|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Insert** | **Bubble** | **Select** | **Heap** | **Quick** | **Merge** |
| **Dane losowe** | 11954907 | 27757015 | 14604356 | 1231681 | 1381271 | 2878718 |
| **Dane**  **posortowane**  **rosnąco** | 42667 | 20484139 | 12252032 | 528965 | 12533735 | 878523 |
| **Dane posortowane**  **malejąco** | 21076 | 5124119 | 6278177 | 454426 | 12573318 | 947406 |

**Raport**

**Wniosek**

Ponieważ czas wykonania zależy od kompletacji komputera, to możemy powiedzieć, że w obliczeniu czasu jest mała rozbieżność.

Jak widać z tej tablicy, dla danych losowych najdłuższe działał algorytm Bubble sort. Najszybciej wykonał się algorytm Heap sort.

Dla danych posortowanych rosnąco najszybciej oczywiście wykonał się Insert sort, ponieważ nie było zrobiono żadnej zamiany elementów. Najdłuższym okazał się algorytm Bubble sort.

Dla danych posortowanych malejąco najszybszym okazał się Insert sort. Najdłuższym jest Quick sort. Złożoność algorytmu zależy od wyboru elementu rozdzielającego; jeśli podziały są zrównoważone algorytm jest tak szybki jak sortowanie przez scalanie, czyli O (n log n) w przeciwnym przypadku może działać tak wolno jak sortowanie przez wstawianie O (n\*n). Średni czas działania przy losowym wyborze elementu rozdzielającego, dorównuje przypadkowi optymistycznemu.