# Algorytmy i struktury danych

*Projekt nr 2. Wykonał Daniel Głowacki gr. K35.2*

## Cel zadania:

Dana jest poniższa implementacja algorytmu badania czy zadana liczba jest pierwsza:

```
bool IsPrime(BigInteger Num)
{
  if (Num < 2) return false;
  else if (Num < 4) return true;
  else if (Num % 2 == 0) return false;
  else for (BigInteger u = 3; u < Num / 2; u += 2)
          if (Num % u == 0) return false;
  return true;
}
```

Celem projektu jest zaproponowanie bardziej efektywnego algorytmu przy zachowaniu niezmienionego interfejsu podprogramu. Przeprowadzić analizę za pomocą instrumentacji i pomiarów czasu. Przyjąć, że operacją dominującą jest dzielenie modulo (%).

W sprawozdaniu przedstawić dla obu algorytmów:

- kod źródłowy przed instrumentacją
- kod źródłowy po instrumentacji
- zebrane wyniki w postaci tekstu i wykresów
- wnioski z analizy zebranych danych (ocena złożoności)

Badanie przeprowadzić dla następującego zbioru punktów pomiarowych (liczb pierwszych):

{ 100913, 1009139, 10091401, 100914061, 1009140611, 10091406133, 100914061337, 1009140613399 }

## Uwagi:

Dla ostatniej liczby w algorytmie przykładowym, zarówno czas wykonania jak i liczba operacji, zostały obliczone metodą proporcji, ze względu na bardzo długi okres działania programu.

### Dodatkowe informacje:

Programy zostały napisane przy pomocy Visual Studio 2017. Badanie natomiast zostało przeprowadzone na komputerze z procesorem Intel (R) Atom (TM) CPU N2600 @ 1.60 GHz 1.60 GHz.

Średni czas algorytmu przyzwoitego został wyliczony bez wzięcia pod uwagę pierwszego pomiaru dla pierwszej liczby z tablicy. Wydłużony czas mógł wynikać z wykorzystania zasobów komputera do innych procesów.

## Algorytm przykładowy z projektu.

Kod algorytmu przykładowego bez instrumentacji:

```
public static bool IsPrime(BigInteger Num)
{
    if (Num < 2) return false;
    else if (Num < 4) return true;
    else if (Num % 2 == 0) return false;
    else for (BigInteger u = 3; u < Num / 2; u += 2)
            if (Num % u == 0) return false;
    return true;
}
```

Kod algorytmu przykładowego po zastosowaniu instrumentacji:

```
public static bool IsPrimeWithCounter(BigInteger Num)
{
    counter = 0;
    if (Num < 2) return false;
    else if (Num < 4) return true;
    else if (Num % 2 == 0) return false;
    else
    {
        counter++;
        for (BigInteger u = 3; u < Num / 2; u += 2)
        {
            counter++;
            if (Num % u == 0)
            {

                return false;
            }
        }

    }
    return true;
```

Kod dla pomiaru czasu zastosowany w metodzie Main:

```
Console.WriteLine("Algorytm sprawdzania liczb pierwszych + czas");
Console.WriteLine("liczba;prime;czas");
for (int l = 0; l < tab.Length; l++)
{
    sw.Start();
    prime = IsPrime(tab[l]);
    sw.Stop();
    Console.WriteLine(tab[l] + "; " + prime + "; " + sw.Elapsed.TotalMilliseconds);
    sw.Reset();
}
```

Metoda Main:

```
static void Main(string[] args)
{
    bool prime;
    Stopwatch sw = new Stopwatch();
    BigInteger[] tab = new BigInteger[] { 100913, 1009139, 10091401, 100914061, 1009140611, 10091406133, 100914061337, 1009140613399 };
    Console.WriteLine("Algorytm sprawdzania liczb pierwszych z wyliczaniem modulo");
    Console.WriteLine("liczba;prime;modulo");
    for (int k = 0; k < tab.Length; k++)
    {
        Console.WriteLine(tab[k] + "; " + IsPrimeWithCounter(tab[k]) + "; " + counter);

    }
    Console.WriteLine();
    Console.WriteLine("Algorytm sprawdzania liczb pierwszych + czas");
    Console.WriteLine("liczba;prime;czas");
    for (int l = 0; l < tab.Length; l++)
    {
        sw.Start();
        prime = IsPrime(tab[l]);
        sw.Stop();
        Console.WriteLine(tab[l] + "; " + prime + "; " + sw.Elapsed.TotalMilliseconds);
        sw.Reset();
    }
}
```

Wyniki dla algorytmu przykładowego wraz z wykresami:

| liczba | modulo | czas |
|---|---|---|
| 100913 | 25228 | 19,4005 |
| 1009139 | 252284 | 161,2983 |
| 10091401 | 2522850 | 1616,0275 |
| 100914061 | 25228515 | 16159,2451 |
| 1009140611 | 252285152 | 162034,7023 |
| 10091406133 | 2522851533 | 4096872,6777 |
| 100914061337 | 25228515334 | 66555561,7277 |
| 1009140613399 | 252285153347 | 665555617,2961 |


## Algorytm przyzwoity.

Kod algorytmu przyzwoitego bez instrumentacji z zaznaczoną zmianą w pętli, usprawniającą znacząco sprawdzanie czy liczba jest pierwsza:

```
public static bool IsPrimeFaster(BigInteger Num)
{
    if (Num < 2) return false;
    else if (Num < 4) return true;
    else if (Num % 2 == 0) return false;
    else for (BigInteger u = 3; u * u <= Num; u += 2)
            if (Num % u == 0) return false;
    return true;
}
```

Kod algorytmu przyzwoitego po zastosowaniu instrumentacji:

```
public static bool IsPrimeFasterWithCounter(BigInteger Num)
{
    counter = 0;
    if (Num < 2) return false;
    else if (Num < 4) return true;
    else if (Num % 2 == 0) return false;
    else
    {
        counter++;
        for (BigInteger u = 3; u * u <= Num; u += 2)
        {
            counter++;
            if (Num % u == 0)
            {

                return false;
            }
        }

    }
    return true;
}
```

Kod dla pomiaru czasu zastosowany w metodzie Main:

```
for (int i = 0; i < tab.Length; i++)
{
    for (int j = 0; j < repeat + 2; ++j)//pętla powtarzająca każdy pomiar czasu 12 razy
    {
        sw.Start();
        prime = IsPrimeFaster(tab[i]);
        sw.Stop();
        Console.WriteLine(tab[i] + "; " + prime + "; " + sw.Elapsed.TotalMilliseconds);
        sw.Reset();
    }
}
```

Kod metody Main:

```
static void Main(string[] args)
{
    bool prime;
    Stopwatch sw = new Stopwatch();
    BigInteger[] tab = new BigInteger[] { 100913, 1009139, 10091401, 100914061, 1009140611, 10091406133, 100914061337, 1009140613399 };
    Console.WriteLine("Usprawniony algorytm sprawdzania liczb pierwszych + czas");
    Console.WriteLine("liczba;prime;czas");
    for (int i = 0; i < tab.Length; i++)
    {
        for (int j = 0; j < repeat + 2; ++j)//pętla powtarzająca każdy pomiar czasu 12 razy
        {
            sw.Start();
            prime = IsPrimeFaster(tab[i]);
            sw.Stop();
            Console.WriteLine(tab[i] + "; " + prime + "; " + sw.Elapsed.TotalMilliseconds);
            sw.Reset();
        }
    }
    Console.WriteLine();
    Console.WriteLine("Usprawniony algorytm sprawdzania liczb pierwszych z wyliczaniem modulo");
    Console.WriteLine("liczba;prime;modulo");
    for (int k = 0; k < tab.Length; k++)
    {
        Console.WriteLine(tab[k] + "; " + IsPrimeFasterWithCounter(tab[k]) + "; " + counter);
    }
}
```

Wyniki dla algorytmu przyzwoitego wraz z wykresami:

| **liczba** | **modulo** | **czas przyzwoity - uśredniony** |
|---|---|---|
| 100913 | 159 | 0,0922 |
| 1009139 | 502 | 0,2973 |
| 10091401 | 1588 | 0,9086 |
| 100914061 | 5023 | 2,8670 |
| 1009140611 | 15883 | 9,0526 |
| 10091406133 | 50228 | 39,0436 |
| 100914061337 | 158835 | 133,4809 |
| 1009140613399 | 502280 | 428,9581 |



## Wnioski.

Algorytm przyzwoity ma znaczącą przewagę w znajdowaniu liczb pierwszych względem przykładowego.

Dla pierwszej sprawdzanej liczby jest on ok. 159 razy szybszy, natomiast dla ostatniej liczby z zadanego zakresu jest on już ok. 502280 razy szybszy.