Projekt Zespołowy

Etap projektu – projektowanie rozwiązania na zadaną architekturę

Autorzy:
Biernacka Kamila
Kania Dominik
Leśniak Mateusz
Maziarz Wojciech

kwiecień 2021

Abstract

Poniższe sprawozdanie jest wynikiem naszej pracy na drugim etapie projektu zespołowego z implementacji metody indeksu w architekturach GPU. Przedstawimy w nim przygotowane przez nas projekty i rysunki koncepcyjne wymaganych do zaimplementowania algorytmów.

1 Analiza możliwości implementacji algorytmów mnożenia modularnego dużych liczb

Zaproponowanym przez nas algorytmem jest ten odkryty przez rosyjskiego matematyka Anatolija Karacubę. Umożliwia on zmniejszenie złożoności czasowej ($\Theta(n^{log_2 3})$) w porównaniu do mnożenia klasycznego ($\Theta(n^2)$) .

Projekt algorytmu:

Mnożone są dwie n-cyfrowe liczby x i y przy podstawie B, gdzie $n = 2m$. Przetwarzane n może być nieparzyste, a x i y mogą mieć różną liczbę cyfr. W takim przypadku po lewej stronie tych liczb należy dopisać zera. Wartości x i y należy rozpisać jako:

$$x = x_1 B^m + x_2$$
$$y = y_1 B^m + y_2,$$

gdzie x_2, $y_2 < B^m$.
Przemnożenie tych liczb prowadzi do otrzymania równania:

$$xy = (x_1 B^m + x_2)(y_1 B^m + y_2) = x_1 y_1 B^{2m} + (x_1 y_2 + x_2 y_1) B^m + x_2 y_2.$$

Klasycznie problem ten rozwiązuje się poprzez przemnożenie czterech czynników osobno, wykonanie przesunięcia i dodanie ich, co powoduje, że opisany algorytm wykonuje się w czasie $O(n^2)$. Karacuba zaproponował, by zastąpić go trzema mnożeniami:

$$X = x_1 y_1$$
$$Y = x_2 y_2$$
$$Z = (x_1 + x_2)(y_1 + y_2) - X - Y$$

W wyniku tego otrzymuje się równanie:

$$Z = (x_1 y_1 + x_1 y_2 + x_2 y_1 + x_2 y_2) - x_1 y_1 - x_2 y_2 = x_1 y_2 + x_2 y_1.$$

Zatem $xy = XB^{2m} + Y + ZB^m$. Tak więc wystarczy zaledwie kilka dodatkowych dodawań i odejmowań, by zmniejszyć liczbę mnożeń z czterech do trzech.
Algorytm można rozszerzyć i wykonać każde z tych mnożeń m-cyfrowych liczb ponownie w ten sam sposób przy wykorzystaniu rekurencji.

2 Analiza możliwości implementacji algorytmu poszukiwania relacji (oraz faktoryzacji w bazie), dla algorytmu metody indeksu

Algorithm 1: Faktoryzacja, $isFactored$

Input: x, N
Output: $True$ lub $False$
for $i \leftarrow 0;\ i < len(N);\ i \leftarrow i + 1$ **do**
    while $x\%N[i] == 0$ **do**
        $x \leftarrow x/N[i]$
    end
    $i \leftarrow i + 1$
end
if $x == 1$ **then**
    return $True$
end
return $False$

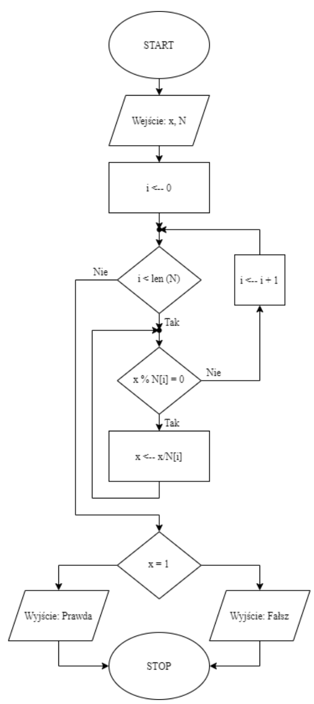

Figure 1: Schemat blokowy algorytmu sprawdzenia faktoryzacji w określonej bazie

Algorithm 2: Poszukiwanie relacji

Input: $\mathbb{B}$, l
Output: N, R

$i \leftarrow 0$
while $p_i \leq \mathbb{B}$, $p_i \in \mathcal{P}$ **do**
    $N[i] \leftarrow p_i$
    $i \leftarrow i + 1$
end
$j \leftarrow 0$
while $j < l$ **do**
    $x \leftarrow random(\mathbb{F}_p^*)$
    if $isFactored(x, N)$ **then**
        $R[j] \leftarrow x$
        $j \leftarrow j + 1$
    end
end
return N, R

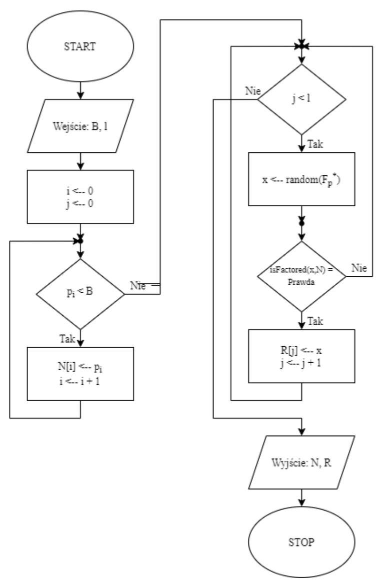

Figure 2: Schemat blokowy algorytmu poszukiwania relacji

3 Analiza możliwości implementacji algorytmu eliminacji Gaussa nad ciałem $\mathbb{F}_p$, dla ciał o dowolnym rozmiarze

Algorithm 3: Eliminacja Gaussa

Input: $\mathbb{A}_{l,k+1}$
Output: x

for $i \leftarrow 0;\ i < l;\ i \leftarrow i + 1$ **do**
    for $j \leftarrow i;\ j < k;\ j \leftarrow j + 1$ **do**
        $A_j \leftarrow A_j - w \cdot A_j$, gdzie $w = [w_0, w_1, \ldots, w_k]$, $w_r = \frac{\mathbb{A}_{r,j}}{\mathbb{A}_{i,i}}$
    end
end
$x \leftarrow A_k$
return x

4 Metoda indeksu

Metoda indeksu podzielona jest na 4 etapy.

1. Konstruowanie zbioru relacji wykorzystując algorytm 2.
2. Rozwiązanie układu równań z wykorzystując algorytm 3.
3. Wyznaczenie wartości r;
4. Wyznaczenie rozwiązania;

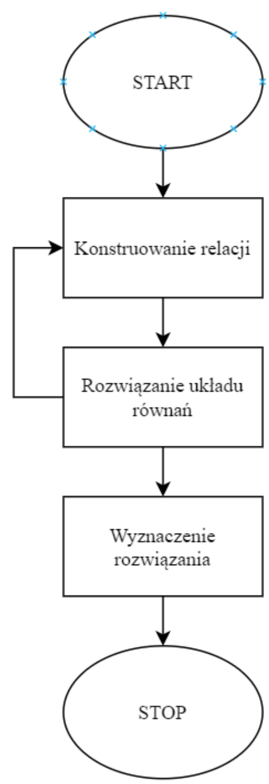

Figure 3: Rysunek koncepcyjny metody indeksu