Sprawozdanie

Projektowanie algorytmów i metody sztucznej inteligencji
Projekt 2

Mikołaj Saramonowicz 252964
Grupa środa, 13:15-15:00

1 Wprowadzenie

Sprawozdanie zapiera opis algorytmów zastosowanych do rozwiązania zadania na ocenę bardzo dobrą oraz omówienie uzyskanych za ich pomocą wyników .

2 Krótki opis stosowanych algorytmów

Każdy z wymienionych sposobów sortowania wykonuje operacje na kluczach sortowanych elementów.

2.1 Merge Sort - sortowanie przez scalanie

Sortowana sekwencja dzielona jest rekurencyjnie na dwie podsekwencje do momentu uzyskania pojedynczych elementów. Następnie struktury te są odpowiednio scalane, tak aby zachować kolejność rosnącą. Algorytm wykorzystuje metodę podziału problemu na łatwe do rozwiązania podproblemy, zwaną też jako "dziel i rządź".

Parametry algorytmu:

- Złożoność obliczeniowa: $O(nlogn)$
- Złożoność pamięciowa: $O(n)$
- Wydajny dla sekwencyjnego dostępu do danych i bardzo dużej ilości elementów

2.2 Quick Sort - sortowanie szybkie

Zaimplementowana wariacja algorytmu rozpoczyna działanie od wyznaczania losowego elementu - pivota. Następnie sortowana sekwencja dzielona jest na trzy podsekwencje. Pierwsza z nich zawiera elementy mniejsze od pivota L, druga elementy mu równe (wliczając pivota) E, a trzecia większe od niego G. Proces dzielenia powtarzany jest aż do uzyskania jednoelementowych sekwencji elementów różnych od pivota - struktury same w sobie nie wymagają już dalszego sortowania. Następnie uzyskane podsekwencje scalane są w sposób rosnący.
W celu optymalizacji procesu sortowania, w przedstawionym rozwiązaniu pivot wybierany jest losowo. Podobnie jak Merge Sort, algorytm wykorzystuje metodę "dziel i rządź".

Parametry algorytmu:

- Złożoność obliczeniowa: $O(nlogn)$
- Pesymistyczna złożoność obliczeniowa (mogąca wystąpić dla znacznych różnic między wylosowanymi rozmiarami podsekwencji L i G): $O(n^2)$
- Złożoność pamięciowa: zależna od implementacji
- Wydajny dla dużej ilości elementów lub konieczności sortowania w miejscu

2.3 Bucket Sort - sortowanie kubełkowe

Algorytm rozpoczyna pracę od utworzenia pomocniczej tablicy pustych sekwencji (kubełków), gdzie każda z nich odpowiada jednej z możliwych wartości klucza. Następnie każdy element z sortowanej struktury zostaje usunięty i przepisany do odpowiadającej jego kluczowi podsekwencji. Ostatnim krokiem jest ponowne zapełnienie struktury wejściowej, dodając do niej elementy z podstruktur w sposób rosnący.

Zaimplementowana wersja różni się nieco od powszechnie spotykanej w literaturze, gdyż przedział wartości kluczy jest stosunkowo niewielki oraz każdy z nich jest liczbą naturalną. Dzięki temu wykorzystanie algorytmu jest proste i wydajne.

Parametry algorytmu:

- Złożoność obliczeniowa: $O(n + N)$, gdzie N oznacza liczbę utworzonych podsekwencji
- Pesymistyczna złożoność obliczeniowa (dla przypadku ogólnego): $O(n^2)$
- Złożoność pamięciowa: $O(n)$
- Wydajny dla względnie niewielkiej liczby możliwych kluczy
- Jeżeli klucze nie są liczbami naturalnymi, to algorytm wymaga modyfikacji, mogących wpłynąć na jego wydajność

3 Przebieg i wyniki eksperymentów

3.1 Wstępne testowanie algorytmów

Testowanie sortowań zrealizowano za pomocą oddzielnego programu, losującego 15 wartości i dającego możliwość wyboru algorytmu. W sprawozdaniu zawarto zrzuty ekranu, potwierdzające poprawną pracę każdego z nich.

```
miksar@DI157559-LD2:~/Desktop/4 semestr/PAMSI 2/final$ ./a.out
Wylosowane dane:
0 10 8 2 4 2 0 8 3 3 2 6 9 10 8
Wybierz sortowanie do przetestowania:
1 - sortowanie szybkie:
2 - sortowanie przez scalanie
3 - sortowanie kubelkowe
Twoj wybor: 1
Wybrano sortowanie szybkie
Posortowane dane:
0 0 2 2 2 3 3 4 6 8 8 8 9 10 10
```

Rysunek 1: Test algorytmu Quick Sort

```
miksar@DI157559-LD2:~/Desktop/4 semestr/PAMSI 2/final$ ./a.out
Wylosowane dane:
0 8 8 7 10 3 4 4 4 4 6 7 8 9 5
Wybierz sortowanie do przetestowania:
1 - sortowanie szybkie:
2 - sortowanie przez scalanie
3 - sortowanie kubelkowe
Twoj wybor: 2
Wybrano sortowanie przez scalanie
Posortowane dane:
0 3 4 4 4 4 5 6 7 7 8 8 8 9 10
```

Rysunek 2: Test algorytmu Merge Sort

```
miksar@DI157559-LD2:~/Desktop/4 semestr/PAMSI 2/final$ ./a.out
Wylosowane dane:
1 4 2 1 6 7 2 7 0 1 7 3 3 0 9
Wybierz sortowanie do przetestowania:
1 - sortowanie szybkie:
2 - sortowanie przez scalanie
3 - sortowanie kubelkowe
Twoj wybor: 3
Wybrano sortowanie kubelkowe
Posortowane dane:
0 0 1 1 1 2 2 3 3 4 6 7 7 7 9
```

Rysunek 3: Test algorytmu Bucket Sort

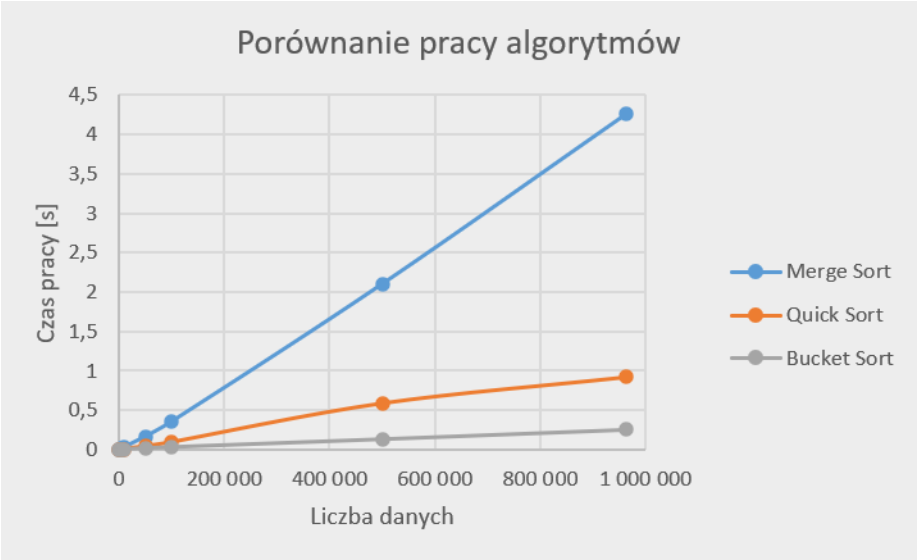
Można zatem stwierdzić, że wszystkie algorytmy zaimplementowano poprawnie.

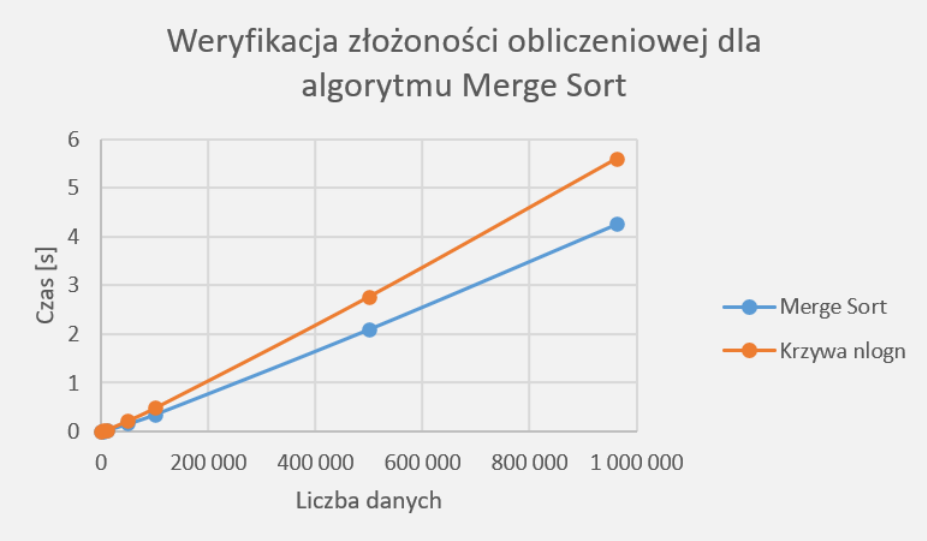
3.2 Wyniki sortowań otrzymanej bazy danych

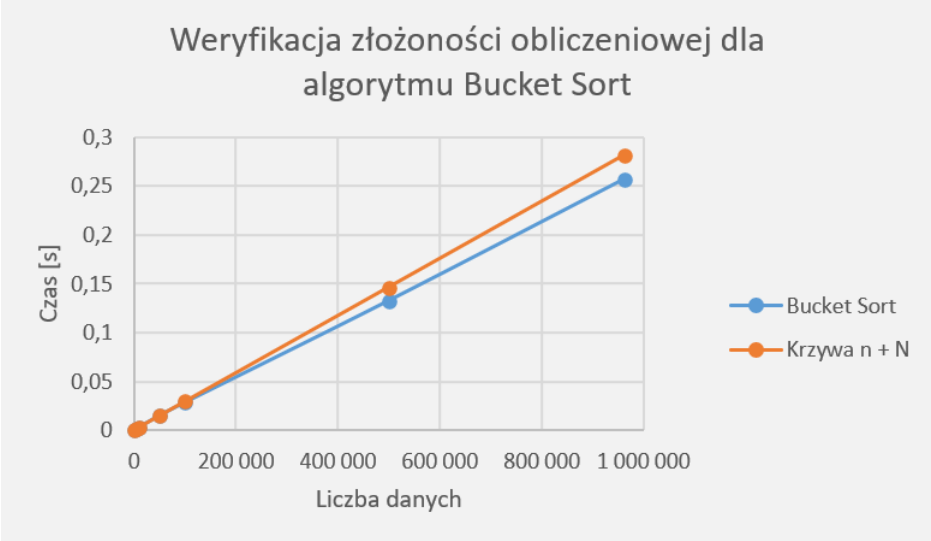
W poniższej tabeli porównano algorytmy pod względem szybkości pracy dla kilku wybranych punktów pomiarowych. Zawarto również wartości mediany oraz średniej kluczy.

Liczba danych	Czas wczytywania do struktury (wliczając filtrację) [s]	Czas zapisywania do pliku [s]	Czas sortowania [s]			Mediana	Średnia
			Merge Sort	Quick Sort	Bucket Sort		
1 000	0,000704	0,000527	0,002916	0,000933	0,000296	5,00	5,42000
5 000	0,002426	0,001583	0,014032	0,004124	0,001546	5,00	5,45480
10 000	0,004719	0,003604	0,031167	0,007107	0,002867	5,00	5,46030
50 000	0,018947	0,015109	0,166329	0,041138	0,014908	5,50	5,49092
100 000	0,053441	0,05639	0,354681	0,09238	0,028246	7,00	6,08993
500 000	0,261605	0,190437	2,10224	0,58589	0,132311	7,00	6,66567
MAX - 962 796	0,505356	0,50959	4,25821	0,918772	0,256815	7,00	6,63666

Dane dotyczące algorytmów przeniesiono na wykresy.







Na podstawie wykresów można stwierdzić, że każdy z algorytmów zachowuje przewidywaną złożoność obliczeniową wyrażoną w notacji dużego O.

4 Wnioski i spostrzeżenia

- W danym przypadku najszybszym algorytmem okazał się Bucket Sort.
  Wynika to z niewielkiej liczby kluczy - jedynie liczby naturalne od 1 do 10.
- Najwolniejszą metodą było wykorzystanie sortowania przez scalanie.
  Nie udało się ustalić dokładnej przyczyny tego zjawiska. Wiadomo, że algorytm został zaimplementowany poprawnie, gdyż spełnia on oczekiwaną złożoność obliczeniową.
- Dla większych zbiorów danych zauważalnie wzrasta czas wczytywania i zapisywania do pliku. Po przeprowadzeniu dalszych eksperymentów ustalono, że wzrost ten ma charakter zbliżony do liniowego, zgodnie z założeniami wynikającymi z wykorzystania sekwencji na liście.
- Wartości średniej oraz mediany zmieniają się w zależności od ilości przetwarzanych elementów - stają się coraz bardziej zbliżone do rzeczywistych.
- Przed filtracją elementów zawartych w pliku było maksymalnie 1010292, zatem spośród nich prawie 50 tysięcy zawierało błędy.

5 Wykorzystana literatura

- dr inż. Łukasz Jeleń, prezentacje wykładowe i materiały dostępne na stronie kursu
- Michael T. Goodrich, Roberto Tamassia i David Mount, Data Structures & Algorithms in C++, John Wiley and Sons Ltd, 2011
- Sortowanie przez scalanie, Wikipedia, Aktualizacja 16 kwietnia 2021,
  dostęp w Internecie:`https://pl.wikipedia.org/wiki/Sortowanie_przez_scalanie`
- Sortowanie szybkie, Wikipedia, Aktualizacja 16 luty 2021,
  dostęp w Internecie:`https://pl.wikipedia.org/wiki/Sortowanie_szybkie`
- Sortowanie kubełkowe, Wikipedia, Aktualizacja 29 maja 2019,
  dostęp w Internecie:`https://pl.wikipedia.org/wiki/Sortowanie_kube%C5%82kowe`