Paweł Pawlaczyk nr. Albumu 102876

Projekt 2

W tym projekcie zajmę się badaniem dwóch algorytmów, których zadaniem jest sprawdzenie czy podana liczba jest liczbą pierwszą. Dla każdego z tych algorytmów będę badał liczbę operacji oraz czas potrzebny na ich zrealizowanie.

Badane liczby to: 100913, 1009139, 10091401, 100914061, 1009140611, 10091406133, 100914061337, 1009140613399.

Pomiary zostały wykonane na jednostce operującej na 3GB pamięci podręcznej, oraz na procesorze: Intel Xeon X5450 3.00GHz.

Różnice w algorytmach

```
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;
}
```

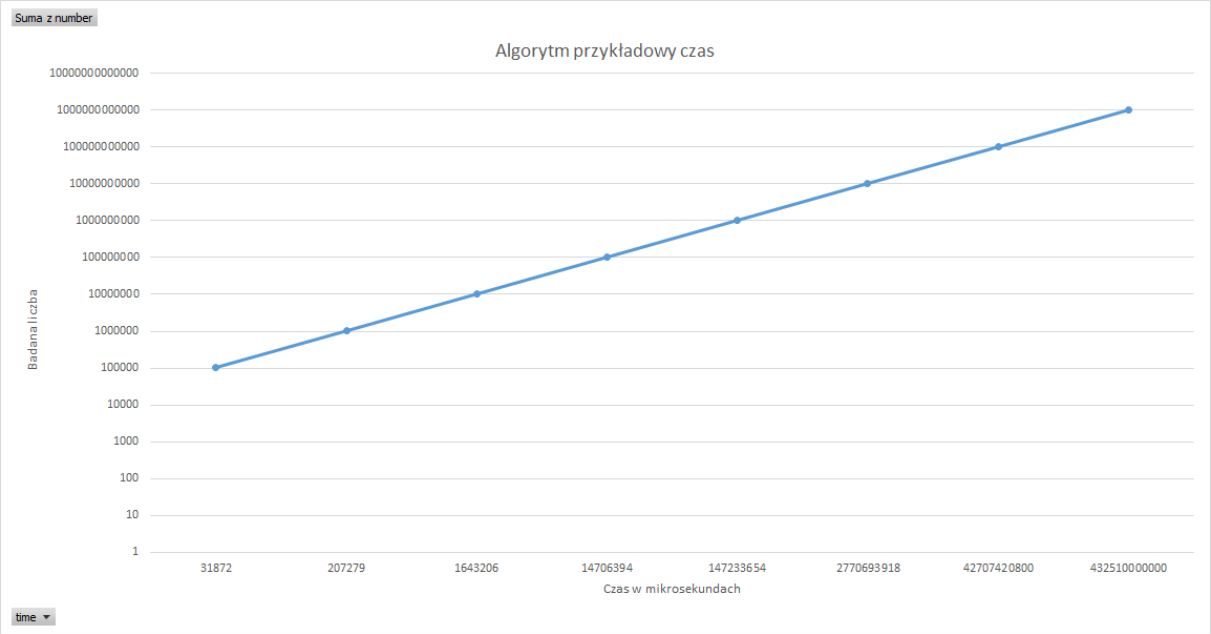
Algorytm przykładowy

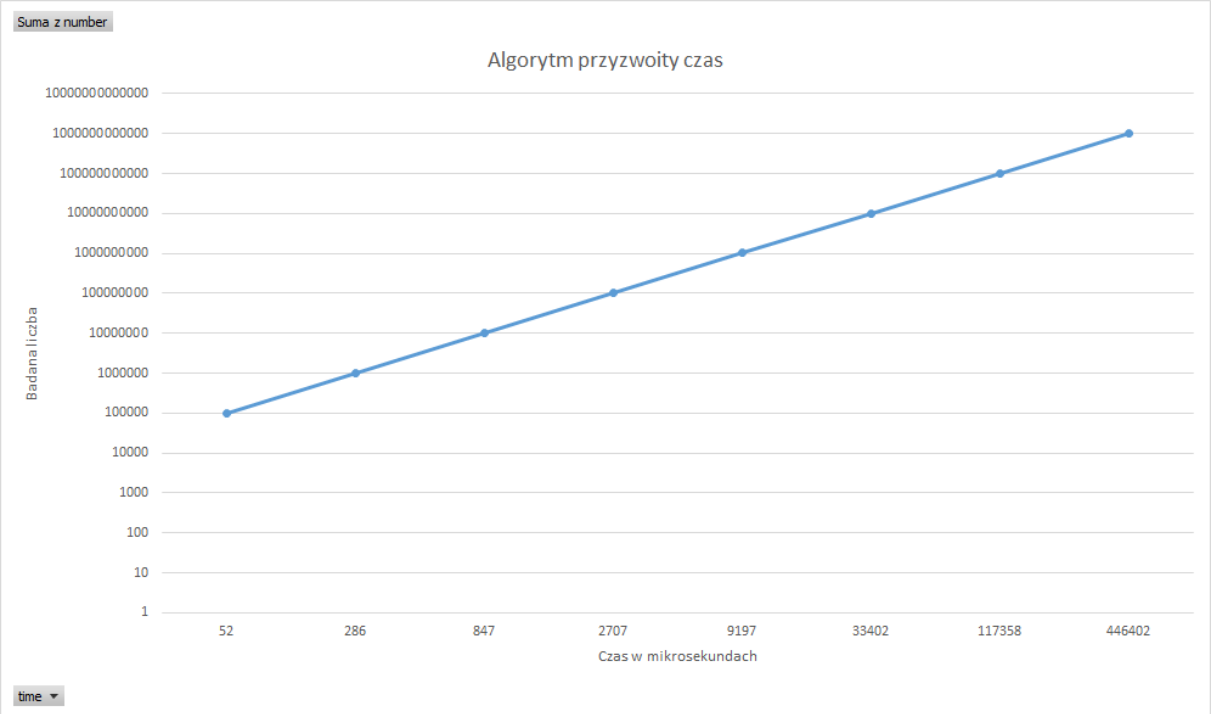
```
static bool IsPrimeBetter(BigInteger Num) {
    if (Num < 2) return false;
    else if (Num < 4) return true;
    else if (Num % 2 == 0) return false;
    else for (BigInteger u = 3; u * u <= Num; u += 2)
            if (Num % u == 0) return false;
    return true;
}
```

Algorytm przyzwoity

Różnica pomiędzy tymi dwoma algorytmami tkwi w deklaracji zakresu dla pętli for. Wraz z iteracją wartości ***u*** badany zakres będzie się zmniejszał w przypadku algorytmu przyzwoitego, natomiast pozostanie stały w przypadku algorytmu przykładowego. Takie działanie powinno znacząco zmniejszyć potrzebą liczbę operacji, jak i czas potrzebny na jego wykonanie.

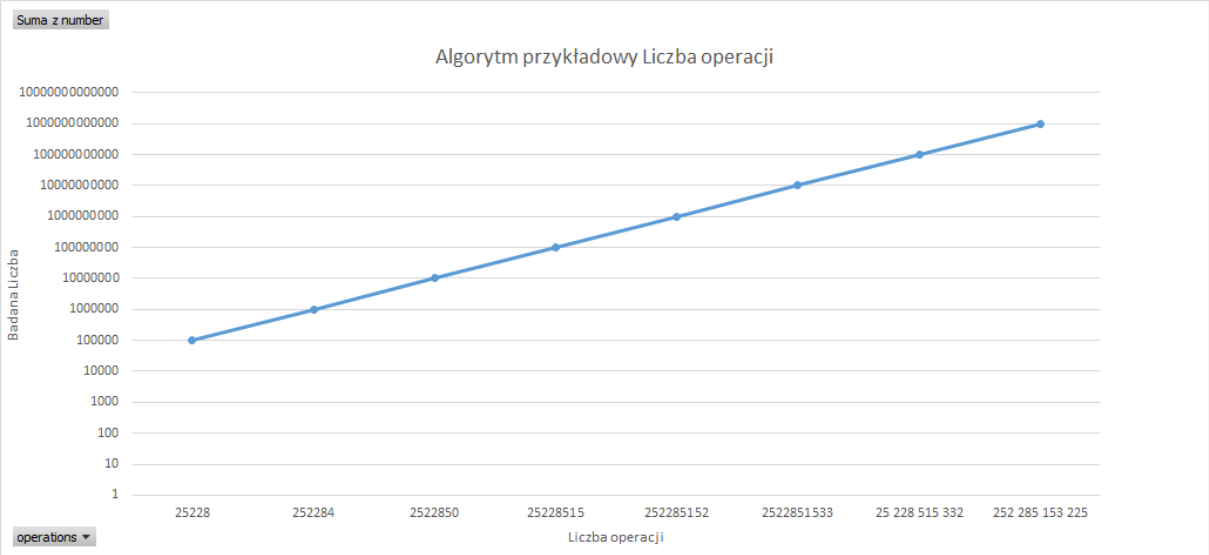
Czas potrzeby na wykonanie

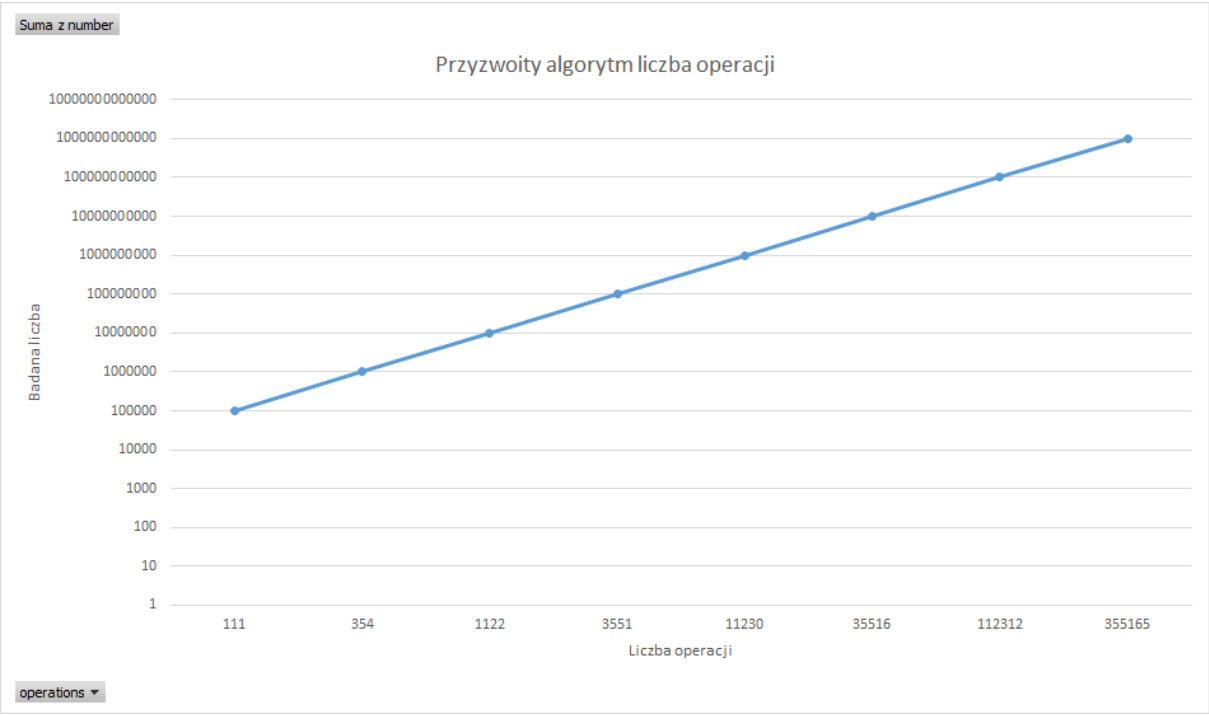




Dla badanych liczb zastosowałem skale logarytmiczną podczas tworzenia wykresu, ponieważ w ten sposób można dokładnie dojrzeć, że obydwa algorytmy przedstawiają funkcję liniową. Jednakże wystarczy spojrzeć na oś X gdzie widnieje czas potrzebny na sprawdzenie czy dana liczba jest liczbą pierwszą i od razu rzuca się w oczy przewaga algorytmu przyzwoitego nad algorytmem przykładowym.

Ilość operacji





W przypadku liczby operacji również zastosowałem skale logarytmiczną dla badanych liczb, dzięki czemu doskonale widać, że zarówno dla potrzebnego czasu jak i liczby operacji obydwa algorytmy przyjmują funkcję liniową. Spoglądając na liczbę operacji możemy zauważyć znaczącą przewagę algorytmu przyzwoitego nad algorytmem przykładowym.

Wyniki przedstawione na tabeli

Badana liczba	Przykładowy czas	Przywoity czas	Przykładowy l. ope	Przywoity l. ope
100913	31872	52	25228	111
1009139	207279	286	252284	354
10091401	1643206	847	2522850	1122
100914061	14706394	2707	25228515	3551
1009140611	147233654	9197	252285152	11230
10091406133	2770693918	33402	2522851533	35516
100914061337	42707420800	117358	25 228 515 332	112312
1009140613399	432510000000	446402	252 285 153 225	355165

Powyższe wykresy doskonale przedstawiły fakt, że oba algorytmy są liniowe. Jednakże moim zdaniem powyższa tabelka najlepiej oddaje przewagę algorytmu przyzwoitego nad algorytmem przykładowym.

Dodatkowo chciałbym odwołać się do tabelki, która widnieje w opisie projektu. W przypadku algorytmu przyzwoitego ilość potrzebnych operacji nie pokrywa się z tą, którą widnieje pdfie z zarysem zadania. Uważam, że nie jest to błąd, ponieważ, moje wartości są mniejsze, dlatego dochodzę do wniosku, że algorytm, który zastosowałem jest bardziej „przyzwoity” niż algorytm, który zastosował autor zadania. Niestety nie mam dostępu do kodu autora, dlatego nie mogę zdiagnozować, czy faktycznie moja interpretacja algorytmu przyzwoitego jest błędna czy lepsza.

Podsumowanie

Ten projekt doskonale ukazał jak odpowiedni algorytm może zmienić świat na lepsze. Przykładowy algorytm sprawdzał czas potrzebny na sprawdzenie czy największa z badanych liczb jest liczbą pierwszą przez ponad dwa dni, natomiast algorytm przyzwoity zdążył sprawdzić wszystkie liczby z zakresu w takim tempie, że zdążyłem go uruchomić trzykrotnie myśląc, że jest jakiś błąd przy kompilacji programu.