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 złożoność wyraża się w $O(nlog_2 n)$

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

   $$T(n) = 1.39nlog_2 n$$

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

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

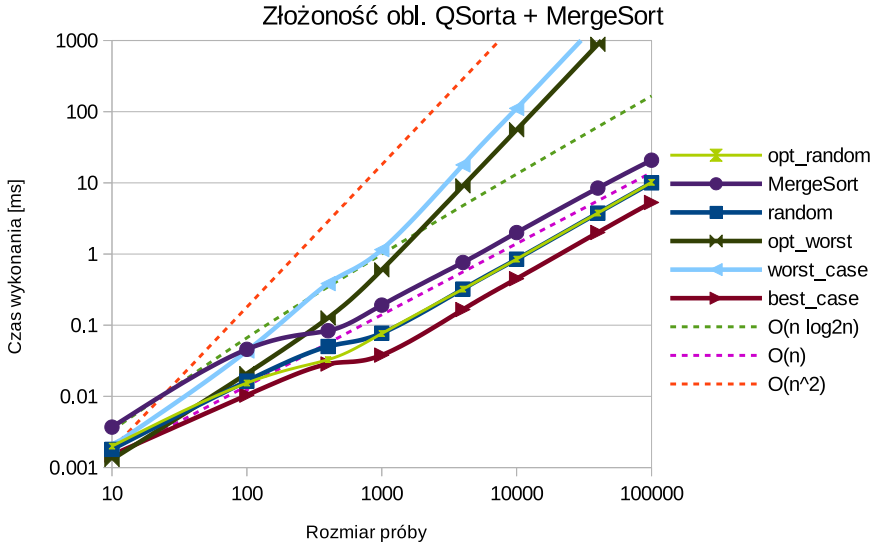
- rozwijając rekurencyjnie powyższy ciąg otrzymujemy

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

- 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)	Best	Worst	Worst(opt)	MergeSort
10^1	0,0018	0,0020	0,0015	0,0020	0,0013	0,0037
10^2	0,0164	0,0154	0,0104	0,0434	0,0209	0,0458
$4 * 10^2$	0,0502	0,0328	0,0285	0,3842	0,1267	0,0836
10^3	0,0773	0,0763	0,0380	1,1534	0,5995	0,1915
$4 * 10^3$	0,3216	0,3212	0,1658	17,8963	9,0689	0,7623
10^4	0,8470	0,8486	0,4494	111,3330	55,7461	2,0032
$4 * 10^4$	3,7611	3,7674	2,0097	1778,4100	885,5880	8,4507
10^5	10,0017	10,1763	5,3017	11104,3000	5534,9700	20,8231



4 Wnioski

- Złożoności obliczeniowe algorytmu sortowania szybkiego otrzymane na podstawie pomiarów i odczytane z wykresu pokrywają się ze zpodziewanymi złożonościami teoretycznymi.
- Optymalizacja algotytmu ze względu na wybór pivota (mediana z trzech wartości) poprawiła dwukrotnie złożoność obliczeniową dla najgorszego przypadku, natomiast na pozostałe ma pomijalnie mały wpływ.
- Algorytm doskonale radzi sobie z dużymi rozmiarami nieposortowanych tablic, natomiast dużo gorsze wyniki otrzymuje dla tablic już posortowanych.
- Złożonośc obliczeniowa algorytmu sortowania przez scalanie otrzymana na podstawie pomiarów jest zgodna z oczekiwaniami teoretycznymi.
- 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, jednak jest algorytmem stabilnym więc jego złożoność zawsze wyraża się w $O(nlogn)$ w przeciwieństwie do konkurenta gdzie złożoność potrafi być kwadratowa.