Sprawozdanie laboratorium 4 - sortowanie

200439

24 marca 2014

Sortowanie jest jednym z podstawowych problemów informatyki. Istnieje wiele algorytmów sortowania, między innymi sortowanie bąbelkowe czy sortowanie przez wstawianie. Wymienione algorytmy mają jednak dużą złożoność obliczeniową (kwadratową) i choć są bardzo proste w budowie i działaniu, czas ich wykonywania dyskwalifikuje je z powszechnego użycia przy większej ilości danych do posortowania.

Lepszymi algorytmami sortowania okazują się między innymi

- sortowanie przez kopcowanie (heap sort)
- sortowanie szybkie (quicksort)

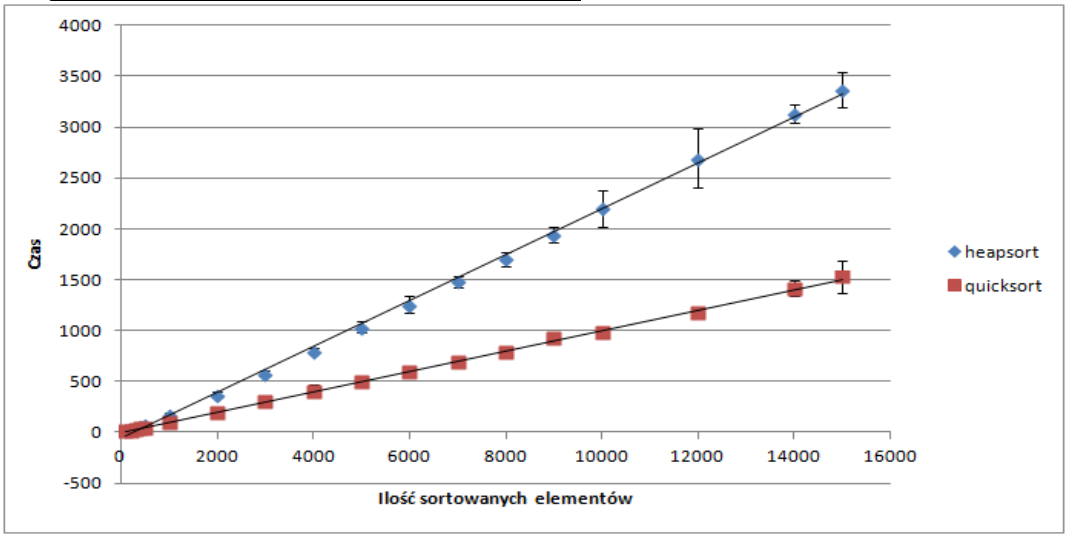
Przy sortowaniu przez kopcowanie wykorzystywana jest kolejka priorytetowa w formie kopca binarnego. Algorytm budowy kopca działa w ten sposób, że buduje najpierw małe kopce, z elementów znajdujących się na końcu tablicy, i później kopce te łączone są w większe, aż do powstania całego kopca. Po utworzeniu kopca następuje właściwe sortowanie. Usuwany jest wierzchołek kopca, który zawiera element maksymalny, a następnie wstawiany w jego miejsce jest element z końca kopca i odtwarzany jest porządek kopcowy. W zwolnione w ten sposób miejsce, zaraz za końcem zmniejszonego kopca wstawia się usunięty element maksymalny. Operacje te powtarza się aż do wyczerpania elementów w kopcu.

Sortowanie szybkie opiera się na zasadzie "dziel i zwyciężaj". Z tablicy wybiera się element rozdzielający, po czym tablica jest dzielona na dwa fragmenty: większe lub mniejsze od elementu rozdzielającego. Potem sortuje się osobno początkową i końcową część tablicy. Algorytm ten powtarzany jest rekurencyjnie do momentu, w którym powstają jednoelementowe tablice. Poszczególne wyniki łączone są w jedną posortowaną tablicę.

Dokonano porównania czasów wykonania sortowania liczb za pomocą dwóch wymienionych algorytmów. Algorytm wykonany został dla każdej ilości elementów tablicy 100 razy, po czym obliczony został średni czas oraz

odchylenie standardowe średniej. Do badań przyjęte zostały: losowe dane, tablica liczb ułożona rosnąco oraz tablica liczb ułożona malejąco. Czasy wykonania oraz złożoność obliczeniowa poszczególnych przypadków sortowania przedstawiona jest na wykresach oraz w tabelach. Czas podany jest każdorazowo w mikrosekundach.

	heap		quick	
liczba elementów	średnia	odchylenie	średnia	odchylenie
100	15,37	4,42	10,66	5,02
200	30,66	9,98	20,15	4,15
300	45,47	5,36	29,67	5,03
400	67,1	8,97	43,52	8,83
500	82,02	8,77	51,3	7,32
1000	174,16	13,21	97,36	4,27
2000	370,86	29,75	196,58	13,94
3000	578,58	28,48	304,17	34,88
4000	800,42	30,68	410,93	54,88
5000	1031,92	56,78	496,28	26,68
6000	1257,58	82,63	593,04	30,64
7000	1481,82	59,23	697,91	42,57
8000	1704,66	67,83	793,26	31,77
9000	1942,1	75,1	923,43	25,34
10000	2200,27	178,31	979,73	35,3
12000	2695,24	290,89	1179,08	52,61
14000	3129,52	89,18	1414,56	79,31
15000	3362,26	173,33	1531,99	160,6



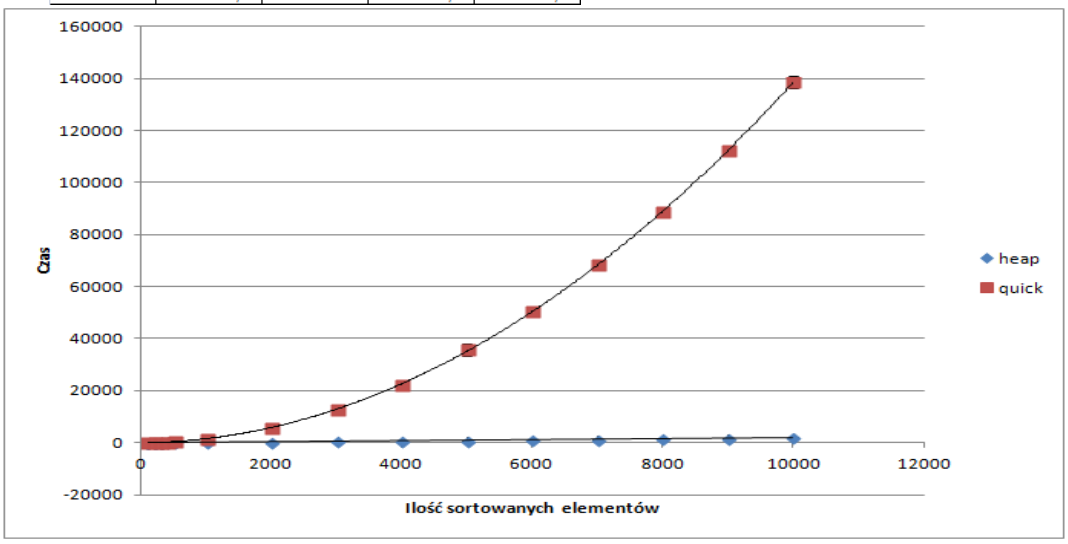
Wykres 1. Czasy sortowania tablicy elementów rozmieszczonych w kolejności losowej

Na powyższym wykresie widać, że algorytm sortowania szybkiego rzeczywiście jest szybszy od sortowania przez kopcowanie, choć złożoność czasowa obu algorytmów to O(n logn).

Kiedy jednak elementy tablicy będą już posortowane (rosnąco lub malejąco), algorytm sortowania szybkiego staje się praktycznie nie do użytku dla większych ilości danych. Jego złożoność obliczeniowa zwiększa się do kwadra-

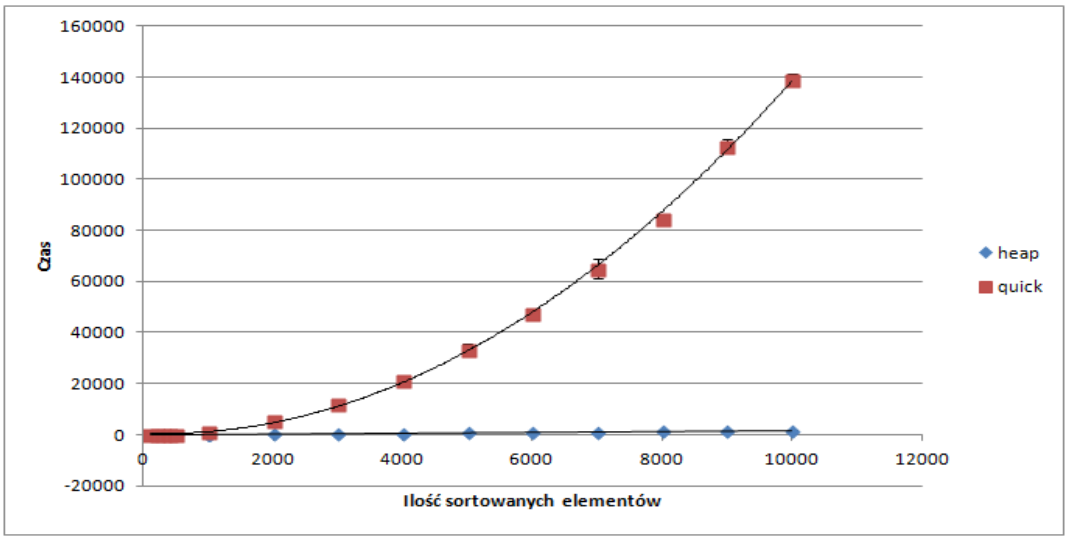
towej, podczas gdy czas sortowania przez kopcowanie nie zmienia się. Widać to dobrze na poniższych wykresach.

	heap		quick	
liczba elemer	średnia	odchylenie	średnia	odchylenie
100	30,11	5,57	39,1	6,65
200	31,44	6,25	69,89	5,02
300	42,16	5,01	138	8,99
400	59,05	5,08	249,39	18,8
500	76,51	11,56	391,56	46,79
1000	157,72	14,57	1445,75	59,83
2000	341,68	54,17	5669,64	193,17
3000	526,32	36,4	12871,16	593,13
4000	707,1	24,02	22454,7	747,45
5000	913,67	40,18	35792,11	2408,67
6000	1137,7	121,86	50360,53	1122,13
7000	1323,91	108,08	68351,44	1243,18
8000	1523,21	61,16	88967,16	1351,53
9000	1725,29	87,87	112200,27	1239,67
10000	1930,89	57	138636,29	2449,03



Wykres 2. Czasy sortowania tablicy elementów rozmieszczonych w uporządkowanej, rosnącej kolejności

Czas sortowania tablicy zawierającej elementy w odwróconej kolejności jest bardzo podobny. Oba te przypadki (uporządkowane tablice) są najgorszymi przypadkami dla algorytmu sortowania szybkiego.



Wykres 2. Czasy sortowania tablicy elementów rozmieszczonych w uporządkowanej, malejącej kolejności

Przedstawione przykłady sortowania tablic pokazują, że czas wykonania algorytmów zależy od początkowego układu elementów, szczególnie w przypadku sortowania szybkiego. Jednak pomijając ekstremalne przypadki sortowania tablic już posortowanych, algorytm quicksort jest rzeczywiście najszybszym algorytmem sortowania.