Sprawozdanie LAB4

Arkadiusz Ziółkowski

09.04.2015r

1 Cel ćwiczenia

Celem ćwiczenia jest zbadanie złożoności obliczeniowej algorytmu Szybkiego Sortowania przed i po jego optymalizacji ze względu na wybór pivota. Dodatkowo implemenatcja algorytmu sortowania przez scalanie.

2 Złożoność obliczeniowa

2.1 Quick Sort

1. Przypadek Optymistyczny

   Jako pivot zawsze wybieramy medianę,
   liczba porównań wyraża się wzorem:

   $$T(n) = (n-1) + 2T\frac{n-1}{2}$$

   Zatem po rozwinięciu złożoność wyraża się w $O(nlogn)$

2. Przypadek pesymistyczny

   Jako pivot zawsze wybieramy element największy lub najmnijeszy, wtedy

   $$T(n) = (n-1) + T(n-1) = \frac{n^2-n}{2}$$

   Więc złośoność obliczeniowa jest w $O(n^2)$

3. Przypadek przeciętny

   Gdy lista z danymi wejściowymi ma równomierny rozkład
   prawdopodobieństwa to złożoność obliczeniowa wynosi

   $$T(n) = 1.39nlogn$$

2.2 MergeSort

- zakładamy, że długość ciągu do posortowania jest potęgą liczby 2,
- ciągi jednoelementowe możemy posortować w stałym czasie,
- sortownie ciągu n elementowego to scalanie dwóch ciągów $\frac{n}{2}$-elementowych, czyli

$$T(n) = 2T(\tfrac{n}{2}) + O(n)$$

- rozwijając rekurencyjnie powyższy ciąg otrzymujemy

$$T(n) = 2(2(...2(T(1) + 2)...) + \tfrac{n}{2}) + n \text{ ,gdzie } n = 2^k$$

- po rozwinięciu otrzymujemy czas $T(n) = 2nlogn$
- zatem złożonośc algorytmu wyrażona jest w $O(nlogn)$

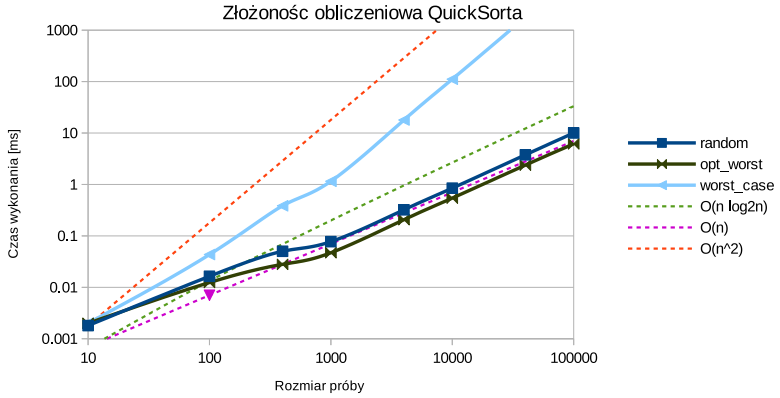
3 Wyniki pomiarów

Rozmiar próby	Średni czas obliczeń [ms]				
	Rand	Rand(opt)	Worst	Worst(opt)	MergeSort
10^1	0,0018	0,0020	0,0020	0,0020	0,0037
10^2	0,0164	0,0154	0,0434	0,0124	0,0458
$4 * 10^2$	0,0502	0,0328	0,3842	0,0280	0,0836
10^3	0,0773	0,0763	1,1534	0,0470	0,1915
$4 * 10^3$	0,3216	0,3212	17,8963	0,2064	0,7623
10^4	0,8470	0,8486	111,3330	0,5429	2,0032
$4 * 10^4$	3,7611	3,7674	1778,4100	2,3750	8,4507
10^5	10,0017	10,1763	11104,3000	6,1393	20,8231

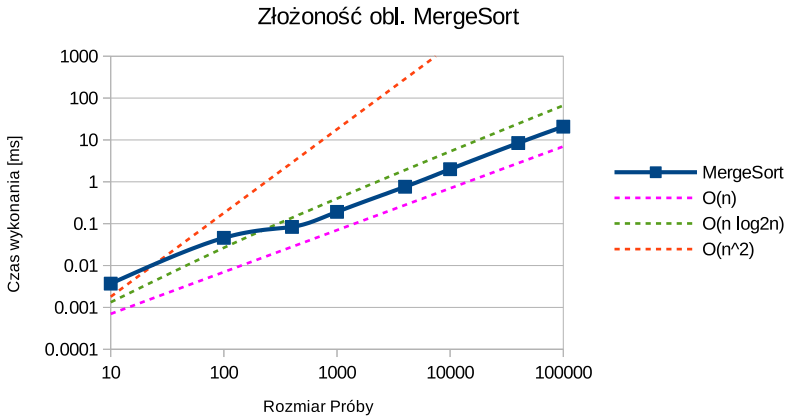
4 Wnioski

- Złożoności obliczeniowe algorytmu sortowania szybkiego otrzymane na podstawie pomiarów i odczytane z wykresu (Rysunek 1.) pokrywają się ze zpodziewanymi złożonościami ukazanamymi w punkcie 2. ($O(nlogn)$ dla przypadku losowego i $O(n^2)$ dla przypadku pesymistycznego).
- Optymalizacja algotytmu ze względu na wybór pivota (mediana z trzech wartości) poprawiła wydajnośc algorytmu w najgorszym przypadku $O(n^2)$ co widać na rysunku 1 - udało się uzyskać przewidywaną złożoność $O(nlogn)$.
- Zmiana czasów dla przypadku losowego dla algorytmu po optymalizacji nie uległa widoczej zmianie w związku z tym nie została przedstawiona na wykresie.
- Złożoność obliczeniowa algorytmu sortowania przez scalanie otrzymana na podstawie pomiarów (rys 2) jest zgodna z oczekiwaniami teoretycznymi - $O(nlogn)$.
- Porównując QuickSort i MergeSort można zauważyć, że MergeSort charakteryzuje się większym zapotrzebowaniem na pamięć operacyjną. Ze względu na implementacje (alokownie podczas sortowania pomocniczej tablicy) algorytm MergeSort wykazuje nieco gorsze czasy w porównaniu do QuickSorta dla losowych danych mimo tej samej klasy złożoności obliczeniowej. Różnice te możemy zaobserować na rysunku nr 3.

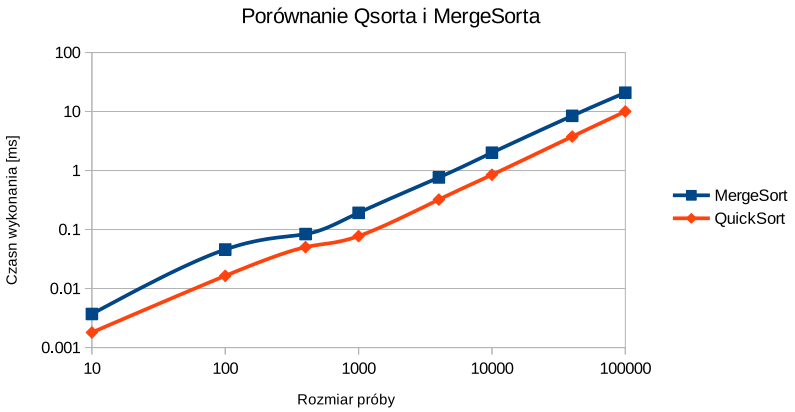
5 Wykresy



Rysunek 1: Wykres czasu od rozmiaru próby różnych przypadków QuickSorta



Rysunek 2: Wykres czasu od rozmiaru próby dla MergeSorta



Rysunek 3: Porównanie QuickSorta i MergeSorta