Sprawozdanie z ćwiczenia laboratoryjnego VI
Sortowanie Hybrydowe

Bartłomiej Ankowski

23.04.2015

1 Wstęp

Celem laboratorium było zaprojektowanie oraz zaimplementowanie algorytmu sortowania hybrydowgo, będącego połączeniem sortowania szybkiego i sortowania przez wstawianie. Dodatkowo należało zaimplementować podczas laboratorium wybrany algorytm sortowania, wybrano sortowanie przez kopcowanie. Należało również usprawnić algorytm doboru pivotu w sortowaniu szybkim, co zostało ukończone podczas wcześniejszego laboratorium(numer 4).

2 Relizacja

Sortowanie hybrydowe pozwala na skrócenie średnich czasów sortowania. WYkorzystyuje sie przy tym fakt, iż algorytmy które posiadają złożoność $O(n^2)$ dla tablic o małym rozmiarze działaja relatywnie szybko w stosunku do algorytmów posiadających złożoność $O(nlogn)$. Należy zatem wywołać szereg rekurencyjnych wywołań Partycjowania(wydzielona metoda z sortowania szybkiego), tak aby tablica wejściowa została podzielona na podzbiory o rozmiarze nie przekraczajacym określonego progu. Każdy z podzbiorów jest rodzielony przez pivot, który jest dobierany przez algorytm szukania mediany z trzech .Dla tak częściowo posortowanej tablicy wywoływane jest sortowanie przez wstawianie, które ma za zadanie posortować wydzielone podzbiory tablicy wejściowej.

3 Złożoność obliczeniowa

3.1 Sortowanie przez kopcowanie

- Jest to algorytm sortowania w miejscu, nie potrzebuje żadnej pomocniczej struktury danych zatem złożoność pomocnicza wynosi $O(1)$, całkowita zaś wynosi $O(n)$, i jest zależna od ilości sortowanych danych
- Pierwszy etap sortowania polega na utworzeniu kopca, należy przy tym pamiętać,iż przy dodaniu nowego elementu, nie może on być większy od swojego rodzica. Wobec tego niezbędne są dodatkowe porównania o

ilości nie większej niż "piętro"drzewa. Należy zatem wykonać n kroków, każdy o logarytmicznym koszcie, złożoność tej częsci wynosi $O(nlogn)$.

- Drugi etap polega na właściwym sortowaniu kopca, polega na usunięciu wierzchołka kopca, zawierający element maksymalny i wstawieniu w jego miejsce elementu z końca kopca i odtworzenie porządku kopca.Usunięty element maksymalny jesy wstawiony na końcu kopca. Wykonuje się zatem n kroków o koszcie logarytmiczny. Złożoność tej częsci wynosi zatem również $O(nlogn)$.

- Zatem całkowita złożoność tego algorytmu wynosi $O(nlogn)$.

3.2 Sortowanie Hybrydowe

- Składa się z sortowania szybkiego i sortowanie przez wstawianie, przy czym tworzenie podzbiorów odbywa się przy pomocy partycjowania majacącego złożonośc $O(nlogn)$,końcowe sortowanie zbiorów odbywa się przy pomocy sortowania przez wstawianie posiadającego złożoność $O(n)$ dla tablic o małym rozmiarze.

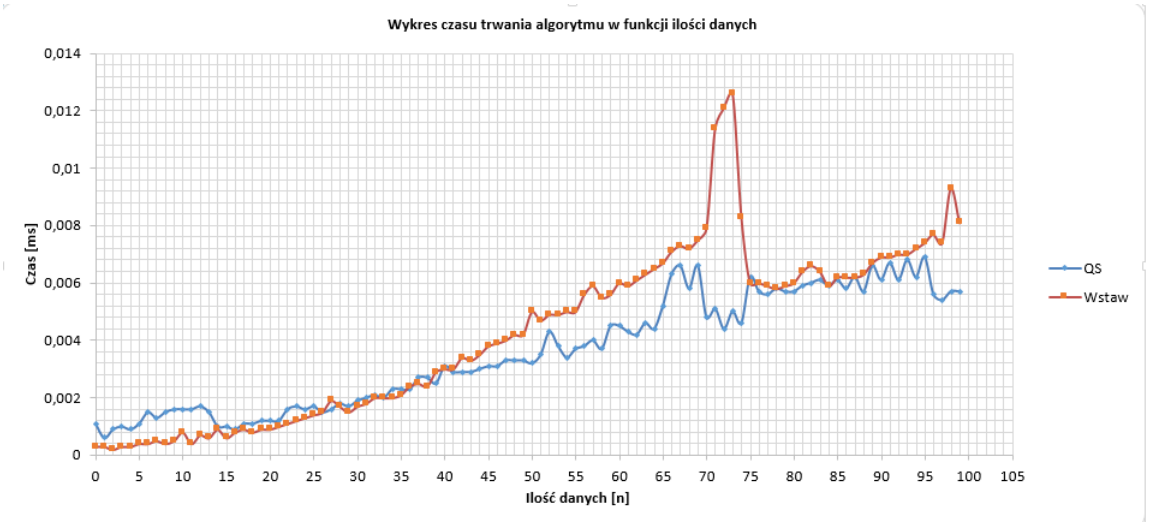
- Zatem całkowita złożoność tego algorytmu wynosi $O(nlogn)$.

4 Wyniki pomiarów

Badania przeprowadzono dla tablic o różnych wielkościach od 100 do 1mln,dla każdego rozmiaru test był przeprowadzany 10-krotnie.Jako końcowy wynik jest przyjmowany czas średni.

4.1 Określenie progu

W ramach określenia progu, dla którego sortowanie przez wstawianie jest szybsze od Quick sort, dokonano testu polegającego na pomiarze czasu sortowania tablic o rozmiarze od 1 do 100.



Rysunek 1: Porównanie sortowania przez wstawianie i szybkiego

Na podstawie przeprowadzonych testów, próg został ustalony na poziomie 13 elemenetów.

4.2 Sortowanie przez Kopcowanie

Wyniki otrzymane dla testowania algorytmu sortowania przez kopcowanie:

Ilość danych[n]	Czas[ms]
100	0,0347
1000	0,5329
10000	2,6395
100000	27,7894
1000000	331,032
10000000	3882,41

Tablica 1: tabela wyników testu sortowania przez kopcowanie

Rysunek 2: Wynik testu sortowania przez kopcowanie

4.3 Sortowanie Hybrydowe

Wyniki otrzymane dla testowania algorytmu sortowania hybrydowego. Testy zostały przeprowadzone dla podzbiorów o rozmiarze 13. Wartości w kolumnie przyrost oznaczają procentowy spadek czasu sortowania w porównaniu do sortowania szybkiego z optymalizacją dobotu pivotu.

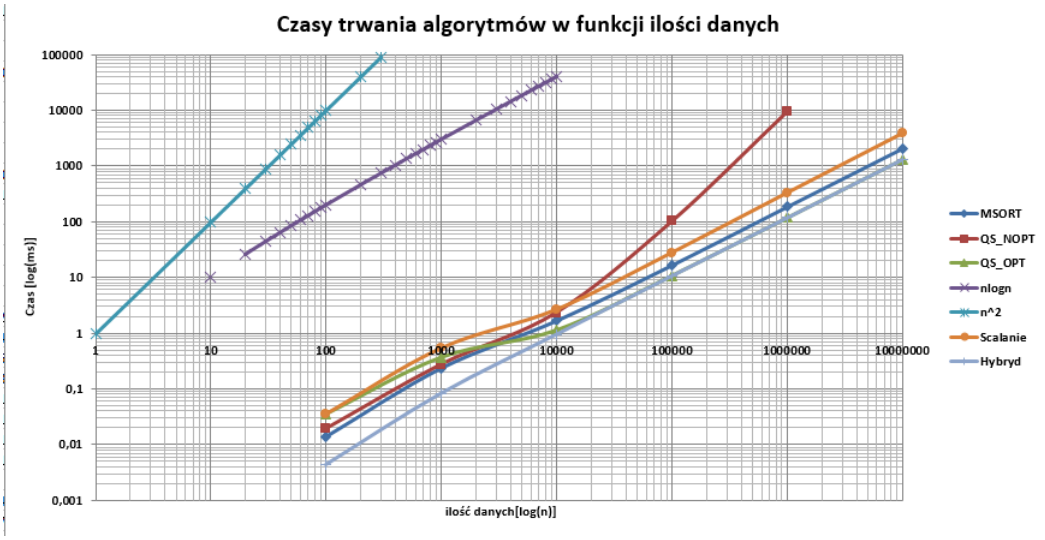
Ilość danych[n]	Czas[ms]	Przyrost[]
100	0,0043	52,2
1000	0,0827	45,6
10000	0,9497	25,0
100000	10,692	6,4
1000000	116,527	4,4
10000000	1305,45	3,52

Tablica 2: Wyniki testu sortowania hybrydowego

Rysunek 3: Wyniki testu sortowania hybrydowego

4.4 Porównanie badanych algorytmów

Zestawienie i porównanie wszystkich zaimplementowanych algorytmów na jednym wykresie:



Rysunek 4: Porównanie badanych algorytmów

5 Wnioski

- Na podstawie rysunku pierwszego został określony próg na poziomie 13, jednakże zwiększanie tego progu powodowało wzrost czasu sortowania dla małych tablic i spadek czasu sortowania dla dużych tablic(rzędu 30%).
- Testy sortowanie przez kopcowanie przyniosły oczekiwany rezutat,algorytm ten posiada złożonośc $O(nlogn)$. Przy czym dla duzych tablic(rzędu 1-10 mln) czasy są duzo gorsze w porównaniu z sortowaniem szybkim.
- Na podstawie tabeli numer 2 i rysunku numer 4 można stwierdzić fakt, iż sortowanie hybrydowe przyniosło spodziewany efekt w postaci poprawy średnich czasów sortowania w sotsunku do sortowania szybkiego. Spdadek procentowego przyrostu wraz ze wzrostem ilości sortowanych elementów prawdopodobnie spowodowane jest wzrostem liczby rekurencyjnych wywołań.
- Rysunek numer 4 jest dowodem na to, iż wszytskie zaimplementowe algorytmy sortowania posiadją złożoność $O(nlogn)$ co jest zgodne z założeniami i teoretyczną złożonościa tych algorytmów.