Projekt 1
Projektowanie i analiza algorytmów
Algorytmy sortujące

Autorzy: Mateusz Władyka 272513
Grupa: Wt 13:15-15:00
Kierunek: Informatyczne systemy automatyki
Prowadzący: dr inż. Łukasz Jeleń

27. marca 2024

1 Wstęp

Projekt polegał na porównaniu czasów sortowań dla wybranych algorytmów. Zdecydowano się na implementację sortowania szybkiego, poprzez scalanie oraz introspektywnego, używając języka C++.

Należało posortować dane pochodzące z *IMDb Largest Review Dataset*, znajdujące się w pliku CSV, przefiltrować, aby nie uwzględniać błędnych danych, a następnie posortować za kryterium biorąc ocenę filmu.

2 Filtracja i algorytmy

2.1 Filtracja

Strukturą danych, z której skorzystano, była tablica dynamiczna, ponieważ pozwala ona na szybki dostęp do dowolnych elementów, a po zaimportowaniu danych z pliku, nie trzeba modyfikować jej rozmiaru. Zaimplementowano mechanizm filtracji, dzięki któremu trwa ona całe **0 sekund**. Stworzono dynamiczną tablicę struktury *Video*, która miała domyślne ustawione wartości, które z całą pewnością nie pojawiłyby się w pliku; następnie do niej zaimportowano tylko te rekordy, które nadpisują każdą domyślną wartość (numeru, tytułu filmu i oceny) - wszystkie inne były niekompletne, przez co trzeba było je odrzucić, albo jak w tym przypadku: nawet nie wczytywać do tablicy.

Później z tablicy, która zawierała wszystkie rekordy, utworzono wycinki tablic 10000-, 100000-, 500000- i 1000000-elementowych. Jednakże, aby zabezpieczyć się przed przypadkiem, gdzie elementów w tablicy ze wszystkimi rekordami jest mniej niż deklarowana ich chciana liczba, napisano mechanizm sprawdzający tę wartość, a następnie pozwalający przyjąć tylko tyle, ile faktycznie może być posortowanych. Zadziałał on w przypadku, gdy chciano posortować tablicę 1000000-elementową - maksymalna liczba została zredukowana do 962893.

2.2 Sortowanie szybkie

Oparty na podobnej filozofii, co sortowanie przez scalanie, ale dzieli on tablice używając tzw. elementu osiowego. Wszystkie elementy mniejsze od tegoż elementu trafiają do jednej tablicy, a większe lub równe do drugiej. Proces powtarza się rekurencyjnie w rezultacie dając posortowaną tablicę.

Średnia złożoność czasowa wynosi: $\Theta(n \log n)$, a w przypadku najgorszym (kiedy element osiowy jest ciągle źle dobierany): $O(n^2)$.

2.3 Sortowanie przez scalanie

Algorytm, wykorzystujący metodę "dziel i rządź", polega na rekurencyjnym dzieleniu tablicy na dwie podtablice, dopóki nie będą one wielkości jednego elementu. Później zaś scala je porównując ze sobą, w ten sposób sortując, a następnie scalając je, aby uzyskać całą posortowaną tablicę.

Złożoność czasowa w każdym przypadku to: $O(n \log n)$.

2.4 Sortowanie introspektywne

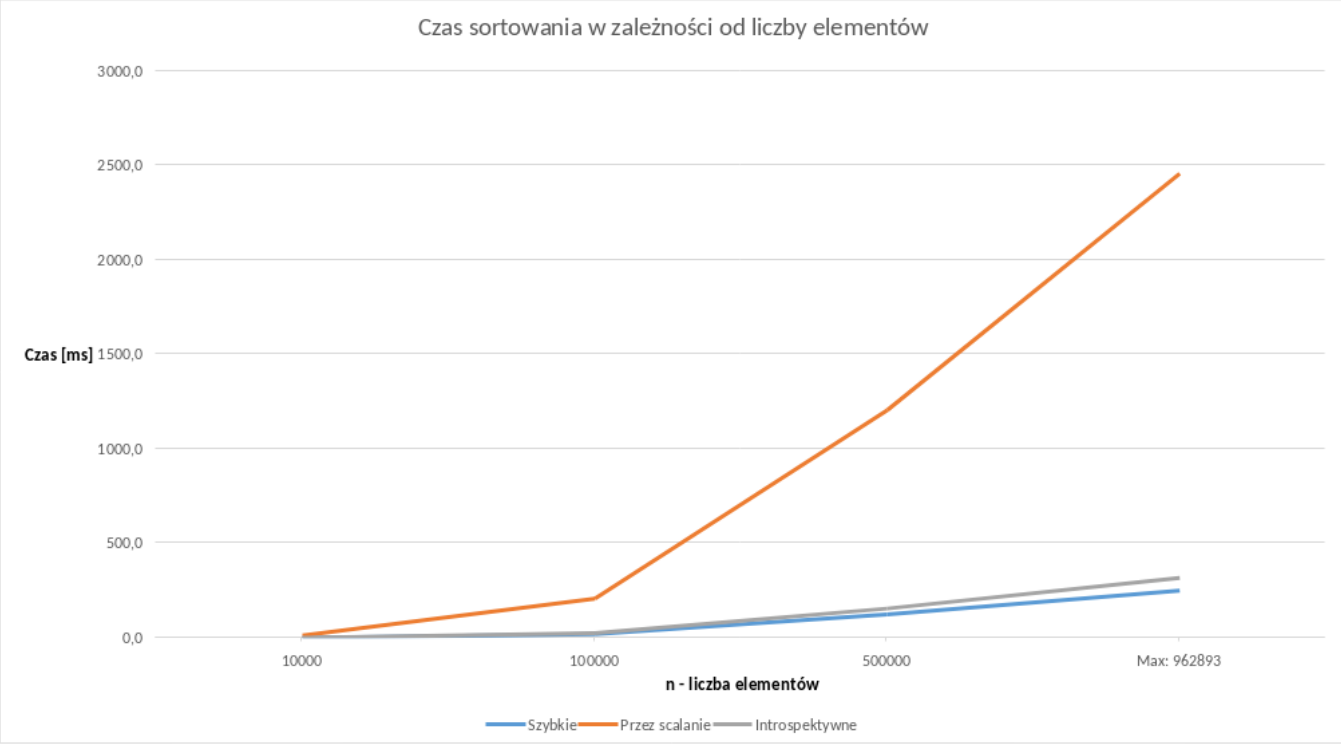
Jest to metoda hybrydowa, łącząca różne algorytmy sortujące, łącząc ich zalety, aby ominąć wady. W projekcie zaimplementowano połączenie sortowania szybkiego, przez kopcowanie oraz wstawianie. Korzysta z niej np. std::sort w C++. Złożoność czasowa dla średniego i najgorszego przypadku wynosi: $O(n \log n)$.

3 Wyniki

Wszystkie czasy zmierzono 3 razy, a do tabeli wstawiono średnią z tych pomiarów w milisekundach.

		Liczba elementów			
		10000	**100000**	**500000**	**Max: 962893**
Sortowanie [ms]	**Szybkie**	1,0	22,0	125,3	251,0
	Przez scalanie	14,3	208,6	1207,3	2459,6
	Introspektywne	1,0	26,0	157,0	319,0
Mediana		5	7		
Średnia		5,46030	6,08993	6,66572	6,63662

Za audyt poprawności filtrowania odpowiada funkcja, która iterując od końca, sprawdza czy ocena i-tego elementu posortowanej tablicy jest większa lub równa ocenie i-1-elementu.



Wykres czasu w funkcji liczby elementów

4 Wnioski

Sortowanie przez scalanie zachowało się zgodnie z oczekiwaniami, uzyskując wyniki tożsamy ze swoją złożonością obliczeniową. Jednakże sortowanie introspektywne i szybkie uzyskały najlepszą możliwą dla siebie wydajność, czyli $\Omega(n \log n)$. Aczkolwiek introsort powinien wypadać lepiej, lub tak samo w każdym przypadku, co quicksort, możliwe że kompilator wprowadził optymalizacje w kodzie maszynowym, albo zaimplementowana wersja introspektywnego sortowania nie jest najwydajniejsza.

5 Bibliografia

1. GeeksforGeeks: platforma naukowa[online]. Indie. GeeksforGeeks 2008.[dostęp 27.03.2024]. Dostęp w internecie: `https://www.geeksforgeeks.org/`