Praca domowa 06 – plane

Michał Szczygieł

Wstęp

Zadanie składa się z 2 plików:

- Main.java
- Plain.java

CEL: Odczytanie danych z bazy danych. Z pobranych danych wyznaczanie największego obszaru płaskiego O, który jest podobszarem obszaru T, w którym dla każdej pary x i y leżącej wewnątrz lub na brzegu obszaru O wartość funkcji jest stała. W końcowym etapie wyświetlenie pola obszaru najmniejszego wielokąta wypukłego rozpiętego nad zbiorem punktów (x, y) należących do obszaru O.

Klasa Main

Zadaniem tej klasy jest wczytanie danych z bazy danych, oraz przekazanie odpowiednio sformatowanych danych do klasy Plain. Połączenie z bazą odbywa się po wcześniejszym poprawnym ustawieniu connection stringa, który podawany jest jako argument wywoławczy programu.

Metody:

`public static void main(final String... args)`- Metoda główna, przyjmuje jeden parametr <connection_string> i przekazuje wczytane dane do klasy Plain, wykonując cel, oraz wyświetlenie wyniku z dokładnością do pięciu miejsc po przecinku.

`public static Map<Float, List<Point2D.Float>> dbconnection(final String connector)` – Metoda ta ma na celu obsłużenie połączenia bazodanowego, oraz wykonanie zapytania SQL dla bazy danych. Oraz zwrócenie poprawnie sformatowanych danych poprzez funkcję umieszczoną poniżej.

`private static Map<Float, List<Point2D.Float>> retriveData(final ResultSet result) throws SQLException` – Metoda ta odbiera dane po wykonaniu zapytania SQL do bazy danych, oraz upakowywuje dane do mapy gdzie klucz stawowi wynik funkcji z =f(x, y), natomiast wartościami są kolekcje przechowujące wbudowany typ w podstawową bibliotekę javy reprezentujący punkt w przestrzeni dwu wymiarowej.

Klasa Plain

Klasa ta, otrzymawszy dane od klasy wywołującej, ustawia te dane w konstruktorze. Klasa ów zawiera także jedną klasę wewnętrzną:

```
private class CoordinateComparator implements Comparator<Point2D.Float> -
```

Klasa ta jest komparatorem niezbędnym do sortowania elementów (Collections.sort()). Powyższa klasa wewnętrzna została opisane w kodzie źródłowym programu, jak i również wszystkie pozostałe metody należące do klasy Plain.

Realizacja algorytmu Plain

W mojej realizacji algorytmu dla klasy Plain zastosowałem następujące kroki.
Tworzę dla danego regionu punktów (tzw. Kolekcja dla takiego samego klucza), otoczkę wypukłą metodą (Montone Chain). Obliczam powierzchnię dla danego obszaru sumując kolejne obszary metodą triangulacji punktów na otoczce wypukłej poniższą metodą:

```
private Double getTrangulizationArea(Point2D.Float firstPoint,
      Point2D.Float secondPoint, Point2D.Float thirdPoint)
```

Końcowym etapem jest wybranie największego regionu oraz wyświetlenie wyniku w postaci zadanej w treści zadania. Złożoność obliczeniowa dla tego programu wynosi:

- Wczytanie danych do mapy – O(n)
- Tworzenie otoczki wypukłej (najmniejszego obszaru ogarniającego wszystkie punkty) metodą monotne chain – O(n log n)

Dyskusja

W przypadku realizacji rozwiązania tego programu, możliwe było zastosowanie innego algorytmu tworzenia otoczki wypukłej, np.:

- **Gift wrapping** aka **Jarvis march** — $O(nh)$
- **Graham scan** — $O(n \log n)$
- **QuickHull**
- **Divide and conquer** — $O(n \log n)$
- **Incremental convex hull algorithm** — $O(n \log n)$
- **The ultimate planar convex hull algorithm** — $O(n \log h)$
- **Chan's algorithm** — $O(n \log h)$

W celu optymalizacji mogło by być stworzenie odpowiedniego zapytania do bazy danych wybierającego już region z największą powierzchnią.