Dokumentacja – Algorytm QuickSort

# Opis ogólny

Aplikacja została stworzona w języku Python przy użyciu biblioteki tkinter w celu wizualizacji działania algorytmu sortowania QuickSort. Program umożliwia użytkownikowi wprowadzenie dowolnej ilości liczb, które zostaną posortowane za pomocą algorytmu. Wyświetlane są szczegóły działania, w tym liczba operacji oraz kolejne etapy rekurencji (historia działania).

# Instrukcja korzystania z aplikacji QuickSort 🔧:

1. Po uruchomieniu aplikacji pojawi się okno z instrukcją, polem do wpisywania danych i przyciskiem 'Sortuj'.  
2. Kliknij w pole tekstowe i wpisz liczby całkowite, które chcesz posortować, oddzielając je przecinkami, np.: 4,2,7,1  
 ❗ Uwaga: Nie wpisuj przecinka na końcu albo dwóch przecinków obok siebie! (np. '1,2,3,' lub ‘1,2,,3,4’ jest błędne i spowoduje błąd działania).  
3. Kliknij przycisk 'Sortuj', aby rozpocząć działanie algorytmu.  
4. W dolnej części aplikacji wyświetlą się:  
 - posortowana lista,  
 - liczba kroków (rekurencji) wykonanych przez algorytm,  
 - historia działania algorytmu – pokazująca każdy etap podziału i sortowania.  
5. W historii:  
 - 'Pivot' to element, względem którego dzielona jest lista,  
 - '<=' oznacza elementy mniejsze lub równe pivotowi,  
 - '>' oznacza elementy większe,  
 - wcięcia pokazują głębokość rekurencji (kolejne poziomy wywołań).  
  
Aplikacja przeznaczona jest do edukacyjnego śledzenia działania klasycznego algorytmu QuickSort.

# Działanie algorytmu QuickSort – krok po kroku

1. Algorytm wybiera pierwszy element z listy jako pivot.  
2. Dzieli pozostałe elementy listy na dwie części:  
 - lewą z elementami mniejszymi lub równymi pivotowi,  
 - prawą z elementami większymi od pivota.  
3. Rekurencyjnie sortuje obie podlisty (lewa i prawa).  
4. Łączy wyniki w jedną posortowaną listę.  
5. Proces powtarzany jest dla każdej podlisty aż do osiągnięcia list jednoelementowych.  
6. Z każdym wywołaniem rekurencji zwiększany jest licznik kroków, a aktualny stan sortowania zapisywany do historii.

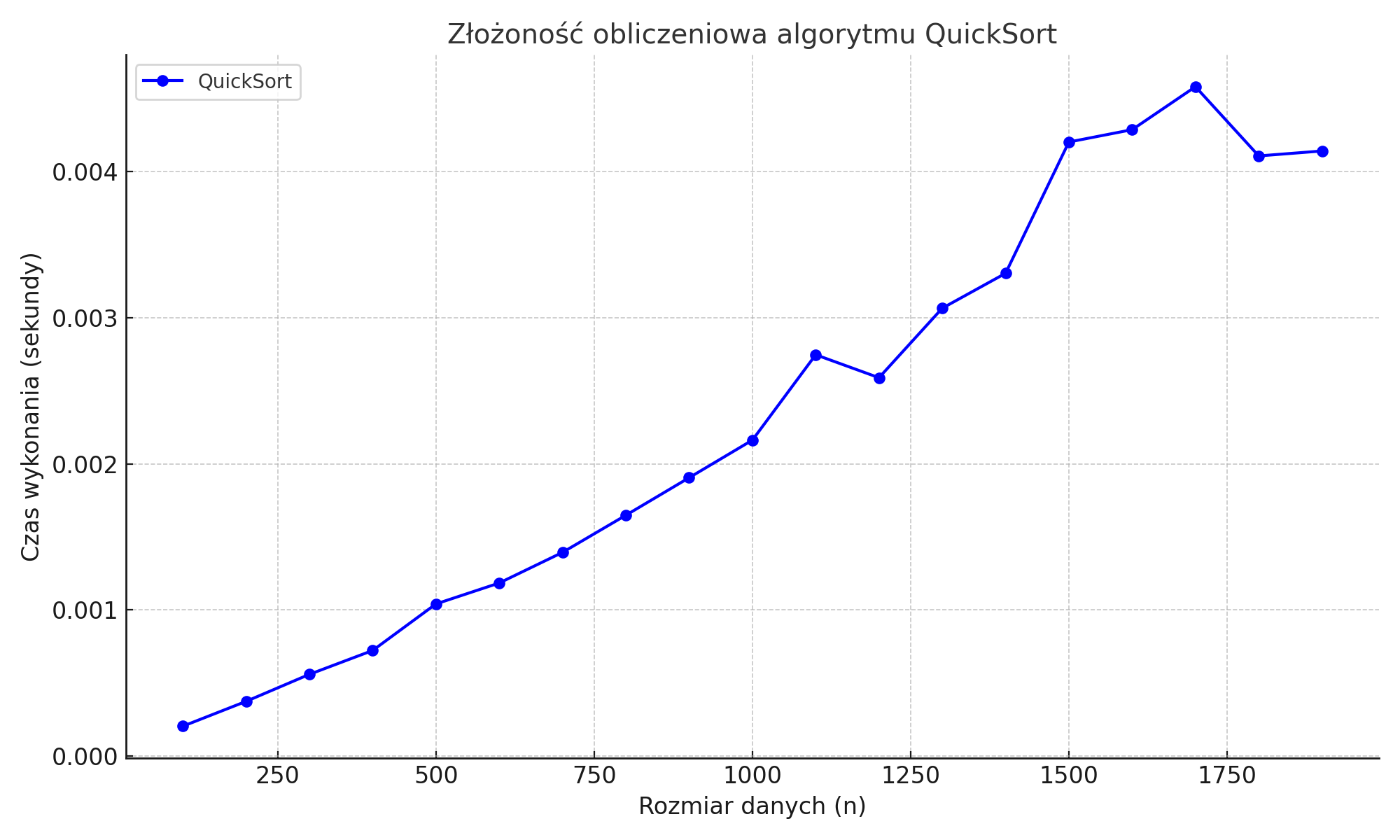
# Złożoność algorytmu QuickSort

QuickSort w przeciętnym przypadku wykonuje się w czasie:  
- Średnia złożoność czasowa: O(n log n)  
- Złożoność pesymistyczna (najgorszy przypadek – dane już posortowane): O(n^2)  
- Złożoność pamięciowa: O(log n) przy implementacji rekurencyjnej

Algorytm działa bardzo wydajnie dla losowo rozłożonych danych. Działa szybciej niż inne algorytmy sortowania w wielu praktycznych przypadkach. Zaletą jest brak potrzeby dodatkowej pamięci oraz prostota implementacji.

# Wykres – Złożoność obliczeniowa

Poniżej przedstawiono wykres porównujący czas wykonania algorytmu QuickSort względem wielkości danych.



Rysunek: Czas działania algorytmu QuickSort w zależności od rozmiaru danych wejściowych.