W12n, AIR Projektowanie Algorytmów i Metody Sztucznej Inteligencji	Sprawozdanie nr 2 Temat: Sortowanie
Jakub Cebula	Wt 15:15, 8 maja 2023 Dr inż. Marek Bazan

Spis treści

1 Opis zadania

1.1 Wybrane algorytmy

- mergesort
- quicksort
- bucketsort

1.2 Przebieg zadania

1. Wczytanie pliku z bazą filmów,
2. Przefiltrowanie filmów bez ocen,
3. Sortowanie według wybranego algorytmu,
4. Zapis posortowanych plików

2 Opis badanych algorytmów

2.1 Sortowanie przez scalanie (mergesort)

Sortowanie przez scalanie jest przykładem algorytmu dzielącego zbiór danych na podzbiory i sortującego coraz to mniejsze zbiory, a następnie łączącym już posortowane zbiory. Jego złożoność obliczeniowa niezależnie od danych wejściowych wynosi $O(n \cdot \log(n))$.

2.2 Sortowanie szybkie (quicksort)

Metoda quicksort jest oparta na podejściu dziel i zwyciężaj.Sortowanie szybkie jest metodą rekurencyjną (tzn. odowłuje się do samej siebie, aż do momentu kiedy otrzyma przypadek podstawowy). Algorytm polega na tym, że dzieląc np. tablice w obojętnie którym miejscu liczby przenoszone są na strone "lewą" lub "prawą" w zależności czy są większe czy mniejsze od elementu osiowego. Dla każdej podtablicy znowu wywołujemy ten algorytm aż do uzyskania przypadku z jednym elementem w tablicy.

2.3 Sortowanie kubełkowe (bucketsort)

BucketSort to algorytm sortowania elementów poprzez umieszczenie ich w odpowiednich kubełkach na podstawie ich wartości, a następnie sortowanie każdego kubełka za pomocą innego algorytmu sortowania (np. quicksort). Po posortowaniu kubełków, elementy są kopiowane z powrotem do oryginalnej tablicy w porządku rosnącym lub malejącym, zgodnie z ich kolejnością w kubełkach. Algorytm ten działa szybko i efektywnie dla rozkładów elementów równomiernie rozłożonych w przedziale wartości. Czas działania BucketSort zależy od liczby kubełków, jakie są używane do sortowania.

2.4 Złożoność obliczeniowa

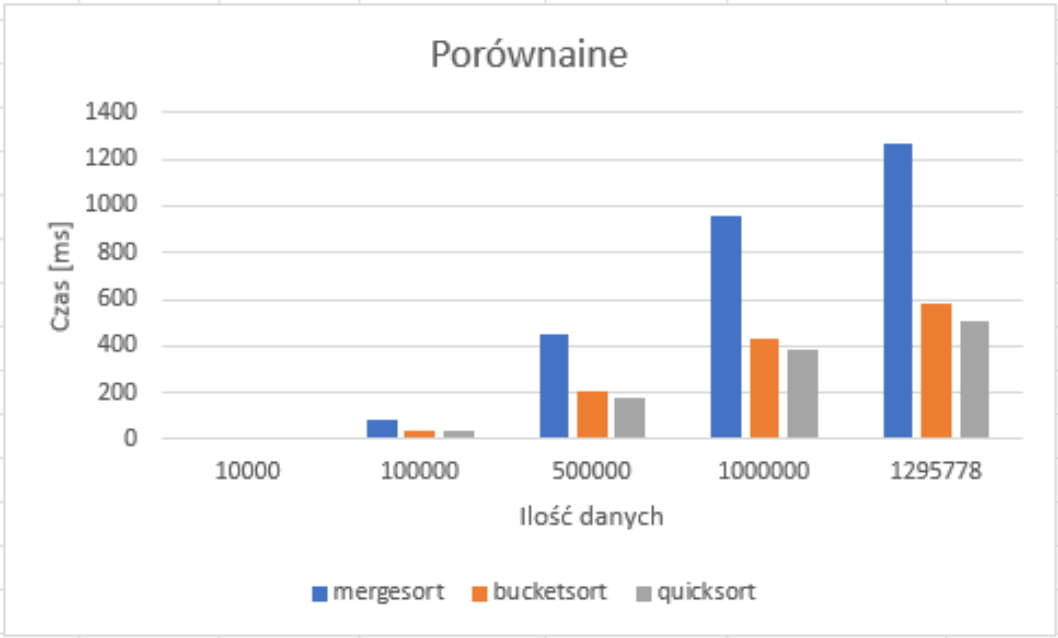
przypadek\typ	mergesort	bucketsort	quicksort
najgorszy	O(nlogn)	O(n^2)	O(n^2)
średni	O(nlogn)	O(n+k)	O(nlogn)

Rysunek 1: Przypadki złożoności obliczeniowej

3 Wyniki testów

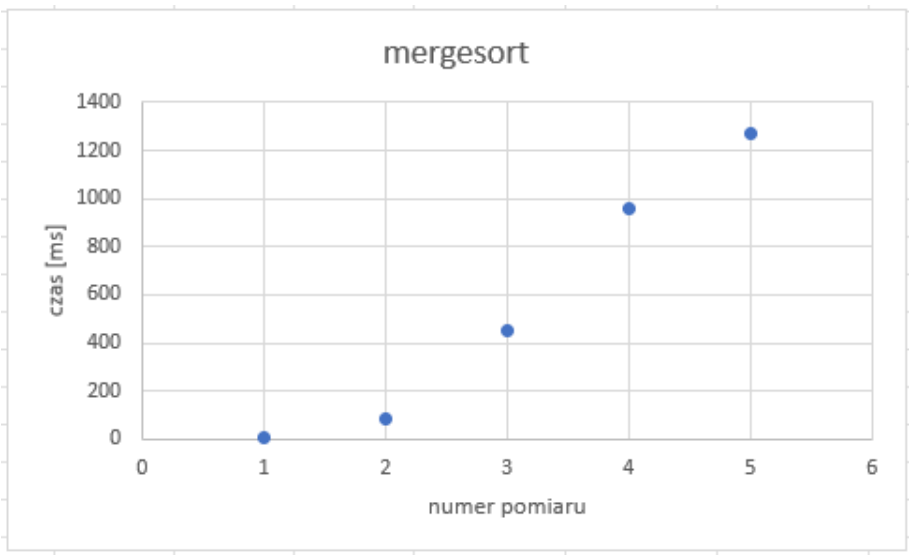
n	sortowanie			
	typ	quicksort	mergesort	bucketsort
10000	czas	2,592ms	6,297ms	3,232ms
	średnia	5,991	5,991	5,991
	mediana	6	6	6
100000	czas	32,559ms	80,903ms	38,159ms
	średnia	6,185	6,185	6,185
	mediana	6	6	6
500000	czas	181,174ms	453,663ms	207,769ms
	średnia	6,753	6,753	6,753
	mediana	6	6	6
1000000	czas	381,549ms	962,113ms	430,981ms
	średnia	6,904	6,904	6,904
	mediana	7	7	7
1295778	czas	504,537ms	1270,545ms	578,650ms
	średnia	6,953	6,953	6,953
	mediana	7	7	7

Rysunek 2: Czas, mediana oraz średnia po przesortowaniu



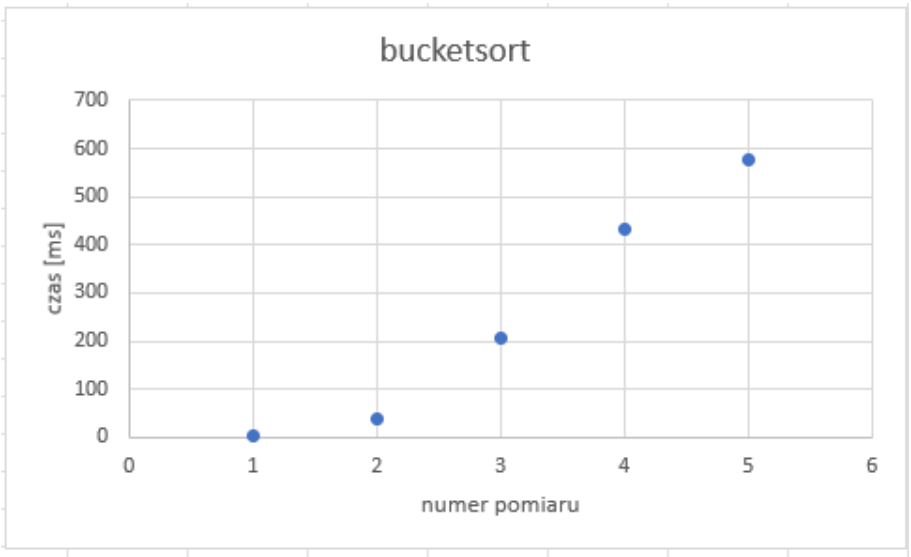
Rysunek 3: Wykres poglądowy

3.1 Mergesort



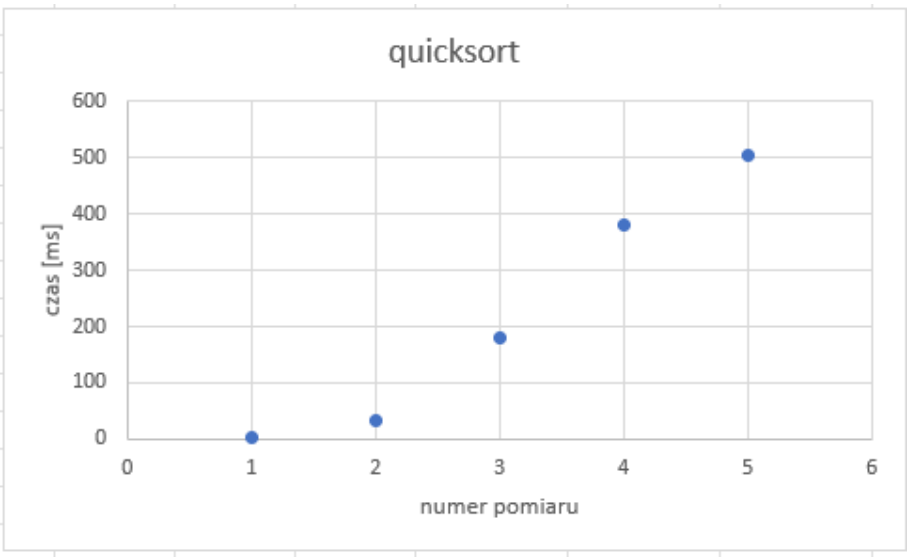
Rysunek 4

3.2 Bucketsort



Rysunek 5

3.3 Quicksort



Rysunek 6

4 Wnioski

- Sortowanie quicksort oraz sortowanie bucketsort jest o wiele szybsze niż sortowanie mergesort.
- Każde z sortowań działa w sposób prawidłowy co możemy zweryfikować w zapisanych plikach.
- Mergesort algorytmem stabilnym i działa w czasie O(n log n), co czyni go efektywnym dla bardzo dużych zbiorów danych.
- Bucketsort jest wydajny dla danych rozproszonych o równomiernym rozkładzie, podczas gdy Mergesort i Quicksort są lepsze dla danych o losowym rozkładzie.
- Sortowanie szybkie przypomina bardziej złożoność liniową na podanym zestawie danych, co może sugerować zbyt małą ilość danych, by zauważyć zależność O(nlogn).

5 Bibliografia

1. `https://www.geeksforgeeks.org/bucket-sort-2/`
2. `https://en.wikipedia.org/wiki/Bucket_sort`
3. `https://www.geeksforgeeks.org/quick-sort/?ref=gcse`
4. `https://pl.wikipedia.org/wiki/Sortowanie_szybkie`
5. `https://www.geeksforgeeks.org/merge-sort/`
6. `https://pl.wikipedia.org/wiki/Sortowanie_przez_scalanie`
7. Michale T. Goodrich, Roberto Tamassia, David M. Mount. Data Structures and Algorithms in C++. Second Edition. John Wiley & Sons, Inc.