

Algorytmy i struktury danych I

Szybkie algorytmy do sortowania

• Zadanie 1

Zaimplementuj jeden (wybrany) z szybkich algorytmów do sortowania:

- Quick sort
- Merge sort
- Heap sort

Dokonaj testów działania programu na zbiorach o następującej ilości elementów: 1000, 8000, 64000. Generuj dane wykorzystując programy testowe przygotowane na stronie przedmiotu.

Dla każdej liczności zbioru, dokonaj następujących testów (zrób tabelkę z wynikami):

1. N_{po} - ilość operacji wykonanych dla sortowania zbioru posortowanego
2. N_{od} - ilość operacji wykonanych dla sortowania zbioru posortowanego odwrotnie
3. N_{pp} - ilość operacji wykonanych dla sortowania zbioru posortowanego z losowym elementem na początku
4. N_{pk} - ilość operacji wykonanych dla sortowania zbioru posortowanego z losowym elementem na końcu
5. N_{np} - ilość operacji wykonanych dla sortowania zbioru z losowym rozkładem elementów

Analizując ilość wykonanych operacji przez algorytm, oceń jaka jest jego złożoność obliczeniowa

- **optymistyczna:** dla zbiorów uporządkowanych (z niewielką ilością elementów nie na swoich miejscach) - na podstawie N_{po} , N_{pp} , N_{pk}
- **typowa:** dla zbiorów o losowym rozkładzie elementów - na podstawie N_{np}
- **pesymistyczna:** dla zbiorów posortowanych odwrotnie - na podstawie N_{od}

• Zadanie 2

Wykorzystując technikę *dziel i zwyciężaj* zaimplementuj algorytm o złożoności obliczeniowej $O(n \log n)$, który zlicza ilość inwersji w podanej serii liczb. Sprawdź poprawność wyniku zliczania inwersji przy pomocy prostego algorytmu o złożoności $Q(n^2)$. Generuj dane wykorzystując zmodyfikowany program `genTest.cpp` przygotowany na stronie przedmiotu.

• Zadanie 3

Wykorzystując technikę *dziel i zwyciężaj* zaimplementuj algorytm o złożoności obliczeniowej $O(n \log n)$, który znajduje parę najbliższych punktów w podanej serii par (x, y) . Sprawdź poprawność wyniku przy pomocy prostego algorytmu o złożoności $Q(n^3)$. Generuj dane wykorzystując zmodyfikowany program `genTest.cpp` przygotowany na stronie przedmiotu.

1 Zaliczenie zestawu

Umieszczenie w systemie PEGAZ 1 pliku w formacie .gzip w którym dla każdego zadania przygotowana została podkartoteka z następującą zawartością:

- Plik z kodem programu: program main sterujący procedurą wczytywania i wypisywania danych, oraz funkcja sortująca, można wykorzystać kod programu *sortTest.cpp*.
- Przykładowy plik (pliki) inputowy wykorzystany do testowania programu. Do generacji użyj zmodyfikowany kod *genTest.cpp*.
- Plik README oraz makefile
- Plik w formacie .pdf zawierający krótki opis algorytmu oraz wyniki testów czasu działania programu.