Struktury Baz Danych Projekt 2 - Sprawozdanie z indeksowej organizacji plików z użyciem struktury B-drzewa

Kamil Szabłowski, 193248

December 10, 2024

1 Wprowadzenie

Zadaniem projektu była implementacja jednego z wybranych algorytmów organizacji plików indeksowych. Zaimplementowanym przeze mnie algorytmem jest algorytm *B-drzewa*.

Użytym w projekcie typem rekordu pliku są ciągi 10 *liczb* z dodatkowycm kluczem, będącym liczbą natruralną większą od 0. Kryterium sortowania rekordu jest wartość klucza. Za *liczbę* będącą elementem rekordu uznaję 32-bitową liczbę całkowita ze znakiem. Poniżej przedstawione został przykład takiego rekordu:

1	23	−45	67	12	−8	34	0	29	−16	5

Pierwszy element powyższego rekordu to kluczz wynoszący 1, reszta stanowi ciąg 10 liczb.

Rozmiar rekordu wynosi w takim razie 44 bajtów (11 liczb całkowitych 4-bajtowych). Przyjęty rozmiar bloku dyskowego pliku danych na potrzeby tego eksperymentu wynosi 364 bajtów.

2 Opis implementacji

Użytą w tym projekcie metodą organizacji pliku jest użycie struktury B-drzewa.

Alogorytmy wykorzystane do wstawianie, usuwania i wyszukiwania rekordów w B-drzewie są analogiczne do tych przedstawionych na wykładzie. Modyfikacja rekordu polega na usunięciu starego rekordu i dodaniu nowego rekordu z nowym kluczem, chyba że nie dokonujemy zmiany klucza, wtedy modyfikowana jest wartość rekordu bezpośrednio w pliku z danymi, po uwczesnym wyszukaniu klucza.

W mojej implementacji na stronę B-drzewa zapisaną do pliku składają się:

- Wskaźnik na stronę rodzica (liczba całkowita, 4B)
- Liczba rekordów w stronie (liczba całkowita, 4B)

- Lewy skrajny wskaźnik na stronę, z rekordami mniejszymi od wszystkich innych należących do obecnej strony (liczba całkowita, 4B)
- Od 1 do $2d$ węzłów, gdzie na dany węzeł składa się:
  - Wartość klucza rekordu (liczba całkowita, 4B)
  - Offset rekordu o danym kluczu w pliku z danymi (liczba całkowita, 4B)
  - Wskaźnik na stronę z rekordami większymi od klucza w tym węźle, ale mniejszymi od klucza w następnym węźle (liczba całkowita, 4B)

Nie zależnie od ilości węzłów w pliku indeksowym B-drzewa, zapisana (lub odczytana) strona z dysku ma zawsze rozmiar $4 + 4 + 4 + 2d * 12 = 12 + 24d$ bajtów.

Strona w pamięci w programie jest prawie identyczna jak strona zapisana na dysku, z tą różnicą, że lewy skrajny wskaźnik na stronę jest zastąpiony dodatkowym węzłem 0, w którym poza wartością wskaźnika na stronę-dziecko, wartość klucza i rekordu ustawiona jest na NULL

Jeśli chodzi o plik z danymi, to podobnie jak w przypadku pierwszego projektu, rekordy są zapisywane binarnie, jeden po drugim, bez żadnych dodatkowych informacji.

2.1 Opis buforowania

3 Specyfikacja plików testowych

Plikiem testowym jest plik tekstowy (.txt) o określonej poniżej strukturze.

3.1 Struktura pliku testowego

Plik testowy składa się z komend oddzielonych znakime nowej linii.

Komendy dostępne do użycia w pliku testowym są analogiczne co do komend dostępnych w interfejsie tekstowym. Lista dostępnych komend może zostać wyświetlona przy użyciu komendy `help`. Dodatkowe komendy do debuggowania programu są pokazane po użyciu komendy `help debug`.

Output komendy `help`:

Output komendy `help`

```
List of available commands
TIP: Most commands can be used by using
first letters of each word in the command
Example: 'am 5' is the same as 'addmulti 5'
--------------------------------------------------------------------
help                                        Show this help message
help debug                                  Show help relating to debug
setcompensation [true/false]                Toggle compensation
clear                                       Clear all the files
rand [n]                                    Insert n random records
update [key] [value] [newKey: OPTIONAL]     Update record with given key
insert [key] [value]                        Insert record into the file
search [key]                                Search for record with key
print [group] [all]                         Prints all records in db
loadtest [filename]                         Loads test file
```

Output komendy `help debug`:

Output komendy `help debug`

```
List of available debug commands
--------------------------------------------------------------------
dblockstats             Prints block stats
dforceflush             Forces a flush of the files
dgetrecord [n]          Gets a record at offset (data file)n
```

Przykład poprawnie ustrukturyzowanego pliku testowego:

Plik: `text.txt`

```
clear
insert 1 5
insert 2 10
insert 3 15
insert 4 20
insert 5 25
delete 2
search 3
search 2
dblockstats
print all group
print
```

Poniższy plik testowy wykona dokładnie to samo co powyższy:

Plik: `text.txt`

```
c
i 1 5
i 2 10
i 3 15
i 4 20
i 5 25
d 2
s 3
s 2
dbs
p a g
p
```

Tak spreparowany plik może bezproblemu zostać załadowany do interfejsu tekstowego.

3.2 Załadowanie i uruchomienie pliku testowego

Aby załadować plik testowy z dysku, wystarczy w interfejsie tekstowym użyć komendy `loadtest nazwa_pliku.txt`, gdzie zamiast *nazwa_pliku.txt* należy wpisać ścieżkę do docelowego pliku testowego, który ma zostać załadowany. Ścieżka może być bezwzględna, lub względna - zależna od katalogu w którym został uruchomiony interfejs tekstowy.

4 Sposób prezentacji wyników działania programu

Po uruchomieniu programu, organizowany plik indeksowy orazz z danymi jest pusty i nie zawiera żadnych rekordów ani stron.

Aby dodać rekordy do bazy danych należzy użyć jednej z nastepujących komend:

- `insert [key] [value]` - próbuje dodać rekord o danym kluczu i wartości do bazy danych
- `rand [n]` - próbuje dodać n losowych rekordów do bazy danych

Analogicznie, usunięcie rekordu z bazy danych odbywa się przy użyciu komendy `delete [key]`.

Wyszukiwanie rekordu w bazie danych odbywa się przy użyciu komendy `search [key]`.

Modyfikacja rekordu w bazie danych odbywa się przy użyciu komendy `update [key] [value] [new`

Po użyciu komendy `print` bez żadnych parametrów, program wypisuje zawartość bazy danych w formie rekordów, od najmniejszego do największego klucza w następującym formacie: Przykład:

Output komendy `print`

```
(1) 10 10 10 10 10 10 10 10 10 10
(2) 20 20 20 20 20 20 20 20 20 20
(3) 30 30 30 30 30 30 30 30 30 30
(4) 40 40 40 40 40 40 40 40 40 40
(5) 50 50 50 50 50 50 50 50 50 50
```

Liczba w nawiasie to klucz rekordu, podczas gdy liczby po spacji to wartości rekordu, czyli liczby należące do ciągu.

Po użyciu komendy `print` z parametrem `group`, program wypisuje zawartość bazy danych w formie rekordów, ale tym razem grupując rekordy znajdujące się na jednej stronie B-drzewa.

Po użyciu komendy `print` z parametrem `all`, program wypisuje zawartość bazy danych w pełnej formie. Przykład:

Output komendy `print all`

```
RecordOffset: 0 | PageOffset: 0 | ParentPageOffset: 1
| LeftPagePtr: -1 | RightPagePtr: -1
(1) 1 1 1 1 1 1 1 1 1 1

RecordOffset: 1 | PageOffset: 0 | ParentPageOffset: 1
| LeftPagePtr: -1 | RightPagePtr: -1
(2) 2 2 2 2 2 2 2 2 2 2

RecordOffset: 2 | PageOffset: 1 | ParentPageOffset: -1
| LeftPagePtr: 0 | RightPagePtr: 2
(3) 3 3 3 3 3 3 3 3 3 3

RecordOffset: 3 | PageOffset: 2 | ParentPageOffset: 1
| LeftPagePtr: -1 | RightPagePtr: -1
(4) 4 4 4 4 4 4 4 4 4 4

RecordOffset: 4 | PageOffset: 2 | ParentPageOffset: 1
| LeftPagePtr: -1 | RightPagePtr: -1
(5) 5 5 5 5 5 5 5 5 5 5

-----------------------------------
Total records: 5
Total pages count: 3
Height: 2
-----------------------------------
```

Poza wyświetleniem klucza i wartości rekordu, program wypisuje również informacje o stronach B-drzewa, takie jak:

- `RecordOffset` - offset rekordu w pliku z danymi
- `PageOffset` - offset strony w pliku indeksowym
- `ParentPageOffset` - offset strony rodzica w pliku indeksowym
- `LeftPagePtr` - offset strony dziecka z rekordami mniejszymi od klucza w obecnym węźle
- `RightPagePtr` - offset strony dziecka z rekordami większymi od klucza w obecnym węźle

Wartość -1 w polach `LeftPagePtr`, `RightPagePtr` oraz `ParentPageOffset` oznacza, że dany wskaźnik nie wskazuje na nic, czyli węzeł nie ma dzieci, bądź rodzica (tylko korzeń).

Dodatkowo wyświetlana jest informacja o całkowitej liczbie rekordów, liczbie stron oraz wysokości drzewa.

Po użyciu komendy `print all group` program wypisuje zawartość bazy danych w pełnej formie, ale tym razem grupując rekordy po stronach B-drzewa. Przykład:

Output komendy `print all group`

```
-------- Page 0 ----------
RecordOffset: 0 | PageOffset: 0 | ParentPageOffset: 1
| LeftPagePtr: -1 | RightPagePtr: -1
(1) 1 1 1 1 1 1 1 1 1 1

RecordOffset: 1 | PageOffset: 0 | ParentPageOffset: 1
| LeftPagePtr: -1 | RightPagePtr: -1
(2) 2 2 2 2 2 2 2 2 2 2

-------- Page 1 ----------
RecordOffset: 2 | PageOffset: 1 | ParentPageOffset: -1
| LeftPagePtr: 0 | RightPagePtr: 2
(3) 3 3 3 3 3 3 3 3 3 3

-------- Page 2 ----------
RecordOffset: 3 | PageOffset: 2 | ParentPageOffset: 1
| LeftPagePtr: -1 | RightPagePtr: -1
(4) 4 4 4 4 4 4 4 4 4 4

RecordOffset: 4 | PageOffset: 2 | ParentPageOffset: 1
| LeftPagePtr: -1 | RightPagePtr: -1
(5) 5 5 5 5 5 5 5 5 5 5

----------------------------------
Total records: 5
Total pages count: 3
Height: 2
----------------------------------
```

Jak widać powyżej, rekordy są grupowane po stronach B-drzewa, co w przypadku większej ilości rekordów może spowodować, że rekordy nie są wyświetlone w kolejności rosnącej klucza.

5 Eksperyment

5.1 Konstrukcja eksperymentu

Do przeprowadzenia eksperymentu wykorzystane zostały funkcje udostępniane prze interfejs tekstowy programu.

Schemat eksperymentu wyglądał następująco:

1. Wygenerowanie plików testowych z losowymi rekordami
2. Załadowanie i posortowanie plików testowych
3. Spisanie informacji o każdym sortowaniu do pliku tekstowego
4. Wyznaczenie teoretycznych liczb faz i operacji dyskowych na podstawie spisanych informacji
5. Wygenerowanie wykresów na podstawie danych eksperymentalnych i teoretycznych
6. Porównanie otrzymanych wyników

5.2 Przygotowania do eksperymentu

Aby wygenerować pliki testowe przy użyciu interfejsu tekstowego, należało użyć komend:

```
clear
rand N
save randN.txt
```

gdzie N to liczba rekordów do wygenerowania. Na potrzeby tego eksperymentu zostało wygenerowane 7 plików testowych dla $N = \{2^4, 2^6, 2^8, 2^{10}, 2^{12}, 2^{14}, 2^{16}\}$. Komenda `rand` generuje N rekordów, w których każdy elemnt przyjmuje wartość z zakresu od 0 do 999. Do eksperymentu użyłem liczb rekordów, które pozwalają na łatwiejsze obliczenie teoretycznej liczby operacji dyskowych.

Aby wczytać i posortować utworzone pliki, należało skorzystać z poniższych komend:

```
clear
load randN.txt
sort polyphase quiet
```

Opcja `quiet` sprawia, że jedynie informacje o danej fazie są wypisywane w konsoli, gdyż dla większych N wypisywanie zawartość taśm zajmowałoby zbyt długo.

Po każdym sortowaniu, spisałem do osobnego pliku dane potrzebne w dalszej części eksperymentu, czyli liczbę odczytów i zapisów z dysku, liczbę faz, początkową liczbę runów w pliku.

5.3 Wyniki eksperymentu

5.4 Podsumowanie

Przeprowadzony eksperyment potwierdza poprawność implementacji alogrytmu sortowania polifazowego z wykorzystaniem liczb Fibonacciego. Potwierdza to fakt, że obliczone liczby teoretyczne zgadzają się z wynikami otrzymanymi podczas eksperymentu.

Dla dużych plików początkowych jedank, uwidocznił się pewnien błąd względny wynoszący ok. 2.5% przy liczbie operacji dyskowych. Wskazuje on na możliwość wykonywania przez

program zbędnych operacji, lecz analiza kodu programu nie uwidoczniła dlaczego taki błąd zaistniał.

Liczba faz za to wydaje się być w pełni poprawna, gdyż błąd między wartościami teoretycznymi i praktycznymi jest w oczekiwanym zakresie.