Sprawozdanie Struktury Baz Danych Projekt 1

Sebastian Kwaśniak

2024-12-09

Wprowadzenie

Wylosowane przeze mnie typ rekordu to:

29. File records: Right circular cylinders - the radius of the base and the height of the cylinder. Sorting by volume.

Implementacja w języku C++. Przyjąłem, że jeden rekord jest podzielony na cztery liczby, rozmiar rekordu to 16 bajtów (4 bajty dla klucza, 4 bajty dla podstawy, 4 bajty dla wysokości, 4 bajty dla wskaźnika).

Zastosowałem optymalizację polegającą na nie alokowaniu całego obszaru indeksu na raz, rozmiar dostosowuje się do zajętości miejsca. Nie ma to wpływu na algorytm, a głównie na zajętość pamięci.

Opis struktury kodu

Kod został głównie przeniesiony z projektu 1, w którym:

- Klasa `Tape` zajmuje się obsługą zarówno głównej taśmy oraz przepełnienia
- Klasa `Index` zajmuje się trzymaniem indeksów
- Klasa `Cylinder` implementuje typ rekordu

Zasada działania

Łańcuch przepełnień

Łańcuch przepełnień działa na zasadzie podobnej do struktury `linked list`, gdzie w moim wypadku, wskaźnikami jest offset w pliku dodany o 1 (wartość 0 jest u mnie wartością specjalną - wskaźnik nie istnieje).

Insert

Gdy próbujemy umieścić nowy rekord w taśmie, najpierw przeszukujemy index. Index posiada w sobie informacje na której stronie zaczynają się poszczególne klucze, dlatego wystarczy że znajdziemy poprzednika od pierwszego większego znalezionego klucza od tego który chcemy wstawić. Mając stronę, nie musimy przeszukiwać całego pliku a tylko skoczyć do wybranej strony i odczytać ją. W niej szukamy poprzednika i umieszczamy go zaraz po poprzedniku. Jeśli nie ma miejsca w głównej taśme, to umieszczamy go w łańcuchu przepełnień.

Reorganise

1. Tworzymy dwa tymczasowe pliki: dodatkową taśmę i indeks, ze wzoru niżej wyliczamy liczbę stron głównych, gdzie N, V - liczba rekordów w taśme głównej i przepełnieniu, b - liczba rekordów danch na stronę, α - średnie zapełnienie strony po reorganizacji pliku.

$$\lceil \frac{N+V}{b*\alpha} \rceil \tag{1}$$

2. Przechodizmy kolejno przez rekordy zgodnie z rośnięciem kluczy i umieszczamy je na kolejnyhc stronach (respektując α)
3. Usuwamy stare pliki i zamieniamy tymczasowe na nie.

Prezentacja wyników programu

Po włączeniu programu użytkownikowi zostają pokazane wszystkie możliwości:

TODO: przekopiować output tutaj

Głównie są to 2 komendy:

- `insert <key> <base> <height>` - dodanie rekordu do bazy
- `file` - wczytanie komend z pliku (domyślnie plik z nazwą `input.txt`)
- `dump` - wypisanie całej bazy
- `reorganise` - reorganizacja całej bazy

Przykładowe wyjście z programu:

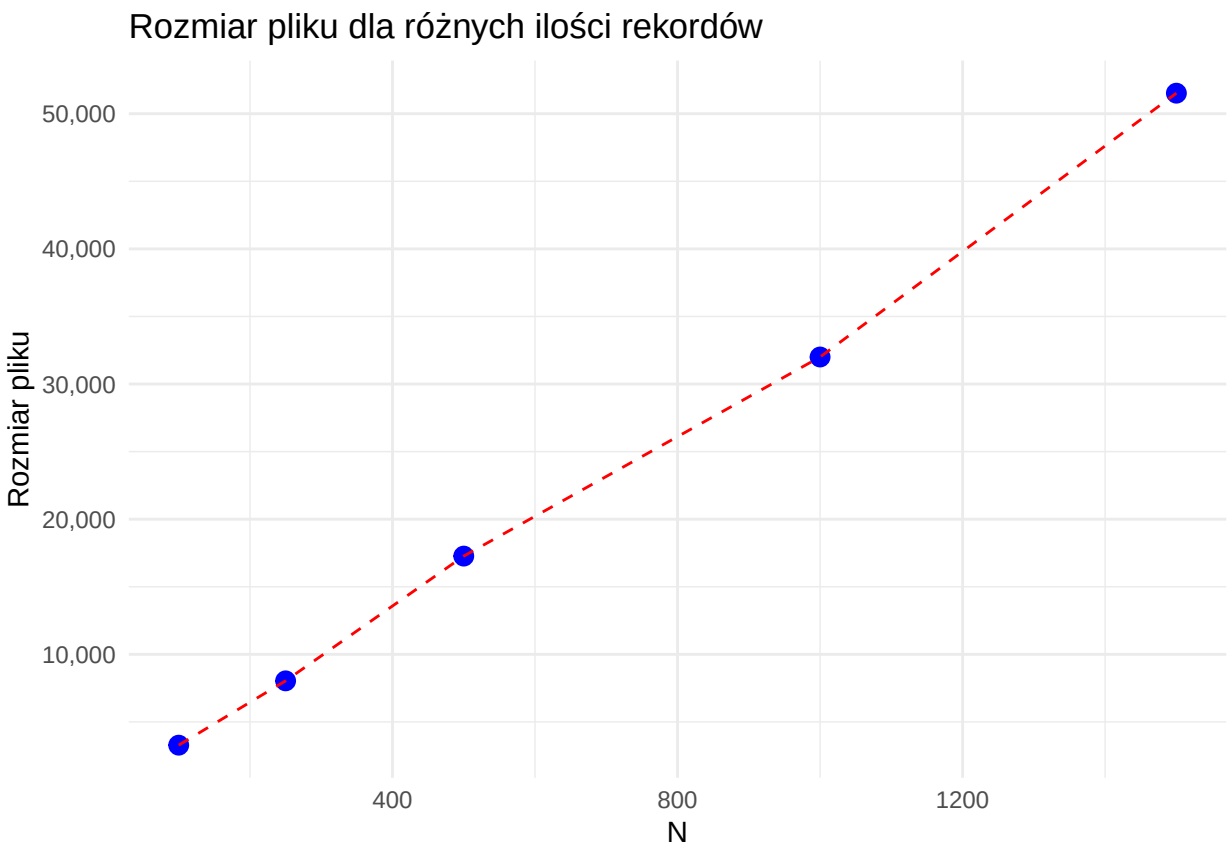
TODO: przykładowe wyjście

Eksperyment

Przeprowadzono eksperymenty na zasadzie wczytywania danych z pliku, gdzie zostało wygenerowanych 1000 operacji `insert`.

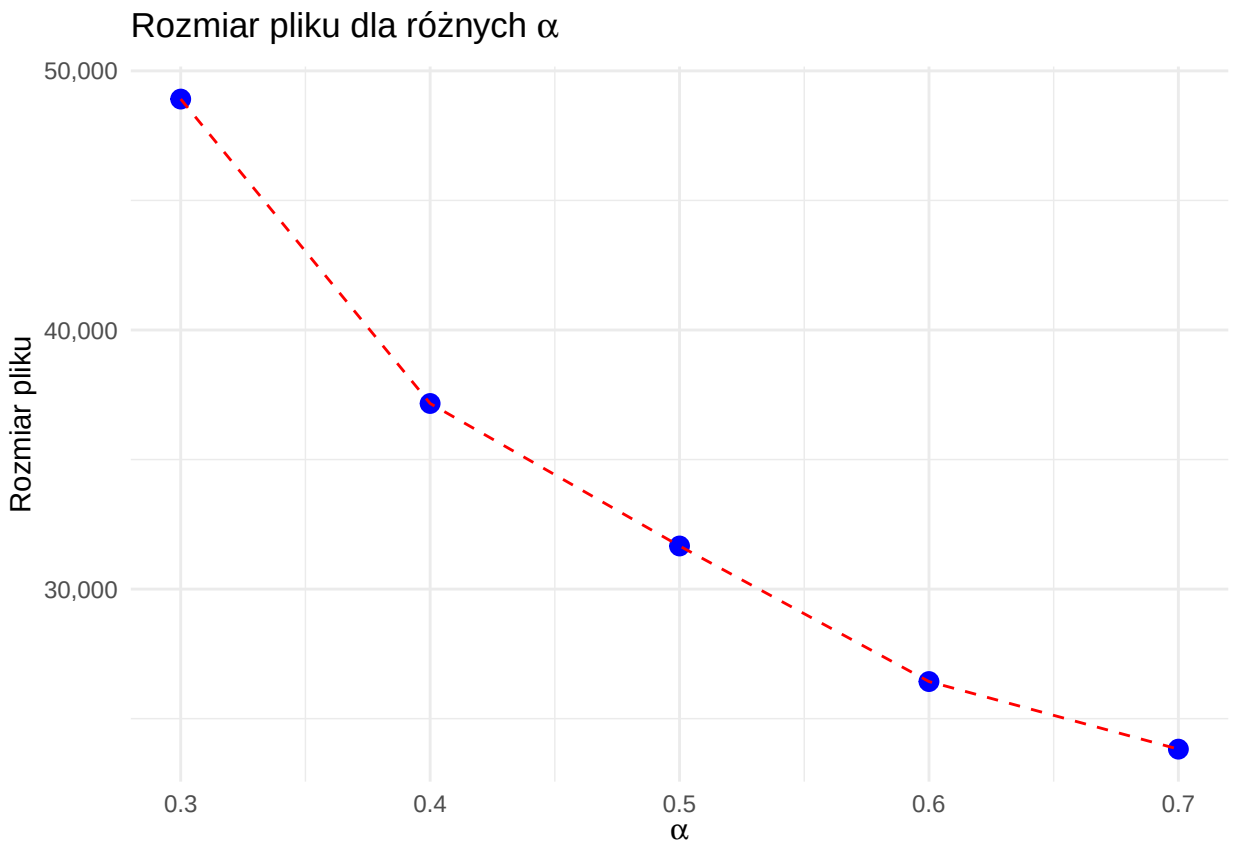
- Ilość rekordów przetrzymywanych w głównej taśmie przyjąłem jako 4
- Rozmiar jednego rekordu w głównej taśmie lub obszarze przepełnienia to 16 bajtów
- Rozmiar jednego rekordu w indeksie to 16 bajtów

N	rozmiar indeksu	rozmiar obszaru głównego	rozmiar obszaru przepełnienia	Suma
100	336	2688	256	3280
250	808	6464	768	8040
500	1856	14838	576	17270
1000	3200	25600	3200	32000
1500	5504	44032	1984	51520



Jak widzimy rozmiar plików rośnie liniowo. Do tego widać zależność rośnięcia plików blokowo.

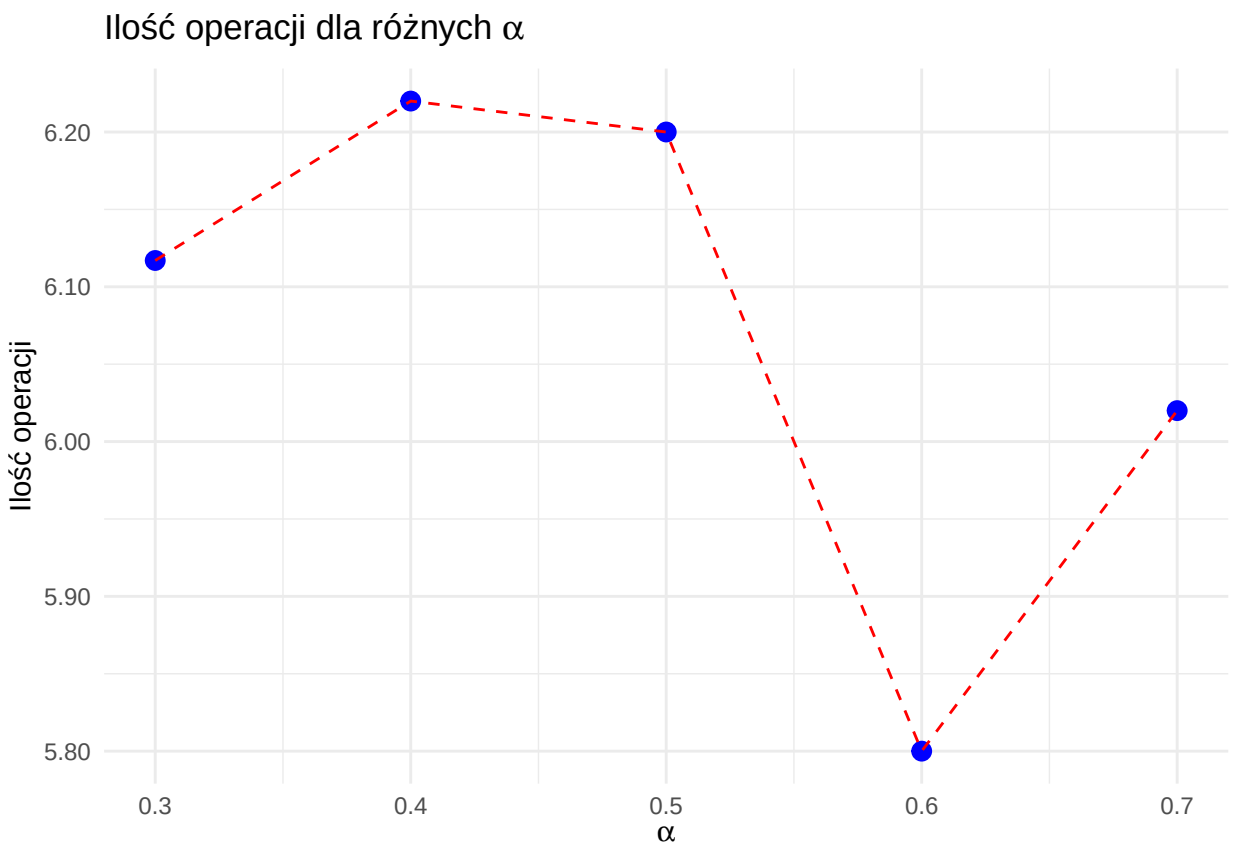
alpha	rozmiar indeksu	rozmiar obszaru głównego	rozmiar obszaru przepełnienia	Suma
0.3	2192	43840	2880	48912
0.4	1164	33280	2720	37164
0.5	1424	28480	1760	31664
0.6	1152	23040	2240	26432
0.7	1104	22080	640	23824



Widzimy, że spadek rozmiaru pliku jes gwałtowny pomiędzy $\alpha = [0.3; 0.4]$ oraz później tendencja jest w mniejszym stopniu. To znaczy, że reorganizacja tworzy mniej stron, przez co będą pełniejsze, ale reorganizacja będzie musiała następować częściej.

W następnym eksperymencie, dla lepszego zobrazowania danych zmieniłem wielkość bloku na 10 rekordów.

alpha	śr. odczyty	śr. zapisy	Suma
0.3	4.15	1.967	6.117
0.4	4.23	1.990	6.220
0.5	4.21	1.990	6.200
0.6	3.85	1.950	5.800
0.7	4.04	1.980	6.020



Widzimy że operacje oscylują w okolicy liczby 6. Zapisy zawsze są blisko 2, a odczyty blisko 4. Ilość odczytów głównie zależy od tego jak bardzo porozrzucane są następne elementy w łańcuchu przepełnień, na co głównie wpływ ma duża losowość.

Podsumowanie

Projekt pozwolił na zrozumienie organizacji indeksowej pliku i dlaczego indeksowanie pozwala znacząco przyspieszyć wszystkie typy operacji, oraz korzyści płynące z posiadania indeksu. Na bazie eksperymentów, można zuważyć zależności pomiędzy α oraz wielkością pliku i ilością operacji.