Projekt 3

do samodzielnego wykonania

I.

1. Porównaj szybkość działania 4 metod sortowania: *Insertion Sort*, *Selection Sort*, *Heap Sort*, *Cocktail Sort* dla tablicy liczb całkowitych (rzędu 50k - 200k elementów) generowanych w postaci: losowej, rosnącej, malejącej, stałej, v-kształtnej.

Przedstaw wykresy *t = f(n)* dla każdej z metod w zależności od postaci tablicy wejściowej, gdzie: t - czas sortowania; n - liczba elementów tablicy.

Liczbę elementów należy dobrać w taki sposób, aby możliwe było wykonanie pomiarów. Wyniki przedstawić na czterech wykresach (przynajmniej 15 punktów pomiarowych).

2. Sformułuj wnioski odnośnie:

- złożoności obliczeniowej badanych metod i ich związku z efektywnością sortowania oraz zajętością pamięciową każdej z nich,
- wpływu postaci ciągów wejściowych na czas sortowania (najgorsze, najlepsze przypadki).

II.

1. Dla różnych typów danych wejściowych porównaj efektywność działania powyższych algorytmów.
   Przedstaw wykresy *t = f(n)* dla każdego typu danych wejściowych i różnych metod sortowania (5 wykresów).
   Liczbę elementów należy dobrać w taki sposób, aby możliwe było wykonanie pomiarów.
2. Sformułować wnioski odnośnie złożoności obliczeniowej i efektywności wykonywania oraz zachowania się algorytmów dla poszczególnych typów danych wejściowych.

III.

1. Zaimplementuj algorytm *Quicksort* w dwóch wersjach: rekurencyjnie oraz iteracyjnie (z własną implementacją stosu).
   Porównaj obie wersje na wspólnym wykresie przy sortowaniu ciągu losowego. Następnie porównaj różne sposoby wyboru klucza do porównania: skrajnie prawego, środkowego co do położenia, losowo wybranego.
   Utwórz wykres porównujący efektywność działania algorytmu (iteracyjnego lub rekurencyjnego) w zależności od wyboru różnego klucza dla A-kształtnego ciągu wejściowego (przynajmniej 15 punktów pomiarowych).
2. Sformułuj wnioski dotyczące złożoność badanych algorytmów. Jak wybór klucza wpływa na działanie algorytmu (najgorsze, najlepsze przypadki)?

Komentarz do zadania:

Losowanie elementów:

- inicjalizacja generatora:
  `Random rnd = new Random(Guid.NewGuid().GetHashCode());`
- losowanie kolejnych liczb:
  `rnd.Next(maxValue);`