Projektowanie i Analiza Algorytmów

Sprawozdanie

Projekt 2 – Algorytmy sortowania

Jakub Piekarek

Indeks 264202

Prowadzący dr inż. Krzysztof Halawa

Kod grupy K00-37d

Poniedziałek 11^{15} – 13^{00}

Wydział Informatyki i Telekomunikacji

1. Opis zadania

Na podstawie polecenia z instrukcji, wybrano trzy algorytmy sortowania z czterech dostępnych na ocenę 5.0 (wyszczególnione wybrano):
- ***przez scalanie***
- ***quicksort***
- ***introspektywne***
- kubełkowe.

Następnie przeprowadzone zostały testy efektywności dla wyżej wymienionych algorytmów. Aby przeprowadzić dany test należało wczytać plik z danymi o rozszerzeniem .csv, zawierał nazwy i oceny filmów. Musiała zostać użyta również filtracja ocen i odrzucenie tych rekordów, gdzie ocen nie było. Ważną rzeczą przy używaniu filtracji było mierzenie czasu jej przeszukiwania. Wariantów przeszukiwanych rekordów było pięć, każda miała n długości, gdzie n = { 10000, 100000, 500000, 1000000, 1010293 }. Dla każdego wariantu należało policzyć czas sortowania każdego typu, medianę i średnią ocen.

2. Opis użytych algorytmów sortowania

2.1 Sortowanie szybkie - Quicksort

Algorytm sortowania szybkiego, podobnie jak mergesort, wykorzystuje metodę "dziel i zwyciężaj" do sortowania danych. W tym algorytmie używa się tzw. elementu osiowego, którym zazwyczaj jest środkowy element zestawu danych. Element osiowy jest kluczowy w procesie sortowania - elementy mniejsze niż osiowy są przenoszone na lewą stronę, a większe na prawą. Złożoność obliczeniowa quicksorta wynosi O(n log n) w przypadkach najlepszym i średnim, ale w najgorszym przypadku, gdy zawsze wybierany jest najmniejszy lub największy element, wynosi O(n^2). Algorytm quicksort sortuje "w miejscu", co oznacza, że nie wymaga dodatkowej, tymczasowej struktury danych, a jego złożoność pamięciowa wynosi O(n) tylko w najgorszym przypadku, którą można zredukować do O(log n).

2.2 Sortowanie przez scalanie - Mergesort

Algorytm mergesort jest oparty na metodzie "dziel i zwyciężaj", która polega na podziale problemu sortowania na mniejsze pod problemy. Aby to zrobić, zestaw danych jest dzielony na dwie równe części, a proces dzielenia jest powtarzany, aż do uzyskania jednoelementowych pod problemów. Następnie, elementy są scalane w porządku nierosnącym, aż do uzyskania w pełni posortowanego zestawu danych. Sortowanie przez scalanie ma złożoność pamięciową O(n) ze względu na potrzebę tworzenia tymczasowej struktury danych, ale złożoność obliczeniowa wynosi O(n log n) we wszystkich przypadkach (najlepszym, średnim i najgorszym).

2.3 Introspektywne - Introsort

Algorytm sortowania introspektywnego jest hybrydowym algorytmem sortowania, który składa się z trzech różnych algorytmów sortowania: szybkiego, przez kopcowanie i przez wstawianie. Przełączanie pomiędzy algorytmami pozwala na uniknięcie problemów złożoności obliczeniowej występujących w sortowaniu szybkim. Algorytm sortowania introspektywnego wykorzystuje element osiowy do podziału zestawu danych i w zależności od głębokości rekurencji, wybiera jeden z trzech algorytmów sortowania. Złożoność obliczeniowa algorytmu w każdym przypadku wynosi O(n log n). Algorytm sortowania introspektywnego nie wymaga dodatkowej, tymczasowej struktury danych podczas procesu sortowania, co skutkuje złożonością pamięciową O(log n). Obecnie standardowa biblioteka języka C++ wykorzystuje algorytm sortowania introspektywnego w funkcji std::sort().

3. Złożoność obliczeniowa algorytmów

Porównanie złożoności pamięciowej algorytmów

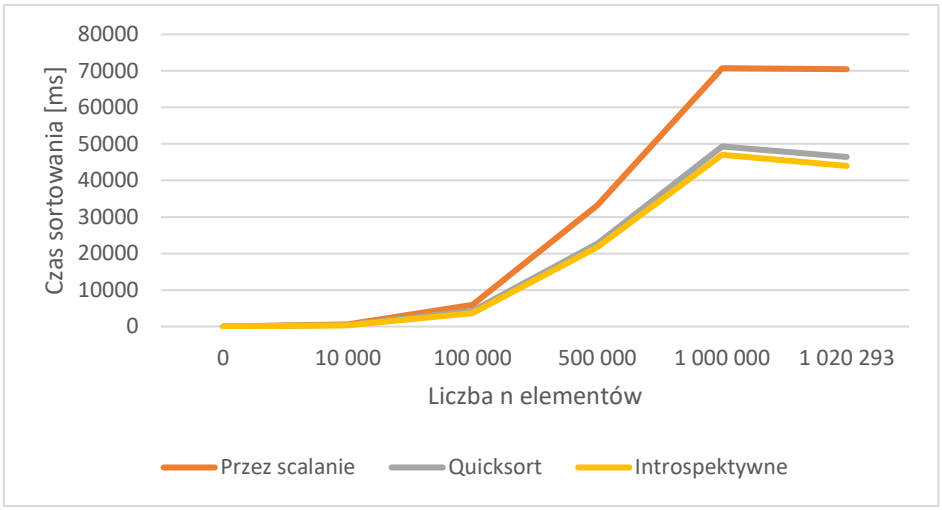
Przez scalanie	Quicksort	Introspektywne
O(n)	O(log n)	O(log n)

Porównanie złożoności obliczeniowej algorytmów

przypadek	Przez scalanie	Quicksort	Introspektywne
najlepszy	O(n log n)	O(n log n)	O(n log n)
średni	O(n log n)	O(n log n)	O(n log n)
najgorszy	O(n log n)	$O(n^2)$	O(n log n)

4. Wyniki testów dla każdego z wariantu

n	Filtrowanie		Sortowanie			
			Co było mierzone	Przez scalanie	Quicksort	Introspektywne
10000	Brak ocen	0	Średnia	5.4603	5.4603	5.4603
	Pozostałych	10 000	Mediana	5	5	5
	Czas filtracji	0 ms	Czas	580.393 ms	435.780 ms	343.830 ms
100000	Brak ocen	4190	Średnia	6.06437	6.06437	6.06437
	Pozostałych	95810	Mediana	6	6	6
	Czas filtracji	86.0834 ms	Czas	5928.71 ms	4038.5 ms	3746.46 ms
500000	Brak ocen	23335	Średnia	6.6555	6.6555	6.6555
	Pozostałych	476665	Mediana	7	7	7
	Czas filtracji	416.454 ms	Czas	33661.8 ms	22932.5 ms	22047.3 ms
1 000000	Brak ocen	43802	Średnia	6.63431	6.63431	6.63431
	Pozostałych	956198	Mediana	7	7	7
	Czas filtracji	842.128 ms	Czas	71261.3 ms	49738.8 ms	47550.2 ms
1020 293	Brak ocen	47459	Średnia	6.6366	6.6366	6.6366
	Pozostałych	962834	Mediana	7	7	7
	Czas filtracji	847.499 ms	Czas	72133.2 ms	47280.5 ms	45194.1 ms



5. Wnioski

   Pięć testów efektywności algorytmów sortowania na różnych zbiorach danych z ilością elementów n ∈ {10 000, 100 000, 500 000, 1 000 000, 1 010 293} zakończyło się sukcesem. Na podstawie wyników czasowych można stwierdzić, że algorytm sortowania introspektywnego zawsze był nieco szybszy od quicksort, a najwolniejszym z testowanych algorytmów był algorytm sortowania przez scalanie. Dla każdego sortowania mediana oraz średnia była jednakowa co potwierdza nam poprawne działanie.