Sprawozdanie z ćwiczenia laboratoryjnego

Bartłomiej Ankowski

7.04.2015

Spis treści

1 Wstęp

Celem przeprowadzonego ćwiczenia była analiza złożoności obliczeniowej algorytmów sortowania, wykorzystując do tego zamodelowane wcześniej struktury danych. Testy zostały przeprowadzone na liście podwajającej, ze względu na najszybszy czas zapełniania tej struktury danymi.

Algorytmy, które były testowane :

- szybkie(z optymalizacją doboru pivot)
- przez scalanie

2 Realizacja

Program pracuję na wygenerowanych pseudo losowych liczbach wczytanych z pliku i użytych do zapełnienia wybranej strktury danych. Zostały zaprojektowane dwie wersję algorytmu sortowania szybkiego. W pierwszym, pivot jest wybierany jako średnia arytmetyczna indeksów skrajnych elementów tablicy danych. W drugim przypadku pivot jest dobierany jako mediana z trzech elementów tj. początkowy, środkowy i końcowy.

W przypadku sortowania poprzez scalanie, algorytm ten rozkłada tablicę na pojedyncze elementy, które z definicji muszą być posortowane, a następnie składa posortowane fragmenty, aż do otrzymania pełnej posortowanej tablicy.

3 Pomiary

Dane do testów były wczytywane z plików zewnętrznych, program pracował na danych typu całkowitego. Próby dokonywane były dla 7 różnych ilości danych wejściowych. Dla większej wiarygodności, dla każdej ilości danych testy były przeprowadzane 10 razy,

Wyjście programu jest zapisywane w pliku zewnętrznym, zawiera on średni czas dla poszczególnej porcji danych

4 Wyniki pomiarów

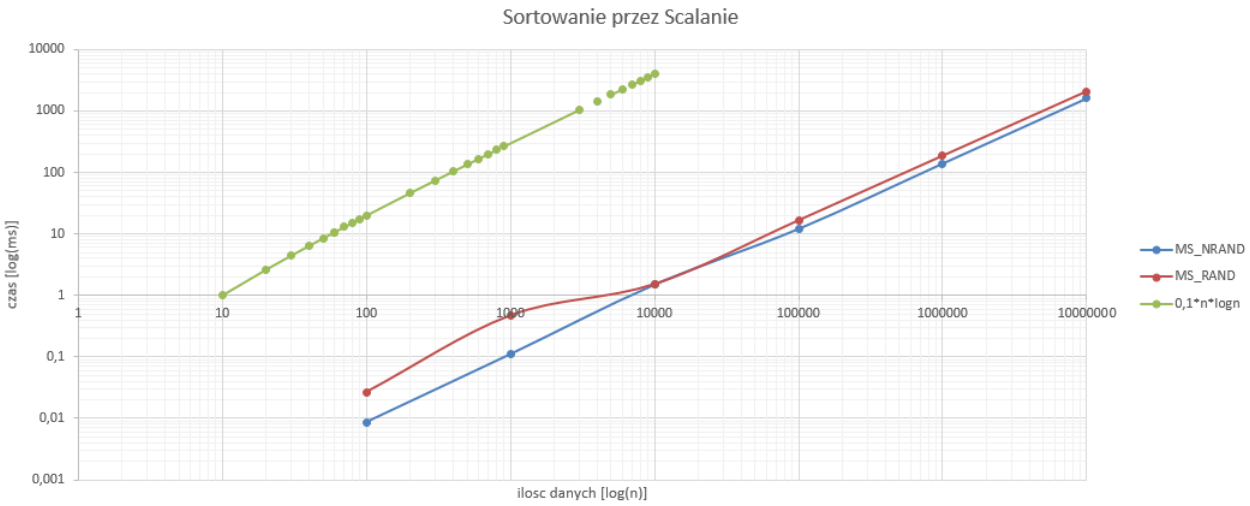
Po przeprowadzeniu serii pomiarów otrzymano wyniki przedstawione w tabelach. Na podstawie wyników utworzono wykresy zamieszczone poniżej.

4.1 Algorytm sortowania przez scalanie

Wyniki pomiarów dla sortowania przez scalanie, zostały przeprowadzone dla przypadków, gdy tablica jest wypełniona losowymi wartościomi i w przypadku, gdy tablica została już posortowana. Czasy podawane są w milisekundach.

Wielkość	MSort(RAND)	MSort(NRAND)
100	0,0263	0,001
1000	0,4638	0,110
10000	1,5013	1,492
100000	16,399	12,067
1000000	184,629	139,486
10000000	2064,23	1605,88

Tablica 1: Sortowanie przez Scalanie



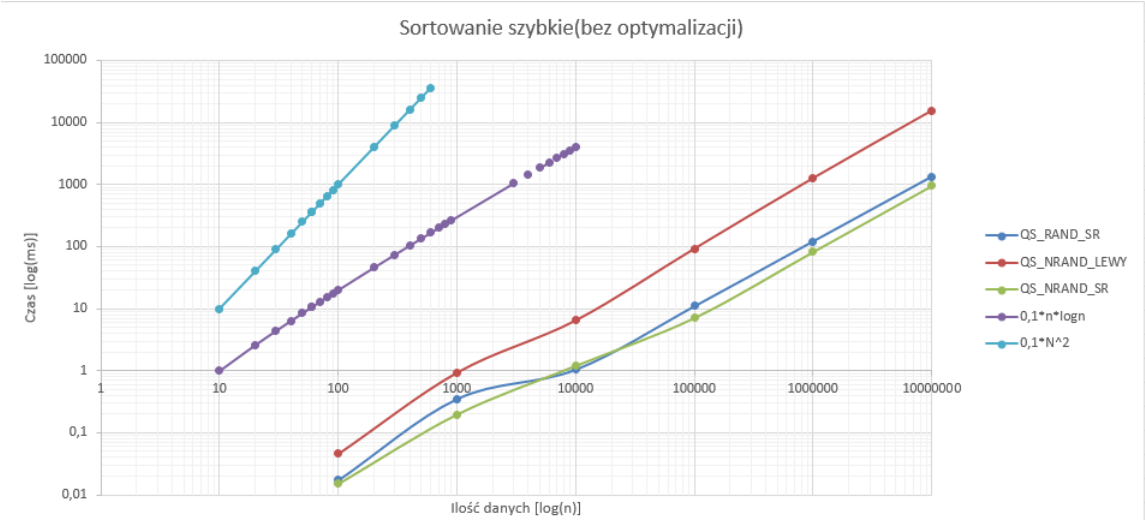
Rysunek 1: Sortowanie przez Scalanie

4.2 Algorytm sortowania szybkiego(bez optymalizacji)

Wyniki pomiarów dla sortowania przez szybkiego, zostały przeprowadzone dla przypadków, gdy tablica jest wypełniona losowymi wartościomi i w przypadku, gdy tablica została już posortowana. Czasy podawane są w milisekundach.

Wielokość	QS(NRAND L)	QS(NRAND SR)	QS(RAND SR)
100	0,045	0,015	0,0175
1000	0,9331	0,197	0,3546
10000	6,4816	1,202	1,052
100000	92,873	7,109	11,045
1000000	1269,03	82,398	121,667
10000000	15473,3	955,508	1340,61

Tablica 2: Sortowanie szybkie(bez optymalizacji)



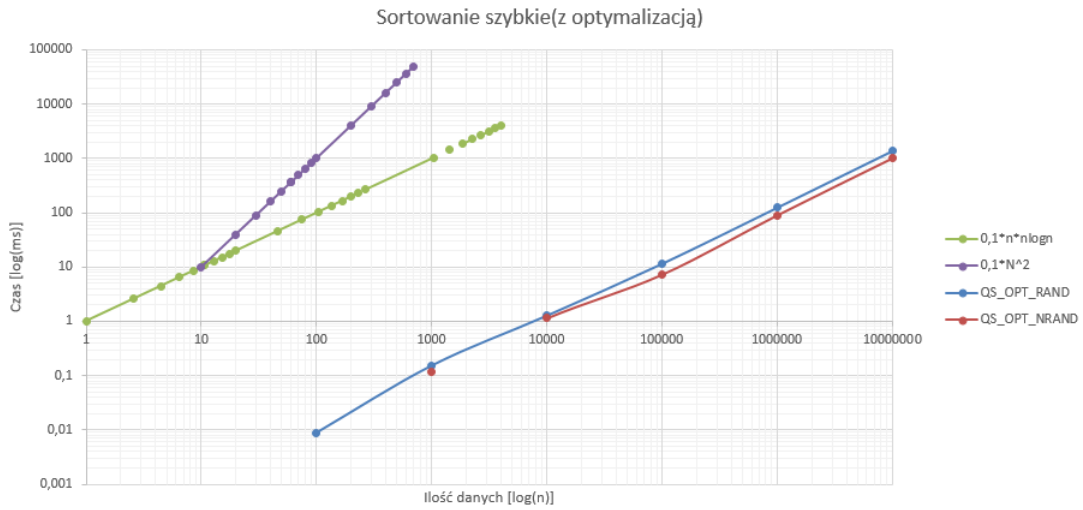
Rysunek 2: Sortowanie szybkie(bez optymalizacji)

4.3 Algorytm sortowania szybkiego(z optymalizacją)

Wyniki pomiarów dla sortowania przez szybkiego, zostały przeprowadzone dla przypadków, gdy tablica jest wypełniona losowymi wartościomi i w przypadku, gdy tablica została już posortowana. Czasy podawane są w milisekundach.

Wielkość	QS(RAND)	QS(NRAND)
100	0,009	0,002
1000	0,152	0,1186
10000	1,266	1,148
100000	11,420	7,269
1000000	123,971	89,341
10000000	1373,93	994,274

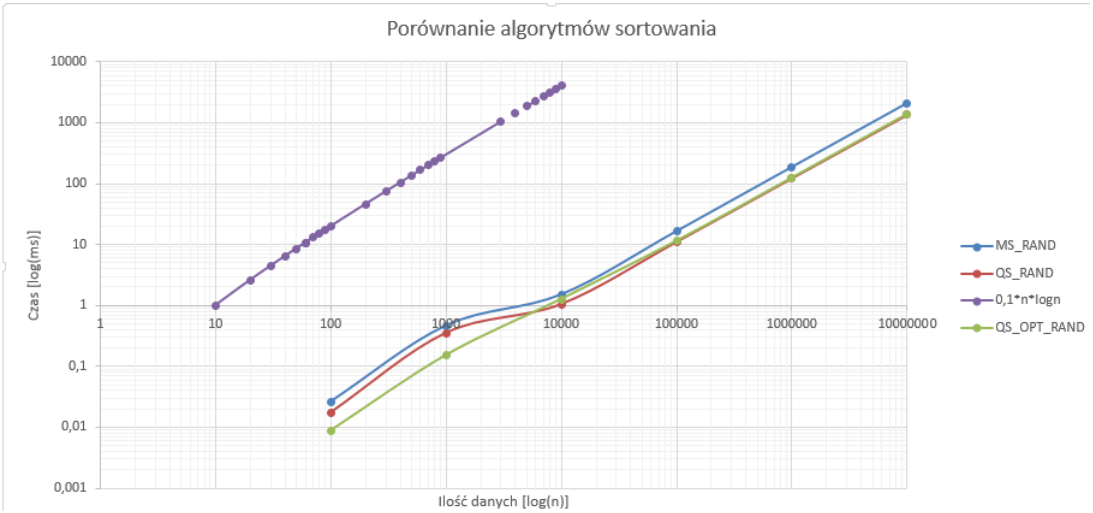
Tablica 3: Sortowanie szybkie(z optymalizacją)



Rysunek 3: Sortowanie szybkie(z optymalizacją

4.4 Porównanie algorytmów

Przedstawienie złożoności obliczeniowej wszystkich badanych algorytmów.



Rysunek 4: Porównanie algorytmów

5 Wnoiski

1. Wszystkie badane algorytmy mają podobną złożoność obliczeniową,wynosząca $nlogn$.

2. Na podstawie Rysunku 1. można wyciągnać wniosek, iż nie ma dużego znacznia dla sortowania przez scalanie fakt, czy wejściowa tablica została już wcześniej posortowana

3. Na Rysunku 2 widać z kolei, iż jeśli tablica wejściowa jest posortowana i wybierany będzię skrajny element tablicy jako pivot(najmniejszy lub największy), wówczas nastąpi gwałtowny wzrost czasu wykonywania operacji.Złożoność jaką posiada ta operacja nie jest właściwa, ponieważ powinna wynosić $O(n^2)$. Pomimo usilnych prób nie udało się uzsykać takiego wyniku.Problem leży najprawdopodoniej w błędnej implementacji.

4. Wprowadzona optymalizacja sortowania szybkiego przyniosła zamierzany efekt.Na podstawie rysunku 3, można dojść do wniosku, iż metoda wyboru pivotu na podstawie mediany z trzech elementów zabezpiecza algorytm przed wyborem najmniejszego elementu jako pivot.

5. Najszybszym badanym algorytmem sortowania dla tablic wypełnionymi losowymi wartościami okazał się refQuick Sort bez optymalizacji wyboru pivotu.Aczkolwiek wynik jest mocno zbliżony dla wersji z optymalizacją.