Uniwersytet Jagielloński w Krakowie
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Wojciech Lasak
Informatyka III rok

Wyznaczanie otoczki wypukłej Graham

Język Python

Projekt wykonany pod kierownictwem
dr hab. Andrzej Kapanowski

2019

1.Spis treści

Spis treści

2.Wprowadznie

Program służy do wyznaczania otoczki wypukłej wokół zadanych punktów, za pomocą algorytmu Grahama.

2.1 Opis działania

Program losuje zadaną liczbę punktów, trafiają one do funkcji wyznaczającej otoczkę wypukłą za pomocą algorytmu Grahama, funkcja ta zwraca listę punktów, które nalezą do otoczki. Następnie za pomocą funkcji *pyplot* z biblioteki Matplotlib, rysuje wszystkie punkty a następnie na podstawie zwróconej list z algorytmu Grahama, łącze te, które należą do otoczki wypukłej.

2.2 Lista kroków

1. Wylosuj wartość x oraz y dla punktów
2. Zainicjuj listę L i dodaj do niej punkty
3. Rozpocznij algorytm Grahama dla zadanej listy L
4. Sprawdź czy lista L zawiera co najmniej 3 punkty
   4.1 Jeśli nie zawiera zwróć listę L jako wynik
   4.2 Jeśli zawiera przejdź dalej
5. Wybierz punkt (ozn. *O)* o najmniejszej wartości współrzędnej y
6. Zamień *O* z pierwszym elementem na liście L
7. Posortuj elementy na liście L wg współrzędnych polarnych oprócz pierwszego elementu
8. Zainicjuj stos S
9. Przeglądaj listę L
   9.1 Od bieżącej pozycji weź trzy kolejne punkty (ozn. A, B, C) i daj na stos S.
   9.2 Jeśli punkt B leży na zewnątrz trójkąta AOC, to może należeć do otoczki wypukłej. Przejdź do następnego punktu na liście.
   9.3 Jeśli punkt B leży wewnątrz trójkąta AOC, to znaczy, że nie należy do otoczki. Usuń punkt B ze stosu S i cofnij się o jedną pozycję (o ile bieżąca pozycja jest różna od początkowej).
10. Zwróć stos jako otoczkę wypukła

3.Opis interfejsu

3.1 Wymagania

Program do uruchomienia wymaga:

- zainstalowanego na komputerze Pythona 2.7
- zainstalowanej pythonowej biblioteki Matplotlib

3.2 Sposób uruchomienia

Pliki programu zawarte są w katalogu *program*. Stamtąd uruchomić go można poleceniem:

python start.py

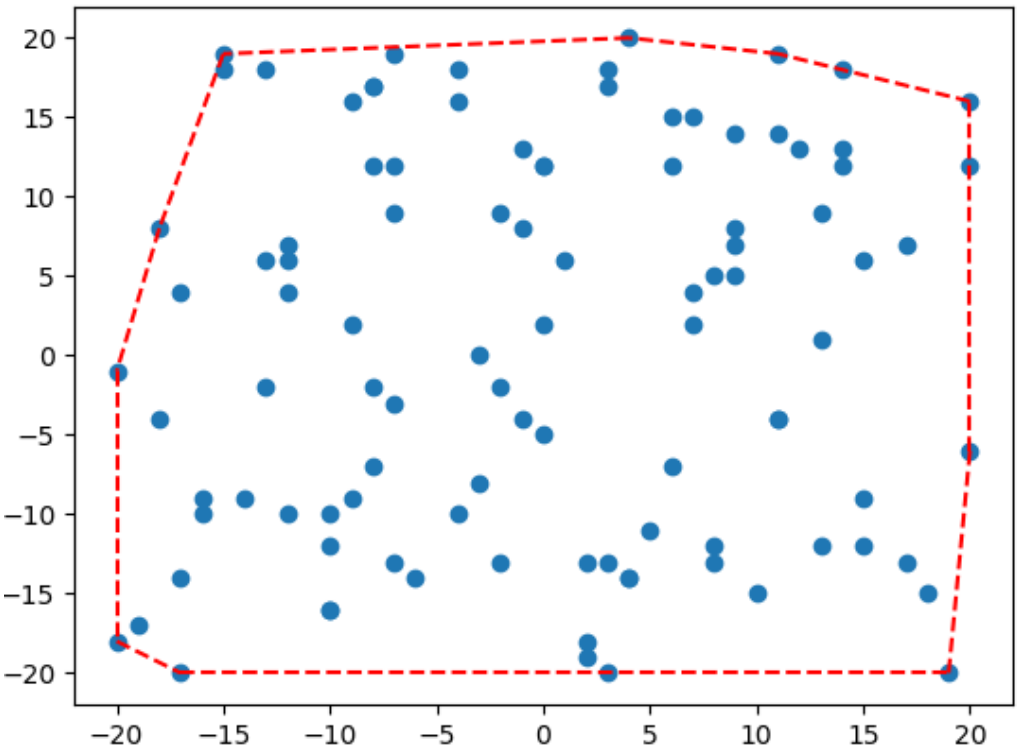
jako wynik zostanie zwrócony układ współrzędnych z zaznaczonymi punktami oraz otoczką, można go zapisać jako plik graficzny przy pomocy GUI.

3.3 Przykładowe uruchomienie

Komenda w konsoli w katalogu z plikiem:

python start.py

Wynik:



4.Implementacja

Kod składa się z trzech plików:

- graham.py
- points.py
- start.py

4.1 points.py

W pliku tym zdefiniowana klasa *Points* reprezentująca punkty na płaszczyźnie

```
class Point:
    """
    Klasa reprezentująca punkty na płaszczyźnie
    """

    def __init__(self, x, y):
        """
        Konstruktor
        """
        self.x = x
        self.y = y

    def __str__(self):
        """
        Reprezentacja jako string
        """
        return "({},{})".format(self.x, self.y)
    def __repr__(self):
        """
        Reprezentacja jako string
        """
        return "Point({},{})".format(self.x, self.y)

    def __eq__(self, other):
        """
        Porownanie
        """
        return self.x == other.x and self.y == other.y

    def __add__(self, other):
        """
        Dodawanie punktow
        """
        return Point(self.x + other.x, self.y + other.y)

    def __sub__(self, other):
        """
        Odejmowanie punktow
        """
        return Point(self.x - other.x, self.y - other.y)

    def length(self):
        """
```

4.2 graham.py

```
def orientation(p, q, r):
    """
    sprawdzanie orientacji 3 punktow
    0 gdy sa wspolliniowe
    1 gdy sa ulozone w kierunku ruchu wskazowek zegara
    2 gdy sa ulozone przeciwnie do ruchu wskazowek zegara
    """
    val = (q.y - p.y)*(r.x - q.x) - (q.x - p.x)*(r.y - q.y)
    if val == 0:
        return 0
    return 1 if (val > 0) else 2


def graham(L):
    """
    Wyznaczanie otoczki wypuklej z podanej listy
    -znalezenie punktu o najmniejszej wsp. y
    -sortowanie listy bez pierwszego elementu wg wspolrzednych polarnych
    -znaleznie punktow nalezacych do otoczki
    """

    if len(L) < 3:
        return L

    bottom = 0
    for i in range (1, len(L)):
        if L[i].y < L[bottom].y:
            bottom = i

    L[0], L[bottom] = L[bottom], L[0]
    pivot = L[0]

    def polar_order(a, b):
        """
        ustalanie polozenia wg wspolrzednych polarnych,
        najpierw kat, potem odleglosc
        """
        order = orientation(pivot, a, b)
        if order == 0:
            return -1 if ((pivot - a).length() < (pivot - b).length()) else 1
        return -1 if (order == 2) else 1


    L[1:] = sorted(L[1:], polar_order)

    stack = []
    stack.append(L[0])
    stack.append(L[1])

    """
    sprawdzanie orientacji danej trojki
    jesli jest wspolniowy lub preciwny do ruchu wskazowek zegara,
    zostawiamy na stosie w przeciwnym wypadku usuwamy ze stosu
    """
    for i in range (2, len(L)):
        while len(stack) > 1 and orientation(stack[-2], stack[-1], L[i]) !=
2:
            stack.pop()
        stack.append(L[i])

    return stack;
```

```
def generate_list():
    """
    przygotowanie listy i losowa generacja punktow
    """
    L = []
    for i in range(0, 100):
        x = random.randint(-20, 20)
        y = random.randint(-20, 20)
        L.append(Point(x, y))
    return L

def draw(L, graham_L):
    """
    odzielne zapisanie czesci ze wspolrzedna x i y z obu list,
    oraz rysowanie
    """
    xL = [p.x for p in L]
    yL = [p.y for p in L]
    xG = [p.x for p in graham_L]
    yG = [p.y for p in graham_L]

    plt.plot(xL, yL, 'o')
    for i in range(0, len(xG)):
        if i == len(xG) - 1:
            temp = 0
        else:
            temp = i + 1
        plt.plot([xG[i], xG[temp]], [yG[i], yG[temp]], 'r--')

    plt.show()
```

4.3 start.py

Wywołuje metody z pliku *graham.py*, przygotowuje listę, wykonuje algorytm grahama i rysuje wynik.

```
L = generate_list()
graham_L = graham(L)
draw(L, graham_L)
```

5.Podsumowanie

Dla klasy *Points* wykonałem testy dla każdej metody.

Wykonałem również testy dla funkcji *graham*. Wykonane testy były dla małych, ale specyficznych danych:

- jeden punkt
- dwa punkty
- trzy punkty
- punkty współliniowe
- punkty zawierające liczby typu float

Dla wszystkich moich testów funkcja zwróciła prawidłowe dane, nie oznacza to jednak, że jest w 100% sprawna i mogą zdarzyć się dane, przy których funkcja może źle zadziałać

6.Literatura

1) https://pl.wikipedia.org/wiki/Algorytm_Grahama
2) https://en.wikipedia.org/wiki/Graham_scan
3) http://www2.lawrence.edu/fast/GREGGJ/CMSC210/convex/convex.html