Sortowanie przez scalanie naturalne

Anna Berkowska

25 listopada 2024

Wstęp

Do zrealizowania projektu wybrałam algorytm do sortowania pliku przez scalanie naturalne na taśmach 2+1. Zaimplementowałam go przy użyciu języka C. Wylosowany przeze mnie rekord zawiera: masę, ciepło właściwe oraz różnicę temperatur. Rekordy te mają być uporządkowanie wg oddanego/pobranego ciepła, które obliczam za pomocą wzoru:

$$Q_{\text{odczuwalne}} = mc\Delta T$$

$Q_{\text{odczuwalne}}$ - ciepło odczuwalne

m - masa substancji

c - ciepło właściwe

ΔT - różnica temperatur

Scalanie naturalne

Sortowanie przez scalanie naturalne przy pomocy użyciu trzech taśm, gdzie dwie z nich są pomocnicze, polega na powtarzaniu kroku dystrybucji i scalania, aż do momentu, kiedy główna taśma jest posortowana. W procesie dystrybucji zaczynamy od jednej z taśm pomocniczych i przepisujemy do niej niemalejący ciąg liczb z taśmy głównej. Kiedy ciąg taki zostanie przerwany, to dalsze liczby przepisujemy do drugiej taśmy pomocniczej. Powtarzamy ten proces aż do momentu, kiedy nie ma już więcej rekordów do przepisania z taśmy głównej. Proces scalania dwóch taśm pomocniczych w jedną taśmę główną, polega na łączeniu kolejnych niemalejących ciągów. Ostatecznie powtarzając te działania, ciągi niemalejące będą się zwiększać, a końcowo otrzymamy jeden ciąg - posortowaną taśmę główną.

Szczegóły implementacji

Format pliku

Plik wejściowy zawiera N rekordów w postaci tekstowej. Każdy rekord składa się z trzech liczb całkowitych - każda z liczb ma stałą szerokość (ilość znaków typu char). Dla poprawnego działania programu w eksperymencie również używam trzech cyfr dla każdej z liczb, żeby zapobiec przepełnieniu. W podanym poniżej przykładzie mamy osiem rekordow, gdzie każda liczba składa się z trzech cyfr (3 bajtów).

008008008007007007006006006005005005004004004003003003002002002001001001

Zapis do pliku

Zapis do pliku następuje poprzez ustawienie się na odpowiednim bajcie w pliku przy użyciu funkcji `fseek()`, dzięki offsetowi strony, i zapis całego bloku (strony) do pliku taśmy. Zwiększany jest wtedy licznik zapisów do taśmy.

Odczyt z pliku

Odczyt z pliku następuje przez odczytanie całego bloku (strony) do pamięci programu z danego pliku taśmy. Strona ta pobierana jest dzięki użyciu `fseek()` i podaniu offsetu strony. Zwiększany jest wtedy licznik odczytów z taśmy.

Działanie programu

Po uruchomieniu programu wyświetlane są metody ładowania danych wejściowych do programu. Do wyboru mamy załadowanie rekordów z pliku, wpisanie ich ręcznie z klawiatury lub wylosowanie podanej przez nas liczby rekordów. Kiedy wybierzemy jedną z opcji i podamy odpowiednie parametry, program wykona sortowanie na danych wejściowych metodą scalania i wypisze stan pliku taśmy głównej (wartości obliczone z rekordów w kolejności, w jakiej znajdują się na taśmie) przed i po sortowaniu, a także po każdej fazie sortowania. Na koniec wypisane zostaną również wartości: liczba odczytów, zapisów do plików, ich suma, ilość faz oraz serii.

Przykład działania programu:

```
1. Load test input from file
2. Load test input from keyboard
3. Load test input generated randomly
4. Exit
> 3
Input number of records to generate:
> 5
___________TAPE1_BEFORE_SORTING___________
--------------------RECORD--------------------
Sensible heat:                       26799840
--------------------RECORD--------------------
Sensible heat:                      163534140
--------------------RECORD--------------------
Sensible heat:                       16982784
--------------------RECORD--------------------
Sensible heat:                      696833760
--------------------RECORD--------------------
Sensible heat:                       65848014
___________TAPE1_AFTER_PHASE_01___________
--------------------RECORD--------------------
Sensible heat:                       16982784
--------------------RECORD--------------------
Sensible heat:                       26799840
--------------------RECORD--------------------
Sensible heat:                      163534140
--------------------RECORD--------------------
Sensible heat:                      696833760
--------------------RECORD--------------------
Sensible heat:                       65848014
--------------------STATUS--------------------
Number of writes:                           6
Number of reads:                            6
Number of writes + reads:                  12
Number of phases:                           2
Series:                                     2
___________TAPE1_AFTER__SORTING___________
--------------------RECORD--------------------
Sensible heat:                       16982784
--------------------RECORD--------------------
Sensible heat:                       26799840
--------------------RECORD--------------------
Sensible heat:                       65848014
--------------------RECORD--------------------
Sensible heat:                      163534140
--------------------RECORD--------------------
Sensible heat:                      696833760
```

Eksperyment

Na potrzeby projektu przeprowadziłam również eksperyment na sortowaniu przez scalanie naturalne. Rekordy losowane są przy użyciu bibliotek `stdlib.h` oraz `time.h`, gdzie używam `srand(time(NULL))` do ustawienia punktu startowego generatora pseudolosowego i losuję liczby przy pomocy `rand()`.

Do obliczenia potrzebnych wartości teoretycznych używam wzorów:

$$M_{O+Z} = \frac{4N\lceil log_2 r \rceil}{b}$$
$$M_F = \lceil log_2 r \rceil$$

M_{O+Z} - maksymalna liczba zapisów i odczytów

M_F - maksymalna liczba faz

N - liczba rekordów

r - liczba serii

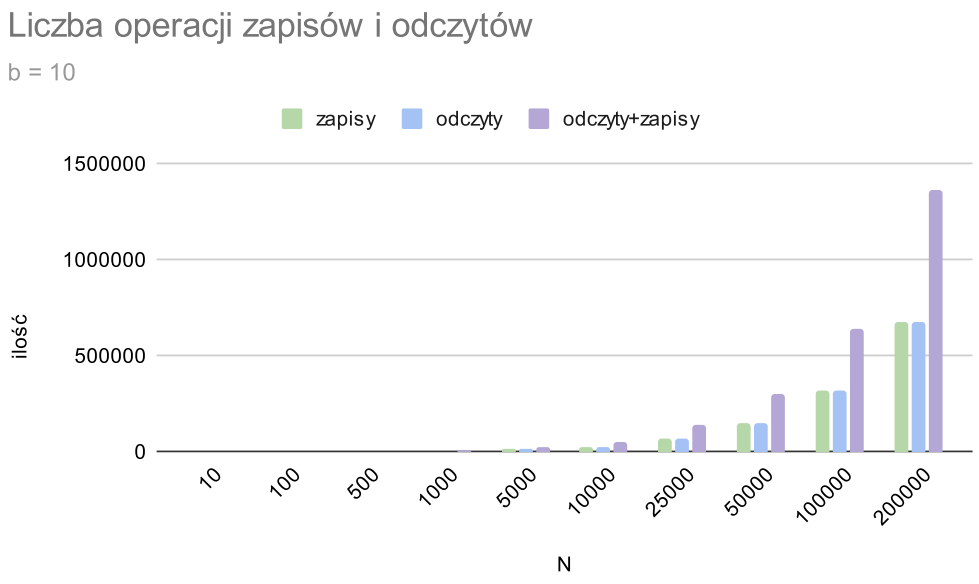
b - ilość rekordów w bloku (stronie)

Eksperyment przeprowadzany jest dla b=10 i b=100.

Wyniki liczbowe

Tabela 1: Tabela operacji dla b=10

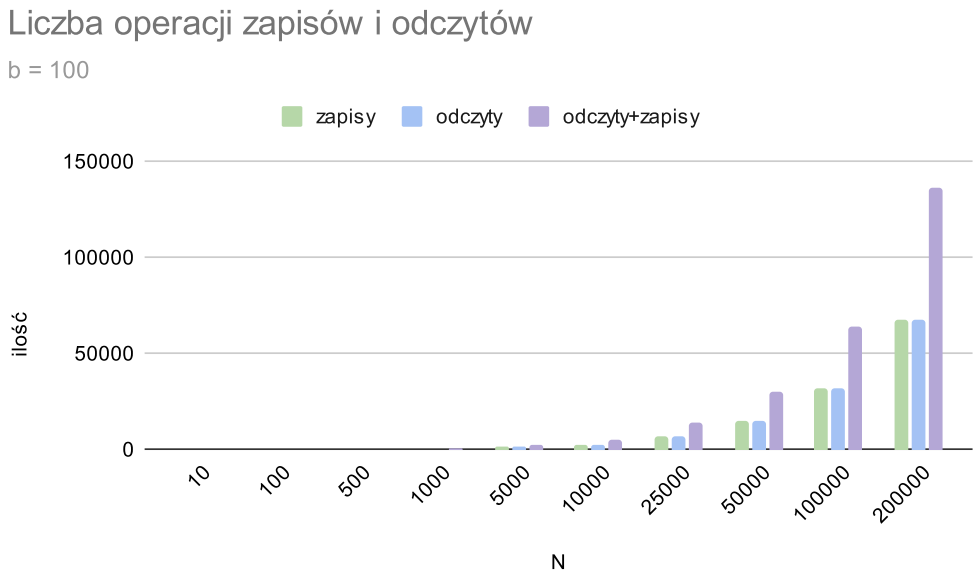
N	**Liczba serii**	**Zapisy**	**Odczyty**	**O+Z**	M_{O+Z}	**Liczba Faz**	M_F
10	5	9	9	18	12	3	3
100	50	123	123	246	240	6	6
500	244	808	808	1616	1600	8	8
1000	489	1809	1809	3618	3600	9	9
5000	2486	12012	12012	24024	24000	12	12
10000	5004	26010	26010	52020	52000	13	13
25000	12558	70010	70010	140020	140000	14	14
50000	25081	150013	150013	300026	300000	15	15
100000	50015	320015	320015	640030	640000	16	16
200000	100101	680014	680014	1360028	1360000	17	17



Rys. 1: Liczba operacji zapisów i odczytów dla b=10

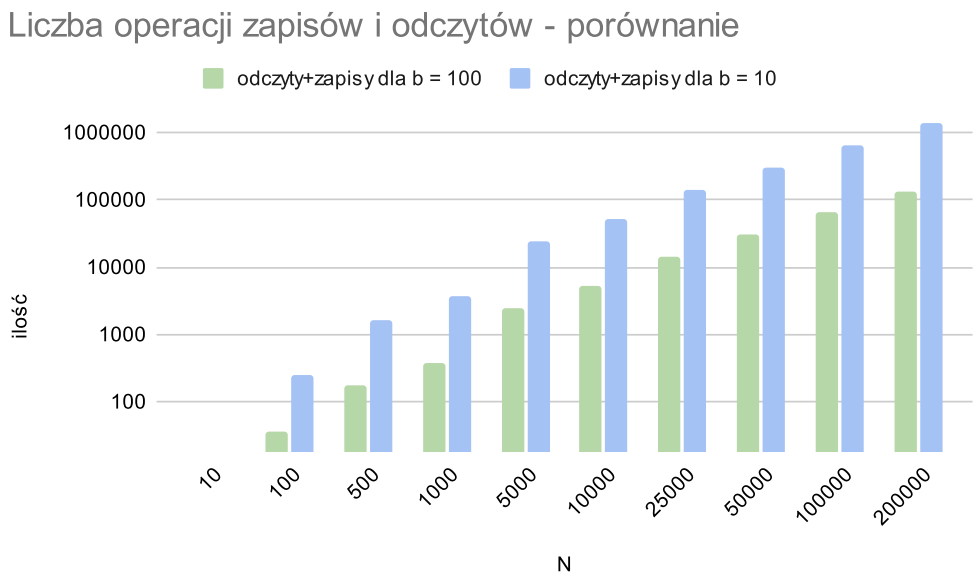
Tabela 2: Tabela operacji dla b=100

N	**Liczba serii**	**Zapisy**	**Odczyty**	**O+Z**	M_{O+Z}	**Liczba Faz**	M_F
10	5	9	9	18	1.2	3	3
100	53	18	18	36	24	6	6
500	244	88	88	176	160	8	8
1000	509	189	189	378	360	9	9
5000	2480	1212	1212	2424	2400	12	12
10000	5041	2613	2613	5226	5200	13	13
25000	12507	7013	7013	14026	14000	14	14
50000	25045	15015	15015	30030	30000	15	15
100000	50105	32016	32016	64032	64000	16	16
200000	99929	68017	68017	136034	136000	17	17



Rys. 2: Liczba operacji zapisów i odczytów dla b=100

Liczba faz jest identyczna do teoretycznego M_F. Łączna ilość operacji zapisów i odczytów nieznacznie rożni się od teoretycznego M_{O+Z}. Mamy do czynienia z lekkim nadamiarem. Według otrzymanych wyników widać, że liczba faz jest niezależna od współczynnika b. Zależy ona od liczby serii. Od czynnika b zależna jest jednak suma operacji zapisu i odczytu, tak jak wynika to ze wzoru.

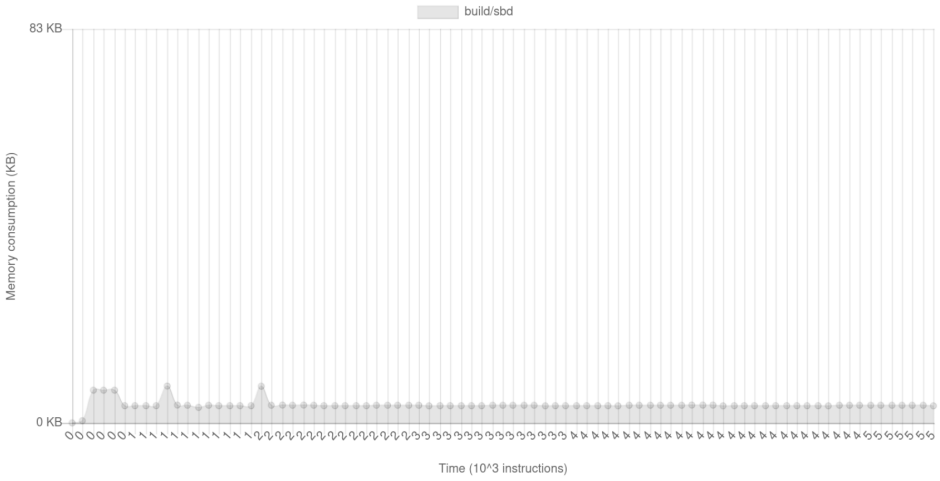


Rys. 3: Liczba operacji zapisów i odczytów - porównanie w skali logarytmicznej

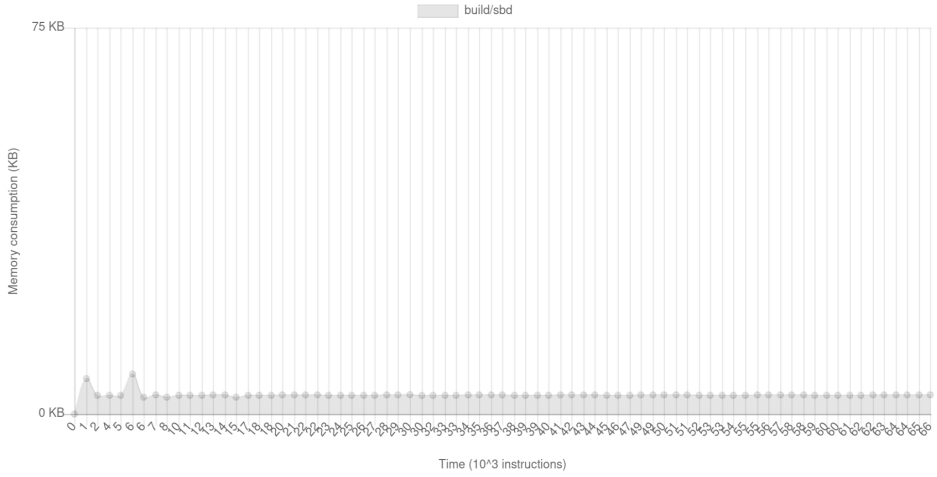
Porównując otrzymane wyniki dla b=10 i b=100 można zauważyć, że liczba operacji zapisów i odczytów proporcjonalnie maleje do ilości rekordów w bloku. Im większa strona, tym większe zużycie pamięci, ale za to mniej operacji zapisów i odczytów. Dla N=10 mamy do czynienia tylko z jedną stroną, więc zapisów i odczytów będzie tyle samo dla obu przypadków.

Zużycie pamięci

Zużycie pamięci jest stałe i niezależne od liczby rekordów. Na poniższych wykresach przedstawione zostało zużycie pamięci odpowiednio dla 100 i 1000 wylosowanych losowo rekordów. Nastąpiły pewne skoki zużycia pamięci, ale nie przekraczały one 7.2 KiB. Znaczna część programu działała na zużyciu 3.2 KiB. Oba te wykresy wygenerowane zostały dla b=10.



Rys. 4: Zużycie pamięci dla N=100



Rys. 5: Zużycie pamięci dla N=1000

Zakończenie

Wykonanie projektu pozwoliło mi zapoznać się z działaniem algorytmu sortowania przez scalanie naturalne. Ciekawym dla mnie aspektem projektu było wykorzystanie taśm i blokowe zapisywanie i odczytywanie danych, gdzie używana była stała pamięć, a także pozwoliło to na mniejsze zużycie pamięci programu, poprzez nieładowanie wszystkich danych do pamięci operacyjnej. Wyniki eksperymentu pokrywają się z obliczonymi wartościami teoretycznymi. Liczba operacji odczytu i zapisu zmniejsza się proporcjonalnie do zwiększenia ilości rekordów w bloku. Ilość rekordów w stronie nie ma znaczenia dla ilości faz.