Paweł Janic 242521

Sprawozdanie z algorytmów sortowania

1. Opis danych: 50000 losowych liczb całkowitych, losowanych z całego przedziału Integera. W innych przypadkach nie mieliśmy wystarczającej dokładności by zobaczyć różnice w czasach wykonywania się algorytmów lub musieliśmy czekać zbyt długo na wypisywanie się wyników.

2. Tabela czasów wykonywania się algorytmów dla tych samych danych losowych:

	insertSort	bubbleSort	selectSort	quickSort	heapSort	mergeSort
Dane losowe	316	4591	1686	7	10	14
Dane posortowane rosnąco	1	1702	664	2	7	5
Dane posortowane malejąco	928	1458	1075	2	6	5
Złożoność czasowa algorytmu	O(n^2)	O(n^2)	O(n^2)	O(n logn)	O(n logn)	O(n logn)

3. Wnioski:

Najmniejszą złożonością obliczeniową algorytmów charakteryzują się algorytmy quickSort, heapSort oraz mergeSort, czyli powinny być zdecydowanie szybsze od pozostałych przy danych losowych. Możemy to bardzo dobrze zobaczyć na wynikach mojego programu implementującego powyższe algorytmy sortujące. Różnice w czasie wykonywania się algorytmów o złożoności O(n logn) są w granicach kilkuset do kilku tysięcy razy krótsze dla przyjętej przeze mnie liczby danych (50000). Czas wykonywania się algorytmu bubbleSort dla danych posortowanych malejąco może być krótszy niż dla danych posortowanych rosnąco, ponieważ przy zamianie miejscami ich elementów, uzyskujemy taką sytuację, że przy okazji otrzymujemy element największy w ostatniej komórce tablicy. Możemy również zaobserwować, że w każdym przypadku bubbleSort jest najgorszy, ponieważ wywołuje największą liczbę porównań i zamian miejscami.