Jacek Łoboda gr 6, Pon. 16:45

Algorytmy geometryczne, ćwiczenie 2

1. Dane techniczne

- Procesor: Intel Core i7-7700HQ 2.80GHz
- RAM: 16GB 2133 MHz
- System operacyjny: Windows 10 Home x64
- Środowisko: Visual Studio Code
- Język: Python 3.13.5

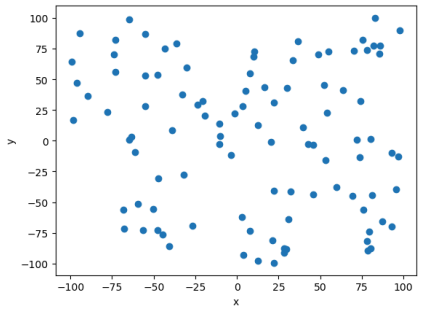
Użyto następujących bibliotek: numpy, pandas, matplotlib, random, functools, time. Do graficznej wizualizacji wykorzystano narzędzie koła naukowego BIT.

2. Cel ćwiczenia

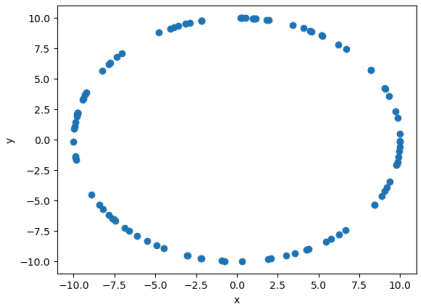
Celem ćwiczenia było zapoznanie się z algorytmem Grahama oraz Jarvisa do wyznaczania otoczki wypukłej oraz porównanie ich wydajności na różnych zbiorach punktów na płaszczyźnie.

3. Generowanie zbiorów

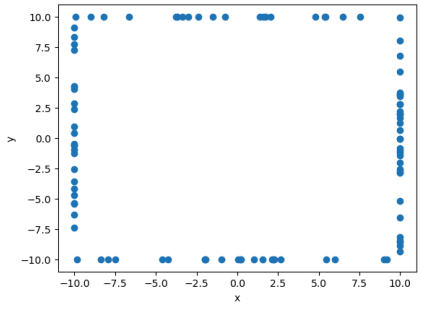
Przy pomocy funkcji random.uniform() przygotowano 4 typy zbiorów. Do pomiaru czasu działania algorytmów wykorzystano warianty tych zbiorów różniące się całkowitą liczbą punktów.



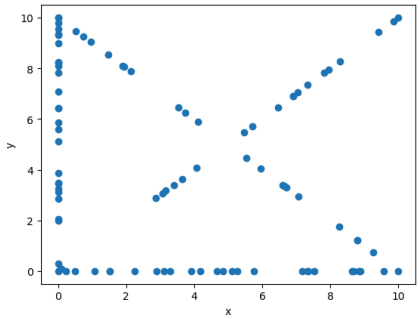
Rys. 1: Zbiór A
100 losowo wygenerowanych punktów o współrzędnych z przedziału [−100, 100].



Rys. 2: Zbiór B
100 losowo wygenerowanych punktów leżących na okręgu o środku (0, 0) i promieniu R=10.



Rys. 3: Zbiór C
100 losowo wygenerowanych punktów leżących na bokach kwadratu o wierzchołkach (-10, −10), (10, −10), (10, 10), (-10, 10).

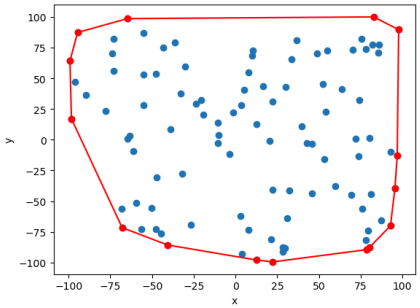


Rys. 4: Zbiór D
4 wierzchołki kwadratu (0, 0), (10, 0), (10, 10), (0, 10), oraz punkty wygenerowane losowo: po 25 punktów na bokach kwadratu leżących na osiach i po 20 punktów leżących na przekątnych.

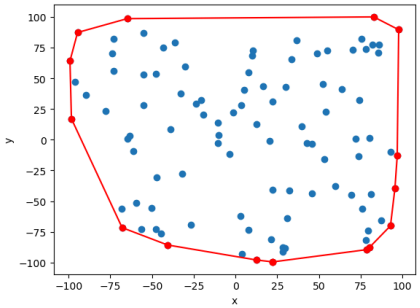
4. Wyniki

Rysunki przedstawiające wyniki działania algorytmów na zbiorach grupują punkty kolorami w następujący sposób: niebieskie - punkt ze zbioru nie należący do otoczki wypukłej, czerwony - punkt ze zbioru należący do otoczki wypukłej. Dodatkowo punkty czerwone połączono czerwonymi liniami tworząc wizualizacje otoczki

4.1 Zbiór A



Rys. 5: Otoczka wypukła wyznaczona przez Algorytm Grahama na zbiorze A.



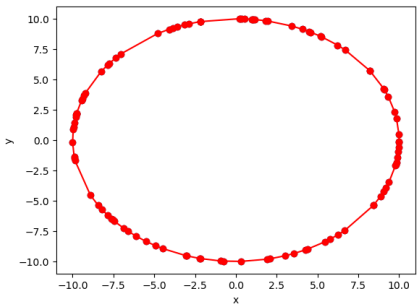
Rys. 6: Otoczka wypukła wyznaczona przez Algorytm Jarvisa na zbiorze A.

liczba pkt.	graham pkt. otoczki	graham czas	jarvis pkt. otoczki	jarvis czas
1,00E+02	14	0,0004665852	14	0,0007052422
1,00E+03	21	0,0089628696	21	0,0094487667
1,00E+04	27	0,0905530453	27	0,1563122272
1,00E+05	29	1,2850613594	29	1,2365109921
1,00E+06	37	12,2891380787	37	13,0771617889

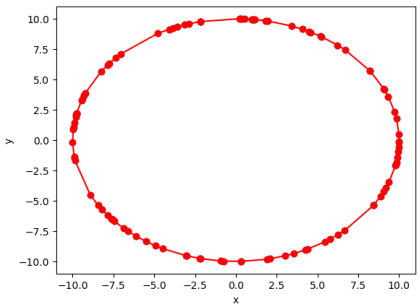
Tabela 1: Wyniki pomiaru czasu dla zmodyfikowanego zbioru A.

Oba algorytmy poprawnie wyznaczyły taką samą otoczkę wypukłą (Rys. 5 i 6). Z danych zawartych w Tabeli 1 wynika, że dla większości przypadków algorytm Grahama był nieznacznie szybszy, z wyjątkiem n=1,00E+05, gdzie szybszy okazał się algorytm Jarvisa.

4.2 Zbiór B



Rys. 7: Otoczka wypukła wyznaczona przez Algorytm Grahama na zbiorze B.



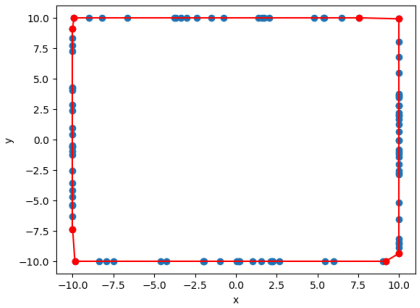
Rys. 8: Otoczka wypukła wyznaczona przez Algorytm Jarvisa na zbiorze B.

liczba pkt.	graham pkt. otoczki	graham czas	jarvis pkt. otoczki	jarvis czas
1,00E+02	1,00E+02	0,00349615	1,00E+02	0,01207923
1,00E+03	1,00E+03	0,02321264	1,00E+03	0,921331882
1,00E+04	1,00E+04	0,186848164	1,00E+04	93,9279151
1,00E+05	1,00E+05	2,12314558	1,00E+05	7651,250855

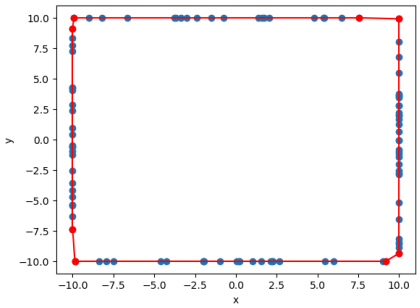
Tabela 2: Wyniki pomiaru czasu dla zmodyfikowanego zbioru B.

Oba algorytmy poprawnie wyznaczyły taką samą otoczkę wypukłą (Rys. 7 i 8). Z danych zawartych w Tabeli 2 wynika, że Algorytm Grahama wykonał zadanie znacznie szybciej niż Algorytm Jarvisa. Jest to spowodowane zestawem danych, W zbiorze B wszystkie punkty należą na otoczki wypukłej co sprowadza Algorytm Jarvisa do złożoności kwadratowej $O(n^2)$, ponieważ $k = n$.

4.3 Zbiór C



Rys. 9: Otoczka wypukła wyznaczona przez Algorytm Grahama na zbiorze C.



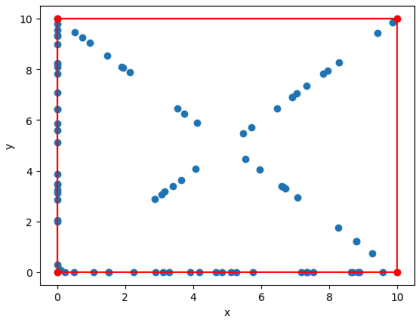
Rys. 10: Otoczka wypukła wyznaczona przez Algorytm Jarvisa na zbiorze C.

liczba pkt.	Graham pkt. otoczki	Graham czas	Jarvis pkt. otoczki	Jarvis czas
1,00E+02	8	0,00049687	8	0,00076854
1,00E+03	8	0,01055128	8	0,00928568
1,00E+04	8	0,13121649	8	0,05941864
1,00E+05	8	1,865250111	8	0,49017357
1,00E+06	8	19,42502737	8	4,9485116

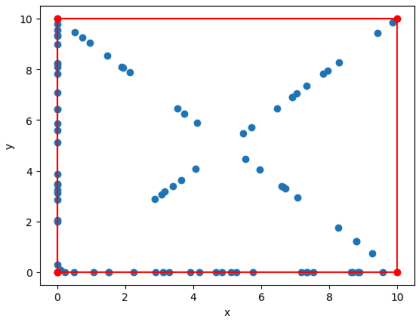
Tabela 3: Wyniki pomiaru czasu dla zmodyfikowanego zbioru C.

Oba algorytmy poprawnie wyznaczyły taką samą otoczkę wypukłą (Rys. 9 i 10). Z danych zawartych w Tabeli 3 wynika, że Algorytm Jarvisa wykonał zadanie szybciej niż Algorytm Grahama, zwyjątkiem n=1,00E+02. Powodem takiego rozkładu czasu jest zestaw danych, w zbiorze C do otoczki zawsze należeć będzie maksymalnie 8 punktów więc Algorytm Jarvisa zadziała liniowo.

4.4 Zbiór D



Rys. 11: Otoczka wypukła wyznaczona przez Algorytm Grahama na zbiorze D.



Rys. 12: Otoczka wypukła wyznaczona przez Algorytm Jarvisa na zbiorze D.

liczba pkt. osie	liczba pkt. przek.	graham otoczka	graham czas	jarvis otoczka	jarvis czas
25	20	4	0,00068983	4	0,00023519
250	200	4	0,01016736	4	0,00236435
2500	2000	4	0,12687397	4	0,01964497
25000	20000	4	2,310040951	4	0,197634935
250000	200000	4	23,74582243	4	2,074540138

Tabela 4: Wyniki pomiaru czasu dla zmodyfikowanego zbioru D

Oba algorytmy poprawnie wyznaczyły taką samą otoczkę wypukłą (Rys. 11 i 12). Z danych zawartych w Tabeli 4 wynika, że Algorytm Jarvisa wykonał zadanie szybciej niż Algorytm Grahama. Tak jak w przypadku zbioru C Algorytm Jarvisa zadziała liniowo. Dodatkowo ten zbiór sprawdza, czy Algorytm Grahama odpowiednio rozpoznaje i usuwa punkty współliniowe z (0, 0) ponieważ w zbiorze znajduje się ich więcej od tych nie współliniowych.

5. Wnioski

Testy wykazały, że wybór algorytmu powinien zależeć od struktury danych ponieważ bezpośrednio przekłada się ona na ich wydajność.

- Algorytm Grahama: Jest bardziej uniwersalny. Jego złożoność $O(n \log n)$ najlepiej sprawdza się dla danych losowych i rozproszonych, gwarantując stabilną wydajność niezależnie od stopnia skomplikowania struktury.
- Algorytm Jarvisa: Działa najlepiej, gdy otoczka wypukła składa się z niewielu punktów k lub dane są w większości współliniowe. Jego złożoność $O(nk)$ staje się wtedy liniowa i szybsza od Grahama.