

Projekt Zespołowy

Etap projektu – projektowanie  
rozwiązania na zadaną  
architekturę

Autorzy:  
Biernacka Kamila  
Kania Dominik  
Leśniak Mateusz  
Maziarz Wojciech

kwiecień 2021

## Streszczenie

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.

## Spis treści

<b>1</b>	<b>Mnożenie modularne dużych liczb</b>	<b>3</b>
<b>2</b>	<b>Poszukiwanie relacji i faktoryzacja w bazie</b>	<b>4</b>
2.1	Szybkie potęgowanie modularne . . . . .	4
2.2	Faktoryzacja w bazie . . . . .	4
2.3	Budowa relacji . . . . .	4
<b>3</b>	<b>Eliminacja Gaussa w pierścieniu <math>\mathbb{Z}_{p-1}</math></b>	<b>4</b>
3.1	Algorytm Euklidesa . . . . .	4
3.2	Rozszerzony algorytm Euklidesa . . . . .	4

# 1 Mnożenie modularne dużych liczb

W celu wykonania mnożenia dużych liczb  $a, b \in \mathbb{F}_p$  wykorzystamy algorytm 2. Pierwszym krokiem jest przedstawienie liczb  $a, b$  w postaci  $a = x_1 \cdot 2^{32} + y_1$  oraz  $b = x_2 \cdot 2^{32} + y_2$ .

Wtedy

$$r_p(ab) = r_p((x_1 \cdot 2^{32} + y_1)(x_2 \cdot 2^{32} + y_2)) = r_p(x_1 x_2 \cdot 2^{64}) +_p r_p(x_1 y_2 \cdot 2^{32}) +_p r_p(x_2 y_1 \cdot 2^{32}) +_p r_p(y_1 y_2).$$

Do wyznaczenia pośrednich wartości  $r_p$  wykorzystywany jest algorytm 1. Algorytm mnożenia pośredniego działa analogicznie do algorytmu 2. Różnicą jest przedstawienie czynników jako  $x \cdot 2^{16} + y$ .

---

## Algorithm 1: Mnożenie pośrednie, *halfMult*

---

**Input:**  $a, b$  - dwie liczby całkowite,  $p$  - modułnik

**Output:** *result* - wynik mnożenia

```

1  $x_1 \leftarrow a \gg 16$ 
2  $y_1 \leftarrow a \&\& 0xffff$ 
3  $x_2 \leftarrow b \gg 16$ 
4  $y_2 \leftarrow b \&\& 0xffff$ 
5  $half_a \leftarrow (((x_1 x_2) \% p) \cdot r_p(2^{32})) \% p$ 
6  $half_b \leftarrow (((x_1 y_2) \% p) \cdot r_p(2^{16})) \% p$ 
7  $half_c \leftarrow (((x_2 y_1) \% p) \cdot r_p(2^{16})) \% p$ 
8  $half_d \leftarrow (y_1 y_2) \% p$ 
9 return  $(half_a + half_b + half_c + half_d) \% p$ 
```

---



---

## Algorithm 2: Pełne mnożenie modularne dwóch liczb

---

**Input:**  $a, b$  - dwie liczby całkowite,  $p$  - modułnik

**Output:** *result* - wynik mnożenia

```

1  $x_1 \leftarrow a \gg 32$ 
2  $y_1 \leftarrow a \&\& 0xffffffff$ 
3  $x_2 \leftarrow b \gg 32$ 
4  $y_2 \leftarrow b \&\& 0xffffffff$ 
5  $half_a \leftarrow (halfMult(x_1, x_2) \cdot r_p(2^{64})) \% p$ 
6  $half_b \leftarrow (halfMult(x_1, y_2) \cdot r_p(2^{32})) \% p$ 
7  $half_c \leftarrow (halfMult(x_2, y_1) \cdot r_p(2^{32})) \% p$ 
8  $half_d \leftarrow halfMult(y_1, y_2)$ 
9 return  $(half_a + half_b + half_c + half_d) \% p$ 
```

---

## 2 Poszukiwanie relacji i faktoryzacja w bazie

### 2.1 Szybkie potęgowanie modularne

### 2.2 Faktoryzacja w bazie

### 2.3 Budowa relacji

## 3 Eliminacja Gaussa w pierścieniu $\mathbb{Z}_{p-1}$

### 3.1 Algorytm Euklidesa

### 3.2 Rozszerzony algorytm Euklidesa