|  |  |  |  |
| --- | --- | --- | --- |
| **Nazwa** | **Dane losowe** | **Dane posortowane** | **Dane posortowane odwrotnie** |
| Quicksort | 0.2857 | failed | failed |
| Heapsort | 0.7585 | 0.7525 | 0.6887 |
| Cocktailsort | 0.01 | 0.009 | 0.009 |
| Mergesort | 0.4164 | 0.4044 | 0.4054 |

Tabela wyrażająca czas działania(sec.) algorytmu dla tablicy **100k** elementowej.

|  |  |  |
| --- | --- | --- |
| **Nazwa** | **Dane losowe** | **Wzrost przy 5x podniesieniu:** |
| Quicksort | 1.6586 | 580% (80%) |
| Heapsort | 4.52 | 595% (95%) |
| Cocktailsort | 0.0628 | 628% (128%) |
| Mergesort | 2.5662 | 616% (116%) |

Próba kontrolna wyrażająca czas działania(sec.) algorytmu dla tablicy **500k** elementowej.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nazwa** | **Dane losowe** | **Dane posortowane** | **Dane posortowane odwrotnie** |
| Quicksort | 0.2857 | failed | failed |
| Heapsort | 0.7585 | 0.7525 | 0.6887 |
| Cocktailsort | 0.01 | 0.009 | 0.009 |
| Mergesort | 0.4164 | 0.4044 | 0.4054 |

|  |  |  |
| --- | --- | --- |
| **Wnioski:** | **Czas Działania** | **Ogólne tempo wzrostu:** |
| Quicksort | To drugi najszybszy algorytm, aczkolwiek jego złożoność obliczeniowa wzrosła tak mocno, iż **przestał działać** dla najgorszych przypadkach. | **najmniejsze** |
| Heapsort | Okazał się być najwolniejszym algorytmem. Czas działania **zmienił się wyraźnie** przy posortowanej odwrotnie tablicy. | **umiarkowane** |
| Cocktailsort | Okazał się być najwydajniejszym algorytmem na każdej płaszczyźnie. **Nie zmienił** znacząco czasu działania. | **drastyczne** |
| Mergesort | Był trzecim najszybszym algorytmem, czas działania **zmienił się** **delikatnie** dla danych już posortowanych. | **zdecydowane** |

**Repository:** [YumaMasterpiece/ASD (github.com)](https://github.com/YumaMasterpiece/ASD)