Organizacja pliku w formie B-drzewa

Marcel Grużewski 193589

8 grudnia 2024

1 Wstęp

1.1 Cel projektu

Celem projektu jest zaimplementowanie i przetestowanie jednej z wybranych indeksowych organizacji plików, takich jak B-drzewo, B+-drzewo lub organizacja indeksowo-sekwencyjna. Projekt ma na celu symulację działania wybranej struktury w warunkach, w których zarządzanie dużymi zbiorami danych wymaga wydajnych operacji odczytu i zapisu z dysku. 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. Program umożliwi analizę liczby operacji I/O oraz wizualizację wewnętrznej struktury pliku i indeksu, co pozwoli na przeprowadzenie eksperymentu.

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

1.2 Wybrany algorytm

Wybrany algorytm to organizacja danych w pliku za pomocą B-drzewa.

B-drzewo to zrównoważona struktura drzewiasta, która umożliwia efektywne wstawianie, wyszukiwanie i usuwanie rekordów. Struktura ta jest szczególnie przydatna w przypadku operacji na dużych zbiorach danych, które nie mieszczą się w całości w pamięci operacyjnej, ponieważ minimalizuje liczbę operacji I/O poprzez grupowanie kluczy i wskaźników w węzłach.

Algorytm działania B-drzewa można opisać za pomocą kilku zmiennych:

$$d \text{ – stopień drzewa,}$$
$$h \text{ – wysokość drzewa,}$$
$$N \text{ – liczba kluczy w drzewie.}$$

Każdy węzeł w B-drzewie może zawierać maksymalnie 2d kluczy i posiadać maksymalnie 2d+1 dzieci. Minimalna ilośc kluczy w węźle B-drzewa to d, chyba że to korzeń drzewa, wtedy minimalna ilość luczy wynosi 1. B-drzewa cechują się dużym stopniem rozgałęzienia, co pozwala na efektywne zarządzanie danymi nawet przy bardzo dużej liczbie kluczy. Wysokość drzewa h rośnie logarytmicznie wraz ze wzrostem liczby kluczy, co przekłada się na niewielką liczbę operacji odczytu i zapisu danych z dysku. Dla

$$N \geq 1 \textbf{ oraz } d \geq 2$$

można ją oszacować za pomocą wzoru:

$$h \leq \log_d \left(\frac{N+1}{2} \right)$$

1.3 Wyszukiwanie klucza w B-drzewie

Wyszukiwanie klucza w B-drzewie rozpoczyna się od korzenia i polega na porównywaniu klucza z wartościami przechowywanymi w węźle. Jeśli klucz znajduje się w bieżącym węźle, algorytm kończy działanie. W przeciwnym razie algorytm przechodzi do odpowiedniego dziecka węzła, w zależności od przedziału, w którym znajduje się klucz. Proces ten jest powtarzany aż do znalezienia klucza lub dotarcia do węzła liścia, co oznacza, że klucz nie istnieje w drzewie.

Złożoność wyszukiwania klucza w B-drzewie wynosi O(h), gdzie h to wysokośc drzewa.

1.4 Dodawanie klucza do B-drzewa

Dodawanie klucza do B-drzewa w niektórych przypadkach wymaga dodatjowych operacji przy przepełnieniu węzłów:

- Przepełnienie korzenia drzewa - tworzymy nowy węzęł, dodajemy do niego część kluczy z poprzedniego korzenia, a środkowy klucz przenosimy do nowego korzenia. Wysokość drzewa zwiększa się o 1,
- Przepełnienie węzła (kompensacja) - gdy węzeł jest przepełniony, a jeden z jego sąsiadów (lewy lub prawy) ma wolne miejsce, można dokonać kompensacji. Polega ona na przeniesieniu klucza z rodzica do sąsiada oraz przesunięciu skrajnego klucza z przepełnionego węzła do rodzica, co zwalnia miejsce na nowy klucz w przepełnionym węźle.
- Przepełnienie węzła (podział) - gdy przepełniony węzeł nie ma niepełnych sąsiadów, wykonuje się jego podział. Polega to na przeniesieniu klucza środkowego do rodzica, a pozostałe klucze i wskaźniki są rozdzielane między nowy węzeł a oryginalny, przepełniony węzeł.

2 Implementacja algorytmu

2.1 Ważne aspekty działania programu

- Najpierw użytkownik w argumentach programu podaje w jaki sposób ma działać program:
  - `interactive` - użytkownik sam wpisuje komendy,
  - `commands` - program wykonuje komendy z pliku (domyślnie "commands.txt"),
  - `mixed` - program najpierw wykonuje komendy z pliku, a następnie daje możliwość ich wprowadzenia użytkownikowi,

  gdy użytkownik nie poda żadnego argumentu program bedzie działał w trybie `interactive`. Przy podstawowym działaniu programu dane przechowywane są w pliku "data.txt", a węzły B-drzewa w folderze "Pages",
- Podczas dodawania elementów do drzewa, program rekurencyjnie schodzi do odpowiedniego węzła, dbając o to, żeby nie doszło do przepełnienia węzła, poprzez kompensację i podział węzłów.
- Użytkownik ma także dostęp do:
  - Szukania klucza, po znalezieniu go w B-drzewie program również wyświetli rekord, który mu odpowiada.

  - Wyświetlenia drzewa z jego widoczną strukturą, każdy węzęł wyświetli również jego flagi, oraz klucze, które się w nim znajdują.
  - Wyświetlenia wszystkich kluczy w posortowanej kolejności.

- Program ma dwa rodzaje blokowania: dla węzłów (1 blok = 1 węzeł) oraz dla rekordów (bufferSize rekordów na blok, zależnie od wartości parametru).

2.2 Opis najważniejszych struktur zaimplementowanych w programie

- 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,
- NodePage - węzęł drzewa, przechowuje flagi pomocnicze (Identyfikator węzła, informacja o tym, czy jest liściem, ile posiada kluczy, ile maksymalnie może mieć kluczy), klucze oraz ich indeksy oraz identyfikatory węzłów podrzędnych,
- BTree - główny obiekt w programie, w nim znajdują się metody do wykonywania operacji na drzewie.

2.3 Format pliku testowego

Plikiem testowym są komendy, każda w osobnej linii, zaczynające się od przedrostek nazwy komendy (insert), nastepnie po spacji znajduje się rekord zgodnie z formatem:

Student i jego wyniki z 3 kolejnych kolokwiów z pewnego przedmiotu.

Rekord zawiera zatem 3 liczby ze zbioru **[2, 3, 3.5, 4, 4.5, 5]**, każda oddzielona spacją, na końcu po spacji znajduje się klucz, należący do zbioru liczb naturalnych (maksymalnie $2^{31} - 1$), przykładowa komenda:

```
insert 2 3.5 4 123
```

Gdzie `insert` to komenda, `2 3.5 4` to rekord, a `123` to klucz.

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

3 Eksperyment

3.1 Opis eksperymentu

Eksperyment ma na celu zbadanie wpływu parametrów implementacyjnych, takich jak stopień drzewa, na złożoność operacji w B-drzewie. Do testów użyto różnych wartości stopnia drzewa, który kontroluje liczbę kluczy przechowywanych w węzłach. Ponadto, eksperymenty obejmowały również różną liczbę rekordów przechowywanych w pliku, aby ocenić, jak zmieniają się koszty operacji w zależności od skali danych.

3.2 Liczba odczytów i zapisów względem zmiennej d

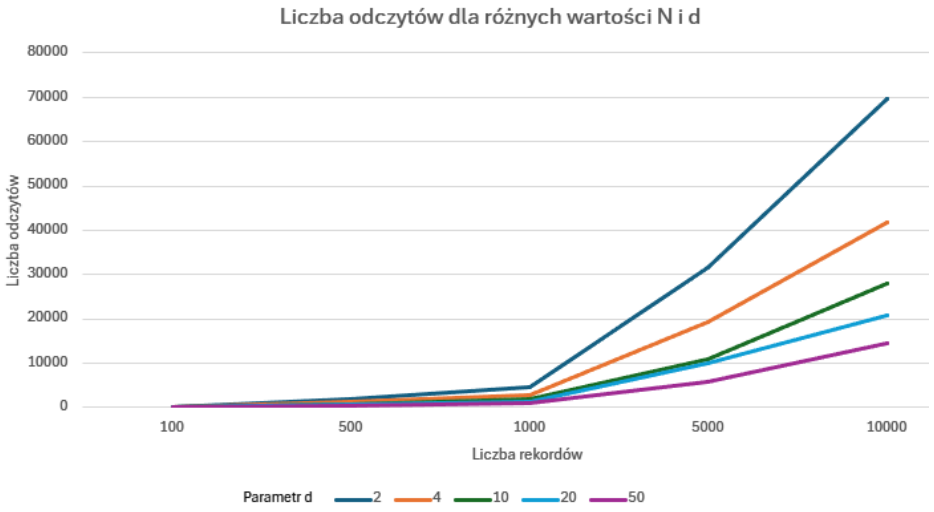
Poniżej znajdują się liczby odczytów i zapisów dla różnych wartości stopnia drzewa (d) oraz liczby rekordów (N).

Tabela 1: Tabela przedstawiająca liczbę odczytów dla różnych wartości d i N.

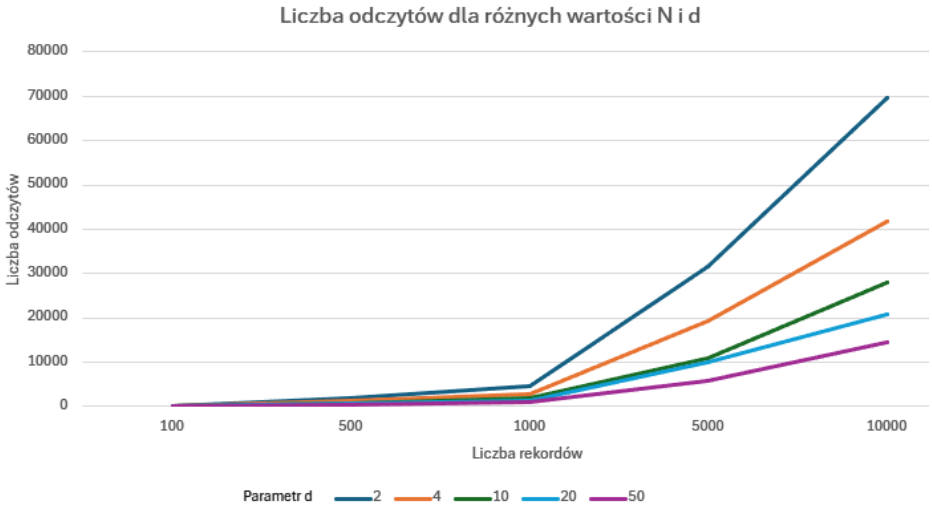
d/N	100	500	1000	5000	10000
2	264	2039	4757	31538	69700
4	175	1287	2946	19383	41859
10	101	777	1951	11044	28096
20	68	594	1171	9947	20927
50	0	508	1132	5805	14598

Tabela 2: Tabela przedstawiająca liczbę zapisów dla różnych wartości d i N.

d/N	100	500	1000	5000	10000
2	238	1448	2815	14261	28789
4	226	1031	2179	10790	21922
10	163	893	1867	8948	18259
20	127	884	1609	8399	16246
50	100	824	1687	7616	15208



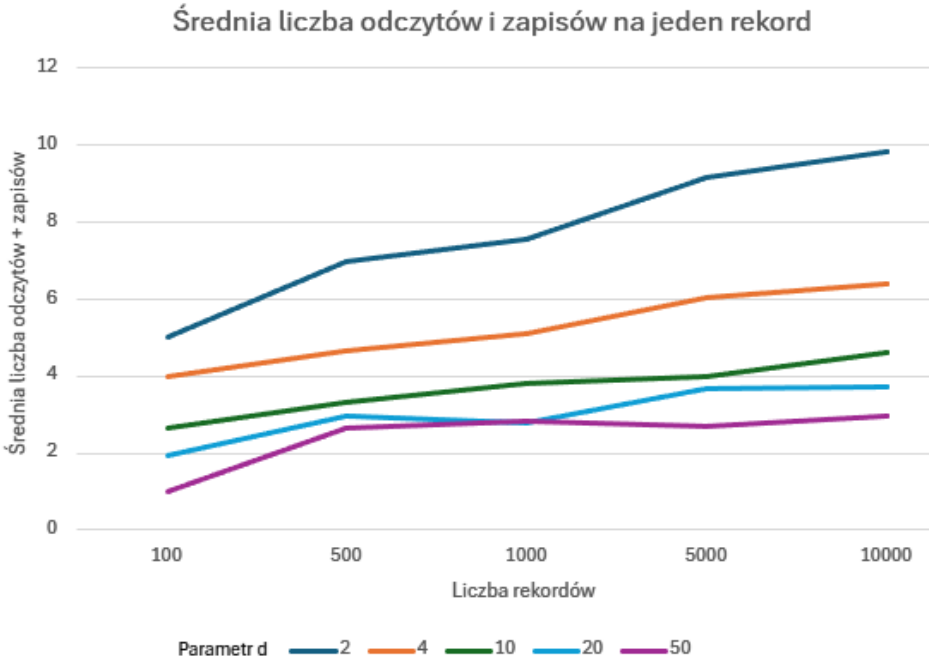
Rysunek 1: Wykres przedstawiający liczbę odczytów dla różnych wartości d i N.



Rysunek 2: Wykres przedstawiający liczbę zapisów dla różnych wartości d i N.

Tabela 3: Tabela przedstawiająca średnią liczbę zapisów i odczytów dla różnych wartości d i N.

d/N	100	500	1000	5000	10000
2	5.02	6.974	7.572	9.1598	9.8489
4	4.01	4.636	5.125	6.0346	6.3781
10	2.64	3.34	3.818	3.9984	4.6355
20	1.95	2.956	2.78	3.6692	3.7173
50	1	2.664	2.819	2.6842	2.9806



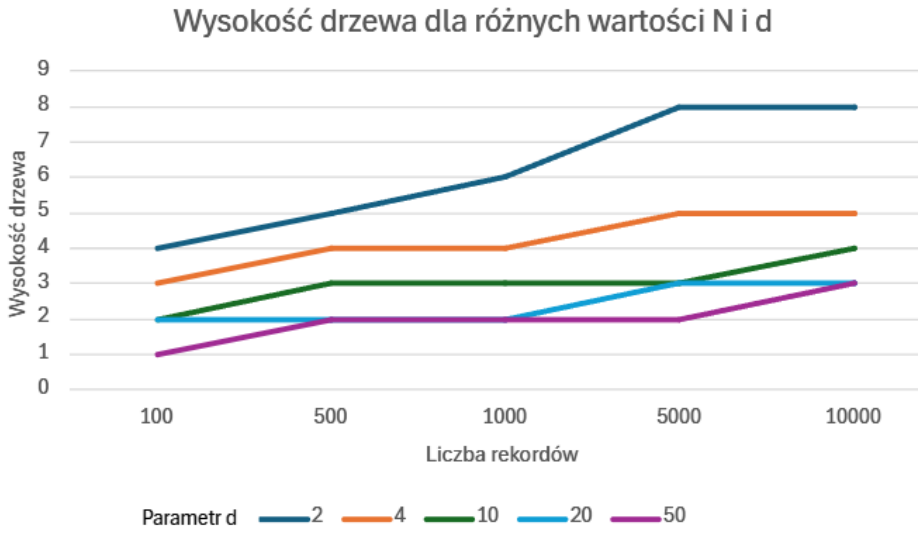
Rysunek 3: Wykres przedstawiający średnią liczbę zapisów i odczytów dla różnych wartości d i N.

3.3 Wysokość drzewa względem zmiennej d

Poniżej znajdują się wysokości drzewa dla różnych wartości stopnia drzewa (d) oraz liczby rekordów (N)

Tabela 4: Tabela przedstawiająca wysokości drzewa dla różnych wartości d i N.

d/N	100	500	1000	5000	10000
2	5.02	6.974	7.572	9.1598	9.8489
4	4.01	4.636	5.125	6.0346	6.3781
10	2.64	3.34	3.818	3.9984	4.6355
20	1.95	2.956	2.78	3.6692	3.7173
50	1	2.664	2.819	2.6842	2.9806



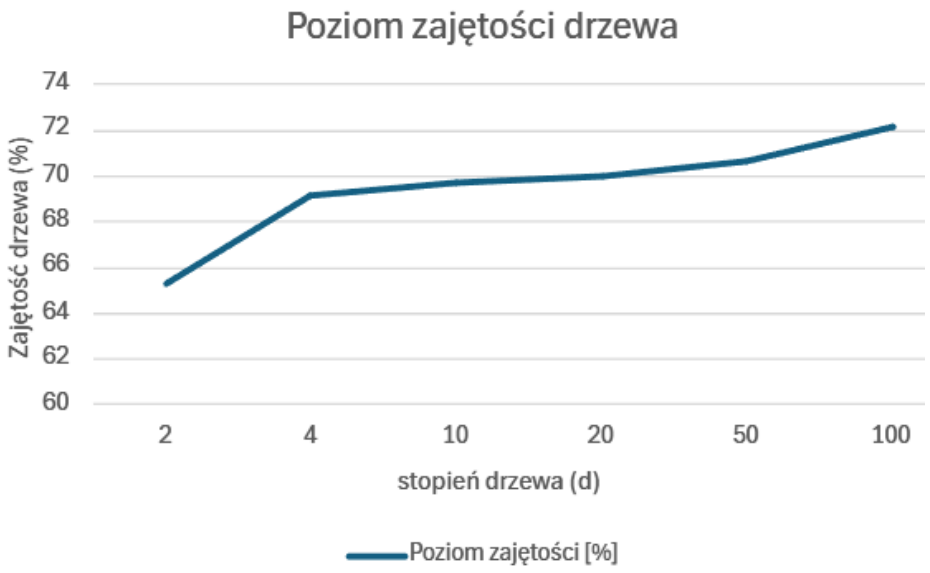
Rysunek 4: Wykres przedstawiający wysokość drzewa dla różnych wartości d i N.

3.4 Poziom zajętości drzewa względem zmiennej d

Po wywołaniu programu 6-krotnie dla stałej liczby rekordów (2000), została wyciągnięta średnia zajętość drzewa dla różnych wartości stopnia drzewa (d).

Tabela 5: Tabela przedstawiająca poziom zajętości dla różnych wartości d.

d	2	4	10	20	50	100
Poziom zajętości [%]	65.3	69.12	69.72	70	70.64	72.2



Rysunek 5: Wykres przedstawiający zajętość drzewa dla różnych wartości d.

4 Wnioski

Eksperyment udowodnił, że odpowiednie dobranie stopnia drzewa (paramter d) ma kluczowe znaczenie przy optymalnizacji B-drzewa. Im większy parametr d, tym mniejsza liczba odczytów i zapisów była potrzebna do wykonania operacji dodawania elementu do B-drzewa. Jednak trzeba pamiętać o tym, że jednocześnie przy większym parametrze d, pobieramy i zapisujemy większe bloki danych. Jeśli chodzi o wysokość drzewa to ponownie, im większy parametr d, tym mniejsza wysokość drzewa, to jednak oznacza, że większe obciążenie przy złożoności jest na algorytmie dostępu do konkretnego klucza w danym węźle. Ostatnim eksperymentem było sprawdzenie poziomu zajętości B-drzewa względem wartości parametru d. Eksperyment ten wykazał tendecję wzrostową przy zapełnieniu drzewa, szybkość wzrostu była jednak znacząco mniejsza dla większych wartości d.

Podsumowując, warto zadbać o odpowiednią wartość stopnia B-drzewa, ponieważ ma duży wpływ na złożoność operacji. Trzeba wziąć pod uwagę złożoność wewnętrznych operacji oraz wielkość pobieranych i zapisywanych bloków, podczas próby zoptymalizowania liczby zapisów i odczytów na dysku. Najlepszym podejściem zatem będzie dostosowanie parametru d do ilości danych i innych potrzeb.

5 Źródła

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Wprowadzenie do algorytmów, Wydawnictwo Helion, Rozdział 18: B-drzewa, 2022.