Sprawozdanie Projekt 2 - Algorytmy		
Rafał Jasiński 259384	15.05.2022r	Dr hab. inż. Andrzej Rusiecki
Zadania na ocenę bdb (5.0)		
Link do repozytorium: `https://gitlab.com/JasinskiR259384/pamsi-2022\`		

1 Wstęp

Celem zadania było napisanie trzech wybranych algorytmów sortowania. Sortowaniu miał podlegać dostarczony plik .csv z filmami (tytuł oraz ocena). Przed przystąpieniem do samego sortowania należało przefiltrować dane w poszukiwaniu wyników niespełniających wymagania.
Po posortowaniu danych zostają podane informacje:

- czas sortowani
- mediana ocen
- średnia ocen

2 Proces filtrowania

Czysty plik z danymi *projekt2_dane.csv* zawiera 1 010 292 rekordów. Podczas filtracji rekordy nie zawierające liczby porządkowej, tytułu lub też rankingu zostały odrzucane (o ile występują).
W wyniku tego, otrzymano 962 903 rekordy spełniające powyższe kryteria. (Powielone rekordy zostały pozostawione).
Niniejszym testy dla 1 000 000 danych, które są opisane w §2 instrukcji do zadania są niemożliwe do spełnienia.

3 Wybrane algorytmy

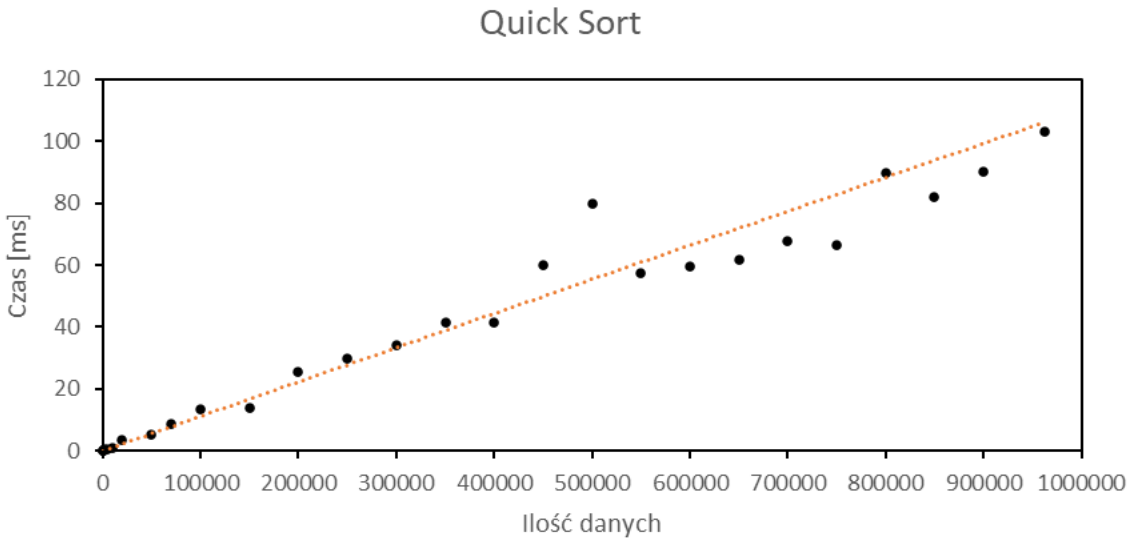
W sprawozdaniu zostaną przedstawione wyniki doświadczeń sortowania oraz analiza następujących algorytmów sortowania:

- Quick Sort
- Bin / Bucket Sort
- Merge Sort

3.1 Quick Sort

Quick Sort działa na zasadzie wyboru piwotu według, którego będą porównywane dane, a następnie przenoszone na odpowiednią stronę. Działa on na zasadzie rekurencji. Dla ilości danych powyżej 500 000 głębokość rekurencji była na tyle duża, że dostępna pamięć, dla tego programu została wyczerpana i dostawałem komunikat *Segmentation fault*. Przy wywołaniu programu poprzez *valgrind –leak-check=full* dostawałem komunikat o przepełnieniu stosu na głównym wątku - przy czym nie było żadnych wycieków pamięci.
Dlatego postanowiłem zrealizować 3-Way QuickSort, który dzieli dane na 3 części - mniejsze, równe, większe piwotowi.

Wykres przedstawiający czas sortowania od ilości danych



Rysunek 1: Wykres t = f(n) - czasu sortowania od ilości danych

Analiza złożoności obliczeniowej

- Przypadek średni
  Dla nieposortowanych danych złożoność algorytmu Quick Sort wynosi O(nlog(n)) w przypadku 3-way Quick Sort. Jednakże uogólniając przypadki tych samych wartości złożoność wyniesie $2n \ln n + \Theta(n)$
  Jeżeli przyjmiemy $i < j$ oraz X_{ij}, któe będzie zmienną przyjmującą 1, jeśli istnieje porównanie dwóch liczb ze zbioru w trakcie działania algorytmu oraz 0 w przeciwnym przypadku. Całkowita liczba porównań wyniesie, więc:

$$\sum_{i-1}^{n-1} \sum_{j=i+1}^{n} X_{ij}$$

  wartość oczekiwana tej sumy będzie zatem

$$E[X] = E\left[\sum_{i-1}^{n-1} \sum_{j=i+1}^{n} X_{ij}\right]$$

  co możemy przekształcić do postaci

$$E[X] = \sum_{i-1}^{n-1} \sum_{j=i+1}^{n} E[X_{ij}]$$

  Zatem prawdopodobieństwo, że z pośród dwóch wybranych elementów jeden z nich jest pierwszym piwotem ze zbioru danych, jest prawdopodobieństwem, że X_i j = 1, wynosi 2/(j i + 1).
  Przyjmując $k = j - i + 1$ dostaniemy

$$E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j - i + 1}$$

  Podstawiając wyżej podany warunek

$$= \sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{2}{k}$$

  Wykorzystując zamianę zakresu sum dostaniemy

$$\sum_{k=2}^{n} \sum_{i=1}^{n+1-k} \frac{2}{k}$$

Ponieważ druga suma nie jest zależna od K, możemy to sprowadzić do postaci

$$\sum_{k=2}^{n}(n+1-k)\frac{2}{k}$$

Przenosząc teraz niezależne człony przed sumę otrzymamy

$$\left((n+1)\sum_{k=2}^{n}\frac{2}{k}\right)-2(n-1)$$

Zmieniając zakres sumy do $k=1$

$$(2n+2)\sum_{k=1}^{n}\frac{1}{k}-4n$$

Sprowadzając teraz sumy $\sum_{k=1}^{n}\frac{1}{k}$ do złożoności otrzymamy $\ln n+\Theta(1)$, co daje nam złożoność końcową równą $2n\ln n+\Theta(n)$ (4)

- Przypadek najgorszy
  Dla posortowanych danych złożoność algorytmu wyniesie O(n^2)
  Jeżeli przyjmiemy piwot jako pierwszy element kontenera danych - n, ilość porównań wyniesie n-1. Następnie operacja podziału zwróci nam dwie podlisty, z których jedna będzie miała rozmiar 0, a druga n-1. Następnym piwot jaki zostanie wybrany będzie n-1. Tak, więc ilość operacji porównania wyniesie w tym momencie n-2. Kontynuując ten przebieg możemy sprowadzić to do zapisu:

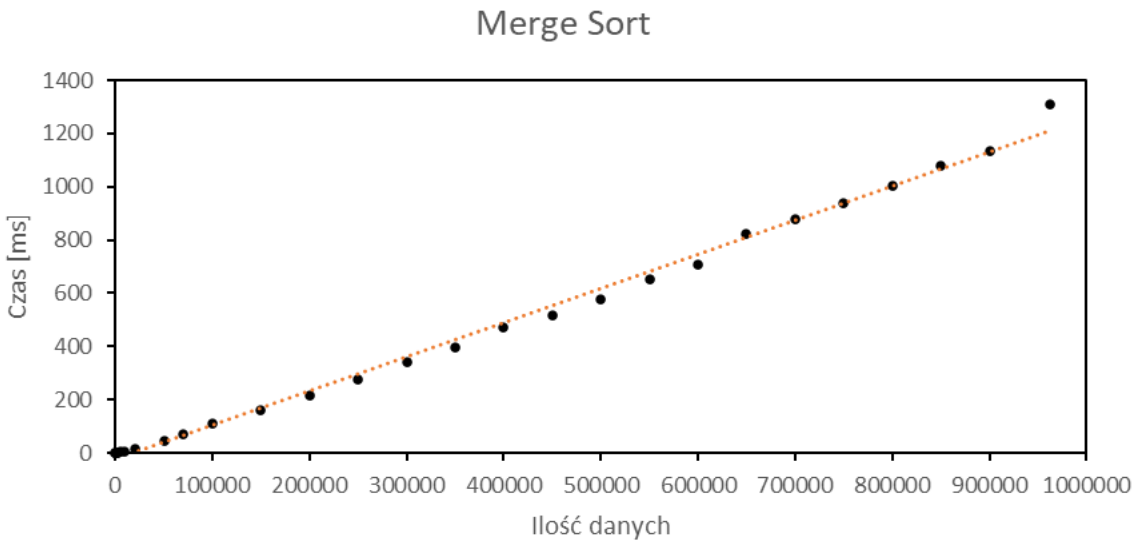
  $$(n-1)+(n-2)+\cdots+2+1=\frac{n(n-1)}{2}\textit{porównań}$$

  (4)

3.2 Merge Sort

Analiza złożoności obliczeniowej
Merge Sort charakteryzuje się tym, że jego złożonośc obliczeniowa zawsze jest stała, niezależnie od przypadku i wynosi ona $O(n\log n)$
Podział tablicy na pół w każdym kroku może być przedstawione jako działanie funkcji logarytmicznej dlatego złożoność tego procesu wynosi $O(\log n)$
Samo znalezienie indeksu, który jest połową pod/tablicy wynosi O(1)
Natomiast połączenie podtalbic w jedną posortowaną tablicę występuje w czasie O(n)
W związku z tym całkowity czas dla merge sort'a wyniesie $O(n\log n+1)$, co daje nam złożoność obliczeniową $O(n\log n)$

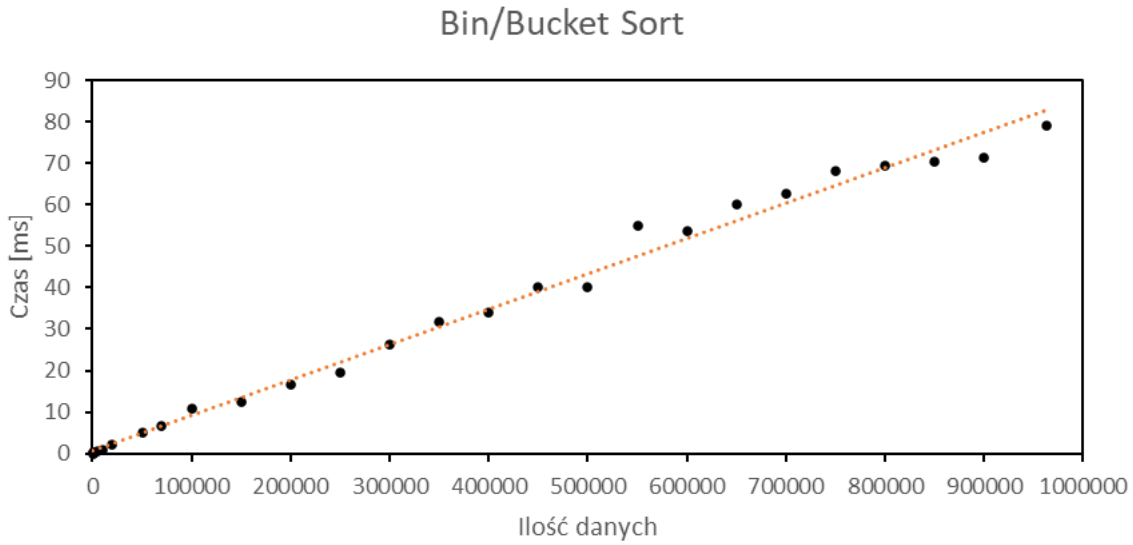
Wykres przedstawiający czas sortowania od ilości danych



Rysunek 2: Wykres t = f(n) - czasu sortowania od ilości danych

3.3 Bin / Bucket Sort

Analiza złożoności obliczeniowej
Ponieważ algorytm bazuje na stworzeniu odpowiednich podkontenerów do przechowywania danych operacja przeniesienia pojedynczego obiektu wynosi O(1). Co dla n elementów sprowadza się do O(n).
Wykres przedstawiający czas sortowania od ilości danych



Rysunek 3: Wykres t = f(n) - czasu sortowania od ilości danych

4 Testy i uwagi

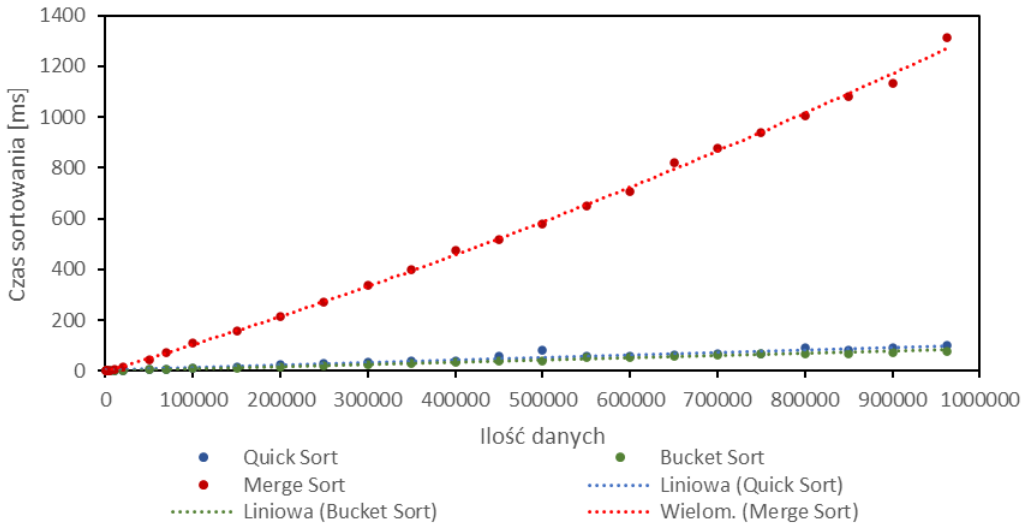
4.1 Testy dla zbiorów

Dla przedstawionych algorytmów oraz podanych danych zostały przeprowadzone testy czasu sortowania dla poszczególnych ilości danych.

Tabela 1: Tabela zbiorcza z czasami sortowań poszczególnych algorytmów

Ilość danych	QS	BS	MS
100	0,0101	0,0367	0,0385
500	0,0514	0,0672	0,2319
1000	0,0977	0,1022	0,547
2000	0,3842	0,1676	1,064
5000	0,4758	0,5423	3,1268
10000	0,977	0,8041	6,6748
20000	3,4414	2,0207	16,4933
50000	5,2292	4,8908	46,1596
70000	8,6278	6,783	70,9287
100000	13,4997	10,9599	108,4741
150000	14,095	12,297	158,8415
200000	25,4106	16,7157	216,2777
250000	29,6717	19,5742	273,3966
300000	34,3127	26,3069	339,0312
350000	41,6231	31,6115	397,3692
400000	41,3755	34,1536	473,2623
450000	60,0533	40,001	515,3928
500000	79,9556	40,081	577,9522
550000	57,2537	55,0633	650,3352
600000	59,765	53,6077	706,6018
650000	61,5542	60,2303	821,1363
700000	67,7999	62,5917	877,6248
750000	66,4373	68,0498	939,7259
800000	89,9121	69,4774	1003,518
850000	82,0852	70,233	1079,312
900000	90,1706	71,3475	1134,942
962903	102,973	78,9274	1310,833

Na jej podstawie został sporządzony wykres zawierający porównanie wszystkich trzech algorytmów



Rysunek 4: Wykres t = f(n) - czasu sortowań poszczególnych algorytmów od ilości danych

Jak już zostało wspomniane wcześniej nie można przeprowadzić testów sortowania dla 1 000 000 danych.

4.2 Filtrowanie danych

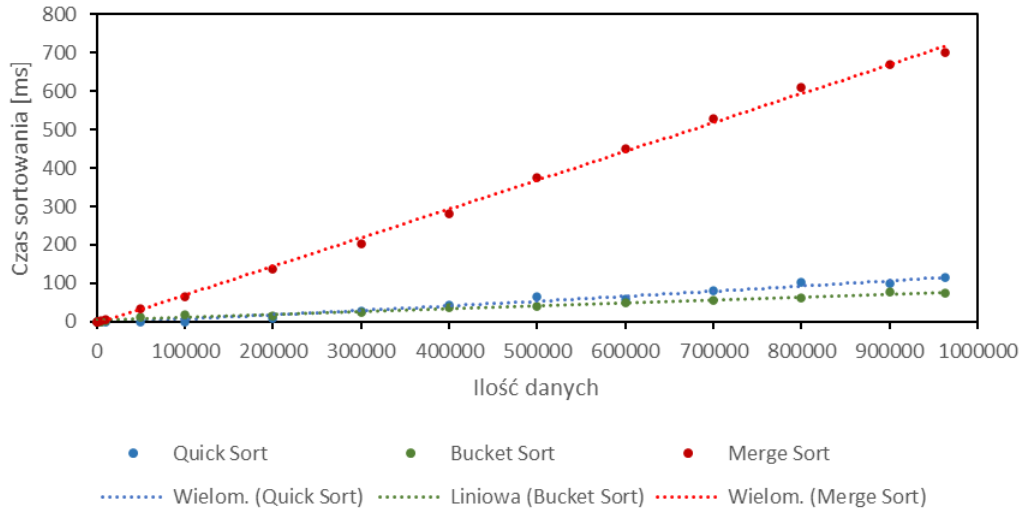
Filtrowanie danych odbywa się dopiero po wczytaniu całego pliku i przebiega ono liniowo.

4.3 Proces sprawdzania

Po każdym sortowaniu wykonywana jest funkcja, która sprawdza czy kolejny element rankingu nie jest mniejszy od aktualnie rozpatrywanego, w celu znalezienia błędy potencjalnie źle zaimplementowanego algorytmu sortowania.

4.4 Testy dla posortowanych danych

Dla posortowanych danych z pliku zostały przeprowadzone testy czasu sortowania.



Rysunek 5: Wykres t = f(n) - czasu sortowań poszczególnych algorytmów od ilości danych

5 Wnioski

1. Quick Sort ze względu na swoją głębokość rekurencji został zastąpiony wersją, która dzieli obszar sortowania na 3 części
2. Algorytm sortowania szybkiego posiada jednak pewną wadę. Nie nadaje się on do sortowania już posortowanych danych, gdyż jego złożoność wzrasta wtedy drastycznie do $O(n^2)$
3. Merge Sort ma stała złożoność obliczeniową i wynosi ona $O(n \log n)$, co może być przydatne przy sortowaniu dużych zbiorów
4. Podobnie jest z Bin / Bucket Sort, który ze względu na podział na kubełki, cechuje się liniową złożonością obliczeniową O(n)
5. Wykresy dla Quick Sort dla pomieszanych jak i posortowanych danych różnią się od siebie, co świadczy o zmiennej złożoności obliczeniowej tego algorytmu.

Literatura

[1] Algorytmy sortujące. `https://eduinf.waw.pl/inf/alg/003_sort/0001.php`. [Online; accessed 15-May-2022].

[2] 3 way quick sort. `https://www.geeksforgeeks.org/3-way-quicksort-dutch-national-flag/`, 2022. [Online; accessed 15-May-2022].

[3] Mergesort - studytonight. `https://www.studytonight.com/data-structures/merge-sort`, 2022. [Online; accessed 15-May-2022].

[4] M. Mitzenmacher, E. Upfal. *Probability and Computing: Randomization and Probabilistic Techniques in Algorithms and Data Analysis*. Cambridge University Press, 2017.

[5] Wikipedia. Bucket sort — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Bucket%20sort&oldid=1071567519`, 2022. [Online; accessed 15-May-2022].

[6] Wikipedia. Merge sort — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Merge%20sort&oldid=1086155116`, 2022. [Online; accessed 15-May-2022].

[7] Wikipedia. Quicksort — Wikipedia, the free encyclopedia. `http://en.wikipedia.org/w/index.php?title=Quicksort&oldid=1087562030`, 2022. [Online; accessed 15-May-2022].