Porównanie sortowań, sortowanie hybrydowe

S. Płaneta

23 kwietnia 2015

1 Zadanie

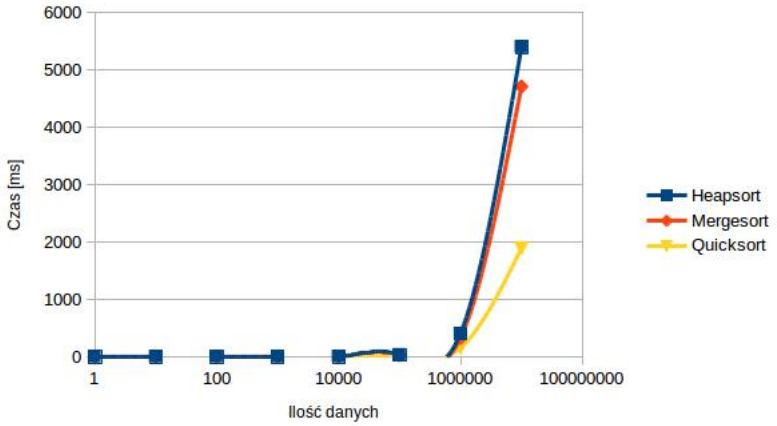
Zadaniem było porównanie quicksort'a, mergesort'a i heapsort'a, a następnie zaimplementowanie algorytmu hybrydowego.

Teoretyczne rozważania:
Złożoność pamięciowa wszystkich trzech algorytmów: $O(n)$ - w obu przypadkach wykorzystywana jedynie lista z danymi, bez dodatkowo alokowanej pamięci na dane (jedynie pojedyncza zmienna pomocnicza, potrzebna do zamiany miejscami dwóch elementów)
Złożoność obliczeniowa quicksort:
średnio $O(n \log n)$
pesymistycznie $O(n^2)$
Złożoność obliczeniowa heapsort:
$O(n \log n)$
Złożoność obliczeniowa mergesort:
$O(n \log n)$

Wszystkie algorytmy znajdują się w klasie $O(n \log n)$, quicksort średnio osiąga najlepsze czasy. Pamiętamy jednak, że mogą zdarzyć się przypadki pesymistyczne heapsort'a - wtedy czas obliczeń gwałtownie wzrasta. Aby temu zapobiec można zaimplementować algorytm sortowania hybrydowego. Zacznijmy od przeanalizowania uzyskanych wykresów.

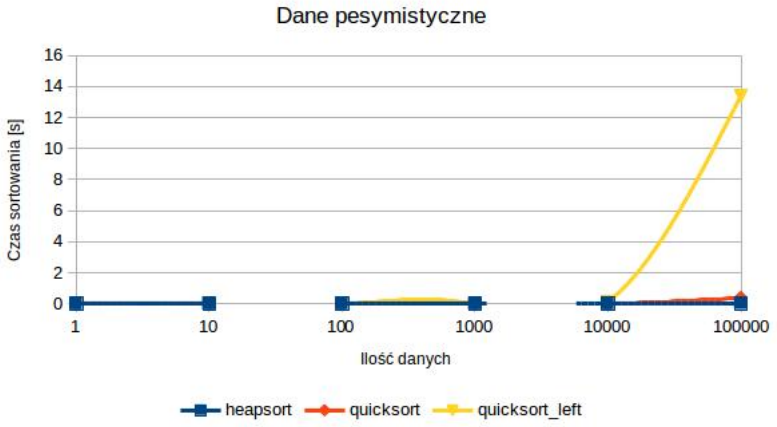
2 Porównanie

Testy przeprowadzono dla losowych danych. Na ich podstawie powstał wykres:



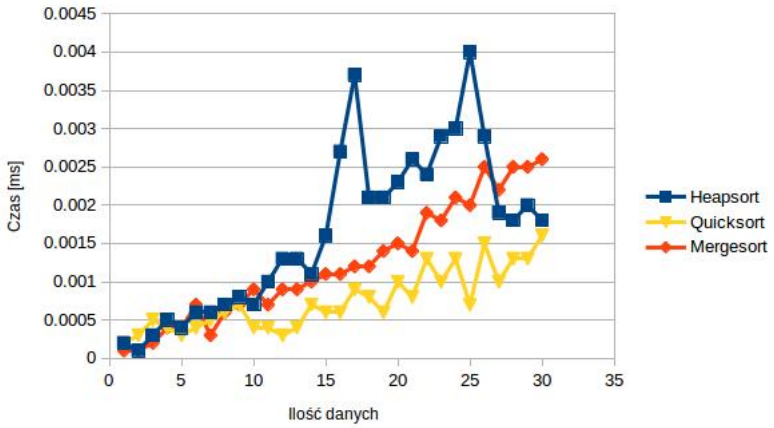
Faktycznie, zgodnie z założeniami quicksort działa najszybciej. Musimy jeszcze zapobiec występowaniu sytuacji, w których złożoność czasowa wzrośnie do $O(n^2)$.

Przyjrzyjmy się teraz przypadkowi pesymistycznemu (dane spreparowane, aby pokazać pesymistyczny przypadek dla quicksort_left)
Przypominam wykres przypadku pesymistycznego ze sprawozdania nr 4:



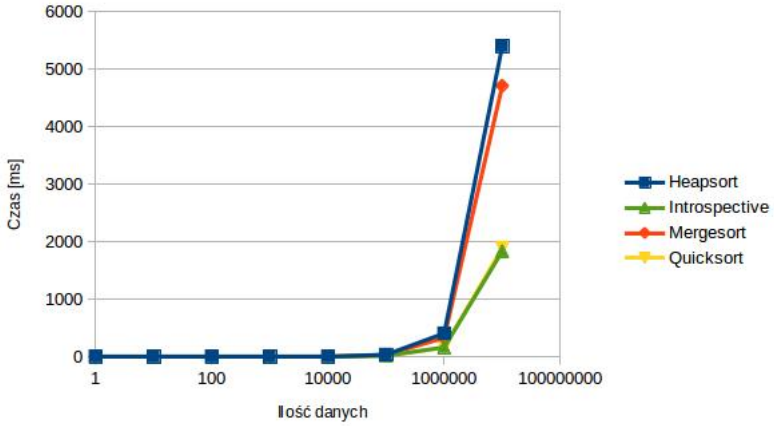
Powodem długiego działania quicksort'a w wypadku pesymistycznym jest zbyt głęboka rekurencja. Ograniczamy zatem maksymalną dozwoloną głębokość rekurencji i wywołujemy sortowanie pomocnicze. Głębokość rekurencji ograniczymy stałą $M = 2\log_2 n$

Aby wybrać sortowanie pomocnicze, przeanalizujmy kolejny wykres:



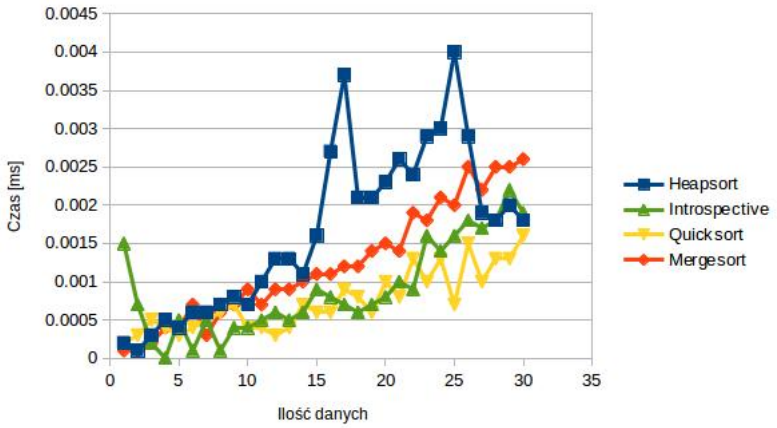
Widzimy, że dla małej ilości danych algorytmy działają podobnie. Jako sortowanie pomocnicze wybrałem heapsort'a, ponieważ jest on odporny na spreparowane dane. Na koniec, kiedy zbiór jest już prawie całkowicie posortowany, algorytm wykona sortowanie przez wstawianie, aby zakończyć cały proces sortowania. Taki rodzaj sortowania - ten algorytm hybrydowy nazywamy "sortowaniem introspektywnym"

Porównanie ze wszystkimi sortowaniami:



Jak widać na wykresie, algorytm zachowuje szybkość działania quicksort'a. Dodatkowo jest odporny na spreparowane dane - jego pesymistyczna złożoność obliczeniowa wynosi również $n \log n$.

Dla małej ilości danych:



również działa tak jak powinien.

3 Wnioski

Spośród testowanych algorytmów, najlepszym okazał się algorytm najbardziej złożony - sortowanie introspektywne. Zachowuje on szybkość quicksort'a, a jednocześnie jest odporny na sytuacje krytyczne, gdy dane są pesymistyczne.

Dobór odpowiedniego algorytmu sortowania zależy jednak od różnych czynników i nie zawsze sortowanie introspektywne będzie najlepszym wyborem. Złożoność jego implementacji, a co za tym idzie czas na to poświęcony może nie być adekwatny do otrzymanych efektów - w przypadku zbiorów małych, lub wstępnie posortowanych może zostać zastąpiony np. przez sortowanie przez wstawianie.

Kolejnym elementem, na który należy zwrócić uwagę jest stabilność algorytmów sortowania. Spośród użytych algorytmów tylko sortowanie przez wstawianie i sortowanie przez scalanie są algorytmami stabilnymi - zachowują kolejność elementów o tych samych wartościach. Pozostałe algorytmy są niestabilne - kolejność ta jest nieokreślona.