Sprawozdanie z projektu nr 2

W ramach projektu zadaniem było zbadanie złożoności obliczeniowej dla algorytmu sprawdzającego, czy podana liczba jest liczbą pierwszą. Wykorzystano dwie wersje tegoż algorytmu (nazwane w dalszej części sprawozdania odpowiednio „niezoptymalizowanym" i „przyzwoitym"), które różnią się od siebie warunkiem wykonania pętli.

Podczas analizy wzięto pod uwagę pomiary przy użyciu:

a) instrumentacji
b) pomiarów czasu

Przy wykorzystaniu instrumentacji do oceny złożoności algorytmu wybrano jako operację dominującą zliczanie operacji dzielenia modulo (%).

Kody źródłowe wykorzystane w eksperymencie można odnaleźć w repozytorium https://github.com/daniel-sobczak-wsb/prime_numbers w katalogu „code", zaś w katalogu „results" zawarto arkusze z wynikami pomiarów. Poniżej znajduje się podsumowanie złożoności obliczeniowych obu wersji algorytmu w postaci wykresów.

Poniższe pomiary zostały wykonane na laptopie Lenovo ThinkPad T440 (procesor Intel Core i5-4300U) z wykorzystaniem edytora Visual Studio Code dostosowanym do użytku z platformą .NET Core.

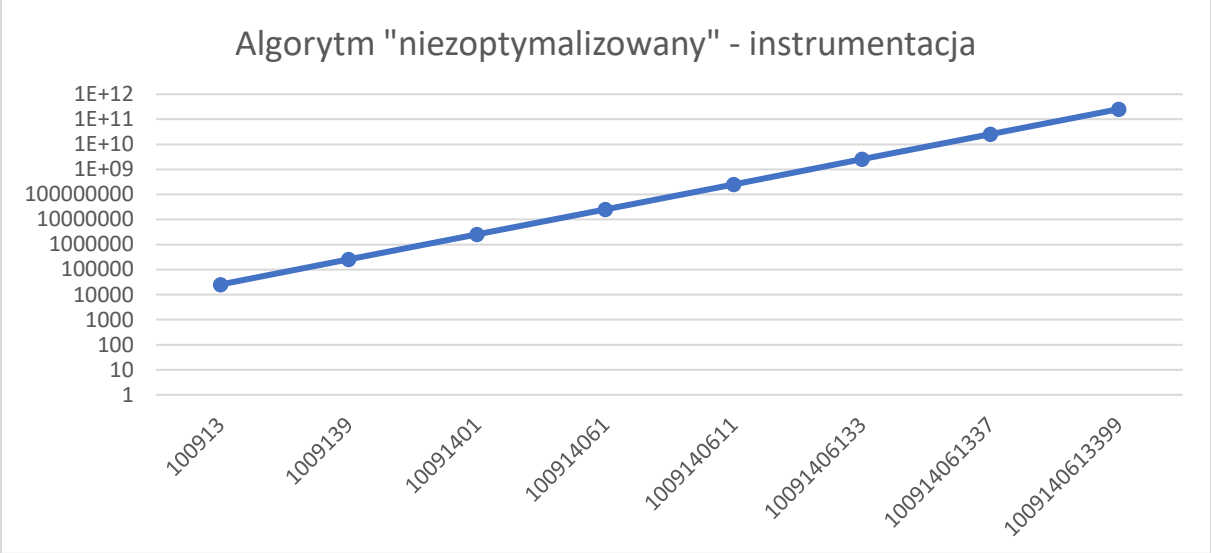
Badanie przeprowadzono do poniższego zbioru punktów pomiarowych:

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

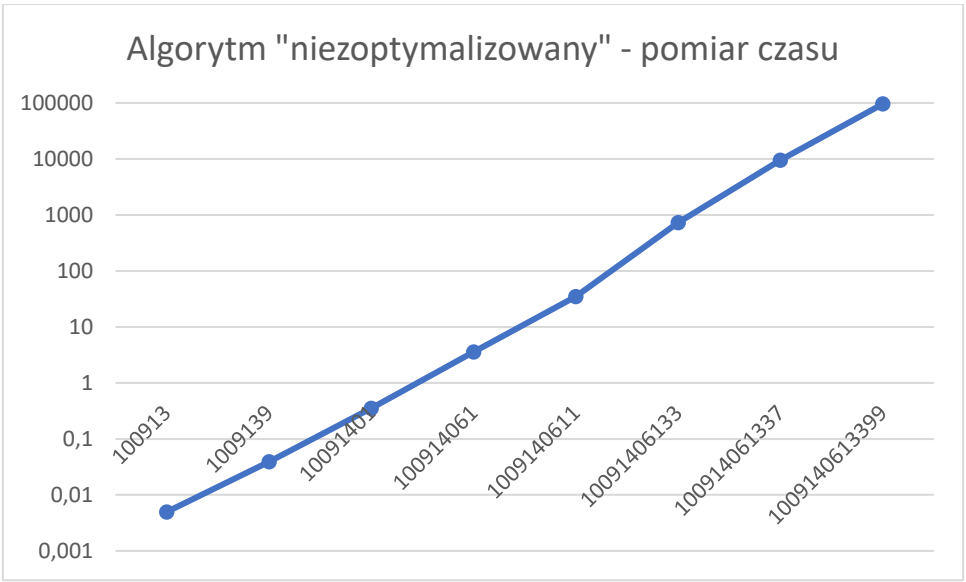
Algorytm „niezoptymalizowany”

Algorytm ten jako warunek pętli ma przeszukiwanie nieparzystych dzielników.

Instrumentacja



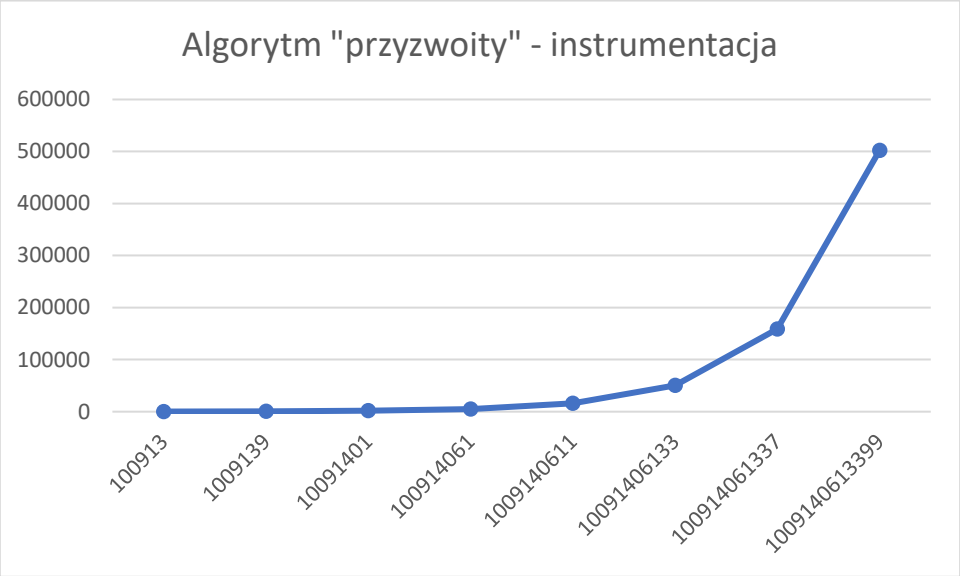
Pomiar czasu



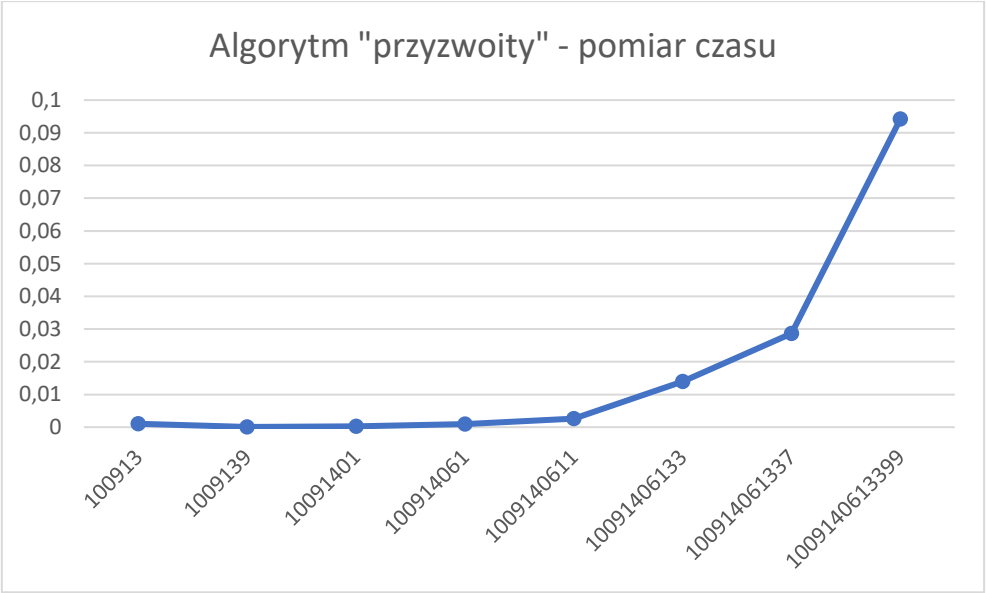
Algorytm „przyzwoity”

Algorytm ten wykorzystuje twierdzenie „każda liczba złożona n ma czynnik p spełniający nierówność $p \leq \sqrt{n}$.

Instrumentacja



Pomiar czasu



Wnioski

Wykonanie projektu udowodniło, że wystarczyła niewielka zmiana w kodzie pętli, żeby otrzymać ogromny zysk wydajności. Pierwsza wersja algorytmu ma liniową złożoność obliczeniową $O(n)$, gdzie bardzo duża ilość operacji (dzielniki do sprawdzenia) działa mocno niekorzystnie na efektywność algorytmu. Druga wersja ma pierwiastkową złożoność obliczeniową $O(\sqrt{n})$, która znacząco redukuje zbiór dzielników, optymalizując działanie programu. O ile dojście do tego rozwiązania wymagało drobnej analizy zagadnienia liczb pierwszych, to szybkość działania stanowczo rekompensuje tą wadę.

Projekt ten ponownie uświadomił, że jednym z najważniejszych kroków w kierunku optymalizacji algorytmu jest możliwie największe zmniejszenie zbioru danych do przetworzenia.