Projektowanie algorytmów i metody sztucznej inteligencji - Raport 2

Damian Ryś

13 czerwca 2021

Spis treści

1 Wstęp

Grupa lab: E12-93c
Termin zajęć: PN 18:45-20:35
Numer indeksu: 252936
Prowadzący: Dr inż. Piotr Ciskowski

2 Zadanie

Podstawionym przed nami problemem jest posortowanie danych uzyskanych z witryny "IMDB" zawierających wyłącznie tytuł oraz ranking. Bazę będziemy sortować względem rankingu przy pomocy poniższym algorytmów:

1. MergeSort
2. QuickSort
3. IntroSort
4. BucketSort

Algorytmy te zostaną omówione dokładniej w dalszej cześci pracy. Co jest warte odnotowania, baza z której korzystamy zawiera błędne wpisy, których musimy się pozbyć przed przystąpieniem do sortowania i pomiarów. Takich wpisów było: 47389.

Po usunięciu takich wpisów pozostało nam 962900 rekordów i to dla takiej liczby były przeprowadzane testy dla miliona wartości z czego mogą wynikać różnicę w medianie lub średniej.

3 Algorytmy

3.1 MergeSort

MergeSort jest algorytmem rekurencyjnym, polegającym na dzieleniu tablicy aż do uzyskania pojedyńczych elementów. Następnie algorytm w przypadku złego ułożenia zamienia elementy miejscami, by kolejne je scalić, aż do uzyskania całej posortowanej tablicy.

3.2 QuickSort

Algorytm wykorzystuje technikę "dziel i zwyciężaj". Zaczynamy w nim od wybrania pivota- będący jednym z elementó sortowanej tablicy. Elementy mniejsze od pivota są przenoszone na lewą stronę, a większę na prawą. Algorytm następnie wywołuje się rekurencyjnie, aż do posortowania się tablicy

3.3 IntroSort

Jest to metoda hybrydowa, będąca połączeniem sortowania szybkiego i sortowania przez kopcowanie. Sortowanie introspektywne pozwala uniknąć najgorszego przypadku dla sortowania szybkiego (nierównomierny podział tablicy w przypadku, gdy jako element osiowy zostanie wybrany element najmniejszy lub największy).

3.4 BucketSort

Ideą sortowania kubełkowego jest podzielenie liczb na k (kubełków) pod przedziałów o równej długości, następnie liczby z sortowanej tablicy do odpowiednich kubełków, posortowanie liczy w niepustych kubełkach i wypisanie po kolei zawartości niepustych kubełków

4 Testy

Testy przebiegały w kilku etapach. Najpierw dla małych tablic zostały sprawdzone wizualnie czy algorytmy prawidłowo sortują. Po stwierdzeniu poprawności zaimplementowanych algorytmów zostały podjęte testy związane z ich wydajnością czasową dla różnej wielkości danych wejściowych. Dane przechowywane były w dynamicznie alokowanej tablicy co pozwoliło na jednoczesne przeprowadzenie testów dla wszystkich algorytmów naraz co eliminowało błędy losowe. Rezultat testów widoczny jest na poniżej:

```
Started sorting
Mergesort time for 10000:11.113ms
Started sorting
Mergesort time for 100000:100.465ms
Started sorting
Mergesort time for 500000:486.225ms
Started sorting
Mergesort time for 962900:937.426ms

Started sorting
QuickSort time for 10000:6.7993ms
Started sorting
QuickSort time for 100000:88.011ms
Started sorting
QuickSort time for 500000:527.492ms
Started sorting
QuickSort time for 962900:1078.98ms
Started sorting
BucketSort time for 10000:0.6404ms
Started sorting
BucketSort time for 100000:8.3213ms
Started sorting
BucketSort time for 500000:33.3534ms
Started sorting
BucketSort time for 962900:64.7212ms
Started sorting
IntroSort time for 10000:0.1572ms
Started sorting
IntroSort time for 100000:1.65ms
Started sorting
IntroSort time for 500000:8.89ms
Started sorting
IntroSort time for 962900:18.26ms
```

Rysunek 1: Pomiary z programu

Wyniki zostały te uporządkowane w poniższej tabeli:

Tabela 1: Tabela pomiarowa dla poszczególnych algorytmów przy różnej wielkości danych wejściowych

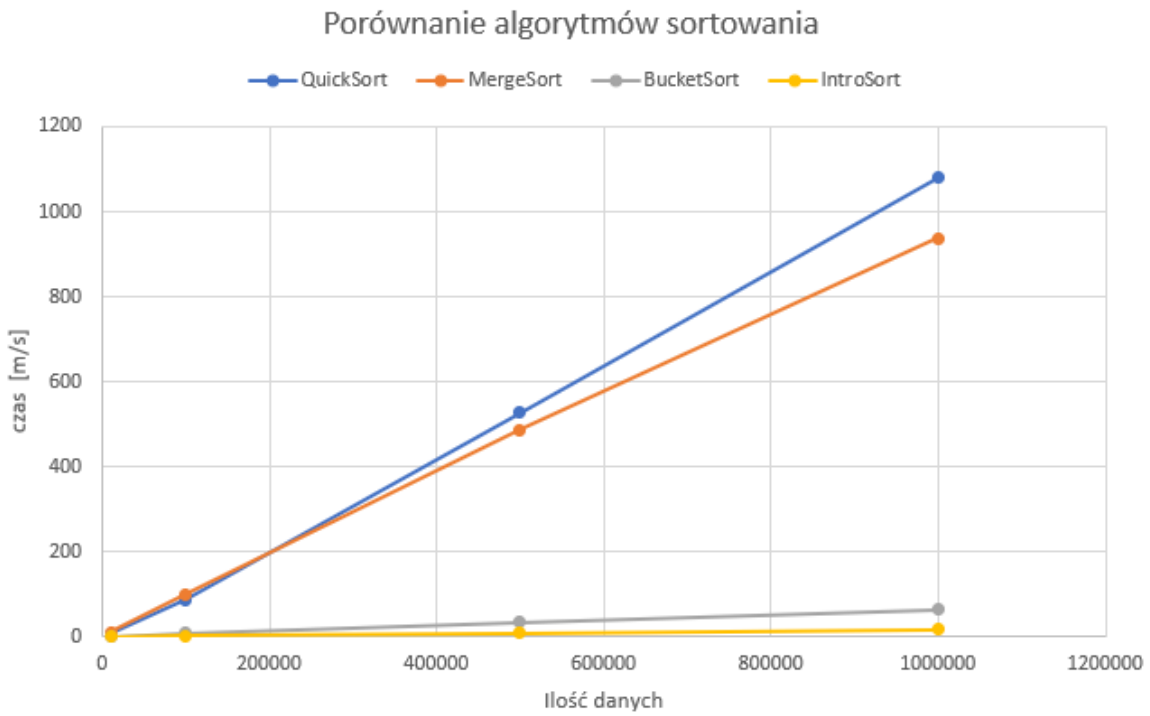
	QuickSort	**MergeSort**	**BucketSort**	**IntroSort**
Ilość danych:	**[ms]**			
10000	6.7993	11.113	0.6404	0.1572
100000	88.011	100.465	8.3213	1.6553
500000	527.492	486.225	33.3534	8.8992
1000000	1078.98	937.426	64.7212	16.261

Zostały również przeprowadzone testy zgodnie z specyfikacją zadania średniej wartości oraz medianę rankingu.

Tabela 2: Tabela pomiarowa mediany i średniej dla poszczególnych algorytmów przy różnej wielkości danych wejściowych

	QuickSort		**MergeSort**		**BucketSort**		**IntroSort**	
	Mediana	**Średnia**	**Mediana**	**Średnia**	**Mediana**	**Średnia**	**Mediana**	**Średnia**
Ilość danych:								
10000	5	5.4603	5	5.4603	5	5.4603	5	5.4603
100000	6	6.01973	6	6.01973	6	6.01973	6	6.01973
500000	6	6.57832	6	6.57832	6	6.57832	6	6.57832
1000000	7	6.56831	7	6.56831	7	6.56831	7	6.56831

W celu łatwiejszego wyciągnięcia wniosków narysowany został również wykres porównujący nasze algorytmy:



Rysunek 2: Wykres liniowy w programie Excel

5 Wnioski

- Na podstawie mediany oraz średniej, która jest identyczna dla każdego sortowania jesteśmy w stanie stwierdzić że nasze algorytmy działają prawidłowo
- Najwięcej czasu w całym programie zajmuje odpowiednie zparsowanie naszego pliku do struktury danych, średni czas ten wynosił około 2.5 min, co znacząco utrudziło debuggowanie programu
- Najszybszym algorytmem jest IntroSort - co nie powinno być dziwnie, przez stosowanie odpowiedniego przez niego odpowiedniego algorytmu w zależności od przypadku
- Sortowanie Szybkie wcale szybkie nie jest. Pomimo swojej nazwy jest to algorytm,który wypadł najgorzej ze wszystkich. Jest to prawdopodobnie spowodowane źle dobranym pivotem.

6 Bibliografia

- http://www.algorytm.org/algorytmy-sortowania/sortowanie-przez-scalanie-mergesort.html
- https://www.geeksforgeeks.org/introsort-or-introspective-sort/
- http://algorytmy.ency.pl/artykul/quicksort
- https://www.interviewbit.com/tutorial/quicksort-algorithm/