Projekt ze „Struktur baz danych”

Plik indeksowo-sekwencyjny

Mateusz Stencel
188676

Wstęp

Celem projektu było zaimplementowanie pliku działającego według organizacji indeksowo-sekwencyjnej. Jest to rozszerzenie organizacji sekwencyjnej poprzez dodanie indeksu, który przyspiesza dostęp do rekordu oraz o dodanie operacji wstawiania nowych rekordów. Implementacje algorytmu wykonano w języku C++. Rekordem pliku jest klucz oraz 3 liczby oznaczające trapezy – długości obu podstaw trapezu i jego wysokość, flaga czy rekord jest usunięty oraz wskaźnik na kolejny rekord w obszarze nadmiarowym jeśli taki jest. Program w trakcie działania używa trzech plików. Obszarem głównym jest plik `FILE`. Jego strony indeksuje plik `IDX`. Jeden wpis w indeksie zawiera wartość pierwszego klucza na stronie oraz numer strony. Gdy niemożliwe jest wstawienie nowego rekordu na stronę z uwagi na brak odpowiedniego miejsca w pliku głównym, wtedy rekord wstawiany jest do obszaru nadmiarowego rozumianego jako plik `OF`. Wówczas tworzony jest odnośnik, który wskazuje na następny rekord w obszarze nadmiarowym. Automatyczna reorganizacja pliku występuje w momencie gdy obszar nadmiarowy zostaje zapełniony. Przy tworzeniu nowego obszaru nadmiarowego przyjęto proporcję $\frac{V}{N} = 0.1$.

Wejście i wyjście programu

Program działa w dwóch trybach. Może działać w trybie interaktywnym – po wykonaniu każdej instrukcji program czeka na kolejną. Może także działać w oparciu o dostarczony plik testowy. Plik testowy wyrażony jest w formie tekstowej. Jest to ciąg kolejnych instrukcji do wykonania przez program. Po każdej instrukcji program wyświetli informacje o liczbie zapisów i odczytów na dysk. W programie zdefiniowano następujące instrukcje:

`+k a b h` : Dodanie rekordu o kluczu k i danych a b h.

`r` : Dodanie rekordu o losowym kluczu z losowymi danymi.

`-k` : Usunięcie rekordu o podanym kluczu.

`?k` : Wyszukanie rekordu o podanym kluczu i wyświetlenie go.

`u k a b h` : Aktualizacja rekordu o podanym kluczu k na dane a b h.

`U k K` : Aktualizacja rekordu o kluczu k na klucz K.

`p` : Wyświetlenie kluczy nieusuniętych rekordów w pliku w kolejności.

`P` : Wyświetlenie struktury pliku głównego oraz obszaru nadmiarowego wraz ze wszystkimi atrybutami.

`i` : Wyświetlenie indeksu.

`o` : Przymusowa reorganizacja pliku.

`c` : Wyczyszczenie pliku.

`q` : Zakończenie działania programu jeśli działał w trybie interaktywnym.

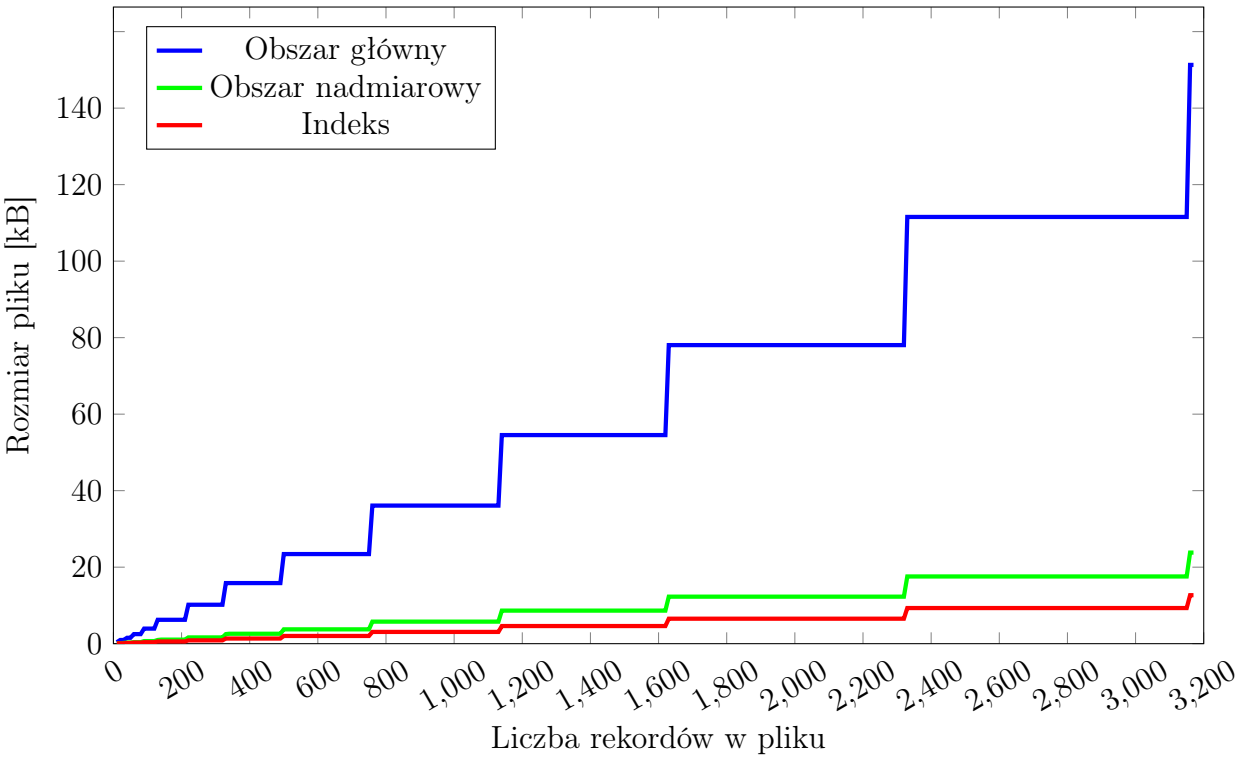
Wyniki eksperymentu

Aby zweryfikować działanie programu wygenerowano dwa pliki testowe. Każdy plik zawiera łącznie 10000 operacji. Zawierają operacje dodawania (5081), usunięcia (2014), wyszukania (934), aktualizacji (1011), aktualizacji klucza (960). Plik `manreorg.txt` zawiera dodatkowo 100 operacji przymusowej reorganizacji. Przy generowaniu rekordów ograniczono wartości klucza do $\langle 1, 7000 \rangle$. Klucze generowano według rozkładu równomiernego. Ponadto zbadano zależność średniej liczby operacji dyskowych w zależności od współczynnika zapełnienia strony α oraz współczynnika blokowania b dla operacji wstawiania rekordu oraz reoganizacji pliku.

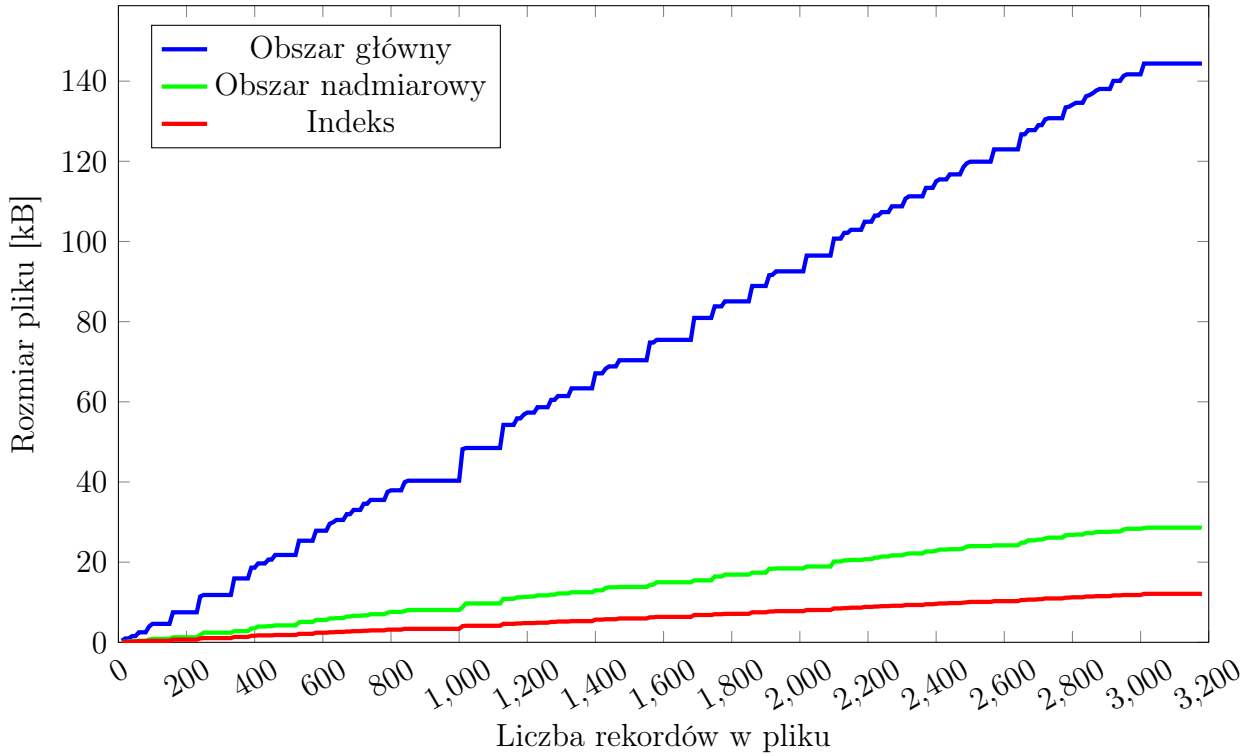
	`autoreorg.txt`	`manreorg.txt`
Dodawanie	65.37	74.90
Usuwanie	63.31	72.64
Wyszukanie	65.90	75.83
Aktualizacja	63.46	72.67
Aktualizacja klucza	118.92	139.93
Reorganizacja	16099.6	40899.28

Tabela 1: Średnia liczba operacji dyskowych dla poszczególnych plików testowych i zadanych komend dla $b = 4$ oraz $\alpha = 0.5$

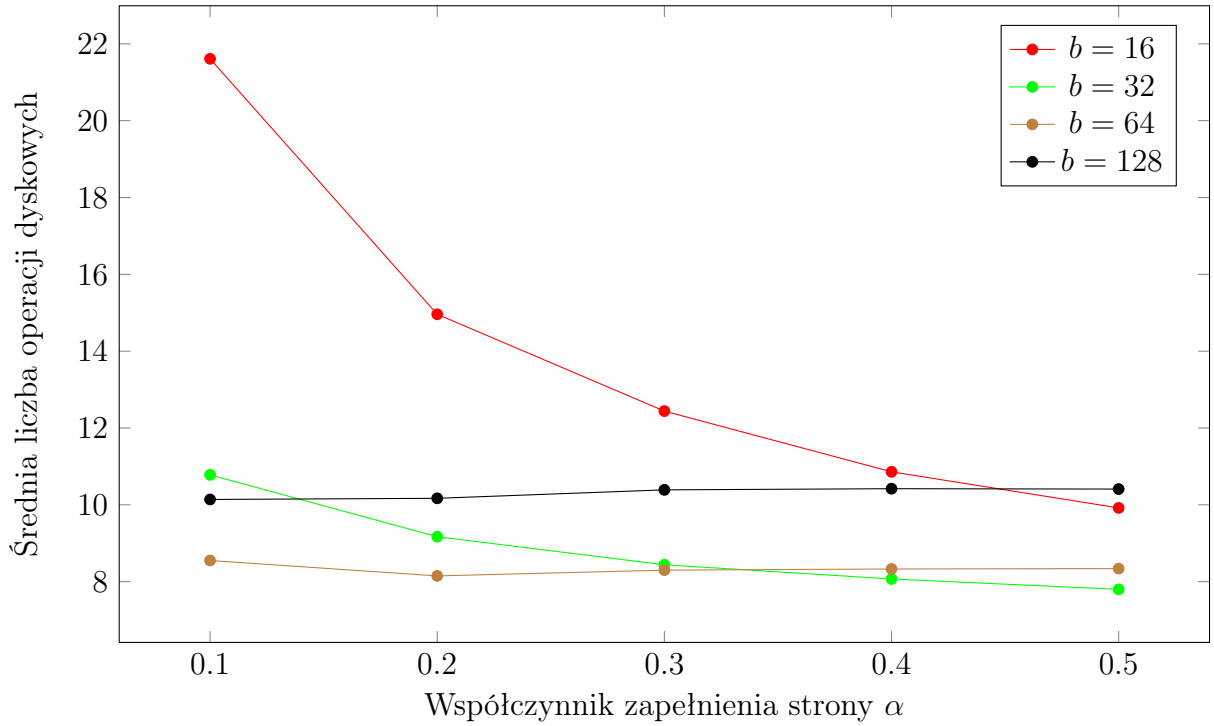
Rozmiar pliku w zależności od liczby rekordów dla $b = 4$ i $\alpha = 0.5$ dla pliku `autoreorg.txt`



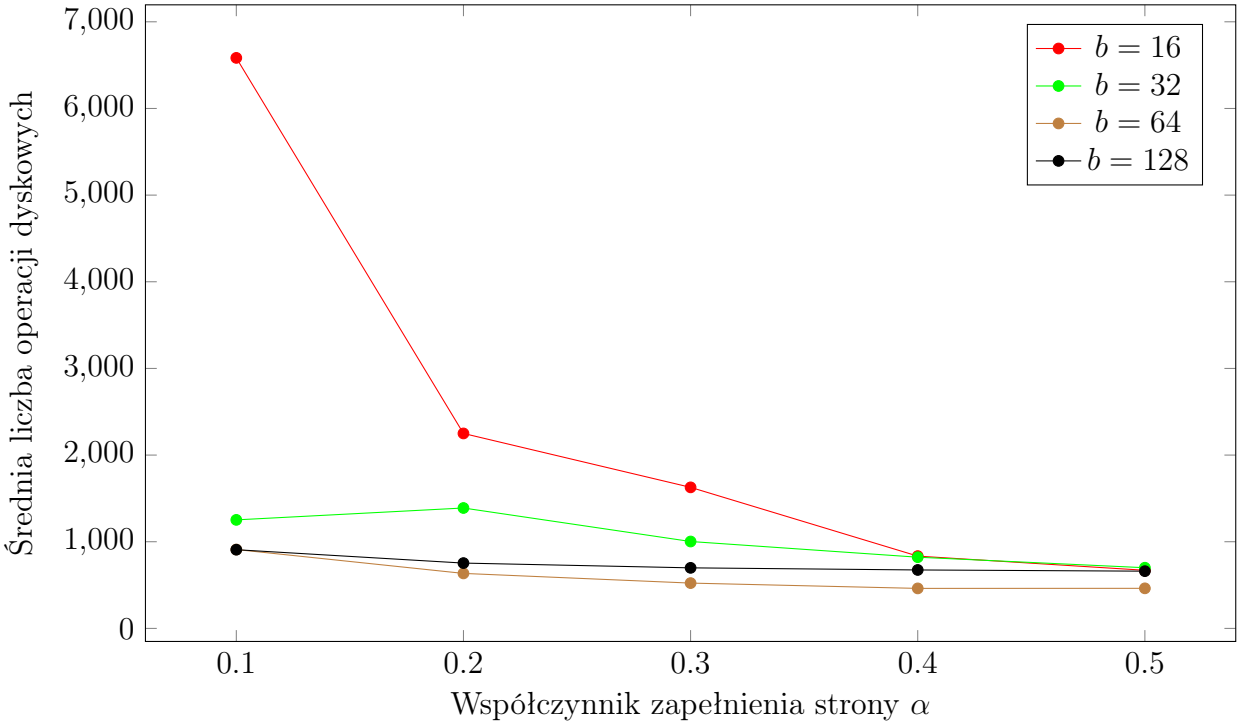
Rozmiar pliku w zależności od liczby rekordów dla $b = 4$ i $\alpha = 0.5$ dla pliku `manreorg.txt`



Średnia liczba operacji dyskowych dla operacji wstawiania rekordu dla pliku `autoreorg.txt`



Średnia liczba operacji dyskowych dla reorganizacji dla pliku `autoreorg.txt`



Wnioski

Porównując wykresy rozmiaru pliku w zależności od liczby rekordów dla obu plików można zauważyć, że dla pliku gdzie reorganizacja odbywała się automatycznie, rozmiary każdego pliku rosły skokowo. Dla pliku testowego, który zawierał przymusowe reorganizacji można zauważyć, że rozmiar obszaru głównego cechował się wzrostem przypominający liniowy. Powodem tego mogło być równomierne rozmieszczenie operacji reorganizacji w pliku testowym. Częstsze reorganizacje zwiększyły średnie liczby operacji dyskowych dla każdej operacji programu. Reorganizacja budowała nowy plik oraz tworzyła większy indeks, który powodował zwiększenie liczby operacji dyskowych.

Analizując wykres średniej liczby operacji dyskowych w zależności od α oraz b dla wstawiania rekordu można zauważyć, że największy wynik osiągnięto dla $\alpha = 0.1$ i $b = 16$. Jest to spowodowane częstą reorganizacją, która kosztowała dużą liczbę operacji dyskowych. Częste przeglądanie dużego indeksu też miało wpływ na zwiększenie operacji. Powodem tego mógł być warunek reorganizacji, który powodował utworzenie nowej strony z niewielką liczbą rekordów na niej. Najmniejsza średnia dla operacji wstawiania jest dla pary $\alpha = 0.5$ i $b = 32$. Zwiększanie rozmiaru strony dyskowej zmniejszało liczbę operacji dla wstawiania rekordu do pewnego momentu. Dla $b = 128$ otrzymano wysokie średnie dla wstawiania, większe niż dla mniejszych b. Może być to spowodowane rzadkim zapełnieniem obszaru nadmiarowego, którego zapełnienie powodowałoby reorganizacje pliku. Nowe rekordy były wstawiane do nadmiaru tworząc kolejne łańcuchy przepełnień. Ich przeglądanie w celu dodania kolejnych rekordów zwiększały średnią liczbę operacji. Zwiększanie wartości α pozwoliło na zmniejszenie liczby operacji dla każdego rozmiaru strony. Spadek ten jest szczególnie widoczny dla $b = 16$. Dla reorganizacji najmniejszy wynik jest dla pary $\alpha = 0.5$ i $b = 64$.