, że reszta z dzielenia 2^a przez *a*, jest z przedziału <0; a-1>, w algorytmie można w prosty sposób obliczyć wyrażenie $2^a mod\ a$. Kolejne potęgi są rozbijane na operacje mnożenia modulo a. Dzięki temu istnieje możliwość wyznaczania dużych potęg 2^a poprzez kolejne wymnażanie reszt. W programie do realizacji algorytmu, zostały stworzone dwie metody: *multiplicationModulo(int num1, int num2, int modNum)* oraz *powerModulo(int num1, int numMod)*. Pierwsza z nich zwraca wynik następującej operacji: $(num1 \cdot num2)\ mod\ modNum$. W swoim działaniu wykorzystuje dodawanie wielokrotności mnożnej modulo *modNum*. Druga z funkcji zwraca wartość wyrażenia $2^{num1}\ mod\ modNum$ i w swoim działaniu wykorzystuje funkcję *multiplicationModulo*. Dla ułatwienia implementacji algorytmu oraz możliwości jakie daje język asembler w zakresie możliwości przesuwania bitów zastosowano w jej działaniu maskę bitową. Maska bitowa znajduje zastosowanie, gdyż w końcowym mnożeniu biorą udział tylko te potęgi liczby 2, które odpowiadają wagom bitów ustawionych na 1 w binarnej reprezentacji wykładnika, czyli parametru num1 przekazanego do funkcji *powerModulo*. Gdy funkcja skończy zwraca wynik, który jest porównywany z liczbą 2 i na tej podstawie jest ustalana pierwszość liczby. Należy również zaznaczyć, że powyższy algorytm ma również wady. Jeśli $2^a mod\ a = 2$, to liczba a jest liczbą pierwszą lub pseudopierwszą przy podstawie 2. Jednakże liczby pseudopierwsze przy podstawie 2 występują bardzo rzadko i istnieje niewielkie prawdopodobieństwo, że algorytm zwróci niepoprawny wynik.

Drugim algorytmem, który został zaimplementowany jest algorytm bazujący na Małym Twierdzeniu Fermata. Na początku sprawdzana jest podzielność wprowadzonej liczby przez liczby pierwsze z przedziału <2;1000>. Umożliwia to wstępną eliminację liczb złożonych oraz liczb pseudopierwszych Carmichaela. Jeśli liczba będzie złożona, to zwracana jest od razu informacja, że nie spełnia ona zadanego warunku. Następnie w pętli sprawdzany jest warunek Fermata. Zostaje wylosowana podstawa *a*, która zawiera się w przedziale od 2 do testowanej liczby pomniejszonej o jeden. Następnie sprawdzane jest czy *a* jest względnie pierwsza z wprowadzoną liczbą *num*. Polega to inaczej mówiąc na sprawdzeniu, czy *NWD (num, a) = 1*. Jeśli tak, to testowany jest warunek: $a^{num\ -\ 1}$ *mod num = 1*. Jeśli zostanie spełniony ten warunek, to liczba może być liczbą pierwszą. Jednak nie jest to na tym etapie pewne. Aby to potwierdzić należy sprawdzić to wykonując test Fermata np. dziesięciokrotnie. Liczba, która została uznana liczbą pierwsza może

być również liczbą pseudopierwsza Carmichaela. Liczby te są jednak bardzo odległe, a stosując podzielność przez liczby pierwsze z przedziału <2;1000> można być pewnym, że algorytm daje poprawne wyniki.

Warto zauważyć, że napisanie algorytmów w języku asembler przysporzyło wielu trudności i wymagało dużej ilości czasu. Co ważne, języki wyższego poziomu pozwalają zapisać warunki w jednej linijce, co w asemblerze jest niemożliwe. Nawet proste porównanie można rozpisać na kilka linijek, co czyni kod bardziej zawiłym i złożonym. Wszystkie algorytmy zostały okomentowane.

W następnym etapie projektu będziemy chcieli zaimplementować kolejny algorytm.