( w przypadku próbek< 100tyś. rozmiary próbki były maksymalne na jakie pozwalał interpreter)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SORTOWANIE** | **LOSOWE(s)** | **POSORTOWANA (s)** | **ODWROTNIE POSORTOWANA(s)** | **rozmiar próbki w tyś.** |
| QUICK | 0,9510 | 2,4230 | 1,1470 | 100/2.9/2.9 |
| HEAP | 2,146000 | 2,1730 | 2,0210 | 100 |
| INSERTION | 10,3420 | 0,0040 | 20,7630 | 100 |
| BUBBLE | 25,7910 | 14,4340 | 37,5730 | 100 |

Biorą pod uwagę wyniki z tabeli, można dojść do wniosku że w średnim przypadku (elementy tablicy są losowe i nieuporządkowane) zdecydowanie najlepiej sprawuje się quicksort.

W przypadku gdy elementy tablicy są już posortowane, najlepiej radzi sobie insertion sort. Przy czym quick sort na bardzo małej próbce (2,9 tyś. elementów) działa wolniej od heapsorta przy 100tysiącach elementów.

Ostatni przypadek czyli tablica odwrotnie posortowana -> w tym przypadku heapsort radzi sobie najlepiej z czwórki. Warto zauważyć że quicksort został użyty znowu na bardzo małej próbce, można śmiało wysnuć tezę według której w tym przypadku byłby najwolniejszym z algorytmów.