КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ

Звіт для **лабораторної роботи №4 "QuickHull"**
з дисципліни КОМП'ЮТЕРНА ГРАФІКА

Галіцької Олени, ІС-3

2018

I. ЗАВДАННЯ

Реалізувати знаходження опуклої оболонки для множини точок за допомогою алгоритму ШвидкОбол - QuickHull.

II. ТЕОРЕТИЧНА ЧАСТИНА

1. ОСНОВНІ ПОНЯТТЯ

   Опукла оболонка множини точок S - найменша опукла множина, що містить S.

2. ОПИС АЛГОРИТМУ

   Метод розбиває множину точок на дві підмножини. Кожна з цих множин міститиме по одній ламаній. З'єднання цих двох ламаних дасть у результаті многокутник опуклої оболонки. Алгоритм є рекурсивним.

   Кроки алгоритму:

   1. У множині точок знаходимо точки з найменшою та найбільшою абсцисами.
   2. З'єднуємо ці точки у пряму. Вона розіб'є нашу множину на дві підмножини. Далі працюємо з обома підмножинами окремо.
   3. Для підмножини знаходимо точку, яка є найвіддаленішою від прямої, побудованою на попередньому кроці. Ці три точки (кінці відрізка, що формують пряму та найвіддаленіша точка) створюють трикутник. Очевидно, всі точки всередині цього трикутника не можуть належати границі опуклої оболонки.
   4. Розглядаємо 2 прямі: ту, яка з'єднує найвіддаленішу точку з мінімальною по абсцисі та ту, що з'єднує найвіддаленішу точку з максимальною по абсцисі.
      З цими двома прямими повертаємось на крок 3. та продовжуємо рекурсивну обробку. Продовжуємо, поки не дійдемо до прямої, для якої всі точки множини будуть знаходитися по один її бік. Відрізок, що формує таку пряму, буде належати опуклій оболонці.

3. АНАЛІЗ АЛГОРИТМУ

   Загальний час роботи алгоритму - O($N \log(N)$), N - потужність множини точок.
   В найгіршому випадку - O(N^2)
   Перевага методу полягає в можливості розпаралелення обчислень.

III. РЕАЛІЗАЦІЯ АЛГОРИТМУ

1. МОВА

   Для реалізації алгоритму була використана мова Python.

2. ІНТЕРФЕЙС ПРОГРАМИ

   Вхід: текстовий файл з координатами точок.
   Вихід: у консоль друкується набір точок, що формують опуклу оболонку.
   Також є можливість переглянути результат роботи програми на графіку.

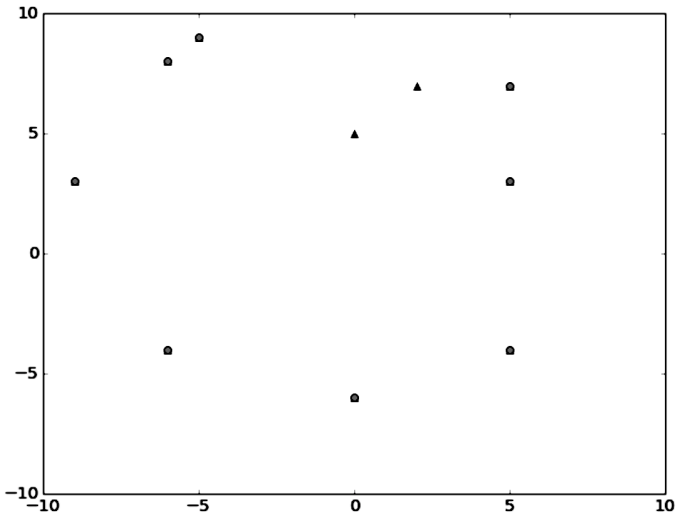
3. ВИКОРИСТАНІ СТРУКТУРИ ДАНИХ
   Для представлення точок використовувався масив. Елементи масиву - об'єкти класу Point, полями якого є координати по осям абсцис та ординат.

   Результат (опукла оболонка) також представлена масивом точок - вершини многокутника опуклої оболонки.

4. ТЕСТОВІ ПРИКЛАДИ
   Точки: Результат (на графіку трикутники - початкові точки, кружечки показують вершини опуклої оолонки) :

-6	-4
0	-6
6	3
5	-4
-6	8
2	7
-9	3
-5	9
0	5
5	7

-6	-4
0	-6
6	3
5	-4
-6	8
-9	3
-5	9
5	7



5. ОСНОВНІ МОДУЛІ ПРОГРАМИ
   1. Початкова функція, яка будує першу пряму та основна рекурсивна функція:

```
def find_hull(points, n):
    if n < 3:
        print("Convex hull can be built only for >= 3 points")
        return

    min_x = min(points, key=lambda point: point.x)
    max_x = max(points, key=lambda point: point.x)

    quick_hull(points, n, min_x, max_x, 1)
    quick_hull(points, n, min_x, max_x, -1)

    for point in hull:
        print(point)
    return
```

```
def quick_hull(points, n, p1, p2, side):
    inner_point = None
    max_distance = 0

    for point in points:
        dist = distance(p1, p2, point)
        if dist > max_distance and on_side(p1, p2, point) == side:
            max_distance = dist
            inner_point = point

    if inner_point is None:
        if p1 not in hull:
            hull.append(p1)
        if p2 not in hull:
            hull.append(p2)
        return

    quick_hull(points, n, inner_point, p1, -1 * on_side(inner_point, p1, p2))
    quick_hull(points, n, inner_point, p2, -1 * on_side(inner_point, p2, p1))
```

IV. ВИСНОВКИ

Алгоритм ШвидкоБол є максимально швидким та ефективним, якщо точки розподілені рівномірно відносно прямої, що з'єднує точки з екстремальними значеннями абсцис.

V. ВИКОРИСТАНІ ДЖЕРЕЛА

Для розбору алгоритму та теоретичної частини використовувала слайди лекцій.