Sortowanie z użyciem wielkich buforów

Marcel Grużewski 193589

24 listopada 2024

1 Wstęp

1.1 Cel projektu

Celem projektu jest zaimplementowanie jednej z metod do sortowania zewnętrznego i przeprowadzenie symulacji odzwierdciedlającej działanie wybranej metody. Do wyboru była trzy metody sortowania zewnętrznego: **scalania z użyciem wielkich buforów**, **scalania naturalnego** oraz **sortowania polifazowego**. Program ma symulować działanie tych algorytmów w warunkach, w których dostępna pamięć RAM jest ograniczona. Dzięki tej symulacji możliwe będzie przeprowadzenie eksperymentów, które odwzorują rzeczywiste scenariusze pracy z dużymi zbiorami danych, które nie mieszczą się w całości w pamięci operacyjnej. Symulacja ma na celu zbadanie, jak algorytm sortowania zewnętrznego zarządza danymi na dysku i jak ograniczona pamięć wpływa na jego wydajność oraz liczbę operacji I/O.

W ramach projektu porównane zostaną teoretyczne i praktyczne wyniki dotyczące liczby faz sortowania oraz operacji dyskowych dla różnych rozmiarów danych, a wyniki zostaną przeanalizowane i przedstawione na wykresach.

1.2 Wybrany algorytm

Wybrany algorytm to sortowanie z wykorzystaniem dużych buforów.

Metoda ta polega na podzieleniu dużego pliku na mniejsze części, zwane runami, które są sortowane w pamięci przy użyciu ograniczonej liczby buforów. Następnie posortowane runy są łączone w większe fragmenty, aż do uzyskania jednego, w pełni posortowanego pliku wynikowego.

Algorytm można opisać za pomocą trzech zmiennych:

N – liczba rekordów w pliku,

b – liczba rekordów mieszczących się w jednym buforze,

n – liczba buforów dostępnych w pamięci.

Całkowity koszt operacji dyskowych w algorytmie sortowania z wykorzystaniem wielkich buforów można oszacować za pomocą wzoru:

$$2\frac{N}{b} + 2\frac{N}{b}\log_n\left(\frac{N}{b}\right)$$

Gdzie:

- pierwszy czynnik $\frac{2N}{b}$ odpowiada za koszt odczytu i zapisu wszystkich rekordów podczas pierwszej fazy algorytmu,

- $2\frac{N}{b}\log_n\left(\frac{N}{b}\right)$ określa liczbę operacji I/O w drugiej fazie algorytmu, czyli koszt jednego cyklu - $\frac{2N}{b}$ pomnożonego razy liczbę cykli potrzebnych do całkowitego scalania wszystkich runów w jeden posortowany plik.

Natomiast liczba cykli potrzebnych w drugiej fazie scalania, aby uzyskać w pełni posortowany plik, może być oszacowana za pomocą wzoru:

$$\left\lceil \log_n\left(\frac{N}{b}\right)\right\rceil - 1$$

1.3 Wylosowany rekord

W zadaniu wylosowany został typ rekordu oraz klucz, według którego wykonane będzie sortowanie:

Rekordy pliku: Student i jego wyniki z 3 kolejnych kolokwiów z pewnego przedmiotu. Uporządkowanie wg oceny średniej.

Rekord zawiera zatem 3 liczby ze zbioru **[2, 3, 3.5, 4, 4.5, 5]**

2 Implementacja algorytmu

2.1 Opis najważniejszych struktur zaimplementowanych w programie

- DataManager - udostępnia różne funckje potrzebne do zarządzania plikami, m.in. generowanie danych, wyświetlanie plików oraz dostarczanie informacji o runach,
- Record - przechowuje rekord o stałym rozmiarze, udostępnia funkcje potrzebne do wyswietlenia go i zarzadzania nim,
- Buffer - odpowiada za przechowywanie rekordow, zawiera rownież funkcje zapisu do pliku,
- Run - jest wykorzystywany tylko za pomocą pośredniczących bufferów, jest posortowaną listą rekordów,
- RAM - zawiera w sobie mechanizm sortowanai z wykorzystaniem wielich bufferów,

2.2 Format pliku

Dla wygody użytkownika oraz łatwej możliwości wyświetlenia pliku, wybrany format to .txt. zarówno dane jak i wyniki przechowywane są w plikach o formacie .txt.

2.3 Opis działania programu

- Najpierw użytkownik w argumentach programu podaje w jaki sposób chce dostarczyć dane ("generate", "file"lub "keyboard"), gdy użytkownik nie poda żadnego argumentu program wygeneruje rekordy zgodnie z ich zdefiniowaną liczbą. Przy podstawowym działaniu programu dane przechowywane są w pliku "data.txt",
- Użytkownik może także za pomoca argumentów programu zdefiniować sposób prezentowania runów po każdej fazie algorytmu. "short prezentuje runy w wersji skróconej, "full prezentuje runy w całości, brak argumentu - nie prezentuje runów.

- Nastepnie odbywa się Faza 1 algorytmu - tworzenie posortowanych runów, polegająca na odczytywaniu danych za pośrednictwem buforów z pliku z danymi, sortowanie ich, aby zapisać je do plików "runXX.txt", gdzie XX oznacza numer runa,
- Na koniec odbywa się Faza 2 algorytmu - faza scalania runów. W cyklach mergowania pobierane jest n-1 runów, które za pomocą n-tego bufora są ze sobą scalane. Procedura ta powtarza się aż do osiągnięcia jednego posortowanego runa ze wszystkimi rekordami.

3 Eksperyment

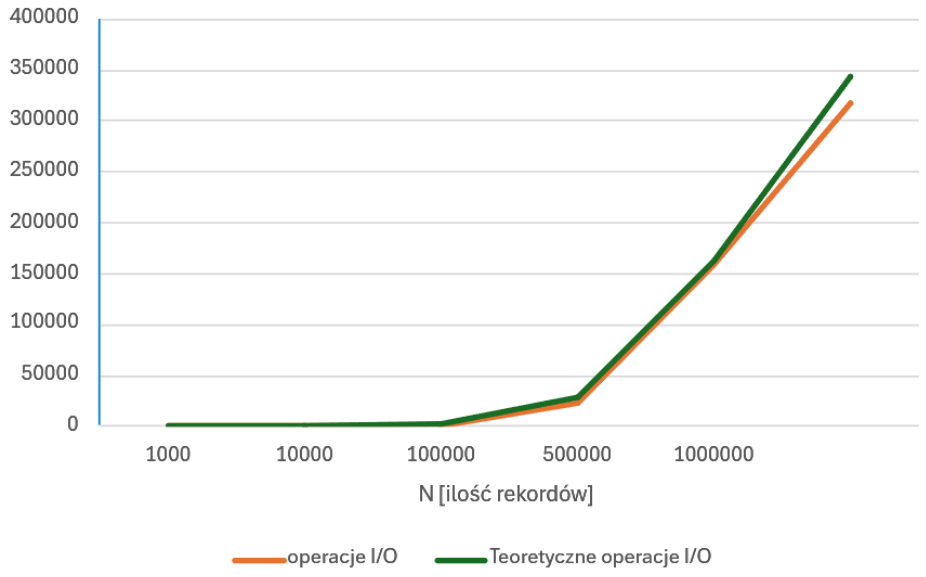
3.1 Opis eksperymentu

Eksperyment ma na celu sprawdzić jak wpływa na liczbe operacji I/O zmiana poszczególnych parametrów oraz porównać jak różnią się teoretyczne liczby operacji I/O od tych praktycznych. Eksperyment będzie polegał na testach ze znacznie różniącą się liczbą rekordów i analizie różnicy między teoretyczną i praktyczną liczbą operacji wejścia i wyjścia na wykresach.

3.2 Przebieg eksperymentu

N	b	n	Operacje I/O	Teoretyczne operacje I/O	Cykle	Teoretyczne cykle
1000	25	25	158	172	1	1
10000	25	25	1584	2289	1	1
100000	25	25	23833	28614	2	2
500000	25	25	159164	163068	3	3
1000000	25	25	318330	343362	3	3

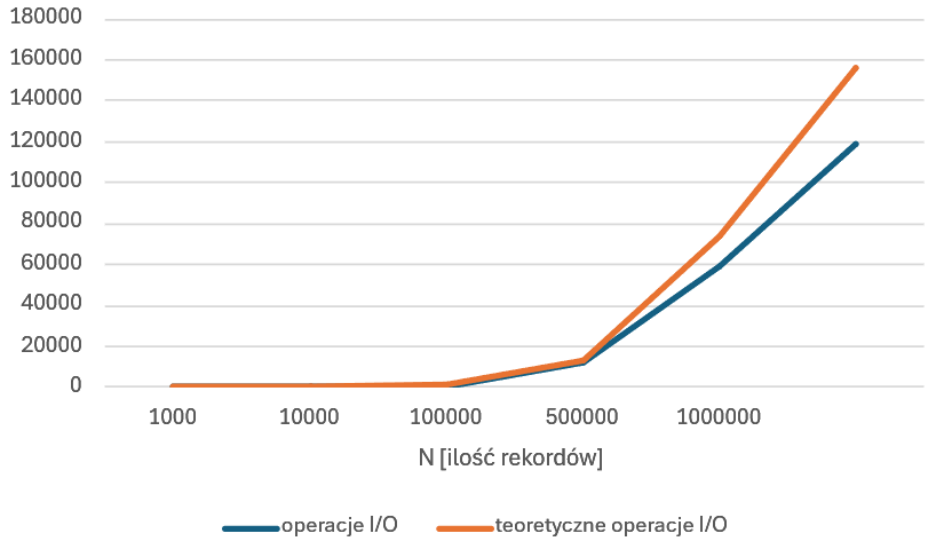
Tabela 1: Porównanie rzeczywistych i teoretycznych wyników operacji I/O oraz cykli dla różnych wartości N.



Rysunek 1: Porównanie rzeczywistych i teoretycznych operacji I/O dla n = 25 oraz b = 25.

N	b	n	Operacje I/O	Teoretyczne operacje I/O	Cykle	Teoretyczne cykle
1000	50	30	40	75	0	0
10000	50	30	793	1023	1	1
100000	50	30	11930	12939	2	2
500000	50	30	59654	74159	2	2
1000000	50	30	119310	156471	2	2

Tabela 2: Porównanie rzeczywistych i teoretycznych operacji I/O oraz cykli dla różnych wartości N, b i n.



Rysunek 2: Porównanie rzeczywistych i teoretycznych operacji I/O dla n = 30 oraz b = 50.

4 Wnioski

Eksperyment potwierdził, że teoretyczne oszacowania liczby operacji I/O i cykli są zbliżone do wyników rzeczywistych, co widać na obu wykresach, dla różnych wartości n i b. Taka zgodność wskazuje, że wzory teoretyczne dobrze opisują działanie algorytmu, również w praktycznej implementacji.

Rozbieżności między wynikami wynikają z mozliwości optymalizacji algorytmu, co obniża liczbę rzeczywistych operacji I/O w porównaniu do teoretycznych przewidywań. Ponadto, wzory teoretyczne upraszczają niektóre obliczenia, stosując zaokrąglenia lub zakładając idealne warunki, które w rzeczywistości mogą nie występować.