Algorytmy geometryczne

Sprawozdanie z ćwiczenia 2.

Paweł Lamża

Dane techniczne urządzenia na którym wykonano ćwiczenie:

Laptop z systemem Windows 10 x64

Procesor: AMD Ryzen™ 5 4600H

Pamięć RAM: 16GB

Środowisko: Jupyter notebook

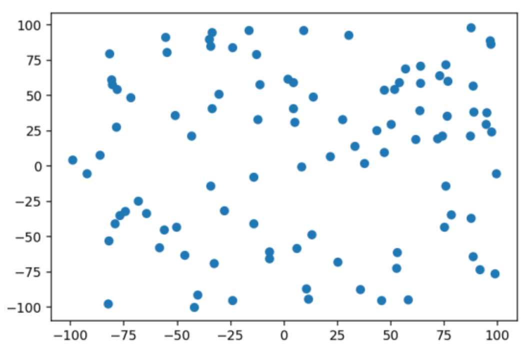
Ćwiczenie zrealizowano w języku Python 3, z wykorzystaniem bibliotek numpy oraz matplotlib

Opis ćwiczenia

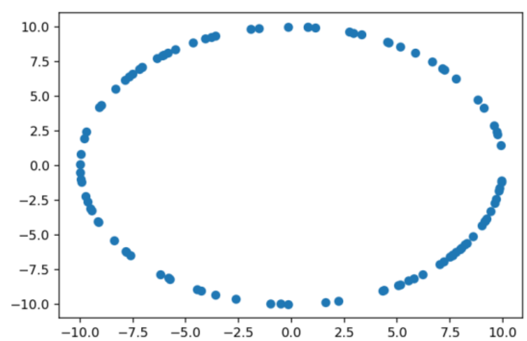
Ćwiczenie polegało na zaimplementowaniu algorytmów Jarvisa i Grahama, wyznaczających otoczkę wypukłą dla różnych zbiorów punktów oraz porównaniu czasów dla różnej liczby punktów w zbiorze.

1. Generacja punktów

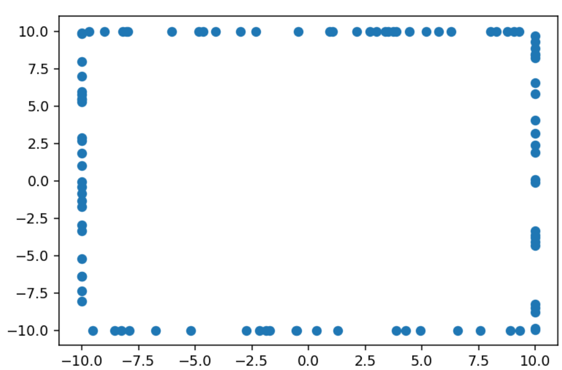
W celu wykonania ćwiczenia wygenerowałem 4 zestawy punktów A, B, C, D z treści ćwiczenia. Dokonałem losowanie położenia punktów z pomocą metody random.random() z biblioteki random. Funkcja ta generuje liczbę zmiennoprzecinkową z przedziału (0,1). Dodatkowo punkty z zestawu B zostały wygenerowane z pomocą funkcji trygonometrycznych, również z biblioteki numpy. Wszystkie zestawy punktów zostały zwizualizowane za pomocą dostarczonego narzędzia graficznego. Poniżej wykresy dla kolejnych zestawów punktów:



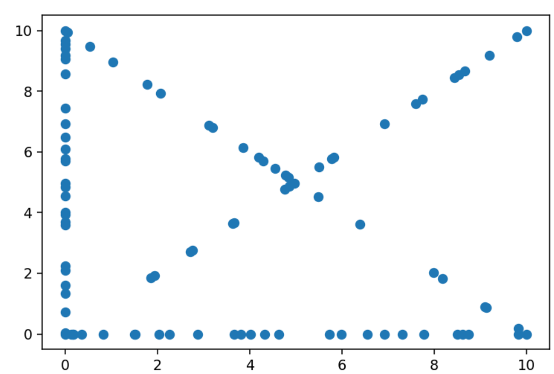
Wykres 1.1, ZESTAW A



Wykres 1.2, ZESTAW B



Wykres 1.3, ZESTAW C



Wykres 1.4, ZESTAW D

2. Metoda obliczania wyznacznika i tolerancja dla zera

Do obliczania wyznacznika użyłem wyznacznika macierzy 3x3 własnej implementacji, a jako tolerancje dla zera przyjąłem 10^-12. Dla niższej tolerancji rzędu 10^-18 algorytm Jarvisa działał czasami niepoprawnie dla zestawu C.

3. Algorytm Grahama

Algorytm polega na systematycznym usuwaniu wierzchołków wklęsłych.

Opis algorytmu:

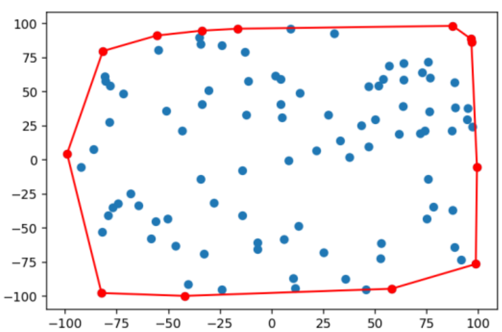
1. Znajdujemy punkt o najmniejszej współrzędnej Y, a w razie tej samej najmniejszej współrzędnej Y wybieramy ten punkt, który ma mniejszą współrzędną X. Będzie to nasz punkt startowy.
2. Sortujemy pozostałe punkty za pomocą sortowania typu quicksort według kąta jaki tworzy wektor, łączący punkt startowy z punktem rozważanym, z dodatnim kierunkiem osi OX. Jeśli dwa punkty tworzą identyczny kąt to pierwszy będzie ten, który znajduje się bliżej punktu startowego.
3. Do stosu wkładamy dwa pierwsze punkty
4. Następnie w głównej pętli:
    - przechodzimy po kolejnych punktach i jeśli punkt leży na tej samej prostej co ostatni punkt na stosie to zamieniamy punkt na stosie z rozważanym punktem
    - jeśli rozważany punkt leży po lewej stronie prostej zbudowanej z dwóch ostatnich punktów na stosie, to dodajemy na stos rozważany punkt
    - jeśli punkt jest po prawej stronie takiej prostej to usuwamy ostatni punkt ze stosu
5. Po zakończeniu pętli usuwamy ostatni punkt jeśli jest niepotrzebny

4. Algorytm Jarvisa – „owijanie prezentu”

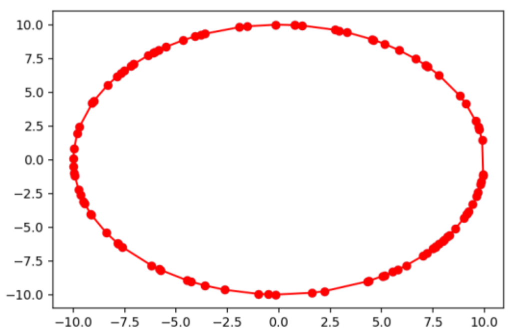
Opis algorytmu:

1. Znajdujemy punkt startowy tak samo jak w algorytmie Grahama.
2. Dodajemy punkt startowy na stos.
3. Tak długo jak nie zakończymy wyznaczania otoczki powtarzamy kolejne operacje:
    - Znajdujemy punkt, inny niż ostatni na stosie, który spełnia warunek, że żaden z pozostałych punktów nie leży po prawej stronie odcinka wyznaczonego przez ostatni punkt na stosie i punkt rozważany (jeśli wiele punktów spełnia ten warunek to wybieramy ten, który jest najdalej od ostatniego punktu na stosie)

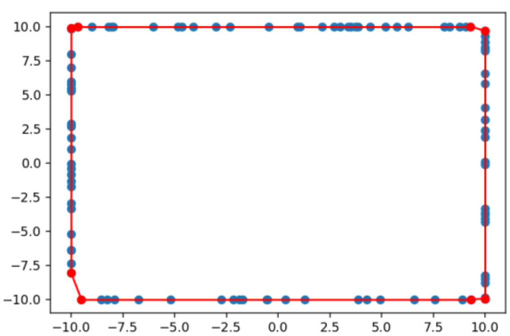
5. Otoczka wyznaczona przez oba algorytmy na zadanych zestawach



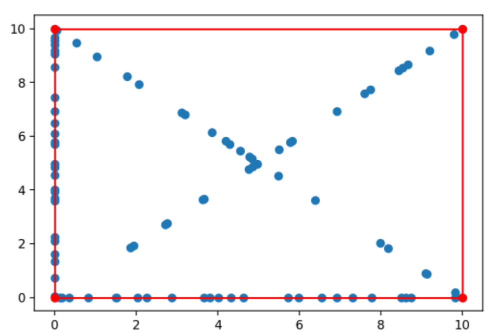
Wykres 5.1 Otoczka Zestaw A



Wykres 5.2 Otoczka Zestaw B



Wykres 5.3 Otoczka Zestaw C



Wykres 5.4 Otoczka Zestaw D

6. Porównanie czasu działania algorytmów w zależności od liczby punktów

Rozważałem działanie algorytmów dla punktów w liczbie 100, 1.000, 2.500, 5.000, 10.000 oraz 20.000 , wyniki zamieściłem w poniższej tabeli:

Zestaw	Algorytm	100 punktów	1000 punktów	2500 punktów	5000 punktów	10000 punktów	20000 punktów
A	Graham	0.028s	0.4519s	0.8228s	1.5884s	2.3723s	5.6340s
A	Jarvis	0.442s	0.6256s	1.3049s	2.9930s	4.1062s	10.1015s
B	Graham	0.0100s	0.1280s	0.4101s	0.8232s	1.8015s	3.8571s
B	Jarvis	0.0741s	9.5319s	82.7934s	485.999s	-	-
C	Graham	0.016s	0.3111s	1.0431s	2.0969s	4.0890s	4.9074s
C	Jarvis	0.016s	0.1680s	0.4601s	0.7962s	1.4324s	1.9455s
D	Graham	0.0202s	0.3086s	1.0367s	2.6451s	4.9208s	6.7322s
D	Jarvis	0.0120s	0.1080s	0.3361s	0.6843s	0.7642s	1.2283s

Tabela 6.1 porównanie czasu działania czasu algorytmów w zależności od liczby punktów dla algorytmów „z flagą”

Zestaw	Algorytm	100 punktów	1000 punktów	2500 punktów	5000 punktów	10000 punktów	20000 punktów
A	Graham	0.0186s	0.2559s	0.8842s	1.5661s	2.6407s	5.1652s
A	Jarvis	0.0282s	0.4881s	1.4489s	3.0733s	4.5661s	8.6242s
B	Graham	0.0089s	0.1142s	0.3924s	0.8022s	1.6425s	3.5122s
B	Jarvis	0.0732s	8.2321s	68.1454s	401,331s	-	-
C	Graham	0.0154s	0.2841s	0.8162s	1.7946s	3.4933s	5.4854s
C	Jarvis	0.0160s	0.1720s	0.3961s	0.7842s	1.4996s	1.8925s
D	Graham	0.0205s	0.3136s	1.0923s	2.0326s	4.8875s	6.5979s
D	Jarvis	0.0090s	0.1320s	0.2400s	0.5722s	0.9803s	1.2991s

Tabela 6.2 porównanie czasu działania czasu algorytmów w zależności od liczby punktów dla algorytmów „bez flagi”

Dla zestawu B, algorytm Jarvisa bardzo szybko zaczął zwiększać czas potrzebny do wyliczenia otoczki co uniemożliwiło mi podanie czasu w tym przypadku dla więcej niż 5000 punktów.

7. Wnioski i spostrzeżenia

- Zarówno algorytm Jarvisa jak i Grahama poprawnie obliczają otoczkę wypukłą dla różnych zestawów danych, również takich które powinny teoretycznie sprawiać im problem
- Dla algorytmu Grahama zmiana liczby punktów w zestawie nie powoduje specjalnego spowolnienia, dla liczby punktów <= 20000 czas wyliczenia nie przekracza 7 sekund
- W przypadku zestawu A – punktów losowych, algorytm Grahama okazuje się około 2 razy szybszy od algorytmu Jarvisa co może prowadzić do wniosku, że dla losowych danych lepiej wybrać algorytm Grahama
- W zestawie B algorytm Jarvisa bardzo drastycznie zwiększa czas obliczeń, w tym zestawie algorytm jest rzucony na najgorszy możliwy dla niego przypadek. Już przy stu punktach jest 7 razy wolniejszy od algorytmu Grahama, przy 2500 punktów jest już prawie 200 razy wolniejszy od drugiego algorytmu, a dla 5000 punktów 500 razy wolniejszy
- Wynika to ze złożoności algorytmu Jarvisa rzędu O(nk) gdzie k to liczba punktów na otoczce. Jak wiadomo w zestawie B liczba k jest równa łącznej liczbie punktów n, a zatem algorytm Jarvisa ma wtedy złożoność rzędu n^2.
- W zestawach C i D, kiedy punktów na otoczce jest niewiele, pokazuje się faktyczny sens algorytmu Jarvisa. Okazuje się on lepszy od Grahama, przy 20 tysiącach punktów Jarvis jest 5 razy szybszy w przypadku zestawu D (3 razy szybszy dla zestawu C).
- Widać zatem, że zastosowanie algorytmu Jarvisa ma sens wtedy, gdy jesteśmy pewni, że punktów leżących na otoczce będzie dużo mniej niż wszystkich punktów w zestawie.
- Jeśli nie wiemy nic o liczbie punktów na otoczce lepiej wybrać algorytm Grahama ponieważ jest on dużo bardziej uniwersalny