Sprawozdanie Struktury Baz Danych Projekt 1

Sebastian Kwaśniak

2024-11-24

Wprowadzenie

Zaimplementowany przeze mnie algorytm to sortowanie przez scalanie w schemacie 2+1. Wylosowane przeze mnie typy rekordów 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 dwie liczby, rozmiar rekordu to 8 bajtów (4 bajty dla podstawy, 4 bajty dla wysokości). Rozmiar strony ustaliłem na 32, a potem na 320.

Sortowanie przez scalanie

Sortowanie przez scalanie jest prostym typem sortowania podzielonym na dwa etapy: `distribute` oraz `merge`. Dystrybucja polega na rozłożeniu głównej taśmy na dwie pomocnicze, według schematu:

1. Weź liczbę
2. Sprawdź czy poprzednia była większa
3. Jak tak, wrzuć ją do aktualnie wybranej taśmy; jak nie, zmień wybraną taśmę i wrzuć do niej

Scalanie polega na schemacie:

1. Weź pierwsze rekordy z taśm pomocniczych
2. Wybierz mniejszy z dwóch rekordów
3. Wpisz go do głównej taśmy
4. Iteruj taśmę z której był wybrany rekord
5. Powtarzaj 2-4 aż do skończenia się jednej z taśm
6. Jeśli coś zostało w jakiejś taśmie, dopisz to na koniec głównej taśmy

Algorytm pozwala na posortowanie danych o rozmiarze większym niż rozmiar pamięci operacyjnej. Zużywamy:

$$S_{\text{dysk}} = NR \tag{1}$$
$$S_{\text{ram}} = B \tag{2}$$

N - ilość rekordów; R - rozmiar rekordów w bajtach; B - rozmiar strony

Czas algorytmu prezentowany jest w operacjach dyskowych, na wykładzie zostały pokazane następujące złożoności:

$$T_{\text{pes}} = \frac{4N\lceil \log_2(N) \rceil}{b} T_{\text{avg}} = \frac{4N\lceil \log_2(N) - 1 \rceil}{b} \tag{3}$$

gdzie $b = \frac{B}{R}$.

Specyfikacja formatu pliku

Plik ma prosty format 8 bajtowych rekordów. Cztery pierwsze bajty to podstawa walca, cztery następne bajty to wysokość walca. Bajty te reprezentują ASCII cyfr zapisanych, aby uprościć analizę i odczyt. Robi to ograniczenie liczb od 1 do 9999, lecz nie jest problemem zamiana długości rekordu i rozszerzeniu ilości bajtów do zapisania tych liczb.

Zapis

Zapis do pliku odbywa się w prosty sposób, dodając zera na początku liczby aby później łatwiej było ją przekształcić w programie.

Odczyt

Odczyt działa na zasadzie wzięcia pierwszych 4 bajtów, zamiany ich do liczby, wczytaniu następnych 4 bajtów i również zamiany do liczby.

Sposób prezentacji wyników

W programie mamy następujące komendy:

```
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Commands:
* help - shows this help
* dump - dump mainTape file, by volume
* random <N> - generate tape with random N records
* file - read tape from file (default name: input.txt)
* manual - generate tape from user input
* * * * * * * * * * * * * * * * * * * * * * * * *
```

Mamy trzy główne sposoby na wprowadzenie danych:

- random: generuje N losowych rekordów
- file: zczytuje plik input.txt do wprowadzenia rekordów
- manual: wczytuje input od użytkownika z klawiatury

Mamy także funkcje pomocniczą `dump` która wypisuje wszystkie rekordy z głównej taśmy w postaci objętości (wokół której w zadaniu było sortowanie).

Przykładowy plik input.txt (5 rekordów):

```
00050005000400040003000300002000200010001
```

Plik załadowany, po czym wprowadzona komenda `dump`:

```
> file
> dump
3.14159
25.1327
84.823
201.062
392.699
```

Eksperyment

Implementacja

Wyniki

Wzór zastosowany do obliczenia operacji dyskowych i faz:

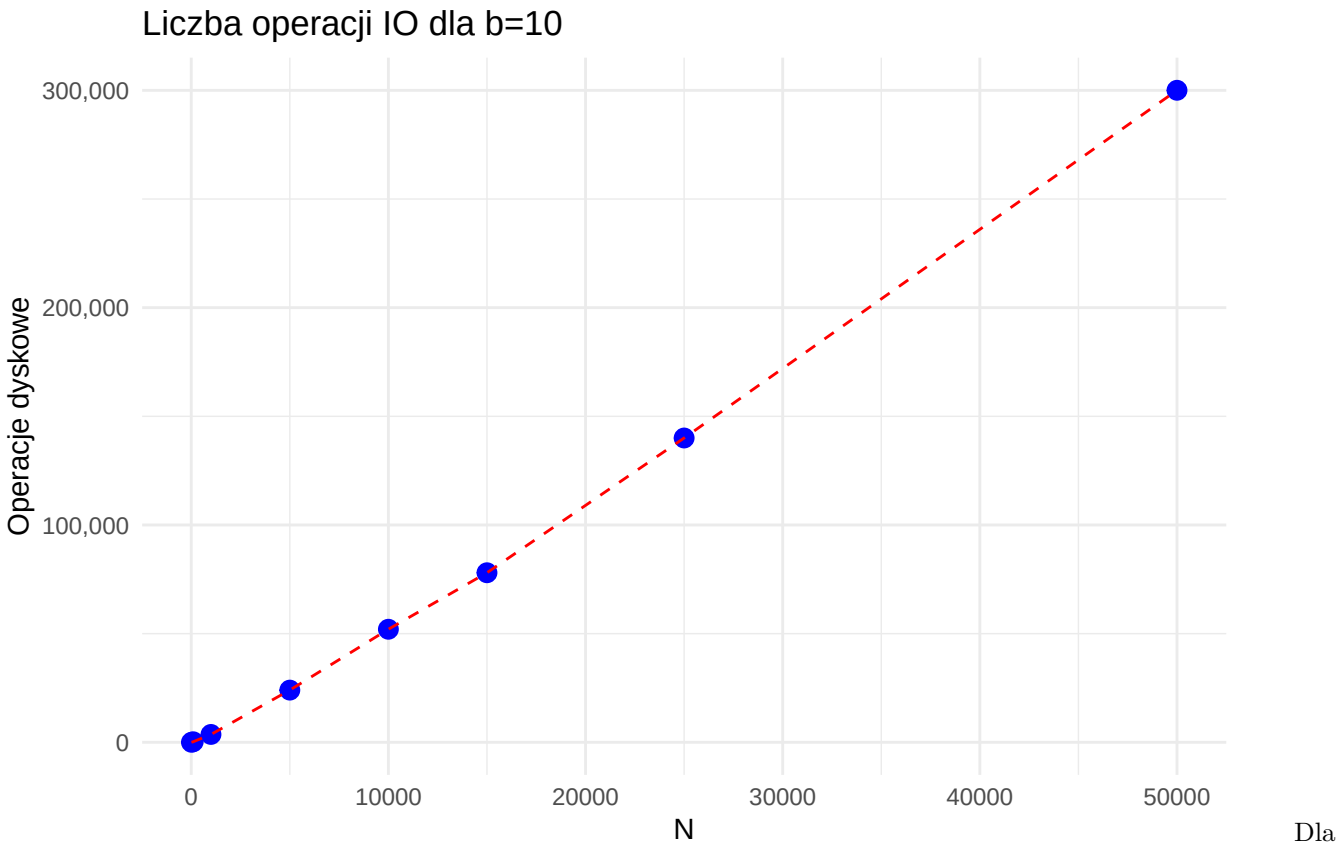
$$T = \frac{4N\lceil \log_2 r \rceil}{b} \tag{4}$$

$$\text{fazy} = \lceil \log_2 r \rceil \tag{5}$$

gdzie r - liczba serii

Dla wielkości strony 10 rekordów:

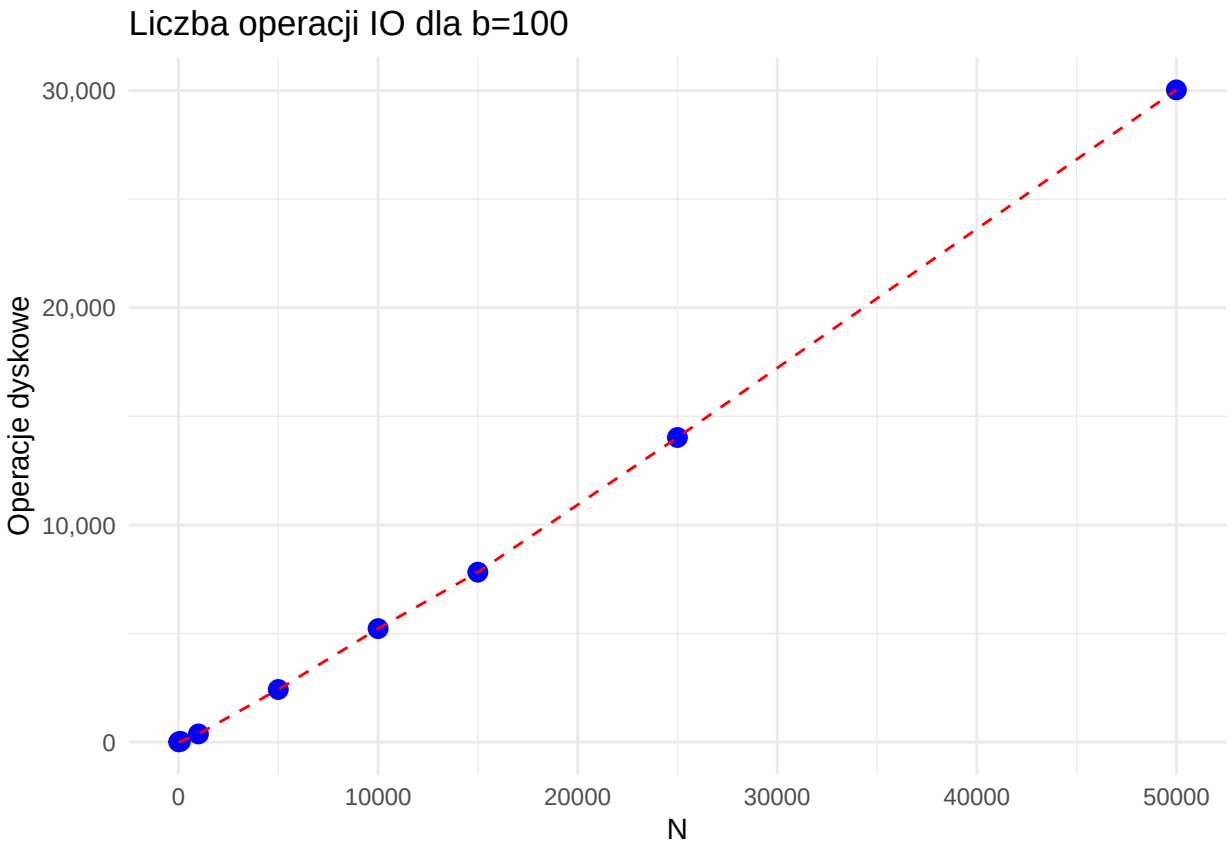
N	Serie	IO	Fazy	Zapisy	Odczyty	Teor.IO	Teor.Fazy
10	5	12	2	6	6	12	3
100	55	250	6	125	125	240	6
1000	505	3616	9	1808	1808	3600	9
5000	2544	24018	12	12009	12009	24000	12
10000	5000	52022	13	26011	26011	52000	13
15000	7534	78022	13	39011	39011	78000	13
25000	12501	140024	14	70012	70012	140000	14
50000	25001	300028	15	150014	150014	300000	15



Dla wielkości strony 100 rekordów:

N	Serie	IO	Fazy	Zapisy	Odczyty	Teor.IO	Teor.Fazy
10	5	12	3	6	6	12	3
100	52	36	6	18	18	240	6
1000	498	378	9	189	189	3600	9
5000	2520	2424	12	1212	1212	24000	12
10000	4991	5226	13	2613	2613	52000	13
15000	7552	7826	13	3913	3913	78000	13

N	Serie	IO	Fazy	Zapisy	Odczyty	Teor.IO	Teor.Fazy
25000	12489	14026	14	7013	7013	140000	14
50000	25014	30030	15	15015	15015	300000	15

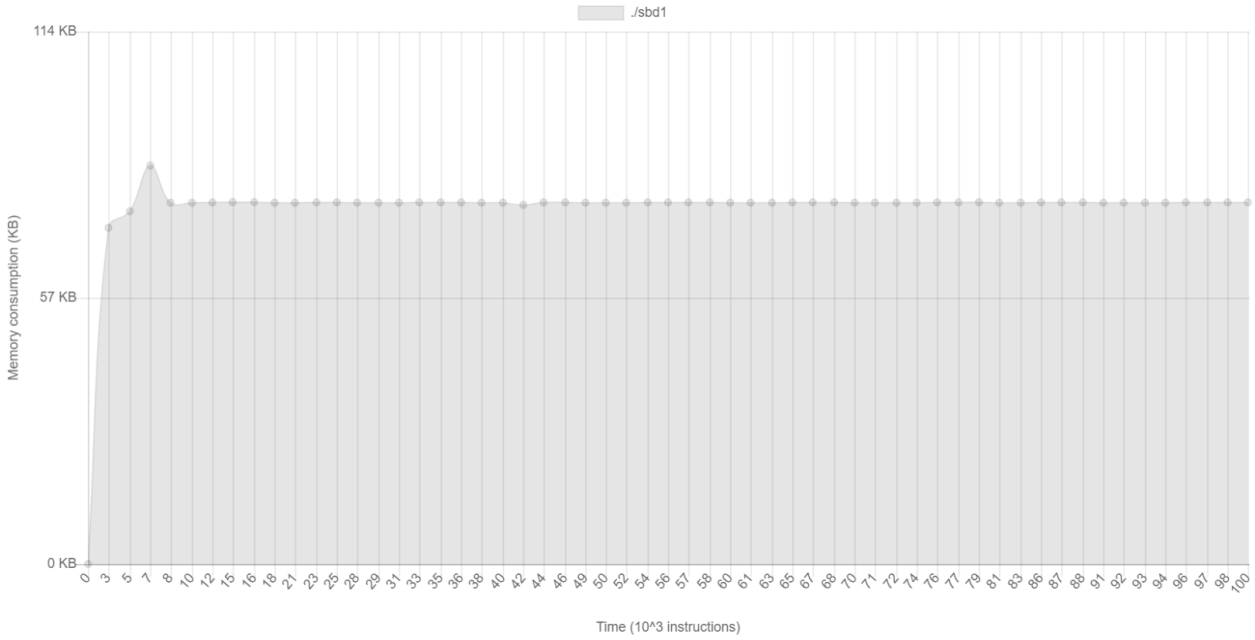


Jak widać, dla większej ilości strony, tym rzadziej ją wymieniamy, przez co znacznie spadają ilości operacji dyskowych. Gdy N jest mniejsze niż rozmiar strony, to nie ma różnicy spowodowanej rozmiarem bloków, bo do zapisu wszystkich danych zostanie wyokrzystana tylko jedna strona na taśmę.

Zużycie pamięci

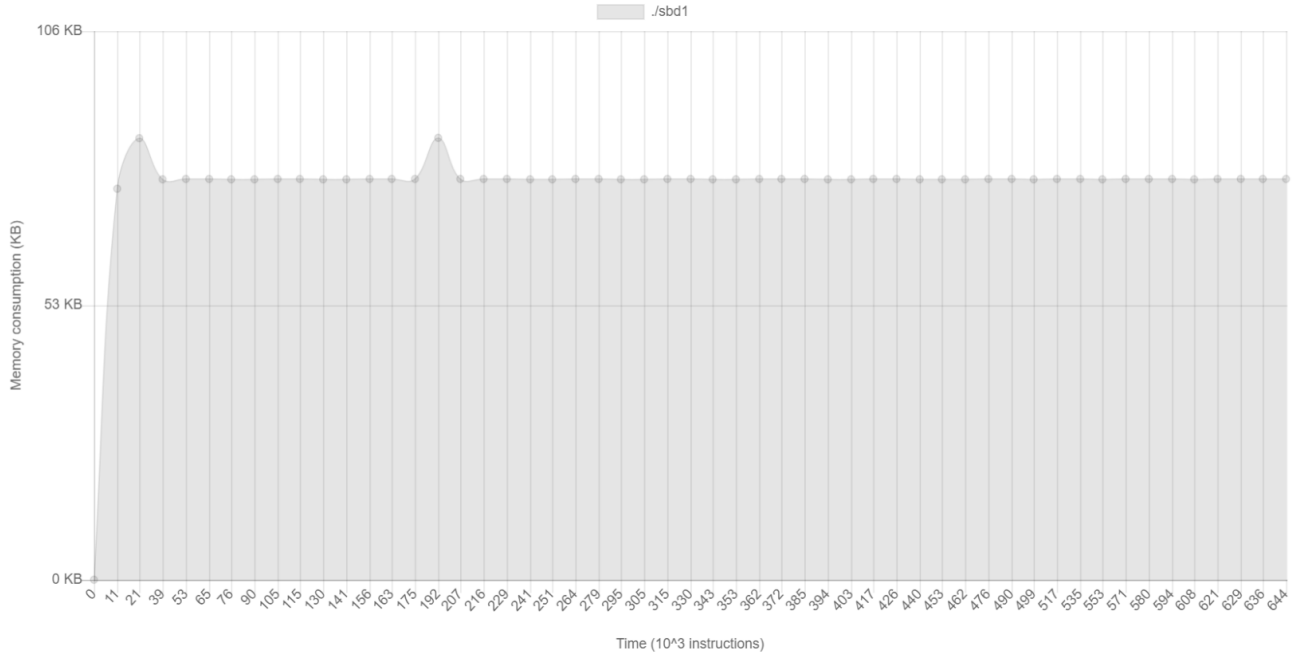
Zużywana pamięć w programie jest stała, jak można zaobserwować na poniższych danych z programu `valgrind --tool=massif`.

Dla 1000 rekordów:



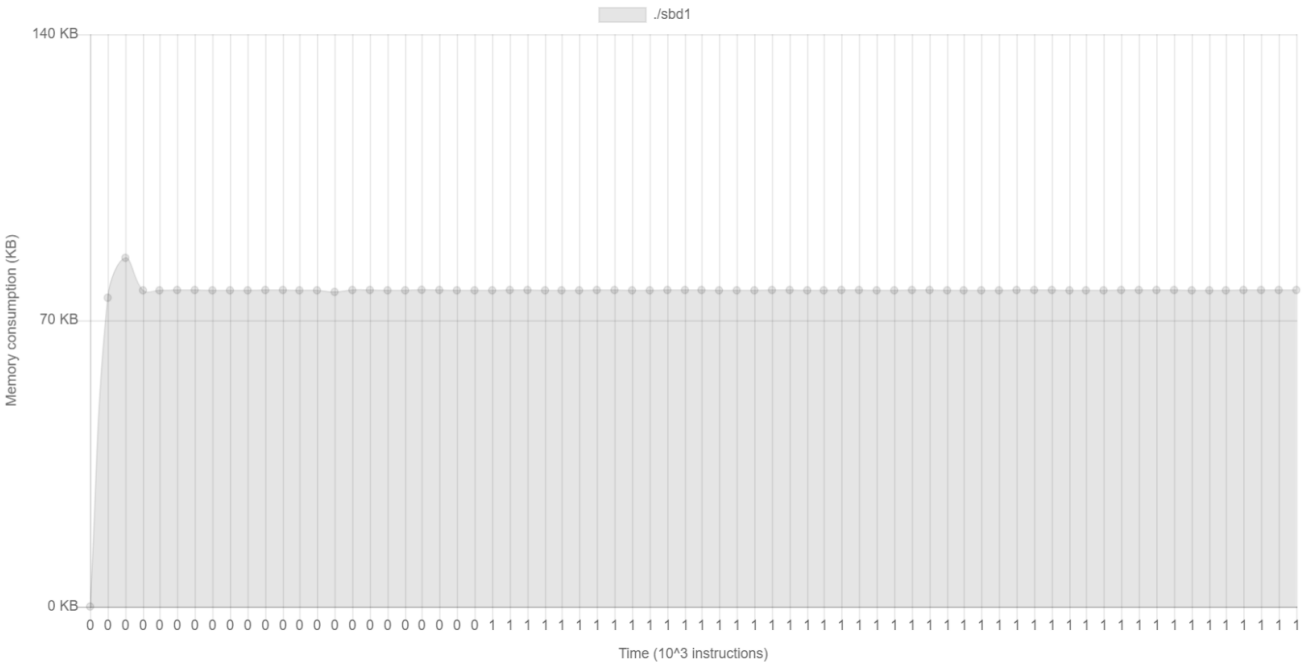
Rys. 1: Mała ilość rekordów

Dla 5000 rekordów:



Rys. 2: Średnia ilość rekordów

Dla 10000 rekordów:



Rys. 3: Duża ilość rekordów

Jak widać, ilość zużytej pamięci operacyjnej nie przekracza 80KB.

Podsumowanie

Projekt pozwolił na zrozumienie jak wielkie pliki są odczytywane z dysku, oraz jak zaimplementować w prosty sposób własny sposób na zapis danych tak, aby nie zużywać za dużo pamięci operacyjnej. Wyniki eksperymentu są zgodne z oczekiwaniami rozważanymi na poziomie teoretycznym jeśli chodzi o ilość operacji dyskowych, oraz zużycie pamięci było stałe, więc zwiększanie liczby rekordów nie zwiększa ilości zużywanej pamięci operacyjnej. Liczba operacji IO zwiększa się proporcjonalnie do zwiększania współczynnika blokowania, ale współczynnik nie ma wpływu na ilość faz algorytmu.