Projekt 2

do samodzielnego wykonania

Dane 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 }

Komentarz do zadania:

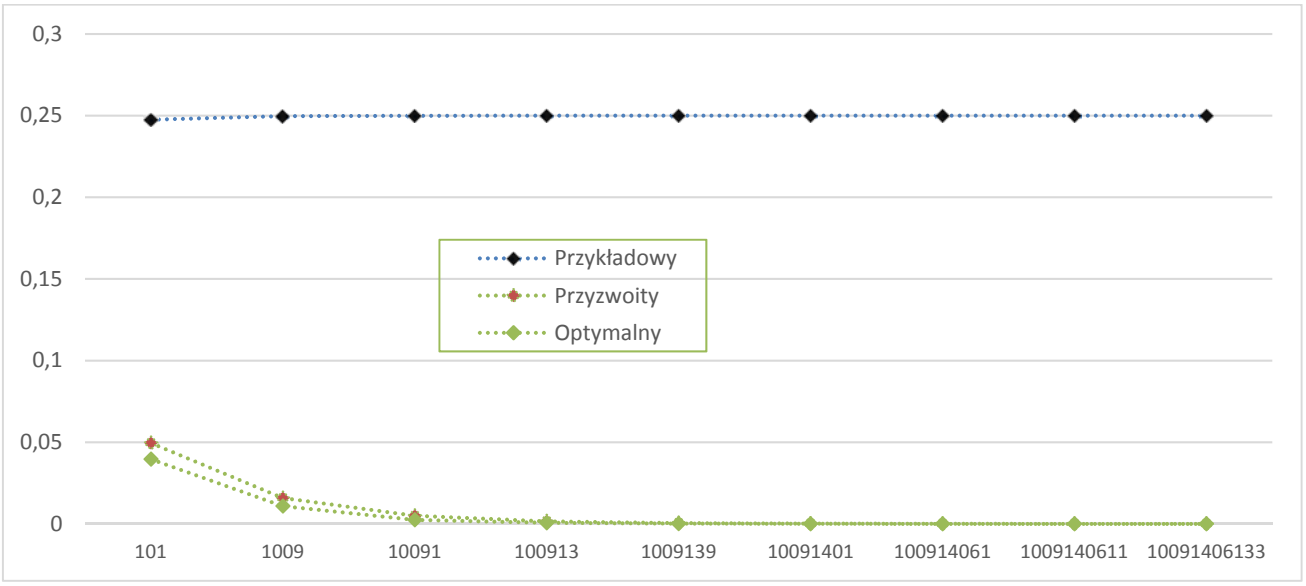
Na ile można usprawnić przedstawiony w przykładzie algorytm ?

Niewielkim nakładem wysiłku, odpowiednio modyfikując pętlę, można uzyskać ogromną poprawę wydajności (zarówno rzędu funkcji jak i współczynników). Jak ? Zasady konstrukcji algorytmu, nazwijmy go „przyzwoitym", są omówione np. w rozdziale 8.1 w http://mmsyslo.pl/content/download/296/1176/version/1/file/Piramidy_szyszki_08.pdf (niestety, nie ma tam kodu źródłowego do wklejenia ☺). Na marginesie: przestrzegam przed wykorzystywaniem liczb zmiennoprzecinkowych (*float*, *double*, pakietu *Math* …) w implementacji.

Ale uwaga: większym nakładem wysiłku można dojść do algorytmu optymalnego – znajdującego odpowiedź w minimalnej liczbie dzieleń. Przyrost wydajności nie jest już tak spektakularny, ale nadal zauważalny. A są jeszcze efektywniejsze podejścia …
Porównajmy liczbę dzieleń wykonywanych w zadanych punktach pomiarowych przez wszystkie trzy algorytmy:

n	przykładowy	przyzwoity	optymalny
101	25	5	4
1009	252	16	11
10091	2522	50	25
100913	25228	159	66
1009139	252284	502	168
10091401	2522850	1588	449
100914061	25228515	5023	1233
1009140611	252285152	15883	3415
10091406133	2522851533	50228	9628

Zdecydowanie lepiej widać charakterystyki poszczególnych algorytmów i poprawę wydajności algorytmu optymalnego nad „przyzwoitym” na wykresie przedstawiającym relatywną liczbę dzieleń:



Można zauważyć również inną, niekorzystną tendencję – wraz ze wzrostem badanej liczby w kolejnych punktach pomiarowych relatywna przewaga algorytmu optymalnego nad „przyzwoitym” maleje. A przecież zależy nam na minimalizacji czasu obliczeń dla dużych liczb …

Oczywiście w kategorii bezwzględnej liczby dzieleń algorytm optymalny jest zdecydowanie wydajniejszy niż „przyzwoity”, jego przewaga bezwzględna rośnie wraz ze wzrostem badanej liczby (ale wolniej niż sama liczba):