Ćwiczenie II - Otoczka wypukła

Jakub Frączek - grupa nr 4

1. Wstęp

Opis ćwiczenia

Ćwiczenie polegało na wygenerowaniu 4 zbiorów punktów tj:
1) 100 losowo wygenerowanych punktów o współrzędnych z przedziału [-100, 100],
2) 100 losowo wygenerowanych punktów leżących na okręgu o środku (0,0) i promieniu R=10,
3) 100 losowo wygenerowanych punktów leżących na bokach prostokąta o wierzchołkach (-10, 10), (-10,-10), (10,-10), (10,10),
4) wierzchołki kwadratu (0, 0), (10, 0), (10, 10), (0, 10) oraz punkty wygenerowane losowo po 25 punktów na dwóch bokach kwadratu leżących na osiach i po 20 punktów na przekątnych kwadratu.

Następnie należało zaimplementować algorytm wyznaczania otoczki wypukłej Grahama i Jarvisa oraz przeanalizować działanie obu algorytmów na powyższych zbiorach danych.

Dane techniczne - software

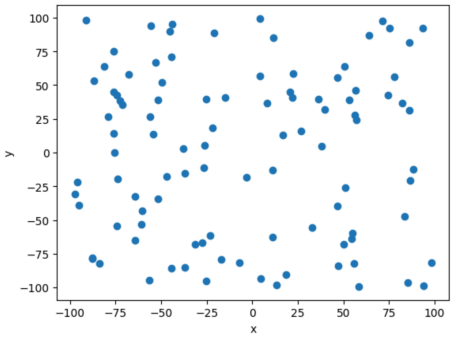
Ćwiczenie zostało zrealizowane w języku Python przy użyciu środowiska Jupyter notebook. Wykorzystane biblioteki to: numpy, random, pandas, matplotlib, bitalg, functools, collections, time.

Dane techniczne - hardware

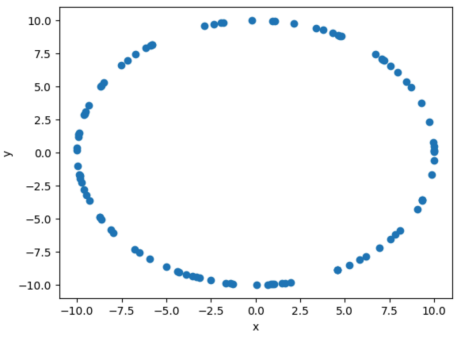
Laptop z systemem operacyjnym Linux Mint, procesorem AMD Ryzen 5 5500U 2.1 GHZ oraz 8 GB pamięci RAM.

2. Generacja punktów

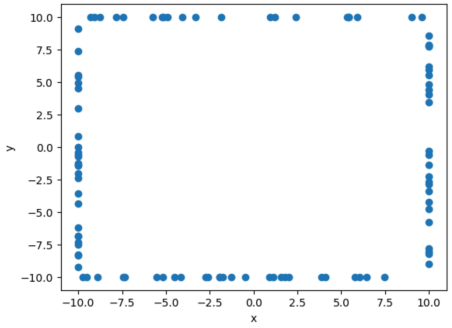
Punkty zostały wygenerowane za pomocą funkcji random.uniform() z biblioteki random. Wizualizacja punktów na poniższych wykresach (Wykres 1, Wykres 2, Wykres 3, Wykres 4) została zrealizowana za pomocą biblioteki bitalg przygotowanej przez koło naukowe BIT.



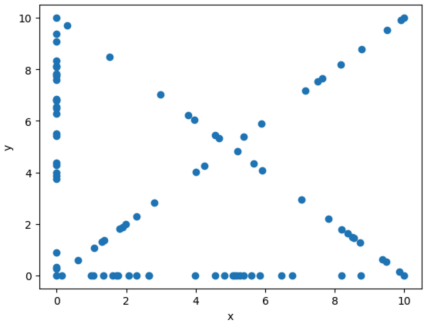
Wykres 1. Punkty ze zbioru 1.



Wykres 2. Punkty ze zbioru 2.



Wykres 3. Punkty ze zbioru 3.



Wykres 4. Punkty ze zbioru 4.

3. Algorytm Grahama

Algorytm Grahama składa się z następujących kroków:
1) Wybranie punktu P o najmniejszej współrzędnej y (dla takich samych y, ten o najmniejszym x-ie)
2) Posortowanie punktów z danego zbioru Q pod względem kąta jaki tworzą z osią OX układu kartezjańskiego oraz usunięcie punktów współliniowych (zostawienie najbardziej odległego punktu od P)
3) Umieszczenie na stosie S punktu P oraz dwóch pierwszych punktów z posortowanego zbioru Q
4) W głównej pętli bierzemy dwa punkty $A,\ B$ ze szczytu stosu S oraz kolejny punkt ze zbioru Q i jeżeli znajduje się on po lewej stronie prostej AB, to to dodajemy go na stos, a jeżeli po prawej to usuwamy punkt B ze stosu.

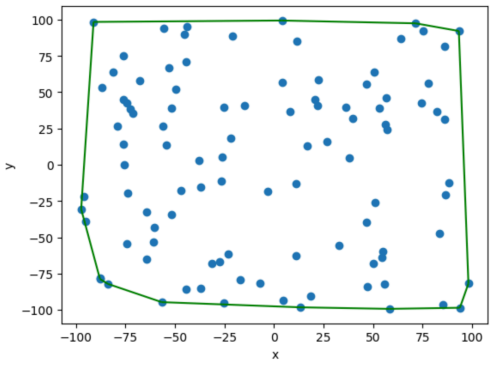
4. Algorytm Jarvisa

Algorytm Jarvisa składa się z następujących kroków:
1) Wybranie punktu P o najmniejszej współrzędnej y (dla takich samych y, ten o najmniejszym x-ie)
2) W głównej pętli wyszukanie takiego punktu A, dla którego pozostałe punkty znajdują się po lewej stronie prostej PA. Następnie jest on dodawany na stos S. Wykonywanie algorytmu kończy się, gdy dotrzemy do punktu wyjściowego P.

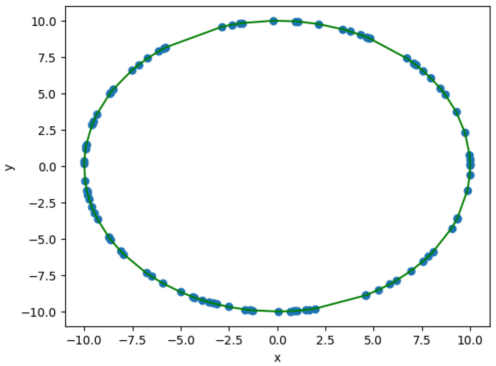
5. Analiza danych i porównanie wydajności algorytmów

Wygenerowane otoczki wypukłe

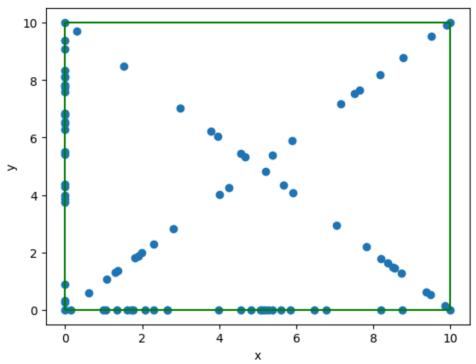
W przypadku obu algorytmów wygenerowane otoczki przedstawione na poniższych wykresach (Wykres 5, Wykres 6, Wykres 7 i Wykres 8) wypukłe są identyczne.



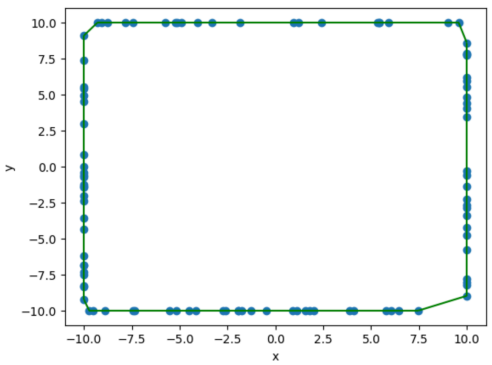
Wykres 5. Otoczka wypukła dla zbioru 1.



Wykres 6. Otoczka wypukła dla zbioru 2.



Wykres 7. Otoczka wypukła dla zbioru 3.



Wykres 8. Otoczka wypukła dla zbioru 4.

Porównanie czasów wykonania

Wykonałem porównanie czasów wykonania dwóch algorytmów dla małych i dużych danych (Tabela 1.). Algorytm Grahama okazał się szybszy przy losowo wygenerowanych punktach i punktach wygenerowanych na okręgu. Z tymi drugimi Algorytm Jarvisa poradził sobie bardzo źle, natomiast okazał się lepszych dla dwóch ostatnich zbiorów (punkty na kwadracie i na dwóch bokach oraz przekątnych).

Średni czas wykonania algorytmów Grahama i Jarvisa					
Liczba punktów	Algorytm	Zbiór 1.	Zbiór 2.	Zbiór 3.	Zbiór 4.
n = 100	Grahama	0.787 ms	0.659 ms	0.685 ms	0.923 ms
	Jarvisa	1.002 ms	7.556 ms	0.787 ms	0.459 ms
n = 500	Grahama	69.100 ms	59.883 ms	72.038 ms	96.258 ms
	Jarvisa	84.366 ms	19578.796 ms	38.960 ms	21.921 ms
n = 10000	Grahama	165.856 ms	118.326 ms	155.6748 ms	212.130 ms
	Jarvisa	212.032 ms	78170.409 ms	77.120 ms	39.896 ms

Tabela 1.

6. Wnioski

- Oba algorytmy poprawie wyznaczają otoczkę wypukłą dla zadanych zbiorów. Dla małej ilości punktów algorytmy działają w dość podobnym czasie. Wyjątkiem jest zbiór punktów wygenerowanych na okręgu, dla którego algorytm Jarvisa ulega ukwadratowieniu.
- Moja implementacja algorytmu Grahama zawierała dwa różne warianty radzenia sobie z punktami współliniowymi:
    - Usuwanie punktów współliniowych podczas sortowania, zrealizowane za pomocą HeapSort
    - Wykrywanie punktów współliniowych podczas głównej pętli w algorytmie Grahama

  Niestety pierwszy wariant algorytmów niepoprawnie wyznaczał otoczkę wypukłą dla testów zaproponowanych przez koło naukowe BIT, czas wykonania był również niezadowalający, dlatego zdecydowałem się na drugi wariant implementacji
- Czas wykonania algorytmu Grahama wynosi O(nlogn), a algorytmu Jarvisa O(kn), gdzie k to ilość punktów otoczki wypukłej, natomiast w najgorszym przypadku $O(n^2)$
- Algorytm Jarvisa może być szybszy od algorytmu Grahama dla niektórych zbiorów