Paweł Augustyn
gr. Poniedziałek 15:00 A

Algorytmy Geometryczne – Labolatorium 2
Otoczka wypukła

1. Dane techniczne

- System operacyjny: Windows 11 (x64-64)
- Procesor: Intel Core 7 150U (1.8 GB)
- Pamięć RAM: 16 GB
- Środowisko: Jupyter Notebook
- Język: Python 3.13.8

Wykorzystane biblioteki: numpy, pandas, random, math, time, narzędzie wizualizacji koła naukowego BIT.
Przyjęta precyzja to float64 dla całego ćwiczenia

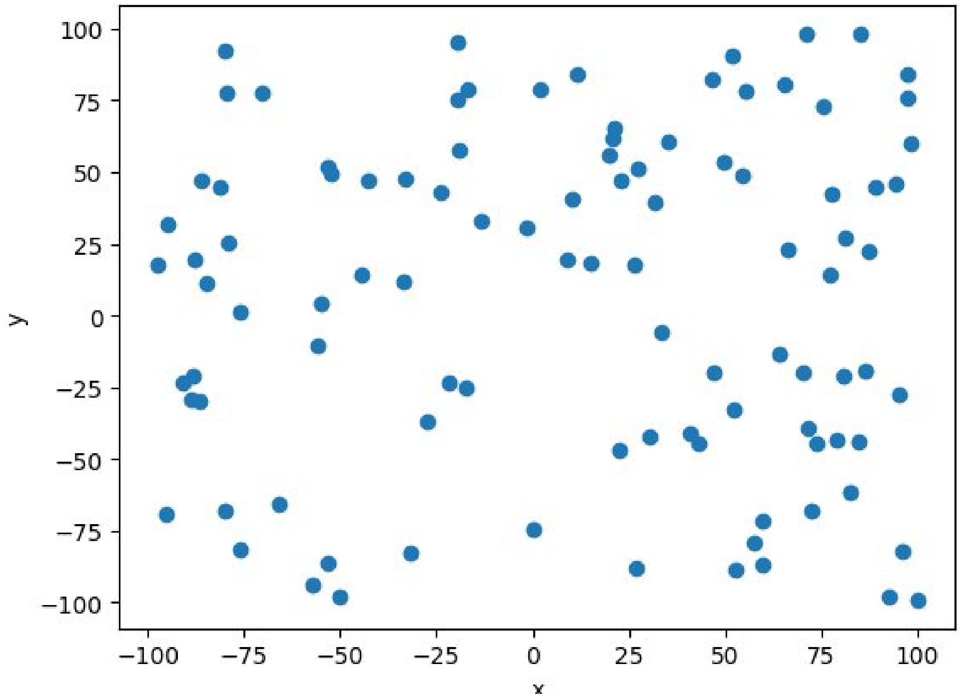
2. Opis ćwiczenia

Ćwiczenie ma na celu wyznaczanie otoczki wypukłej używając algorytmu grahama oraz algorytmu jarvisa, a także porównanie ich czasu wykonania. Częścią tego ćwiczenia jest też implementacja wizualizacji kroków wykonania algorytmów i analiza danych. W tym ćwiczeniu otoczka wypukła jest zdefiniowana jako niajmniejszy zbiór wypukły zawierający podany punkt startowy. W przypadku punktów współliniowych wybieramy tylko ten najbardziej oddalony

3. Przebieg ćwiczenia

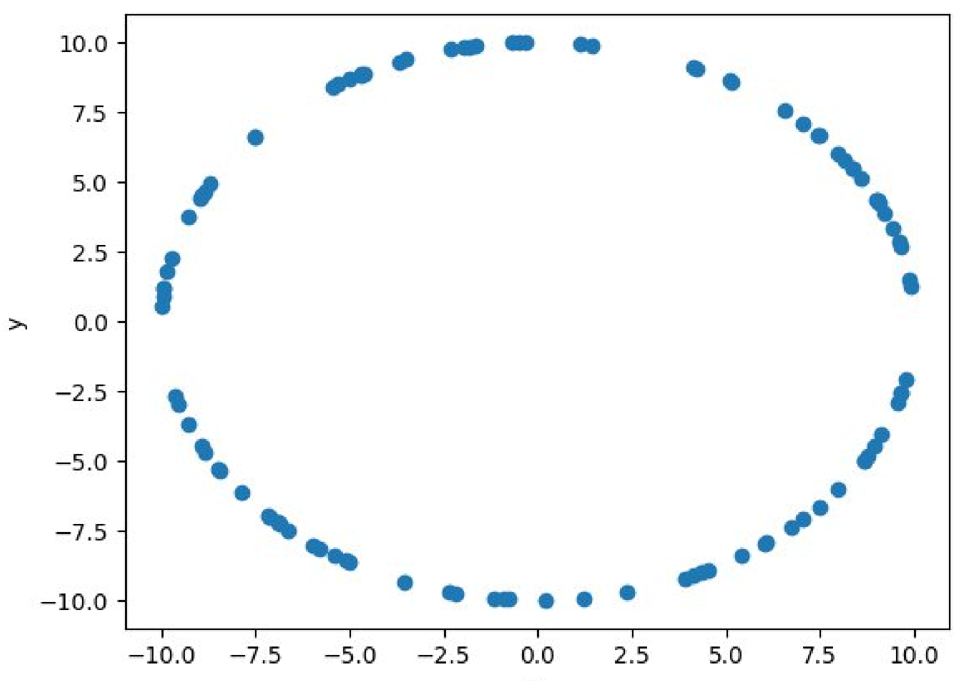
3.1 Generowanie punktów do danych testowych

A. Zbiór A – punkty generowane losowo w obszarze prostokątnym (Przykład: 100 punktów wygenerowanych na obszarze x=[-100, 100], y = [-100, 100])



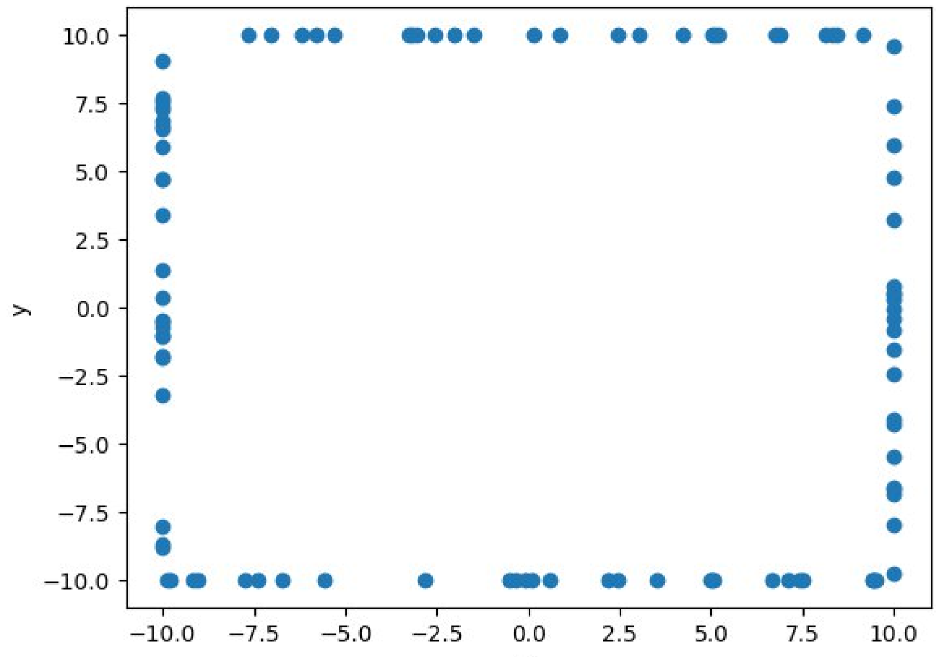
Rysunek 1: Przykładowy zbiór A

B. Zbiór B – punkty wygenerowane losowo na okręgu (Przykład: 100 punktów wygenerowanych na okręgu o środku w punkcie (0,0) i promieniu równym 10)



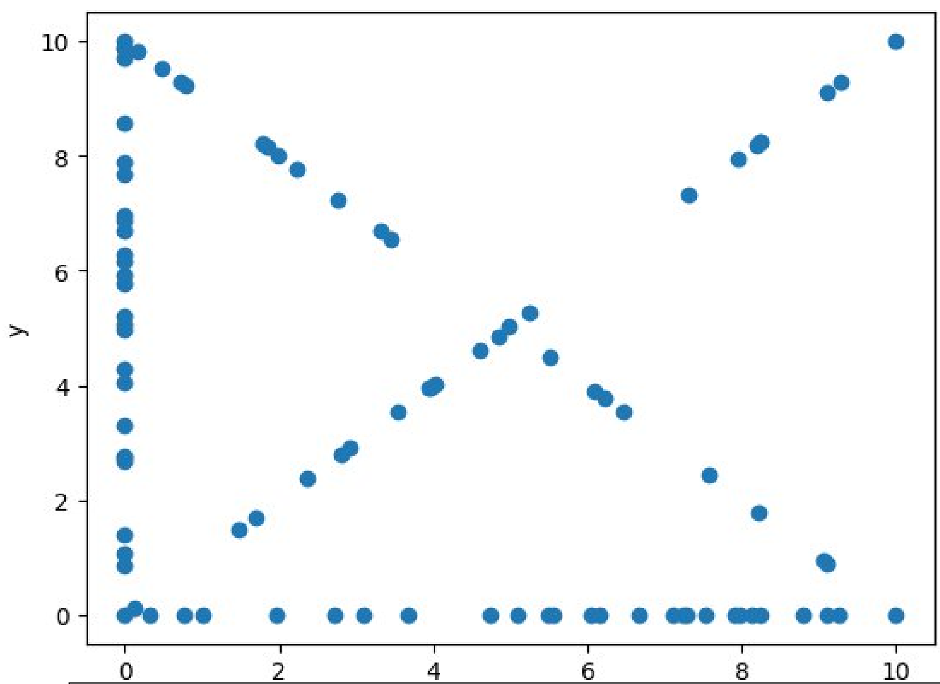
Rysunek 2: Przykładowy zbiór B

C. Zbiór C – punkty wygenerowane losowo na krawędziach prostokąta (Przykład: 100 punktów wygenerowanych na krawędziach prostokąta wyznaczonego przez punkty:
(-10, -10), (10, -10), (10, 10), (-10, 10))



Rysunek 3: Przykładowy zbiór C

D. Zbiór D – punkty wygenerowane losowo na dwóch krawędziach kwadratu leżących na osiach X oraz Y, a także na przekątnych tego kwadratu (Przykład: 25 punktów wygenerowanych na krawędziach, 20 na przekątnych oraz 4 dodatkowe punkty na wierzchołkach kwadratu wyznacznonego na punktach: (0, 0), (10, 0), (10, 10), (0, 10))



Rysunek 4: Przykładowy zbiór D

3.2 Implementacja algorytmów

Algorytm Grahama:

Ten algorytm tworzy otoczkę wypukłą poprzez użycie stosu, na którym znajdują się możliwi kandydaci spośród punktów, które mogą być częścią otoczki. Algorytm wstawia oraz usuwa elementy stosu, dopóki nie zostaną na nim same elementy otoczki wypukłej. W mojej implementacji algorytmu dane są początkowo sortowane według kąta względem punktu startowego, wyznaczonego przez wbudowaną funkcję cotangensa (math.atan2). Jeśli kąt dwóch punktów jest taki sam, ich kolejność jest rozstrzygana przez ich odległość o punktu startowego.

Złożoność tego algorytmu dla n punktów w zbiorze początkowym wynosi O(nlogn), ponieważ każdy punkt jest analizowany tylko raz, dominuje złożoność sortowania.

Algorytm Jarvisa:

Algorytm znany także jako “algorytm owijania prezentów” (eng. Gift wrapping algorithm), w każdym kroku wybiera punkt tworzący najmniejszy kąt z aktualnie wybraną krawędzią. Ostatecznie, algorytm znajdzie poprawne rozwiązanie.

Złożoność algorytmu Jarvisa wacha się znacznie w zależności od podanych danych, ale ogólna złożoność szacowana jest na O(nk), gdzie n to liczba punktów w zbiorze a k to ilość punktów w otoczce

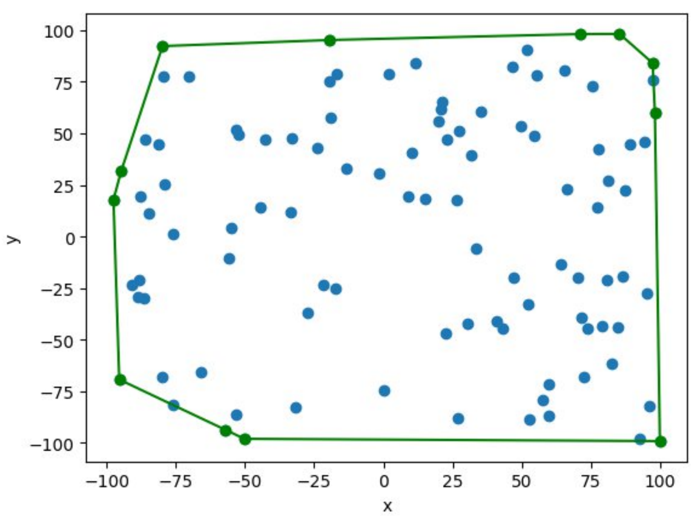
Wizualizacja algorytmów:

Procesy działania algorytmów są załączone w formacie .gif, które zostaną wysłane razem ze sprawozdaniem

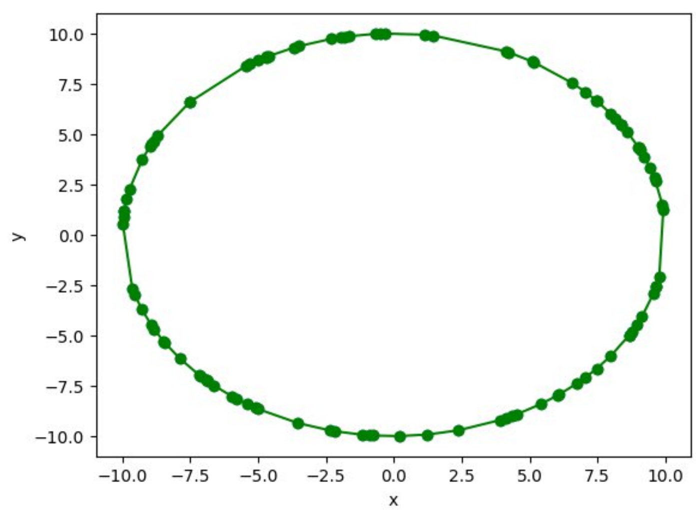
4. Analiza wyników

4.1 Zbiory przykładowe

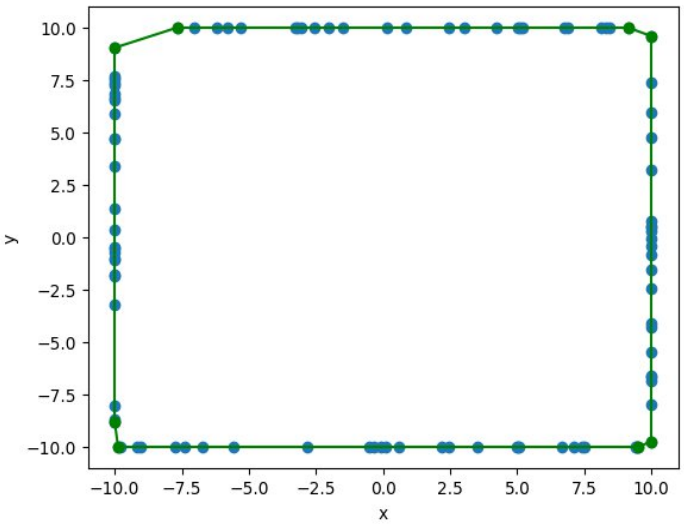
Wizualizacje otoczek otrzymanych przez oba algorytmy. Punkty należące do otoczki oraz krawędzie między nimi zostały zaznaczone na zielono, reszta punktów zaś na niebiesko.



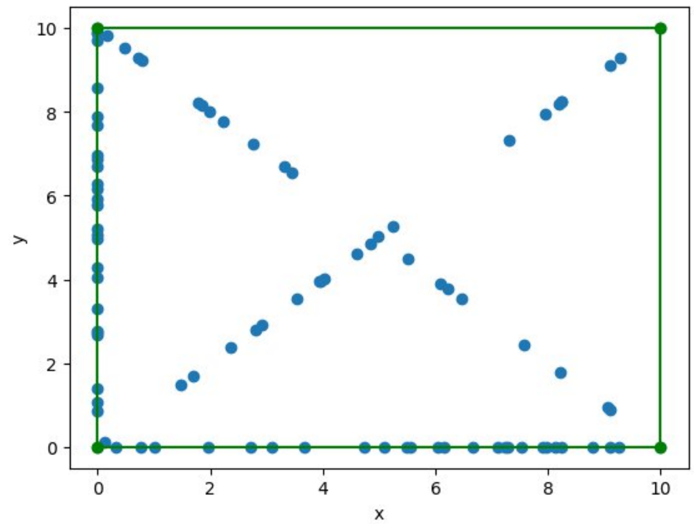
Rysunek 5: Otoczka zbioru A



Rysunek 6: Otoczka zbioru B



Rysunek 7: Otoczka zbioru C



Rysunek 8: Otoczka zbioru D

Zbiór punktów	**Liczba punktów w otoczce**	
	Algorytm Grahama	**Algorytm Jarvisa**
A	12	12
B	100	100
C	8	8
D	4	4

Tabela 1: Wyniki algorytmów dla zbiorów przykładowych

Jak widać wyniki obu algorytmów pokrywają się, co oznacza że oba zostały zaimplementowane poprawnie.
Dla zbioru A widać że otoczka liniowa została wyliczona poprawnie i algorytm nie popełnił żadnych znaczących błędów.
Dla zbioru B wygenerowana otoczka składa się z każdego punktu na obwodzie koła, co także jest wynikiem poprawnym. Warto zauważyć że w tym przypadku przyjęty został epsilon = 0, więc algorytm poprawnie poradził sobie z klasyfikacją punktów jako niewspółliniowych.
Dla zbioru C i D podobnie algorytm zadziałał poprawnie. W szczególności trzeba zauważyć że algorytm prawidłowo uwzględnił punkty współliniowe i pominął je w obliczeniach

4.2 Zbiory generowane

Metody generowania zbiorów:

1. Zbiór A: losowo wygenerowane punkty o współrzędnych z przedziału [-1000, 1000]
2. Zbiór B: losowo wygenerowane punkty leżące na okręgu o środku (0,0) i promieniu równym 1000
3. Zbiór C: losowo wygenerowane punkty leżące na krawędziach prostokąta wyznaczonego punktami (-1000, -1000), (1000, -1000), (-1000, 1000), (1000, 1000)
4. Zbiór D: losowo wygenerowane punkty leżące na dwóch krawędziach i przekątnych kwadratu wyznaczonego punktami (-1000, -1000), (1000, -1000), (-1000, 1000), (1000, 1000)

Liczności punktów w zbiorach:

Ilości punktów w każdej próbie: [100, 500, 1000, 2000, 5000, 10000, 20000]

Niepewność punktów współliniowych:

Do wyznaczania współliniowości punktów przyjmujemy niepewność epsilon = 10**-20

4.3 Analiza zbioru A

Liczność zbioru		Czas działania [s]	
Wszystkie punkty	Punkty otoczki	Algorytm Grahama	Algorytm Jarvisa
100	10	0.000141	0.000179
500	11	0.000481	0.000785
1000	18	0.000856	0.002851
2000	23	0.001858	0.009163
5000	24	0.004777	0.020359
10000	20	0.011559	0.03316
20000	28	0.023113	0.092448
50000	32	0.075623	0.300416

Tabela 2: Wyniki pomiarów zbioru A

Z danych zawartych w tabeli A wynika, że dla zbioru A algorytm Grahama otrzymuje wynik szybciej, ale wciąż oba algorytmy działają w akceptowalnym czasie

4.4 Analiza zbioru B

Liczność zbioru		Czas działania [s]	
Wszystkie punkty	Punkty otoczki	Algorytm Grahama	Algorytm Jarvisa
100	100	0.0001	0.001364
500	500	0.000381	0.039157
1000	1000	0.000825	0.163116
2000	2000	0.001968	0.624915
5000	5000	0.004649	4.138641
10000	10000	0.008467	17.055349
20000	20000	0.02158	174.339159
50000	50000	0.153259	697.964604

Tabela 3: Wyniki pomiarów zbioru B

Z danych w tabeli B wynika, że dla zbioru B algorytm Grahama działa dużo lepiej niż algorytm Jarvisa. Duża ilość punktów w otoczce powoduje ekstremalnie wolne działanie algorytmu Jarvisa, ale nie powoduje większych problemów dla algorytmu Grahama

4.5 Analiza zbioru C

Liczność zbioru		Czas działania [s]	
Wszystkie punkty	Punkty otoczki	Algorytm Grahama	Algorytm Jarvisa
100	8	0.000345	0.000507
500	8	0.001488	0.002529
1000	8	0.002588	0.005462
2000	8	0.006162	0.01108
5000	8	0.015512	0.026828
10000	8	0.039068	0.052846
20000	8	0.069869	0.104313
50000	8	0.197999	0.277921

Tabela 4: Wyniki pomiarów zbioru C

Z danych w tabeli C wynika, że dla zbioru C oba algorytmy działają w podobnym, niskim czasie. Ten zbiór zawiera dużo punktów współliniowych, co daje przewagę algorytmowi Jarvisa, ale dla takiej ilości danych i tylko punktów współliniowych oba algorytmy radzą sobie podobnie

4.6 Analiza zbioru D

Liczność zbioru		Czas działania [s]	
Wszystkie punkty	Punkty otoczki	Algorytm Grahama	Algorytm Jarvisa
100	5	0.000383	0.000433
500	5	0.00158	0.001871
1000	5	0.0037	0.005235
2000	5	0.006429	0.007233
5000	5	0.018369	0.018042
10000	5	0.03742	0.035619
20000	5	0.073978	0.079614
50000	5	0.219623	0.176718

Tabela 5: Wyniki pomiarów zbioru D

Z danych w tabeli D wynika, że dla zbioru D algorytm Jarvisa działa szybciej niż algorytm Grahama. Duża ilość punktów współliniowych gdzie nie wszystkie są kandydatami na punkty w otoczce dają dużą przewagę algorytmowi Jarvisa.

5. Wnioski

Na podstawie wyników ćwiczenia można wyciągnąć następujące wnioski:

1. Wyniki obu algorytmów są identyczne, więc można wywnioskować że implementacja jest poprawna
2. Dla zbiorów losowych algorytm Grahama jest szybszy przez jego niższą złożoność
3. Algorytm Jarvisa ma przewagę dla zbiorów z niską ilością punktów w otoczce, natomiast dla sytuacji odwrotnej (zbiór B) algorytm znacznie spowalnia

Podsumowując, algorytm Grahama i algorytm Jarvisa bardzo dobrze wyznaczają otoczkę, ale który działa lepiej zależy bezpośrednio od charakterystyki punktów.