Porównanie quicksort'a i heapsort'a

S. Płaneta

23 kwietnia 2015

1 Zadanie

Zadaniem było napisanie algorytmów sortowania szybkiego (quicksort) oraz sortowania przez kopcowanie (heapsort), a następnie ich porównanie.

Teoretyczne rozważania:
Złożoność pamięciowa obu metod: $O(n)$ - w obu przypadkach wykorzystywana jedynie lista z danym, 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:
$O(n \log n)$

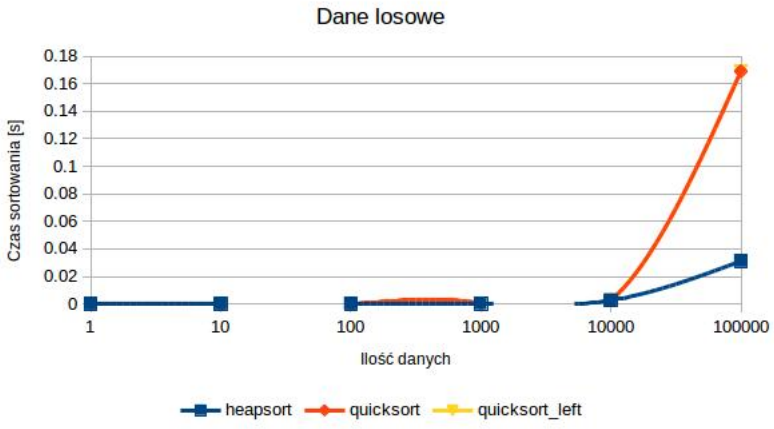
Pomimo tego, że oba algorytmy znajdują się w klasie $O(n \log n)$, quicksort zazwyczaj działa szybciej od heapsorta. Zauważmy, że w przypadku pesymistycznych danych, przy sortowaniu quicksortem można spotkać się z pesymistycznym zbiorem danych do posortowania, a wtedy czas sortowania gwałtownie wzrasta - do klasy $O(n^2)$. Aby zminimalizować szansę trafienia niepożądanego przypadku należy zoptymalizować sposób doboru piwota - zamiast wyboru np. elementu pierwszego z lewej można wykorzystać losowanie elementu czy medianę z trzech elementów. Wybranie za piwota elementu środkowego też dobrze się sprawdza.
Sortowanie przez kopcowanie nie posiada tak pesymisycznych przypadków - złożoność obliczeniowa zawsze znajduje się w tej samej klasie.

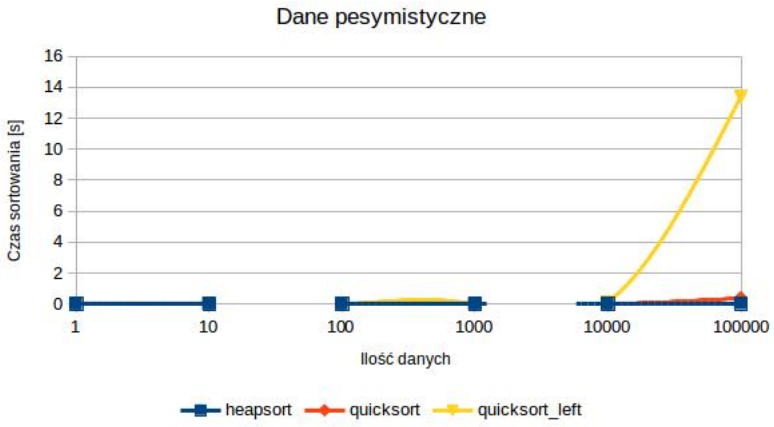
2 Wyniki

Przeprowadzono testy szybkości sortowania dla 3 rodzajów sortowań:
- quicksort_left (piwot pierwszy element tablicy)
- quicksort (piwot srodkowy element tablicy)
- heapsort

Testy przeprowadzono dla dwóch rodzajów danych:
- dane losowe
- dane pesymistyczne (spreparowane, aby pokazać pesymistyczny przypadek dla quicksort_left)

Na podstawie otrzymanych danych sporządzono wykresy:





3 Wnioski

Analizując wykresy można zauważyć, że dla danych losowych, czasy sortowania dla wszystkich algorytmów są bardzo do siebie zbliżone. Nie jestem pewien, dlaczego dla danych losowych heapsort wypadł lepiej niż oba rodzaje quicksorta.
W przypadku danych pesymistycznych, można zauważyć wyraźny wzrost czasu sortowania dla algorytmu quicksort_left. Czasy quicksort i heapsort pozostały podobne.
W przypadku większej ilości danych, wystąpienie przypadku pesymistycznego mogłoby spowodować poważne problemy, dlatego należy zadbać o odpowiednią optymalizację dobierania piwota. Można również opracować algorytm rozpoznawania przypadków pesymistycznych i w tych wyjątkowych wypadkach używać heapsorta.