Mikołaj Mrożewski
145331

Wyznaczanie liczb pierwszych OpenMP

Data	Status Projektu	Uwagi
26.04.2022	Wybór tematu	
6.05.2022	Implementacja algorytmu sekwencyjnego	
7.05.2022	Implementacja algorytmu równoległego	
7.05.2022	Wykonanie pomiarów czasu	
13-25.05.2022	Analiza wyników	

Streszczenie

Liczby pierwsze od dawna są obiektem zainteresowania matematyków oraz informatyków. Algorytm sita Eratostenesa jest jednym z najbardziej znanych algorytmów do ich wyznaczania. Za pomocą OpenMP można go zrównoleglić w celu przyspieszenia przetwarzania.

Opis problemu

Problem polega na wyznaczeniu liczb pierwszych w zakresie <2; n>, gdzie n jest podane przez użytkownika, zapisanie ich do pliku tekstowego, oraz ich zliczenie. Zadanie wykonałem w oparciu o algorytm sita Eratostenesa.

Algorytm sekwencyjny

Dane: zakres do 100 000, 500 000, 1 000 000, 5 000 000, 10 000 000, 50 000 000 oraz 100 000 000

Wyniki:

- dla 100 000: 9 592 liczby zapisane w pliku
- dla 500 000: 41 538 liczb zapisanych w pliku
- dla 1 000 000: 78 498 liczb zapisanych w pliku
- dla 5 000 000: 348 513 liczb zapisanych w pliku
- dla 10 000 000: 664 579 liczb zapisanych w pliku
- dla 50 000 000: 3 001 134 liczby zapisane w pliku
- dla 100 000 000: 5 761 455 liczb zapisanych w pliku

Metoda:

Algorytm zaczyna się od stworzenia tablicy wypełnionej jedynkami. Tablica ma długość równą zakresowi poszukiwania, który jest podany przez użytkownika podczas uruchamiania programu. Każdy indeks tablicy jest kandydatem do bycia liczbą pierwszą. Zaczynając od drugiego indeksu iteruję po tablicy, sprawdzam czy na danym indeksie jest wartość 1. Jeśli tak jest, to indeks zostaje zapisany do pliku

tekstowego, a każde pole w tablicy o indeksie będącym wielokrotnością tego indeksu zamieniane jest na wartość 0. Wówczas zwiększana jest też wartość licznika zliczającego liczby pierwsze.

Złożoność obliczeniowa: O(n * n log(n))
Złożoność pamięciowa: O(n)

Kody

część kodu odpowiedzialna za główną funkcjonalność sita Eratostenesa:

```
for (int i = 2; i < range; i++){
    if (numbers[i] == 1){
        fprintf(file, "%d\n", i);
        prime_counter += 1;
        int j = 0;
        int temp_val = i;
        for(int temp_val = i; temp_val < range; temp_val += i){
            numbers[temp_val] = 0;
        }
    }
}
```

Link do repozytorium:

https://github.com/mmrozewsk/Przetwarzanie_Rownolegle

Opcje kompilacji:

gcc projekt_par.c -o projekt_par -fopenmp
gcc projekt_seq.c -o projekt_seq -fopenmp

Uruchomienie programu:

Do uruchomienia programu wielowątkowego niezbędne jest podanie dwóch parametrów - liczby wątków oraz zakresu przeszukiwania. Poniżej uruchomienie programu szukania liczb pierwszych w zakresie 2 - 1000000 z wykorzystaniem 4 wątków.
./projekt_par 4 1000000

Program sekwencyjny uruchamiany jest z tylko jednym parametrem - zakresem przeszukiwania. Poniżej uruchomienie programu szukania liczb pierwszych w zakresie 2 - 1000000
./projekt_seq 1000000

Testy programów i profilowanie aplikacji

Podzespoły komputera:

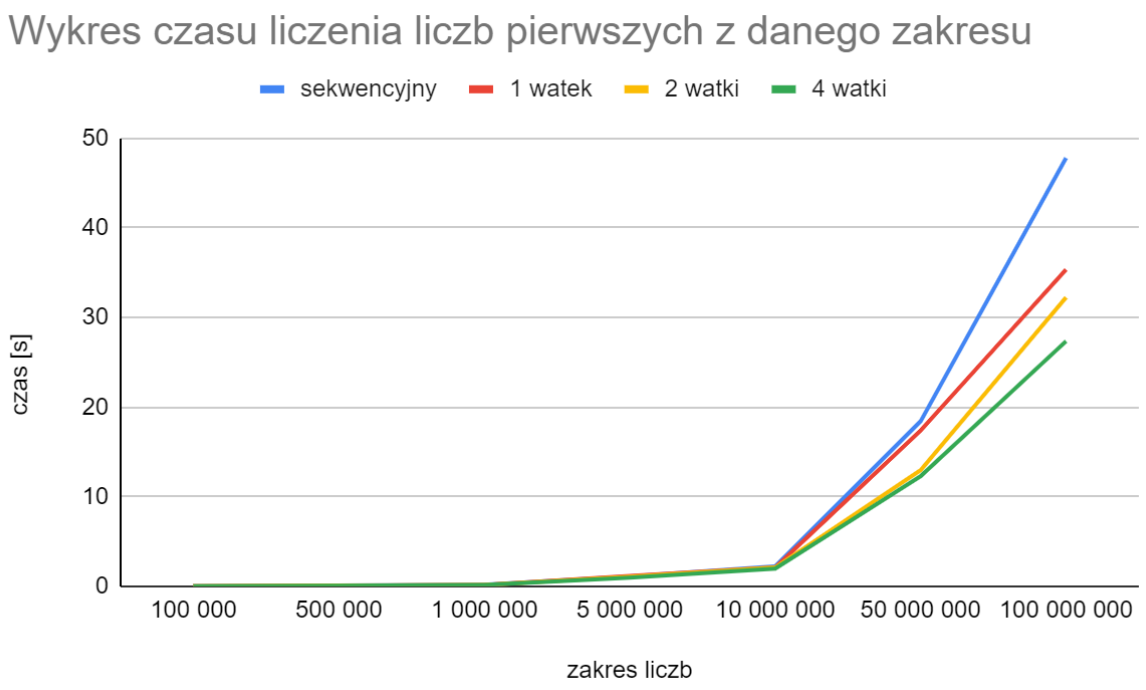
- architektura 64 bitowa
- procesor Intel Core i7-7700HQ CPU @ 2.80GHz
- pamięć RAM 16GB
- program był uruchamiany na maszynie wirtualnej Fedora 31, która miała dostęp do 8GB RAMu

Sprawdzenie zgodności wyników wersji sekwencyjnej i równoległej algorytmów:
wyniki obu algorytmów są identyczne

Pomiary czasu

Zakres	Sekwencyjny	1 wątek	2 wątki	4 wątki
100 000	0,017552	0,022824	0,017217	0,011634
500 000	0,107214	0,086595	0,088513	0,080423
1 000 000	0,179016	0,160924	0,17049	0,143571
5 000 000	1,114565	1,154324	1,076145	0,963832
10 000 000	2,233997	2,107901	2,107223	1,976312
50 000 000	18,423112	17,382199	12,95188	12,28543
100 000 000	47,785559	35,342499	32,236697	27,340734

Wizualizacja pomiaru czasu



Analiza narzutów czasowych i przyspieszenia obliczeń

Przyspieszenie wyliczane w stosunku do czasu wykonywania programu sekwencyjnego

Zakres	1 wątek	2 wątki	4 wątki
100 000	0,7690150719	1,019457513	1,50868145
500 000	1,23810843	1,211279699	1,333126096
1 000 000	1,112425741	1,050008798	1,246881334
5 000 000	0,9655564642	1,035701509	1,156389288
10 000 000	1,059820646	1,060161644	1,130386801
50 000 000	1,05988385	1,422427632	1,499590328
100 000 000	1,352070746	1,482334217	1,747778937

Analiza złożoności pamięciowej

Złożoność pamięciowa wynosi O(n) - korzystamy z n-elementowej listy kandydatów na liczbę pierwszą

Podsumowanie

Liczby pierwsze można zapisywać i zliczać w znacznie krótszym czasie używając do tego wielu wątków. Ku mojemu zaskoczeniu dodanie wielowątkowości do pętli *for* której zadaniem jest eliminacja kandydatów będącymi wielokrotnością innych liczb, przynosi efekt odwrotny do oczekiwanego - zwiększa to czas wykonywania programu. Wynika to z tego, że wielokrotnie inicjalizowane są procesy, a dla większych liczb pozostała część tablicy jest na tyle krótka, że inicjalizacja zajmuje znacznie więcej czasu, niż jest zaoszczędzane dzięki ich współpracy.

Źródła

https://pl.wikipedia.org/wiki/Prawo_Amdahla
https://pl.wikipedia.org/wiki/Sito_Eratostenesa