

Sortowanie - Laboratorium nr 5 z PAMSI

Justyna Klijewska

30 03 2014

Zadanie do wykonania

W istniejącym już programie dodać sortowanie. Dodatkowo zmienić działanie quicksort - ma wybierać losowy element z tablicy.

Program był pisany w środowisku Windows.

Sortowanie	Zł cz optym	Zł cz typowa	Zł cz pesym	Zł. pamięciowa
Quicksort	-	$O(n \log n)$	$O(n^2)$	Zależnie od implementacji
Mergesort	$O(n)$	$O(n \log n)$	$O(n \log n)$	$O(n)$
Heapsort	-	$O(n \log n)$	-	$O(n)$

Tabela 1. Zależności między liczbą elementów w pliku a czasem wykonywania programu.

Liczby do posortowania w tablicach były wybierane losowo randn.

QUICKSORT

Jest to sortowanie szybkie i polega na zasadzie "dziel i zwyciężaj". Jego złożoność obliczeniowa to $O(n \log n)$. Jest jednym z najczęściej używanych ze względu na szybkość wykonywania oraz łatwość implementacji.

Zostały rozważone 3 warianty:

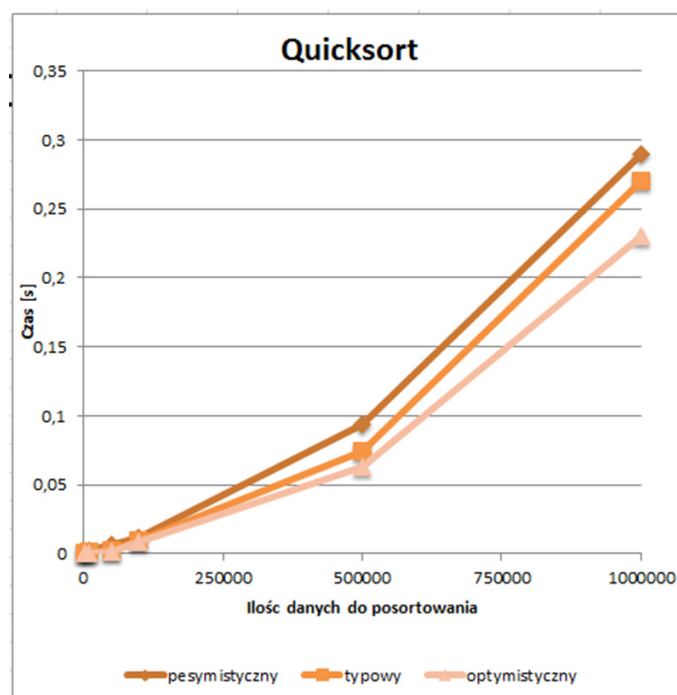
optymistyczny - użycie mediany

typowy - równomierny rozkład prawdopodobieństwa dla wyboru elementu z tablicy

pesymistyczny - zawsze wybieramy element najmniejszy lub największy

Ilość liczb	Wariant optymistyczny	Wariant typowy	Wariant pesymistyczny
1000	0	0	0
10000	0	0,001	0,002
50000	0,001	0,0028	0,0061
100000	0,009	0,01	0,012
500000	0,063	0,074	0,094
1000000	0,23	0,27	0,29

Tabela 2. Zależności między liczbą elementów w pliku a czasem wykonywania programu w przypadku quicksort. Czas podany jest w sekundach.



Rysunek 1: Wykres zależności liczby elementów w pliku od czasu wykonywania sortowania

MERGESORT

Jest to sortowanie przez scalanie i jego złożoność obliczeniowa to $O(n \log n)$. Podobnie jak quicksort korzysta z zasady "dziel i zwyciężaj". Zostały rozważone 3 warianty:

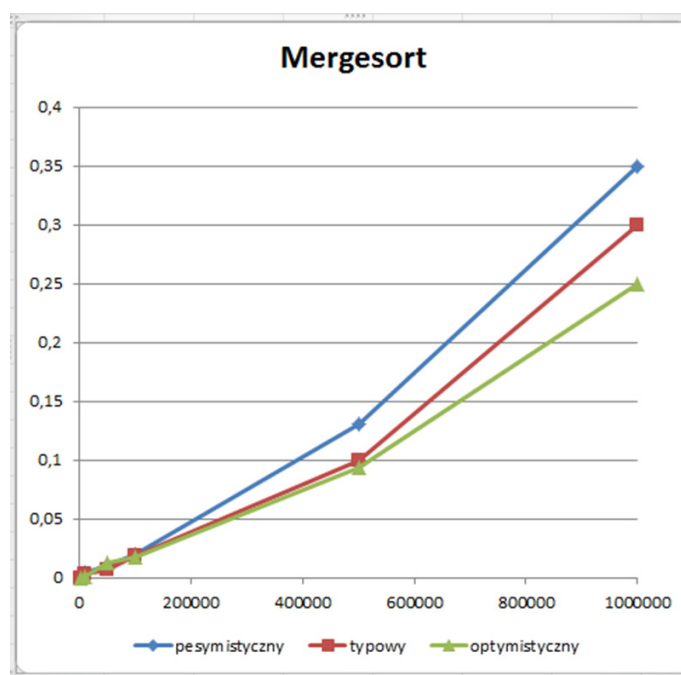
optymistyczny - dla zbiorów w miarę uporządkowanych

dla zbiorów o losowym rozkładzie elementów

pesymistyczny - dla zbiorów posortowanych odwrotnie

Ilość liczb	Wariant optymistyczny	Wariant typowy	Wariant pesymistyczny
1000	0	0	0,001
10000	0,001	0,0035	0,004
50000	0,012	0,0061	0,0098
100000	0,017	0,019	0,02
500000	0,093	0,1	0,13
1000000	0,26	0,3	0,35

Tabela 3. Zależności między liczbą elementów w pliku a czasem wykonywania programu w przypadku mergesort. Czas podany jest w sekundach.



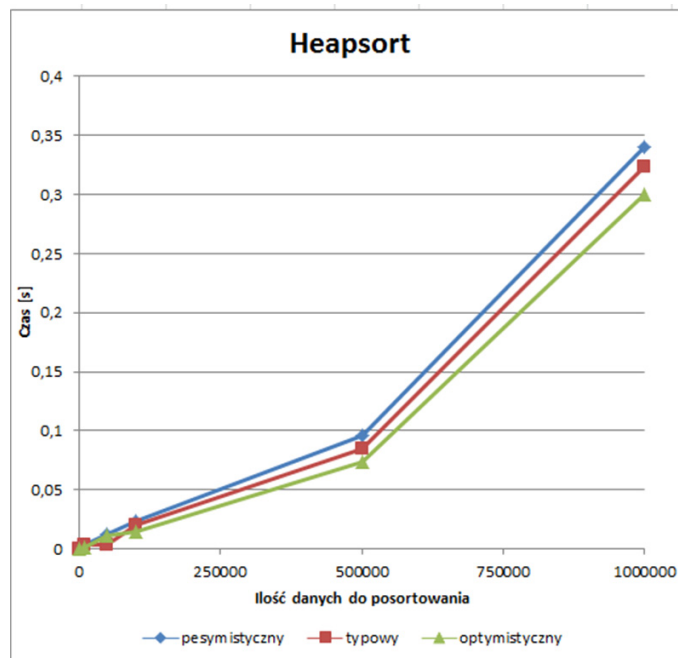
Rysunek 2: Wykres zależności liczby elementów w pliku od czasu wykonywania sortowania

HEAPSORT

Jest to to sortowanie przez kopcowanie. Złożoność czasowa wynosi $O(n \log n)$, a pamięciowa – $O(n)$. Jedną z największych jej zalet jest możliwość użycia tej samej tablicy w której znajdowały się nieposortowane elementy.

Ilość liczb	Wariant optymistyczny	Wariant typowy	Wariant pesymistyczny
1000	0	0	0
10000	0,001	0,003	0,003
50000	0,011	0,0037	0,012
100000	0,015	0,02	0,024
500000	0,074	0,085	0,096
1000000	0,3	0,32	0,34

Tabela 4. Zależności między liczbą elementów w pliku a czasem wykonywania programu w przypadku heapsort. Czas podany jest w sekundach.



Rysunek 3: Wykres zależności liczby elementów w pliku od czasu wykonywania sortowania

WNIOSKI:

Najpopularniejszym i najczęściej używanym sortowaniem jest quicksort. I ciężko się dziwić, gdyż w powyższych testach wykonywał on najszybciej swoje zadanie. Nie zmienia to faktu, że podane sortowania mają podobną złożoność obliczeniową, a co za tym idzie podobny czas wykonywania programu. Gdyby porównać powyższe z sortowaniem np.: bąbelkowym różnice byłyby większe. Różnice pomiędzy wariantami nie są bardzo duże. Na wpływ szybkości algorytmu ma wpływ zawartość tablicy (przy użyciu tablicy posortowanej, ale odwróconej proces ten zachodzi wolniej.