Przykład 1

Ustal wartości:

- M = 7
- N = 77

Zadanie 1. (4 punkty)
Zbadaj złożoność obliczeniową poznanych na ćwiczeniach metod sortowania list.
W tym celu:

- wykonaj kilka testów próbnych dla danej metody, by ustalić minimalny i maksymalny rozmiar listy – *n_min* i *n_max* (minimalny powinien dawać niezerowe czasy sortowania – np. większe od 0,002s, maksymalny rozmiar nie powinien być zbyt duży, by obliczenia nie trwały zbyt długo – dla listy o największym rozmiarze czas powinien być krótszy niż 1s)
- uzupełnij metodę *MierzCzas* w pliku *Sortowania.java* tak, by wykonywała ona po M iteracji algorytmu, który masz zbadać, wykorzystaj instrukcję warunkową *switch*, by później móc korzystać z tej metody dla obu algorytmów sortowania
- spróbuj wywołać metodę *BadajZlozonosc* dla obiektu klasy *Sortowania* – jeżeli jako złożoność dostaniesz wynik *NaN* należy zwiększyć *n_min* (by wyeliminować czasy równe 0)
- narysuj odpowiedni wykres i opisz go, podsumuj wyniki i wyciągnij wnioski z przeprowadzonego doświadczenia

Zadanie 2. (2 punkty)
Porównaj efektywność sortowania losowej listy obiema metodami.

W tym celu:

- ustal maksymalny rozmiar listy *n_max*, tak by obliczenia dla obu metod nie trwały zbyt długo
- jeśli nie zrobiłeś tego w zdaniu pierwszym – zmodyfikuj metodę *MierzCzas* tak, by wykonywała ona po M iteracji obu algorytmów, które masz zbadać
- narysuj odpowiedni wykres wywołując metodę *PorownajMetody* i skomentuj go (czerwone punkty to pierwsza z metod, niebieskie – druga)
- podsumuj wyniki i wyciągnij wnioski z przeprowadzonego doświadczenia.

W raporcie zamieść także kod metody *MierzCzas* oraz zapisz argumenty z jakimi powołałeś do istnienia obiekt klasy *Sortowania* w obu zadaniach.