Sprawozdanie Struktury Baz Danych Projekt 2: Bdrzewo

Radosław Piotrowicz 193251

1. Wprowadzenie:

Celem projektu było zaimplementowanie indeksowej metody organizacji pliku. Do wyboru były metody: Indeksowo-sekwencyjna, Bdrzewo, B+-drzewo. W moim projekcie użyte zostało Bdrzewo. Bdrzewa bardzo dobrze nadają się do indeksowej organizacji pliku, ze względu na to że drzewa te z założenia mają być szerokie, i nie zbyt wysokie, co zmniejsza ilość dostępów do pliku przy ich przeszukiwaniu.

2. Implementacja:

Program został zaimplementowany w języku C++.
Program będzie pracował na dwóch plikach:

- pliku indeksowym gdzie znajduje się struktura Bdrzewa
- pliku z danymi gdzie zapisywane są dane

Program pozwala na:

- wyszukanie rekordu w Bdrzewie.
- dodanie nowego rekordu.
- usunięciu rekordu.
- zaktualizowaniu wartości w pliku z danymi (wyszukanie w Bdrzewie a potem zaktualizowanie strony z danymi).
- wczytanie sekwencji komend z pliku (dodawanie, usuwanie, aktualizacja).
- wyświetlenie Bdrzewa.
- wyświetlenie zawartości pliku z danymi.
- ustawienie rzędu Bdrzewa.
- wyczyszczenie Bdrzewa.
- przeprowadzenie eksperymentu (wyniki są zapisywane do pliku tekstowego).

2.1) Format pliku:

W pliku z danymi każda strona ma nagłówek, który informuje ile miejsc jest zajęte, oraz 10 rekordów na dane, nowe rekordy dodawane są na pierwszą wolną stronę.
Natomiast strona (węzeł Bdrzewa) w pliku indeksowym natomiast zawiera nagłówek zawierający informacje jaki węzeł jest rodzicem danego dziecka oraz ile rekordów jest już zajęte. Następnie występują na przemian numery węzłów potomnych, oraz rekordy, rekordy są posortowane według klucza.
W obu plikach wartości kluczy są wyrównywane do 20, lub 10 znaków, ponieważ maksymalna wartość unsigned long wynosi 20 cyfr, a unsigned int 10 cyfr. W obydwu przypadkach przed numerem klucza / innej wartości umieszczane są zera o potem sama wartość.
Typem rekordu był ciąg znaków o długości od 1 do 30, natomiast ze względu na to że rekordy sortowane są po wartości klucza, kryterium sortowania (porządek leksykograficzny) straciło ważność. Ciągi znaków są wyrównywane do 30 znaków poprzez wypełnienie ich znakami '\0'.
Rekordy w obydwu plikach są wyrównywane w celu ponownego wykorzystania miejsca, ponieważ aby ponownie wykorzystać miejsce musimy mieć "pustą stronę" to strony muszą być jednakowej długości nie zależnie od zawartości.

2.2) Mechanizm cachowania:

Aby zmniejszyć liczbę odczytów z pliku wykorzystany został mechanizm cachowania węzłów Bdrzewa, cache dla węzłów jest zaimplementowany jako stos, ze względu na to że algorytmy dodawania, usuwania oraz wyszukiwania są rekurencyjne, stos ten ma zawsze długość równą wysokości Bdrzewa + 1. Dziki takiej implementacji cacha można bez problemu wyświetlić całe Bdrzewo odczytując każdą stronę dokładnie raz.

Natomiast strony z danymi są cachowane tylko podczas sekwencyjnego wyświetlania pliku, w celu zmniejszenia liczby odczytów z pliku. Maksymalnie w pamięci mogą być 4 strony z danymi.

2.3) Ograniczenia i usprawnienia:

Podczas implementacji zauważyłem że gdy kompensujemy po usunięciu lub dodaniu, to lepiej jest najpierw uzupełnić węzeł obecny a potem dopiero rodzeństwo które kompensuje, ponieważ gdyby przy dodawaniu podzielić rekordy po równo to jeżeli oba węzły mają potomków to może zaistnieć sytuacja że więcej niż jednemu potomkowi trzeba zaktualizować rodzica, a w przypadku gdzie najpierw wypełniamy obecny węzeł a potem sąsiedni, taka aktualizacja występuje tylko raz.
Ograniczenie występuje głównie przy wyświetlaniu, ze względu na to że okno terminala ma ograniczoną wielkość, i gdy drzewo będzie zbyt duże to może się po prostu nie zmieścić.
Jeden dosyć poważny błąd implementacyjny to wyszukiwanie w węźle Bdrzewa załadowanego do pamięci odpowiedniego miejsca sekwencyjne zamiast bisekcją.
Jeszcze jednym problemem nie koniecznie jeżeli chodzi a logikę programu, ale o estetykę kodu jest duplikacja niektórych fragmentów kodu (głownie w klasie App, oraz trochę w BTree).

2.4) Opis klas:

- **BTree -** główny element, to tutaj dodaje i usuwa się elementy, klasa ta zawiera numer węzła korzenia, wysokość Bdrzewa, rząd Bdrzewa, oraz liczbę kluczy, a także liczbę zapisów i odczytów danej operacji, i dwie pomocnie flagi używane przy operacjach. Bdrzewo nie trzyma więcej informacji.
- **Displayer -** zajmuje się wyświetlaniem Bdrzewa oraz sekwencyjnym wyświetlaniem pliku z danymi.
- **Cache -** tutaj cachowane są węzły i strony z danymi.
- **App -** tutaj znajduje się główna pętla, która pozwala na wywoływanie funkcjonalności programu.
- **FileManager -** obsługuje pliku (odczyt, zapis konkretnej strony), zajmuje się odzyskiwaniem miejsca poprzez umieszczanie nowy węzłów lub rekordów danych w pierwszym wolnym miejscu.
- **DataPage -** składa się z ilości zajętych rekordów (unsigned int) oraz z jednego std::vector zawierającego rekordy pliku z danymi (RecordData), wektor ma długość 10 tyle ile wynosi długość strony.
- **Node(węzeł Bdrzewa) -** składa się z numeru rodzica (unsigned long), ilości zajętych rekordów (unsigned int), oraz dwóch std::vector jedn zawiera rekordy pliku indeksowego(Record Index), a drugi numery węzłów potomnych(unsigned long). Pierwszy z wektorów ma długość rząd Bdrzewa * 2, a drugi rząd Bdrzewa * 2 + 1.
- **RecordIndex -** składa się z klucza (unsigned long) i numeru strony w pliku z danymi (unsigned long).
- **RecordData -** składa się z klucza (unsigned long) i wartości (string).

Każda z 5 pierwszych klas jest singletonem, pozostałe 4 to zwykłe struktury do agregacji danych.

3. Eksperyment:

W ramach eksperymentu sprawdziłem ile będzie wynosić całkowita jak i średnia ilość operacji dyskowych dla liczby rekordów: 20, 50, 100, 200, 500, 1000, 10000, dla Bdrzewa rzędu; 2, 5, 10, 50, 100. W tym eksperymencie najpierw budujemy w losowej kolejności Bdrzewo (dodajemy N różnych rekordów), a następnie usuwamy całe Bdrzewo również w losowej kolejności

Wyniki prezentują się następująco:

- Bdrzewo rzędu 2:

Ilość rekordów	Całkowita ilość odczytów	Całkowita ilość zapisów	Średnia ilość odczytów przy wyszukiwaniu	Średnia ilość odczytów przy dodawaniu	Średnia ilość zapisów przy dodawaniu	Średnia ilość odczytów przy usuwaniu	Średnia ilość zapisów przy usuwaniu
20	97	85	1.75	0.85	1.9	0.5	2.35
50	360	268	2.43	1.14	2.22	1.2	3.14
100	867	546	2.845	1.72	2.81	1.26	2.65
200	1903	983	3.48	1.68	2.7	0.875	2.215
500	5385	2391	4.078	1.73	2.824	0.884	1.958
1000	11868	4761	4.576	1.666	2.729	1.05	2.032
10000	150150	37103	6.37445	1.6967	2.7826	0.5694	1.9277

- Bdrzewo rzędu 5:

Ilość rekordów	Całkowita ilość odczytów	Całkowita ilość zapisów	Średnia ilość odczytów przy wyszukiwaniu	Średnia ilość odczytów przy dodawaniu	Średnia ilość zapisów przy dodawaniu	Średnia ilość odczytów przy usuwaniu	Średnia ilość zapisów przy usuwaniu
20	65	51	1.425	0.2	1.2	0.2	1.35
50	210	156	1.76	0.38	1.48	0.3	1.64
100	488	360	1.845	0.59	1.64	0.6	1.96
200	1290	798	2.4275	1.075	2.105	0.52	1.885
500	3649	1884	2.818	1.008	2.024	0.654	1.744
1000	7647	3988	2.8755	1.1154	2.144	0.742	1.844
10000	98896	35283	4.0509	1.1919	2.2252	0.5959	1.3031

- Bdrzewo rzędu 10:

Ilość rekordów	Całkowita ilość odczytów	Całkowita ilość zapisów	Średnia ilość odczytów przy wyszukiwaniu	Średnia ilość odczytów przy dodawaniu	Średnia ilość zapisów przy dodawaniu	Średnia ilość odczytów przy usuwaniu	Średnia ilość zapisów przy usuwaniu
20	40	40	0.975	0.05	1	0	1
50	173	127	1.58	0.12	1.12	0.18	1.42
100	464	340	1.765	0.72	1.78	0.39	1.62
200	952	676	1.8875	0.665	1.69	0.33	1.69
500	2940	1831	2.279	0.886	1.876	0.436	1.768
1000	6737	3665	2.648	0.958	1.952	0.483	1.713
10000	84404	33741	3.4732	1.0291	2.022	0.4649	1.3521

- Bdrzewo rzędu 50:

Ilość rekordów	Całkowita ilość odczytów	Całkowita ilość zapisów	Średnia ilość odczytów przy wyszukiwaniu	Średnia ilość odczytów przy dodawaniu	Średnia ilość zapisów przy dodawaniu	Średnia ilość odczytów przy usuwaniu	Średnia ilość zapisów przy usuwaniu

20	40	40	0.975	0.05	1	0	1
50	100	100	0.99	0.02	1	0	1
100	200	200	0.995	0.01	1	0	1
200	637	480	1.495	0.165	1.31	0.03	1.09
500	2146	1492	1.791	0.48	1.568	0.23	1.416
1000	4555	3206	1.896	0.403	1.45	0.36	1.756
10000	56737	35127	2.2243	0.9045	1.9042	0.3206	1.6085

- Bdrzewo rzędu 100:

Ilość rekordów	Całkowita ilość odczytów	Całkowita ilość zapisów	Średnia ilość odczytów przy wyszukiwaniu	Średnia ilość odczytów przy dodawaniu	Średnia ilość zapisów przy dodawaniu	Średnia ilość odczytów przy usuwaniu	Średnia ilość zapisów przy usuwaniu
20	40	40	0.975	0.05	1	0	1
50	100	100	0.99	0.02	1	0	1
100	200	200	0.995	0.01	1	0	1
200	400	400	0.9975	0.005	1	0	1
500	1651	1090	1.597	0.038	1.064	0.07	1.116
1000	4293	2947	1.789	0.536	1.624	0.161	1.323
10000	52561	36271	1.97795	0.9452	1.9524	0.355	1.6747

Wnioski z eksperymentu:

Widać że dla Bdrzew wyższego rzędu zarówna średnia jak i całkowita liczba odczytów i zapisów rośnie wolniej niż dla tych o niższym rzędzie. Co się zgadza ponieważ Bdrzewa mają być niskie ale za to bardzo szerokie w celu zmniejszenia ilości dostępów do dysku.
Można też zauważyć że jeżeli rekordów jest mniej niż wynosi rząd Bdrzewa to ilość odczytów i zapisów wynosi tyle ile było operacji. Ilość odczytów mniejsza od 1 przy szukaniu wynika z tego że gdy Bdrzewo jest puste to nie ma jak przeczytać pierwszej strony.