



GROUP

31



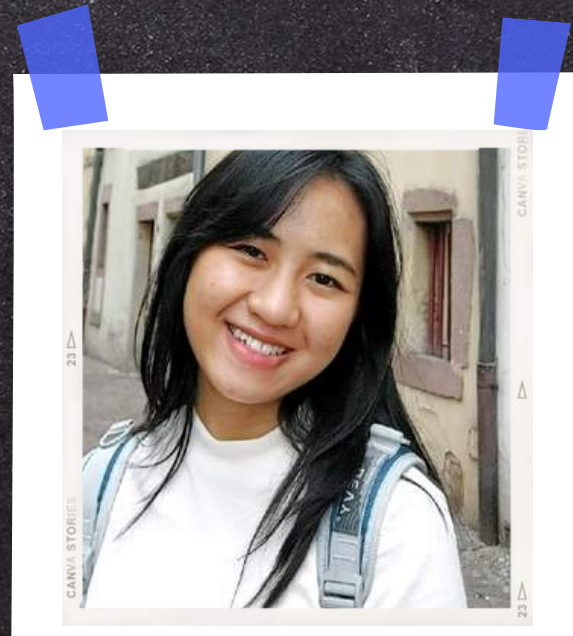
# The team



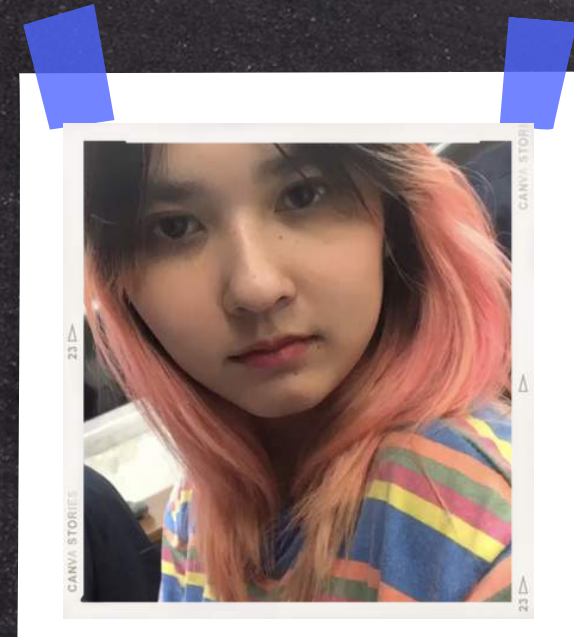
บุริศ เสรีวัตตะนะ  
64010462



คิวกกร สุริยะะ  
64010850



สุพิชญา พรหมपालิต  
64010926



สุรางคนางค์ เกตุยั้งยืนวงศ์  
64010936



อรัชฌา ปิ่นประยูร  
64010994



รณกฤต ธรรมภาณพินิจ  
64011131





# Table of contents

01

version 0

04

version 1

08

version 2

12

version 3

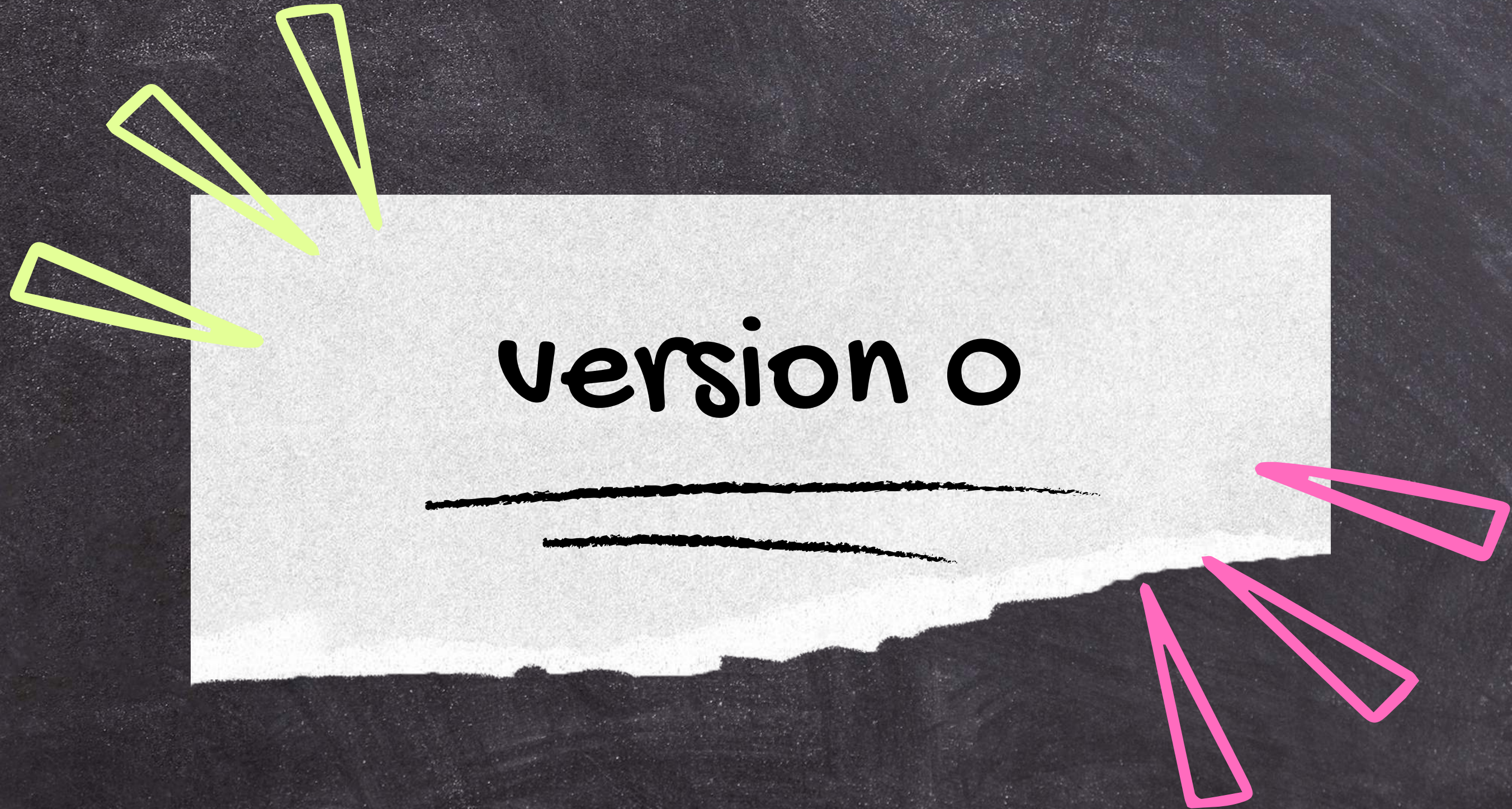
16

version 4





version 0







**ผลลัพธ์**

```
Data read...Complete.
```

```
Working...Done.
```

```
Summation result: 888701676
```

```
Time used: 10645ms
```



**ปัญหาที่พบ**

มีการใช้ระยะเวลาในการรันมากเกินไป  
ต้องการใช้น้อยกว่าเดิม



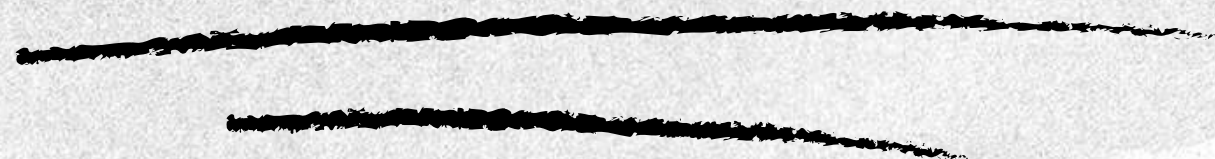


# ข้อสันนิษฐาน

มีการรับ for loop ใน function main() เพื่อเรียก function sum()  
ทำให้มีการ pop, push ของ instruction address ใน stack หลายรอบ  
ทำให้ใช้เวลาในการรันมากขึ้น



version 1





## สิ่งที่แก้ไข

จาก version 0 มีการเปลี่ยนแปลงจากการใช้ for loop ใน function main() เป็นการใช้ for loop ใน function sum() แทน โดยเปลี่ยนจาก  
for (int i = 0; i < 10000000000; i++) ใน static void main() ไปอยู่ใน static void sum()

```
static void sum()
{
    for (int i = 0; i < 10000000000; i++)
    {
        if (Data_Global[G_index] % 2 == 0)
        {
            Sum_Global -= Data_Global[G_index];
        }
        else if (Data_Global[G_index] % 3 == 0)
        {
            Sum_Global += (Data_Global[G_index] * 2);
        }
        else if (Data_Global[G_index] % 5 == 0)
        {
            Sum_Global += (Data_Global[G_index] / 2);
        }
        else if (Data_Global[G_index] % 7 == 0)
        {
            Sum_Global += (Data_Global[G_index] / 3);
        }
        Data_Global[G_index] = 0;
        G_index++;
    }
}
```

```
/* Start */
Console.WriteLine("\n\nWorking...");
sw.Start();
//for (i = 0; i < 10000000000; i++)
    sum();
sw.Stop();
Console.WriteLine("Done.");
```



ผลลัพธ์

```
Data read...Complete.
```

```
Working...Done.
```

```
Summation result: 888701676
```

```
Time used: 10645ms
```

ปัญหาที่พบ


แม้ว่าการประมวลผลจะเร็วกว่า version 0 แต่เวลาในการ execute ของ version 1 ยังใช้เวลานานอยู่ ซึ่งเร็วกว่า version ก่อนหน้าเพียง 1 วินาที





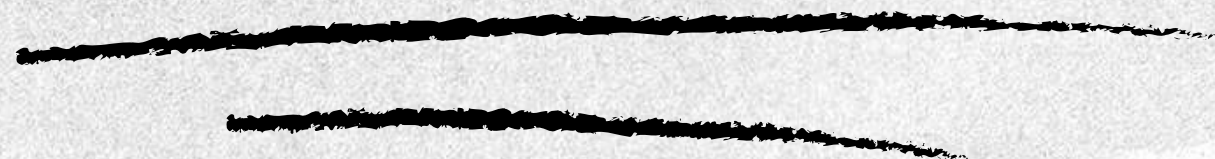
# ข้อสันนิษฐาน

หลังจากได้ทำการปรับแต่งโค้ดที่แต่ละการเรียกใช้ function แล้ว แต่การทำงานของโปรแกรมยังเป็นแบบ single thread อยู่ โดยใช้เพียง thread เดียวในการ loop จำนวน 1 พันล้านรอบเพื่อคำนวณคำตอบ จึงใช้เวลาค่อนข้างนาน





version 2





# สิ่งที่แก้ไข

- เพิ่มเป็น 2 THREADS ให้ทำงานพร้อมกัน
- เปลี่ยน INDEX ของ DATA\_GLOBAL ให้ใช้ค่า I ในการชี้ INDEX แทน เนื่องจากการใช้ G\_INDEX++ จากทั้ง 2 FUNCTION อาจทำให้ค่า INDEX ของ DATA\_GLOBAL ไม่ถูกต้อง
- สร้าง function ขึ้นมาใหม่สำหรับการทำงานของ Threads
  - sum\_1() สำหรับ for (int i = 0; i < 5000000000; i++)
  - sum\_2() สำหรับ for (int i = 5000000000; i < 10000000000; i++)

```
static void sum1()
{
    for (int i = 0; i < 5000000000; i++)
    {
        if (Data_Global[i] % 2 == 0)
        {
            Sum_Global -= Data_Global[i];
        }
        else if (Data_Global[i] % 3 == 0)
        {
            Sum_Global += (Data_Global[i] * 2);
        }
        else if (Data_Global[i] % 5 == 0)
        {
            Sum_Global += (Data_Global[i] / 2);
        }
        else if (Data_Global[i] % 7 == 0)
        {
            Sum_Global += (Data_Global[i] / 3);
        }
        Data_Global[i] = 0;
        //G_index++;
    }
}
```

```
static void sum2()
{
    for (int i = 5000000000; i < 10000000000; i++)
    {
        if (Data_Global[i] % 2 == 0)
        {
            Sum_Global -= Data_Global[i];
        }
        else if (Data_Global[i] % 3 == 0)
        {
            Sum_Global += (Data_Global[i] * 2);
        }
        else if (Data_Global[i] % 5 == 0)
        {
            Sum_Global += (Data_Global[i] / 2);
        }
        else if (Data_Global[i] % 7 == 0)
        {
            Sum_Global += (Data_Global[i] / 3);
        }
        Data_Global[i] = 0;
        //G_index++;
    }
}
```

```
Thread th1 = new Thread(sum1);
Thread th2 = new Thread(sum2);

th1.Start();
th2.Start();

th1.Join();
th2.Join();
```



ผลลัพธ์

```
Data read...Complete.
```

```
Working...Done.
```

```
Summation result: 532689246
```

```
Time used: 5174ms
```

ปัญหาที่พบ

ค่า Sum\_Global ไม่ถูกต้อง





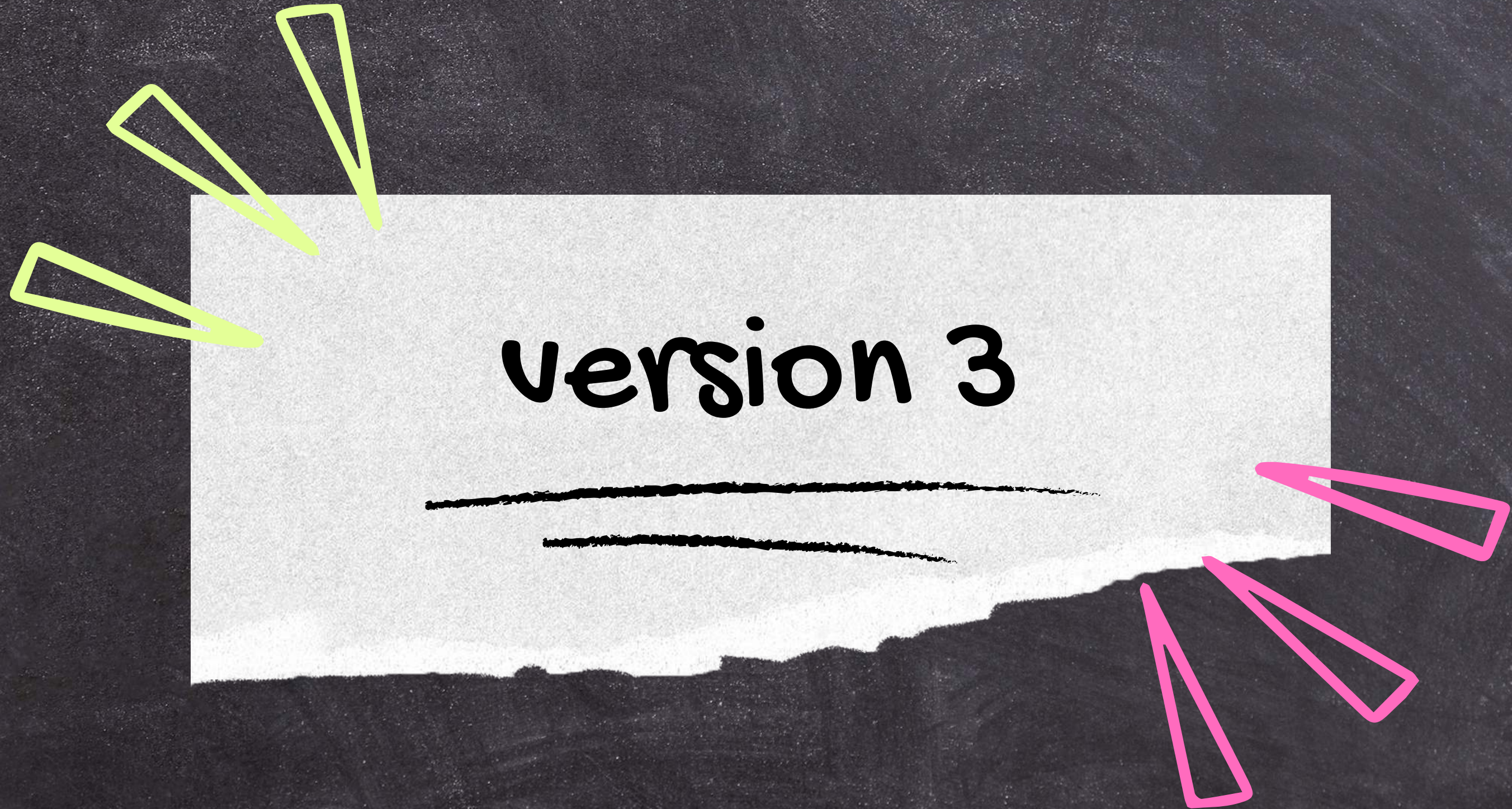
# ข้อสันนิษฐาน



โปรแกรมนี้มี Thread มากกว่า 1 ทำงานพร้อม ๆ กัน ดังนั้นค่า Sum\_Global  
จึงอาจจะมีการเข้าถึงหรืออัปเดตค่ามากกว่า 1 thread ทำให้เมื่อถูกอัปเดตค่า  
พร้อมกันจากหลาย ๆ thread ทำให้ค่า Sum\_Global ผิดพลาด และส่งผลให้  
Function sum1 และ sum2 คำนวณผลลัพธ์ออกมาได้ไม่ถูกต้อง



version 3





# สิ่งที่แก้ไข

- เพิ่มตัวแปร SUM\_LOCAL แทนใน FUNCTION SUM1 และ SUM2 เพื่อให้แต่ละ THREAD ใช้งาน ตัวแปรของตัวเอง เพื่อรวมค่าผลลัพธ์สำหรับการคำนวณใน DATA\_GLOBAL
- เมื่อรวบรวมค่าผลลัพธ์เสร็จ จึงค่อยนำไปรวมกันกับ ตัวแปร SUM\_GLOBAL ในขั้นตอนสุดท้ายของ FUNCTION

```
static void sum1()
{
    long sum_local = 0;
    for (int i = 0; i < 500000000; i++)
    {
        if (Data_Global[i] % 2 == 0)
        {
            sum_local -= Data_Global[i];
        }
        else if (Data_Global[i] % 3 == 0)
        {
            sum_local += (Data_Global[i] * 2);
        }
        else if (Data_Global[i] % 5 == 0)
        {
            sum_local += (Data_Global[i] / 2);
        }
        else if (Data_Global[i] % 7 == 0)
        {
            sum_local += (Data_Global[i] / 3);
        }
        Data_Global[i] = 0;
        //G_index++;
    }

    Sum_Global += sum_local;
}
```

```
static void sum2()
{
    long sum_local = 0;
    for (int i = 500000000; i < 1000000000; i++)
    {
        if (Data_Global[i] % 2 == 0)
        {
            sum_local -= Data_Global[i];
        }
        else if (Data_Global[i] % 3 == 0)
        {
            sum_local += (Data_Global[i] * 2);
        }
        else if (Data_Global[i] % 5 == 0)
        {
            sum_local += (Data_Global[i] / 2);
        }
        else if (Data_Global[i] % 7 == 0)
        {
            sum_local += (Data_Global[i] / 3);
        }
        Data_Global[i] = 0;
        //G_index++;
    }

    Sum_Global += sum_local;
}
```



ผลลัพธ์

```
Data read...Complete.
```

```
Working...Done.
```

```
Summation result: 888701676
```

```
Time used: 4995ms
```

ปัญหาที่พบ

ผลลัพธ์ของโปรแกรมถูกต้องแต่ความเร็วดีขึ้นเพียงเล็กน้อย จึงอยากแก้ไขโปรแกรมให้คำนวณได้เร็วขึ้นกว่าเดิม





# ข้อสันนิษฐาน



จากการทดลองแก้ไขโปรแกรมทำให้เราได้ทราบว่า จำนวน Thread มีผลต่อเวลาในการคำนวณ ดังนั้น จึงสันนิษฐานได้ว่า หากจำนวน Thread มีมากกว่า 2 จะทำให้โปรแกรมสามารถประมวลผลได้เร็วกว่าเดิม



version 4

~~version 3~~



# สิ่งที่แก้ไข

```
* number_of_thread = Environment.ProcessorCount;  
* PREFERRED_DATA_SIZE = MAX_DATA_SIZE + number_of_thread - (MAX_DATA_SIZE % number_of_thread);
```

## คำนวณขนาดของ DATA

คำนวณขนาดของ data ที่ใช้ต่อ 1 Thred โดยกำหนดให้ PREFERRED\_DATA\_SIZE มีค่าเท่ากับ MAX\_DATA\_SIZE + number\_of\_th - (MAX\_DATA\_SIZE % number\_of\_th) และทำการจำกัดให้การวน loop สูงสุดที่ 1000 ล้านรอบ

```
static void sum(int start, int end)  
{  
    int Sum_Local = 0;  
    for (int i = start; i < end; i++)  
    {  
        Sum_Local += i;  
    }  
}
```

```
// สร้าง Thread จำนวน Thread ที่ต้องการ  
Thread[] th = new Thread[number_of_thread];  
int dataSizePerThread = PREFERRED_DATA_SIZE / number_of_thread;  
  
for (i = 0; i < th.Length; i++)  
{  
    int start = i * dataSizePerThread;  
    int end = (i + 1) * dataSizePerThread;  
    if (i == th.Length - 1)  
    {  
        end = MAX_DATA_SIZE;  
    }  
    else {  
        end = (i + 1) * dataSizePerThread;  
    }  
    // ทำให้ Thread ทำงานแบบขนาน โดยไม่ parameter  
    th[i] = new Thread(() => sum(start, end));  
}
```

## หาจุด START และ END

ใช้ for loop ในการกำหนดช่วงการทำงานของแต่ละ Thread ผ่าน function sum โดยมีการระบุพารามิเตอร์ start และ end ซึ่งใช้เป็นตัวแปรในการกำหนดช่วงของ index เริ่มต้นและสิ้นสุดในการทำงานของแต่ละ Thread ตามลำดับ





**ผลลัพธ์**

```
Data read...Complete.
```

```
Working...Done.
```

```
Summation result: 888701676
```

```
Time used: 1412ms
```



# ตารางเปรียบเทียบ

version	0	1	2	3	4
summation result	888701676	888701676	532689246	888701676	888701676
Time used (s)	11.316	10.645	5.174	4.995	1.412
Thread used (s)	1	1	2	2	ตามจำนวน thread ในเครื่อง





Thank  
you

