# Sprawozdanie

Daniel Majchrzycki, 218476

1. Program alokuje listę o n polach, które wypełnia wartościami. Następnie przeszukuje tę listę w poszukiwaniu ostatniego elementu, podając czas w jakim tego dokonał.

   Testy zostały przeprowadzone dla liczb:
   a) 10
   b) 100
   c) 1 000
   d) 10 000
   e) 100 000
   f) 1 000 000
   g) 10 000 000
   h) 100 000 000 – tylko dla algorytmu quicksort

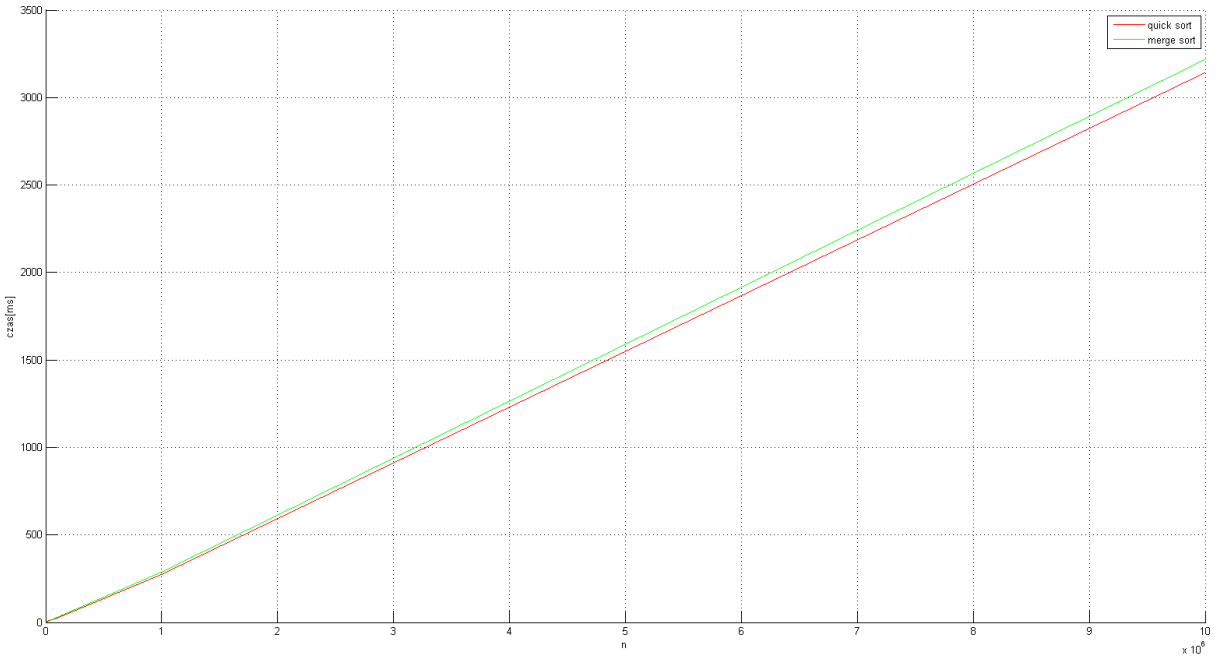
| | 10 | 100 | 1000 | 10000 | 100000 | 1000000 | 10000000 | 100000000 |
|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 3 | 25 | 282 | 3149 | 35083 |
| | 1 | 1 | 1 | 4 | 25 | 278 | 3134 | 33761 |
| | 1 | 1 | 1 | 8 | 24 | 274 | 3134 | 34200 |
| | 1 | 1 | 1 | 5 | 25 | 277 | 3144 | 33733 |
| | 1 | 1 | 1 | 7 | 24 | 264 | 3102 | 32435 |
| | 1 | 1 | 1 | 2 | 28 | 263 | 3086 | 33773 |
| | 1 | 1 | 1 | 3 | 28 | 274 | 3156 | 34585 |
| | 1 | 1 | 1 | 6 | 27 | 270 | 3182 | 34431 |
| | 1 | 1 | 1 | 4 | 23 | 266 | 3095 | 33965 |
| | 1 | 1 | 1 | 7 | 26 | 273 | 3247 | 34123 |
| średnia | 1 | 1 | 1 | 4.7 | 25.3 | 272.1 | 3142.9 | 34008.9 |

*Tabela wyników dla sortowania quicksort.*

| | 10 | 100 | 1000 | 10000 | 100000 | 1000000 | 10000000 | 100000000 |
|---|---|---|---|---|---|---|---|---|
| | 1 | 1 | 1 | 2 | 25 | 277 | 3139 | – |
| | 1 | 1 | 1 | 6 | 26 | 292 | 3265 | – |
| | 1 | 1 | 1 | 3 | 29 | 289 | 3250 | – |
| | 1 | 1 | 1 | 2 | 26 | 289 | 3250 | – |
| | 1 | 1 | 1 | 6 | 25 | 291 | 3284 | – |
| | 1 | 1 | 1 | 2 | 26 | 292 | 3396 | – |
| | 1 | 1 | 1 | 4 | 28 | 284 | 3187 | – |
| | 1 | 1 | 1 | 3 | 25 | 286 | 3138 | – |
| | 1 | 1 | 1 | 4 | 31 | 277 | 3117 | – |
| | 1 | 1 | 1 | 3 | 30 | 278 | 3132 | – |
| średnia | 1 | 1 | 1 | 3.5 | 27.1 | 285.5 | 3215.8 | – |

*Tabela wyników dla sortowania mergesort.*

Wykresy:




Obydwa algorytmy osiągają zbliżoną wydajność dla liczb z zakresy $10 - 10^7$, którą można oszacować jako $O(n \cdot \log n)$). Różnica polega na złożoności pamięciowej. Algorytm MergeSort wykorzystuje drugą tablicę buforową wielkości $n$, co bardzo niekorzystnie wpływa na wykorzystywaną pamięć, a w rezultacie nie pozwoliło mi sprawdzić czasu działania algorytmu dla $n = 10^8$. Wszystkie pomiary zostały przeprowadzone dla list wypełnionych losowymi liczbami.