LABORATORIUM 2

Algorytmy geometryczne

Ćwiczenie 2 – sprawozdanie

Hieronim Koc

09.11.2022

grupa nr 4, czwartek_11_np_4

Dane techniczne specyfikacji komputerowej:

- Procesor : AMD Ryzen 5 4600H with Radeon Graphics 3.00 GHz
- RAM: 8.00 GB
- Typ system: 64-bitowy system operacyjny
- Środowisko: Jupyter notebook
- System operacyjny: Ubuntu 22.04 LTS Jammy Jellyfish (2022-04-21)

Wszystkie testy i ich implementacje były wykonywane w języku programowania Python 3.9.12, z wykorzystaniem zewnętrznych bibliotek takich jak: *numpy*, *matplotlib.* Oraz wewnętrznej biblioteki jaką jest *random* oraz *time*.

Cel ćwiczenia

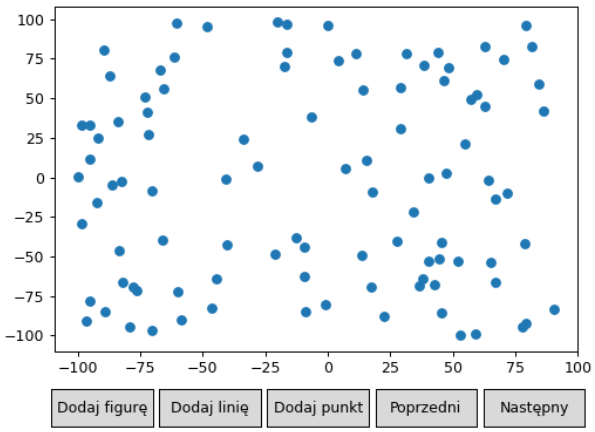
Celem ćwiczenia było zapoznanie się z podstawowymi algorytmami do wyznaczania otoczki wypukłej jak: algorytm Grahama oraz algorytm Jarvisa z równoczesną implementacją zbioru punktów, na którym owe algorytmy bazowały, wraz z wizualizacją ich działania jak i zmierzeniem wydajności pod względem znajdywania otoczki dla różnych wprowadzanych danych.

Opis ćwiczenia

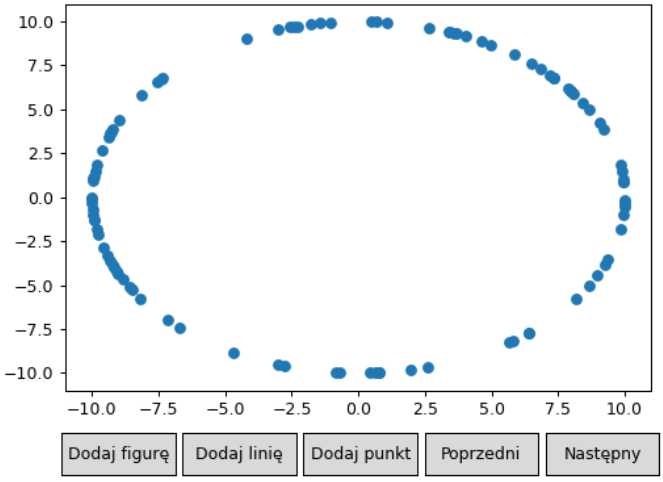
Ćwiczenie składało się z implementacji dwóch algorytmów, wygenerowaniem punktów, jak i opisu graficznego dla kolejnych kroków działania algorytmów.

Generowanie punktów na płaszczyźnie

Dla zestawu I 100 punktów zostało rozłożonych losowo dla przedziału [-100,100]. Przykład takiego rozłożenia przestawia Rys.1.1.

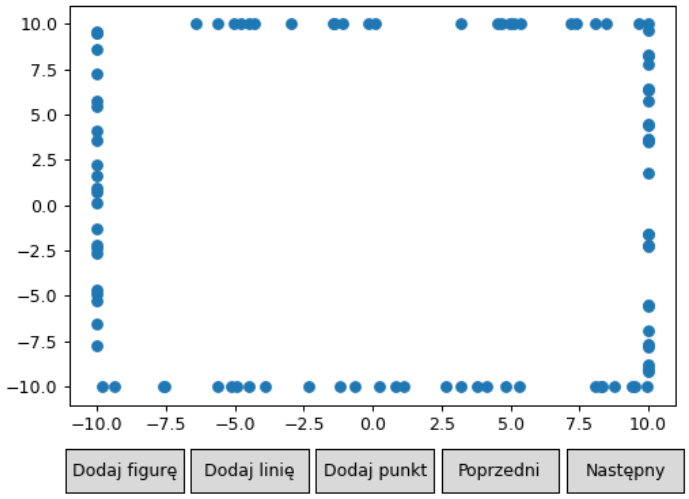


Dla zestawu II 100 punktów zostało rozłożonych losowo na okręgu o promieniu R = 10 o środku (0,0). Przykład takiego rozłożenia przedstawia Rys.1.2.



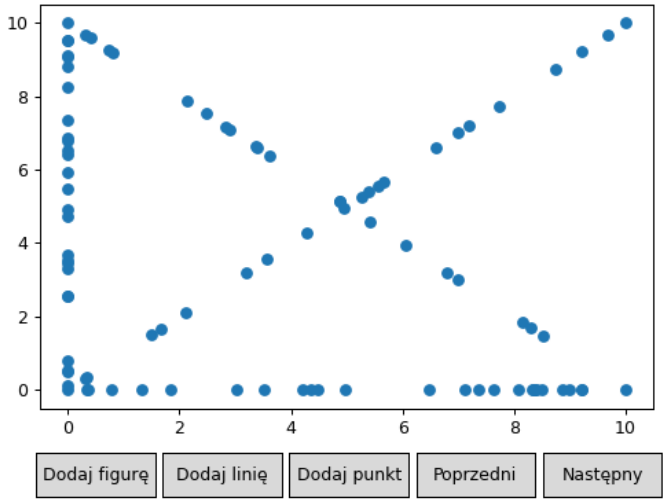
Rys.1.2

Dla zestawu III 100 punktów zostało rozłożonych losowo na bokach prostokąta o wierzchołkach (-10, 10), (-10,-10), (10,-10), (10,10). Przykład takiego rozłożenia przedstawia Rys.1.3.



Rys.1.3.

Dla zestawu IV 25 punktów zostało rozłożonych losowo na dwóch bokach kwadratu leżących na osiach i 20 punktów na przekątnych kwadratu o wierzchołkach (0, 0), (10, 0), (10, 10), (0, 10). Przykład takiego rozłożenia przedstawia Rys.1.4.



Rys.1.4

Algorytm Grahama i Jarvisa

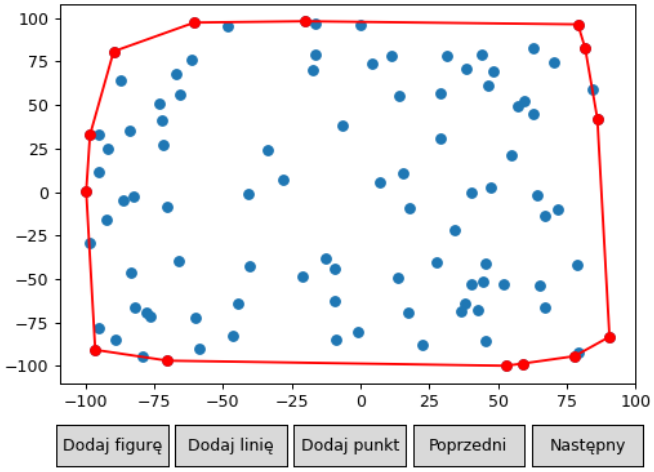
Dla algorytmów Grahama oraz Jarvisa do wyznaczania punktu względem prostej został użyty iloczyn wektorowy. Dla wszystkich przypadków $\varepsilon = 10\text{e-}12$. Przybliżenie czasu, które zostało zastosowane dla algorytmów wynosiło 5 miejsc po przecinku danego czasu. Dla algorytmu Grahama dodatkowo została zaimplementowana funkcja sortująca na bazie znanego już algorytmu sortującego *quicksort.* Otoczka wypukła dla każdego zbioru punktów wyznaczona tymi algorytmami była taka sama.

Wynik programu dla zestawu I przedstawia Rys.2.1. Dla punktów z zestawu I oba algorytmy szybko znalazły otoczkę wypukłą. Była to mała ilość punktów, jak i zarazem mała ilość punktów współliniowych.

Czas działania algorytmów dla danego zbioru punktu wyszedł odpowiednio:

-- algorytm Jarvisa: 0.00802 s

-- algorytm Grahama: 0.00199 s



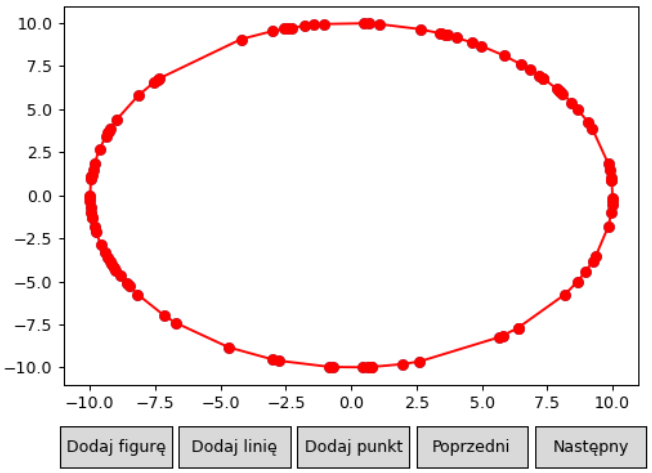
Rys.2.1

Wynik programu dla zestawu II przedstawia Rys.2.2. Algorytm Grahama znalazł szybko otoczkę wypukłą. Problem napotkał algorytm Jarvisa, dla którego złożoność czasowa wyniosła $O(n^2)$, ponieważ wszystkie punkty na okręgu okazały się należeć do otoczki i ów algorytm musiał wiele razy sprawdzać następne punkty.

Czas działania algorytmów dla danego zbioru punktu wyszedł odpowiednio:

-- algorytm Jarvisa: 0.48308 s

-- algorytm Grahama: 0.00299 s



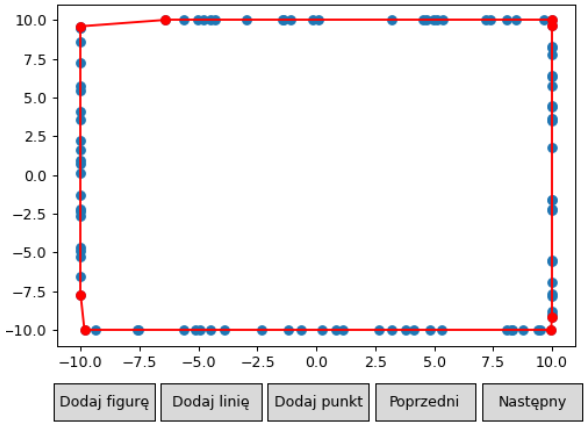
Rys.2.2

Wynik programu dla zestawu III przedstawia Rys.2.3. Oba algorytmy dla zadanego zbioru danych poradziły sobie bezproblemowo. Został tu poruszony problem współliniowości. Jak można zobaczyć po czasach wykonania, ani pierwszy, ani drugi od odstawał od siebie za bardzo i w dość szybkim czasie wykonały zadanie.

Czas działania algorytmów dla danego zbioru punktu wyszedł odpowiednio:

-- algorytm Jarvisa: 0.00500 s

-- algorytm Grahama: 0.00101 s



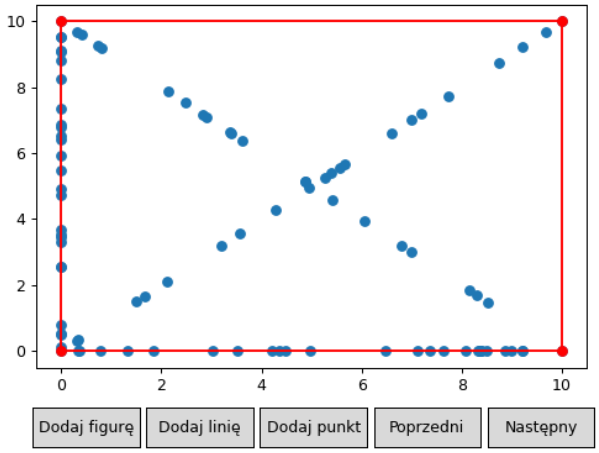
Rys.2.3

Wynik programu dla zestawu IV przedstawia Rys.2.4. W tym zestawie również został włączony problem współliniowości, jednakże w połączeniu z przypadkiem wracania się. Tylko najdalszy punkt współliniowy został zaliczony do otoczki, dzięki prawidłowemu zaimplementowaniu wyliczania wyznacznika dla danych punktów. Algorytmy wyznaczyły otoczkę szybko i poprawnie.

Czas działania algorytmów dla danego zbioru punktu wyszedł odpowiednio:

-- algorytm Jarvisa: 0.00300 s

-- algorytm Grahama: 0.00200 s



Rys.2.4

Czasy działania algorytmów, dla różnych ilości punktów

Dla lepszego wyznaczenia, czy algorytmy działają poprawnie dla różnej ilości punktów w danych zbiorach i w jakim czasie zostały wyznaczone otoczki wypukłe, ów algorytmy zostały ponownie uruchomione dla tych samych zbiorów.

Ilości punktów, które zostały wykorzystane to 10, 100, 1000 i 10000.

Dla zbioru punktów z zestawu I oba algorytmy zachowywały się podobnie, jednak algorytm Grahama był minimalnie szybszy, większą różnicę można zobaczyć dla 10000 punktów. W zestawie II czasy wykonania bardzo się różniły, gdyż algorytm Jarvisa posiadał złożoność obliczeniową $O(n^2)$ ze względu na sprawdzanie wszystkich punktów wiele razy, gdyż każdy z nich należał do otoczki wypukłej, gdzie algorytm Grahama posiadał złożoność obliczeniową rzędu O(nlogn). Natomiast dla zestawów III i IV algorytm Jarvisa przeważał nad algorytmem Grahama przy większej liczbie punktów, ponieważ algorytm Grahama musiał sprawdzać wiele razy punkty współliniowe i wybierać najbardziej oddalony od ostatniego punktu otoczki. Wszystkie czasy dla każdego zbioru punktów zostały zamieszczone w Tabeli 1.

Zbiór punktów	Algorytm	Ilość punktów: 10	Ilość punktów: 100	Ilość punktów: 1000	Ilość punktów: 10000
I	Grahama	0.00004 s	0.00036 s	0.00581 s	0.07943 s
	Jarvisa	0.00010 s	0.00088 s	0.01216 s	0.15455 s
II	Grahama	0.00004 s	0.00051 s	0.00796 s	0.12026 s
	Jarvisa	0.00019 s	0.01236 s	1.20264 s	120.17902 s
III	Grahama	0.00004 s	0.00063 s	0.00967 s	0.15476 s
	Jarvisa	0.00014 s	0.00083 s	0.00816 s	0.09183 s
IV	Grahama	0.00022 s	0.00338 s	0.05095 s	0.73809 s
	Jarvisa	0.00016 s	0.00095 s	0.00939 s	0.09437 s

Tabela 1

Wnioski

Algorytm Jarvisa i Grahama prawidłowo wyznaczyły otoczki wypukłe dla wszystkich zbiorów punktów. Jak można zauważyć algorytm Grahama otrzymuje podobne wyniki dla różnych zbiorów danych. Jest to spowodowane tym że złożoność obliczeniowa nie zależy od ilości punktów otoczki wypukłej, jak występuje to w algorytmie Jarvisa. Jednakże dla zbiorów punktów, które mają więcej współliniowych punktów wybór algorytmu Jarvisa jest naturalnym wyborem. Przy uruchomieniu algorytmu Jarvisa przy 10**6 punktów dla okręgu przy 80 min wykonywania, program został przerwany. Można wywnioskować z tego, że jeżeli nie wiadomo z jakim rozłożeniem punktów przyjdzie pracować, warto wybrać algorytm Grahama, pomimo że algorytm Jarvisa jest asymptotycznie szybszy.