

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:

$$\begin{aligned}x &= x_1 B^m + x_2 \\ y &= y_1 B^m + y_2,\end{aligned}$$

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:

$$\begin{aligned}X &= x_1 y_1 \\ Y &= x_2 y_2 \\ Z &= (x_1 + x_2)(y_1 + y_2) - X - Y\end{aligned}$$

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 $xy = X B^{2m} + Y + Z B^m$. 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*

```

1 for  $i \leftarrow 0; i < \text{len}(N); i \leftarrow i + 1$  do
2   while  $x \% N[i] == 0$  do
3      $x \leftarrow x / N[i]$ 
4   end
5    $i \leftarrow i + 1$ 
6 end
7 if  $x == 1$  then
8   return True
9 end
10 return False

```

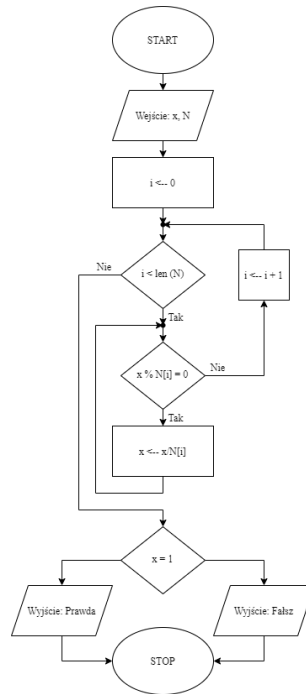


Figure 1: Schemat blokowy algorytmu sprawdzenia faktoryzacji w określonej bazie

Algorithm 2: Poszukiwanie relacji

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

```
1  $i \leftarrow 0$ 
2 while  $p_i \leq \mathbb{B}, p_i \in \mathcal{P}$  do
3    $N[i] \leftarrow p_i$ 
4    $i \leftarrow i + 1$ 
5 end
6  $j \leftarrow 0$ 
7 while  $j < l$  do
8    $x \leftarrow \text{random}(\mathbb{F}_p^*)$ 
9   if  $\text{isFactored}(x, N)$  then
10     $R[j] \leftarrow x$ 
11     $j \leftarrow j + 1$ 
12  end
13 end
14 return  $N, R$ 
```

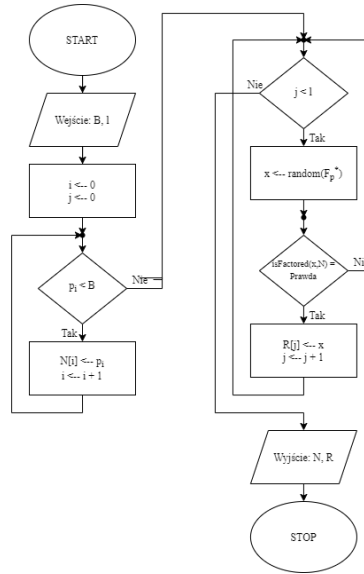


Figure 2: Schemat blokowy algorytmu poszukiwania relacji

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

```
1 for  $i \leftarrow 0; i < l; i \leftarrow i + 1$  do
2   for  $j \leftarrow i; j < k; j \leftarrow j + 1$  do
3      $A_j \leftarrow A_j - w \cdot A_i$ , gdzie  $w = \frac{\mathbb{A}_{k,j}}{\mathbb{A}_{i,i}}$ 
4   end
5 end
```

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;

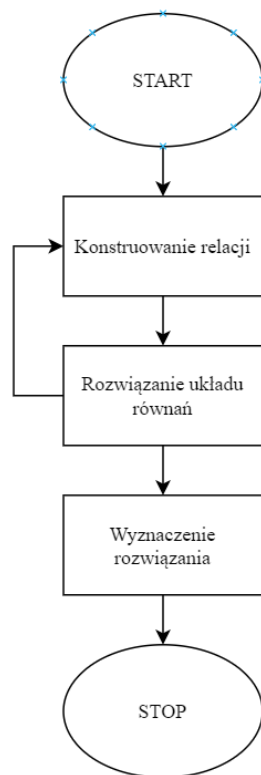


Figure 3: Schemat blokowy algorytmu metody indeksu