Sprawozdanie LAB6

Arkadiusz Ziółkowski

23.04.2015r

1 Cel ćwiczenia

Celem ćwiczenia jest implemetacja algorytmu Sortowania przez Kopcowanie, oraz Sortowania Hybrydowego. Następnie należy zbadać ich złożoności obliczeniowe i porównać.

2 Teoretyczna złożoność obliczeniowa

2.1 Sortownie Szybkie

- Zgodnie ze sprawozdaniem do lab4 złożonośc algorytmu Sortowania Szybkiego (zoptymalizowanego ze względu na wybór pivota - mediana z trzech) wyraża się dla każdego przyapdku w $nlogn$.

2.2 Sortowanie przez Kopcowanie

- Podczas sortowania tworzone jest drzewo binarne. Jak wiadomo złożonośc takiej operacji jest liniowa: $O(n)$
- Dodatkowo musi zostać zachowana struktura kopca (nowy element nie może być większy od swojego przodka). Wobec czego muszą zajśc dodatkowe porównania, a ich maksymalna ilośc jest równa wysokości drzewa, która wyraża się wzorem $log_2 n$. Więc porównania generują dodatkową złożonośc $O(logn)$.
- Zatem całkowita złożoność obliczeniowa Sortowania przez Kopcowanie wyraża się w $O(nlogn)$.

2.3 Sortowanie Hybrydowe

- Sortowanie Hybrydowe wykorzystuje algorytm Sortowania Szybkiego jako główny oraz, algorytm Sortowania przez Wstawianie jako pomocniczy.
- Dla dużych rozmiarów tablic kluczową rolę odgrywa algorytm Sortowania Szybkiego zatem spodziewana złożonośc obliczeniowa wyraża się w $O(nlogn)$.

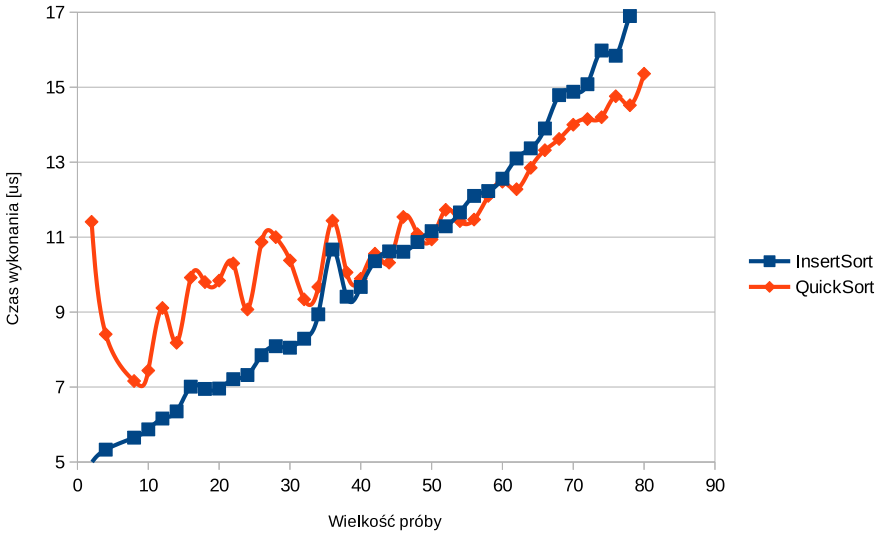
3 Wyniki pomiarów

Pomiary czasu wykonywania algorytmów zostały wykonane na podstawie losowo wygenerowanych danych wejściowych po 50-100 razy dla każdej ich liczności.
Czasy obliczeń są średnimi czasami otrzymanymi z serii pomiarów.
Liczność jest to ilość elementów do posortowania
Δ - wyrażony w procentach przyrost wydajności między Sortowaniem Szybkim a Hybrydowym

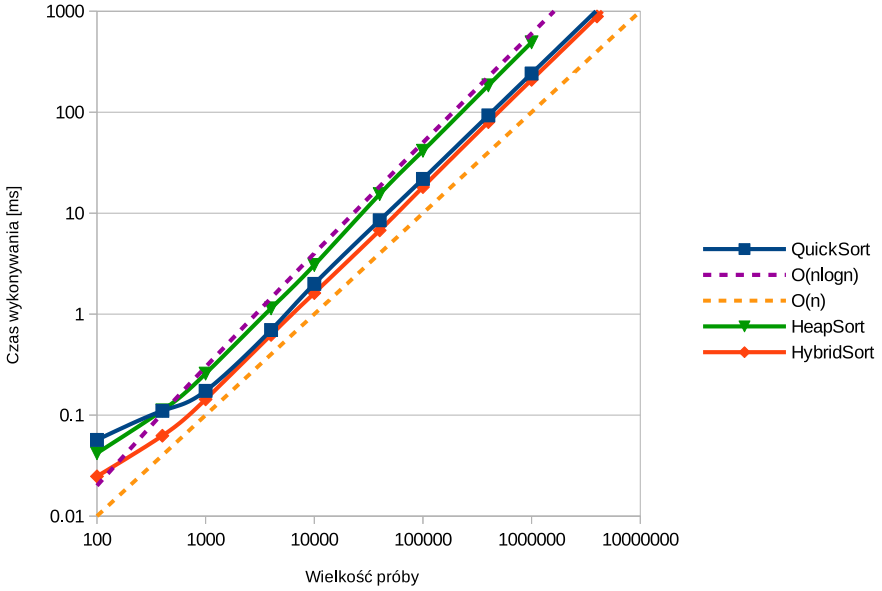
Liczność	**Czas obliczeń [us]**	
	Insert	**Quick**
2	4,89	11,41
4	4,98	11,41
6	5,33	8,41
8	5,65	7,16
10	5,87	7,44
12	6,16	9,11
14	6,35	8,18
16	7,01	6,95
18	6,95	9,80
20	6,96	9,84
22	7,21	10,30
24	7,32	9,07
26	7,85	10,87
28	8,09	11,00
30	8,05	10,38
32	8,29	9,34
34	8,94	9,67
36	10,67	11,44
38	9,41	10,06
40	9,67	9,89
42	10,36	10,56
44	10,62	10,32
46	10,61	11,54
48	10,87	11,08
50	11,16	10,64
52	11,29	11,73
54	11,66	11,43
56	12,10	11,47
58	12,23	12,10
60	12,56	12,48
70	14,88	14,00
80	18,03	15.36
90	19,08	15,99
100	21,31	17,69

Liczność	**Czas obliczeń [ms]**			**Δ**[%]
	Quick	**Hybrid**	**Heap**	
10^2	0,05682	0,02474	0,04197	56,46
$4 * 10^2$	0,05682	0,02474	0,04197	56,46
$4 * 10^2$	0,11026	0,06254	0,11144	43,28
10^3	0,17382	0,14322	0,25760	17,60
$4 * 10^3$	0,69606	0,62268	1,14079	10,54
10^4	1,99428	1,64904	3,07001	18,82
$4 * 10^4$	8,51302	6,76212	15,4335	30,57
10^5	21,84270	18,16020	41,4926	16,86
$4 * 10^5$	93,00520	79,73590	184,274	14,27
10^6	241,86000	208,98800	491,435	13,59
$4 * 10^6$	1025,02000	886,96800	-	13,47
10^7	2686,60000	2364,55000	-	11,99

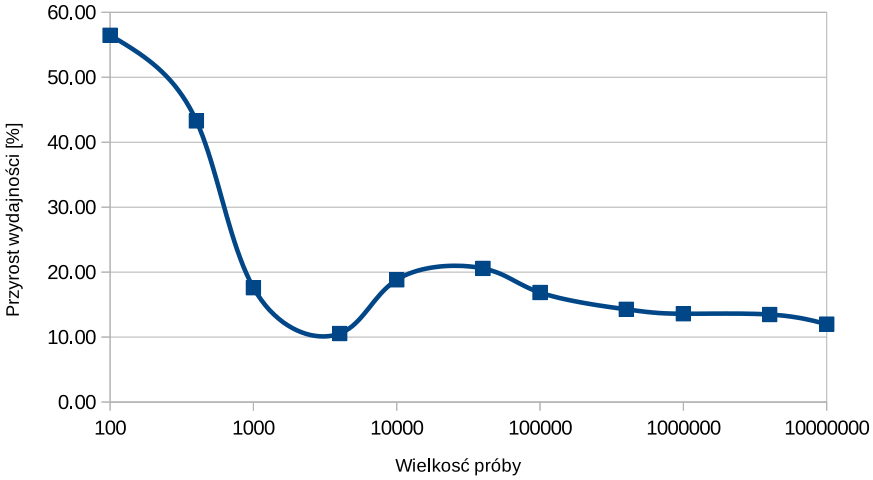
Tabela 1. Wyniki pomiarów



Rysunek 1. Wykres czasu od ilości danych dla małych rozmiarów próby



Rysunek 2. Wykres czasu od wielkości próby (porównanie algorytmów)



Rysunek 3. Wykres procentowego przyrostu wydajności od wielkości próby

4 Wnioski

- Na rysunku nr 1 widać, że dla rozmiaru listy mniejszego od ok. 40 elementów Sortowanie przez Wstawianie okazuję się sybsze od Szybkiego Sortowania. W związku z tym do Sortowania Hybrydowego zastosowałem próg równy 32 elementy, poniżej którego zamiast rekurencyjnego wywołania algorytmu Szybkiego Sortowania zostanie wywołany algorystm Sortowania przez Wstawianie.
- Z wykresu na rysunku nr 2 wynika, iż złożoności wszystkich trzech algorytmów sortownia (Szybkiego, Hybrydowego i przez Kopcowanie) zgadzają się z oczekiwaniami teoretycznymi i wyrażają się w $O(nlogn)$.
- Z rysunku nr 2 widać również, że Sortowanie przez Kopcowanie w potównaniu do Sortowania Szybkiego dla małych rozmiarów listy jest wydajniejsze czasowo, natomiast dla dużych rozmiarów problemu potrzebuje więcej czasu.
- Rysunek 2. przedstawia również różnicę między Sortowaniem Szybkim a Hybrydowym. Widać, że Sortowanie Hybrydowe potrzebuje nieco mniej czasu do postortowanie listy.
- Zgodnie z rysunkiem 3. przyrost wydajności czasowej Sortowania Hybrydowego względem Szybkiego nie jest stały i dla małych rozmiarów problemu osiąga wartości ok. 60%, i wraz ze zwiększaniem liczności elementów do posortowania maleje i dla 10^7 elementów wynosi ok 10%.