**Analiza porównawcza quickSort i mergeSort na podstawie liczby porównań, przepisań i zamian.**

1. **Zbiór losowy:**

Tabela wyników:



Wykres:

1. **Zbiór odwrotny do posortowanego:**

Tabela wyników:



Wykres:

1. **Zbiór już posortowany:**

Tabela wyników:



Wykres:

Podsumowanie:

We wszystkich trzech próbach quickSort wypada lepiej na tle przepisań/zamian, traci za to przy porównaniach gdzie lepszy jest mergeSort.

**Analiza porównawcza pod kątem czasu działania programu**

Wartości w tabelach są średnią trzech kolejnych pomiarów, a czas jest mierzony   
w milisekundach

1. **Zbiór losowy**

Tabela wyników:



Wykres:

Niewielka różnica między mergesortem i heapsortem. Quicksort traci, jest to spowodowane złym wyborem pivota. (wybieram losowy element jako pivot)

1. **Zbiór odwrotny do posortowanego**

Tabela wyników:



Jak widać znów na czele jest mergesort. Tym razem na drugie miejsce wskoczył quicksort.

1. **Zbiór już posortowany**

Tabela wyników:



Wykres:

Tym razem to quicksort okazał się najlepszy, przed mergesortem, a na trzecim miejscu ponownie znalazł się heapsort.

Podsumowanie:

Wszystkie trzy algorytmy sortują bardzo szybko duże zbiory liczb, a czas działania jest zależny od zestawu danych jaki otrzymają. We wszystkich trzech zestawieniach mergesort jest lepszy od heapsorta. Najbardziej pokrzywdzony dla zestawu liczb losowych jest quicksort, ale w jego wypadku wybór pivota nie był optymalny. Z tego też powodu ciężko stwierdzić, na którym ostatecznie miejscu powinien się znaleźć quicksort.