Projekt ze „Struktur baz danych”

Sortowanie plików sekwencyjnych

Mateusz Stencel
188676

Wstęp

Celem projektu było zaimplementowanie metody sortowania plików sekwencyjnych. Do realizacji zadania użyto algorytmu scalania naturalnego w schemacie 2+1. Plik sekwencyjny jest plikiem, w którym rekordy są mają ustaloną strukturę a dane są uporządkowane zgodnie z wartością klucza. Sortowanie pliku polega na sortowaniu pliku seriami czyli ciągiem elementów ułożonych w zadanym porządku. W przypadku sortowania w wariancie 2+1, kolejne serie są dystrybuowane naprzemiennie na 2 taśmy. Następnie serie są scalane, tworząc dłuższe serie. Algorytm powtarza się tak długo, aż otrzyma jedną serię długą jak cały plik. W takim przypadku zawartość pliku została posortowana. Implementacje algorytmu wykonano w języku C++.

Dane

Plik testowy wyrażony jest w formie binarnej. Ma to na celu łatwiejsze przetwarzanie danych w pliku. Kolejne rekordy zapisywane są bezpośrednio po sobie. Rekordem pliku są trapezy – długości obu podstaw trapezu i jego wysokość. Rekordy są porządkowane niemalejąco wg. pola trapezu. Każde pole rekordu wynosi 4 bajty, co daje łącznie 12 bajtów na jeden rekord.

Wejście i wyjście programu

Program przyjmuje komendy z wiersza poleceń. W zależności od wpisanej opcji, program będzie wyświetlał dodatkowe informacje dotyczące działania algorytmu sortowania. Zakończenie działania algorytmu spowoduje wyświetlenie informacji o liczbie faz sortowania, zapisów i odczytów na dysk. W programie zdefiniowano następujące opcje:

- `-h` : Powoduje wyświetlenie pomocy i zakończenie programu.
- `-f` *`nazwa pliku`* : Powoduje wykonanie sortowania na pliku testowym o podanej nazwie.
- `-s` : Wyświetlenie rekordów w pliku na początku i na końcu działania programu. Rekordy są przedstawiane jako pola trapezów.
- `-r` : Wyświetlenie rekordów w pliku za każdym przebiegiem pętli. Rekordy są przedstawiane jako pola trapezów.
- `-o` : Wygenerowanie pliku testowego o podanych rekordach i wykonanie sortowania.

Wyniki eksperymentu

W celu sprawdzenia działania algorytmu, przeprowadzono testy na wygenerowanych plikach o znacznie różniących się wielkościach z losową zawartością aby określić liczbę faz sortowania oraz sprawdzić zależność pomiędzy liczbą rekordów, a ilością operacji dyskowych.

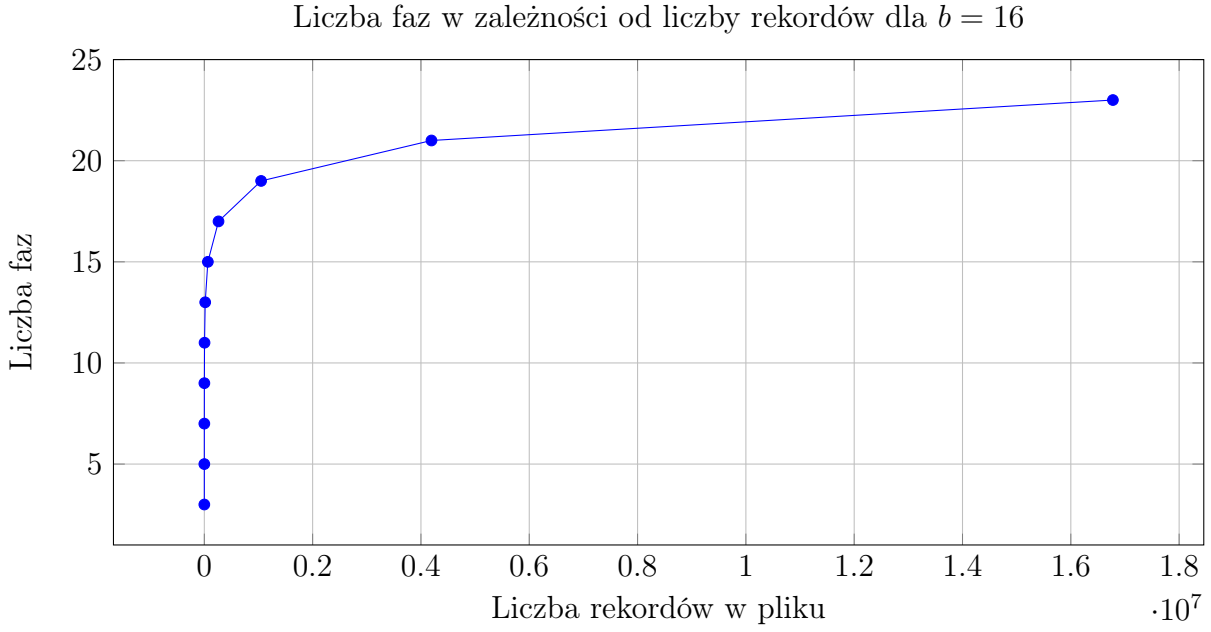
Liczba faz

Liczba faz potrzebnych do posortowania rekordów w pliku rośnie logarytmicznie wraz z początkową liczbę serii. Zależność można opisać wzorem:

$$liczba\ faz = \lceil log_2 r \rceil$$

Liczba rekordów	Początkowa liczba serii	Zmierzona liczba faz	Oczekiwana liczba faz
16	8	3	3
64	33	5	6
256	127	7	7
1024	509	9	9
4096	2070	11	12
16384	8123	13	13
65536	32728	15	15
262144	131272	17	18
1048576	514193	19	19
4194304	2096604	21	21
16777216	8387382	23	23

Tabela 1: Porównania liczby faz dla $b = 16$.



Oczekiwana liczba faz dobrze przybliża rzeczywistą liczbę faz. Przypadki, w których pomierzona liczba faz jest mniejsza niż oczekiwana może wynikać ze zjawiska sklejania serii. Efekt ten powoduje zmniejszenie liczby faz. Początkowa liczba serii w pliku wynosi w przybliżeniu połowę liczby rekordów w pliku.

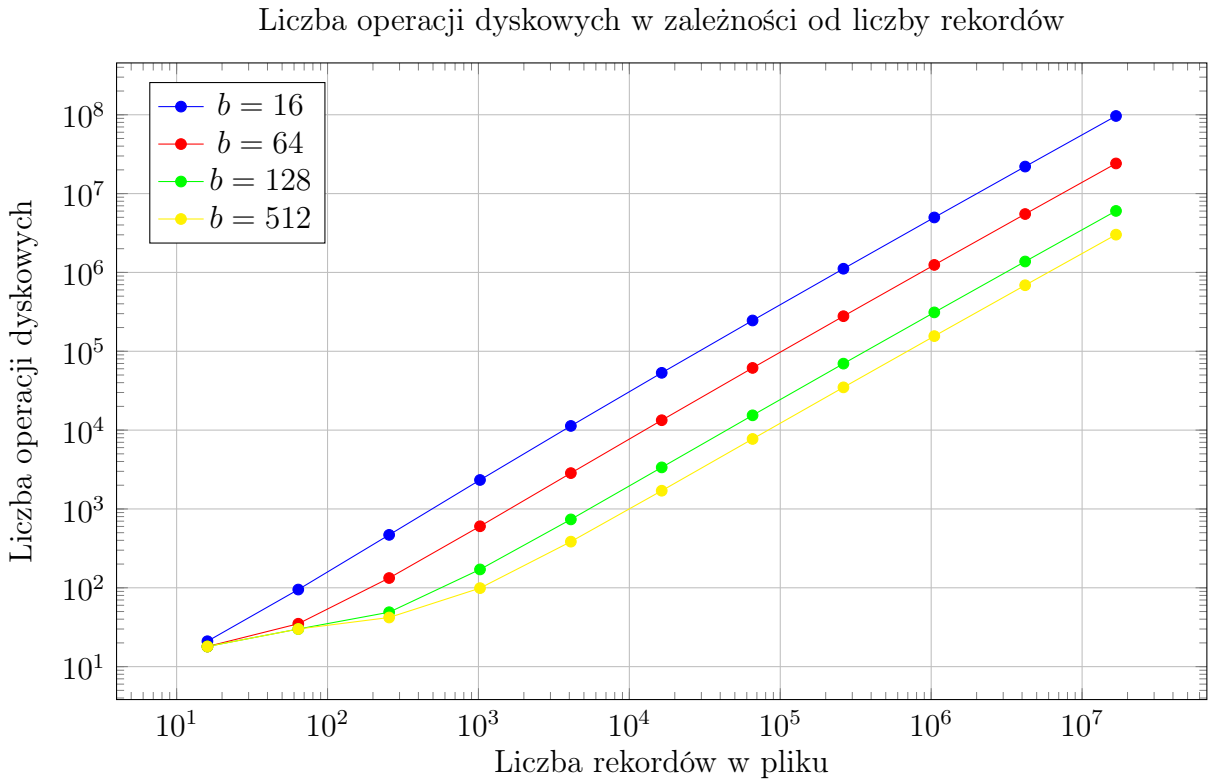
Liczba operacji dyskowych

Liczbę operacji dyskowych w średnim przypadku można oszacować na:

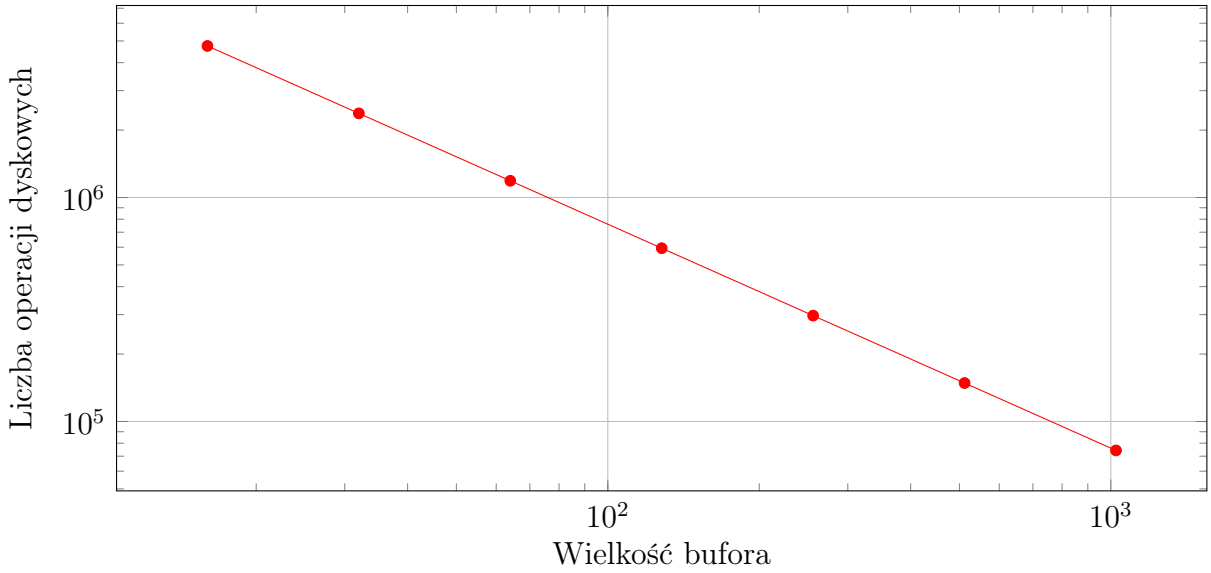
$$liczba\ operacji\ dyskowych = 4\frac{N}{b}\lceil log_2 r \rceil$$

Liczba rekordów	Zmierzona liczba operacji	Teoretyczna liczba operacji	Błąd względny
16	21	12	$7{,}5{\cdot}10^{-1}$
64	95	96	$1{\cdot}10^{-2}$
256	469	448	$4{,}6{\cdot}10^{-2}$
1024	2331	2304	$1{,}1{\cdot}10^{-2}$
4096	11297	12288	$8{\cdot}10^{-2}$
16384	53287	53248	$7{,}3{\cdot}10^{-4}$
65536	245805	245760	$1{,}8{\cdot}10^{-4}$
262144	1114163	1179648	$5{,}5{\cdot}10^{-2}$
1048576	4980793	4980736	$1{,}1{\cdot}10^{-5}$
4194304	22020159	22020096	$2{,}8{\cdot}10^{-6}$
16777216	96469061	96468992	$7{,}1{\cdot}10^{-7}$

Tabela 2: Porównanie liczby operacji dyskowych dla $b = 16$.



Liczba operacji dyskowych dla pliku z milionem rekordów w zależności od wielkości bufora



Porównując rzeczywistą liczbę operacji dyskowych z teoretyczną liczba dla rozmiaru buforu $b = 16$ można zauważyć, że wraz ze wzrostem liczby rekordów przybliżenie staje się lepsze. Można to zaobserwować dla bardzo dużej liczby rekordów. Duża rozbieżność dla pierwszej próbki może wynikać z jej małej wielkości.

Analizując wykres rzeczywistych liczby operacji dyskowych w zależności od liczby rekordów dla różnych rozmiarów buforów można zauważyć, że wraz ze wzrostem rozmiaru liczby rekordów w pliku, zwiększa się liczba operacji dyskowych. Coraz większe rozmiary bufora pozwalają zmniejszyć liczbę operacji w trakcie pojedynczej fazy. Ze względu na ich wielkość, zmniejsza się liczba stron, które są ładowane do pamięci oraz zapisywane na dysk. Można to zaobserwować dla trzech pierwszych próbek dla $b = 512$.

Coraz większe rozmiary bufora powodują zmniejszenie całkowitej liczby operacji dyskowych nawet o rząd wielkości. Znaczne zmniejszenie całkowitej liczby operacji przekłada się na szybsze posortowanie pliku. Odbywa się ono kosztem znacznego zwiększenia pamięci przydzielonej buforowi.