Algorytmy geometryczne

Sprawozdanie z laboratorium 2

Otoczka wypukła

Iwo Zowada

Gr. 6 Wtorek 15:00-16:30 A

Dane techniczne

Komputer z systemem Windows 10
Procesor Intel Core i7-3770 CPU 3.40GHz
Pamięć RAM 24GB
Program napisany w języku Python w środowisku Jupyter Notebook z wykorzystaniem bibliotek numpy, random, matplotlib oraz narzędzia przygotowanego przez KN Bit

Opis ćwiczenia

Celem ćwiczenia jest implementacja algorytmów Grahama i Jarvisa do znajdowania otoczki wypukłej i znajdowanie otoczki dla wybranych zbiorów losowo wygenerowanych przez nas punktów oraz wizualizacja algorytmów.

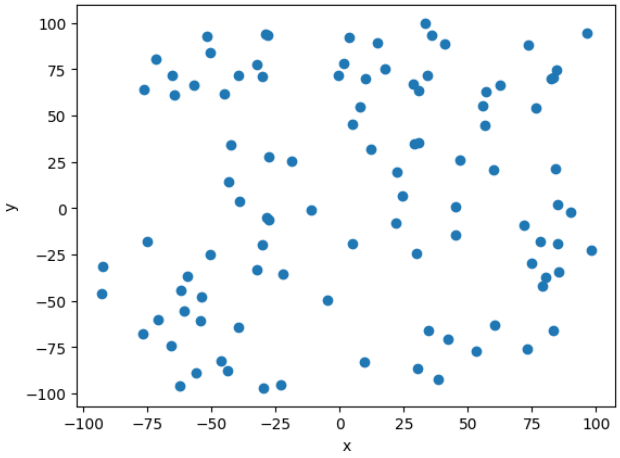
Realizacja ćwiczenia

1. **Generowanie punktów**

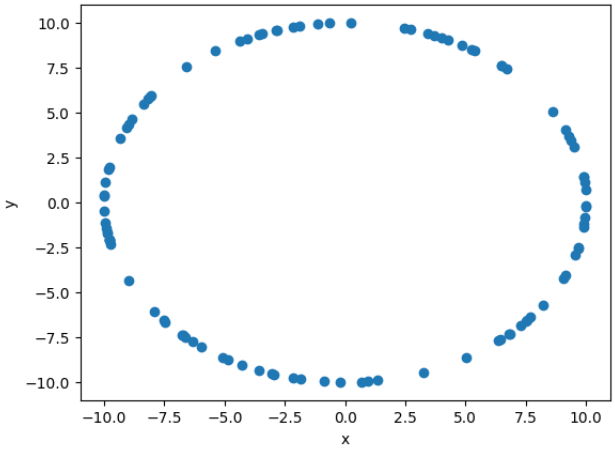
   Przygotowałem 4 zbiory punktów (2D, współrzędne rzeczywiste typu double), punkty losowe były generowane przy użyciu funkcji random.uniform() oraz random.randint() z biblioteki numpy

   a) 100 losowych punktów o współrzędnych (x,y) z przedziału [-100;100],
   b) 100 losowych punktów leżących na okręgu o środku w punkcie (0,0) i promieniu R = 10
   c) 100 losowych punktów leżących na obwodzie prostokąta, którego wyznaczają wierzchołki: (-10,-10), (10,-10), (10,10), (-10, 10)
   d) -25 punktów leżących na dwóch bokach kwadratu leżących na osiach,
      -20 punktów leżących na przekątnych kwadratu,
      Zawierające punkty wyznaczające kwadrat: (0,0), (10,0), (10,10), (0,10).

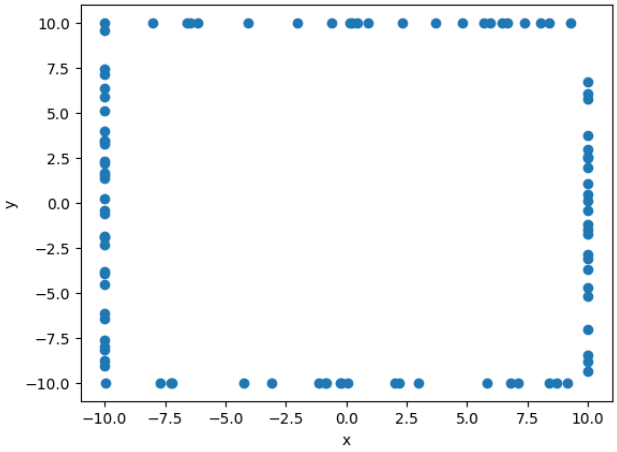
Przedstawienie zbiorów punktów na wykresach prezentuję poniżej:



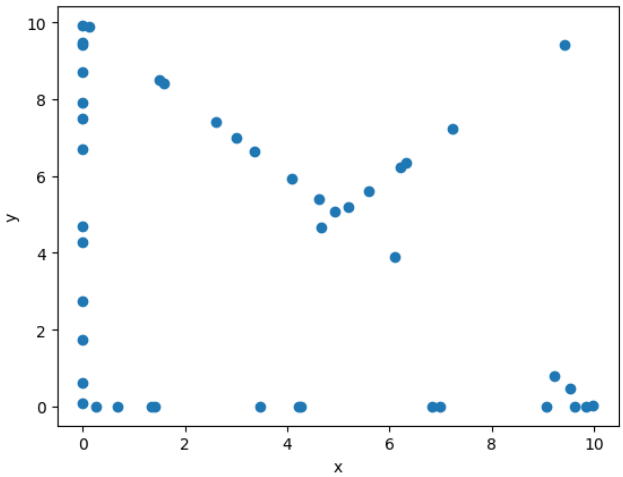
(Wykres 1.1): Zbiór A



(Wykres 1.2): Zbiór B



(Wykres 1.3): Zbiór C



(Wykres 1.4): Zbiór D

2. **Algorytmy Grahama i Jarvisa**
   Algorytm Grahama jak i Jarvisa są używane do wyznaczania otoczki wypukłej dla zbiorów punktów. Poniżej przedstawię na czym one polegają oraz ich różnice.

   Algorytm Grahama

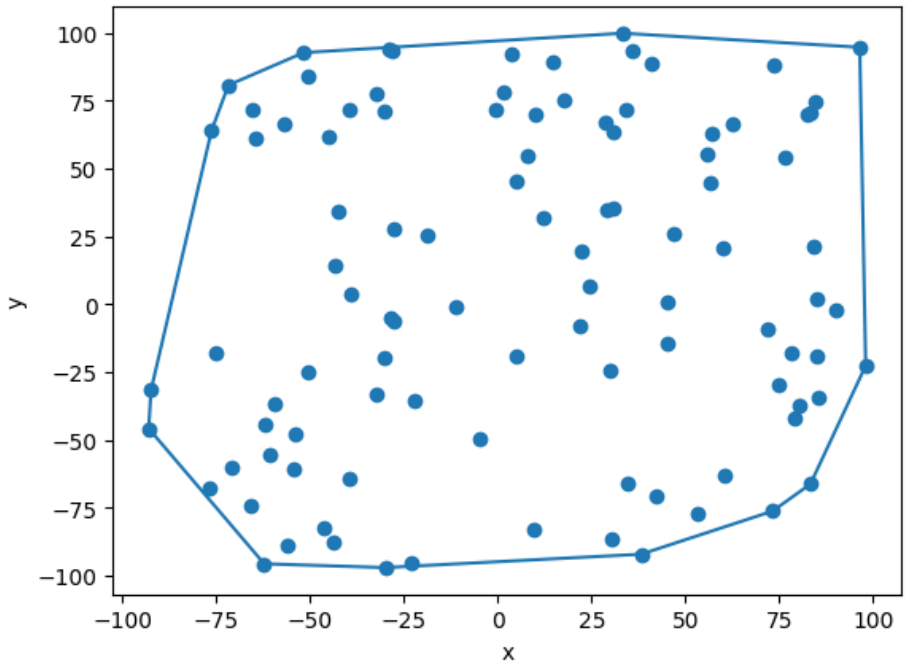
   1. **Działanie**:
      - Wybieramy punkt startowy o najmniejszej współrzędnej y, a w przypadku punktów z takim samym najmniejszym y wybieramy punkt o najmniejszej współrzędnej x,
      - Sortujemy wszystkie punkty według kąta, jaki tworzą z punktem startowym i osią x.
      - Przechodzimy przez posortowane punkty, dodając je do stosu (reprezentującego otoczkę) i odrzucamy po kolei te, które powodują prawo-skrętność (czyli nie tworzą wypukłości).
   2. **Złożoność**: O(n*logn)

   Algorytm Jarvisa

   1. **Działanie**:
      - Wybieramy punkt startowy o najmniejszej współrzędnej y, a w przypadku punktów z takim samym najmniejszym y wybieramy punkt o najmniejszej współrzędnej x,
      - Następnie znajduje kolejny punkt na otoczce, wybierając ten, który tworzy najbardziej "lewo-skrętny" kąt względem poprzedniego.
      - Kontynuuje dodawanie punktów na obwodzie, aż wróci do punktu początkowego.
   2. **Złożoność**: O(n*h), gdzie n to liczba punktów, a h to liczba punktów na otoczce wypukłej.

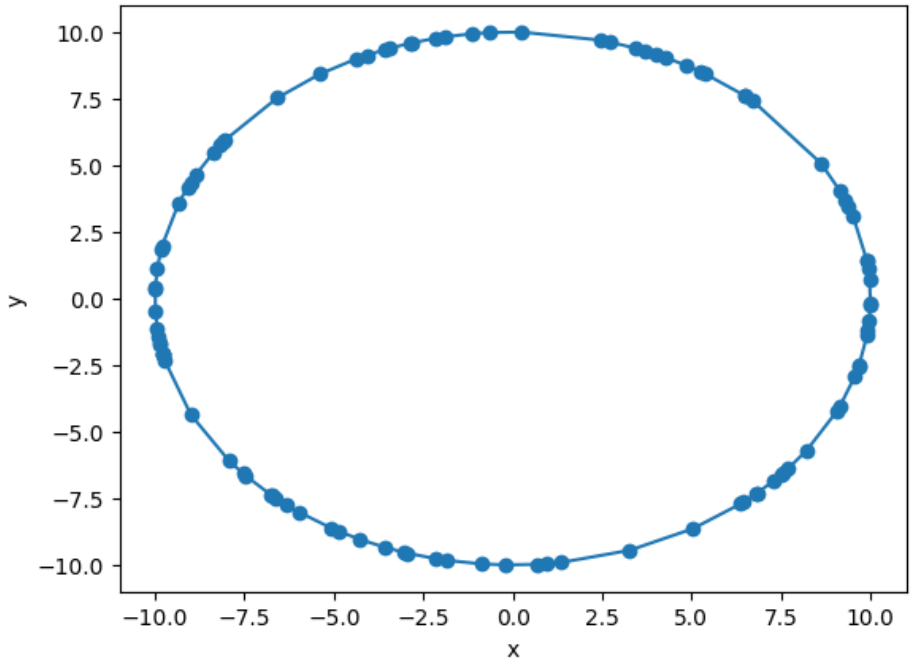
3. Otrzymane wyniki

Poniżej przedstawię zbiory punktów z zaznaczoną otoczką wypukłą wyznaczoną przez zaimplementowane przeze mnie algorytmy Grahama i Jarvisa.



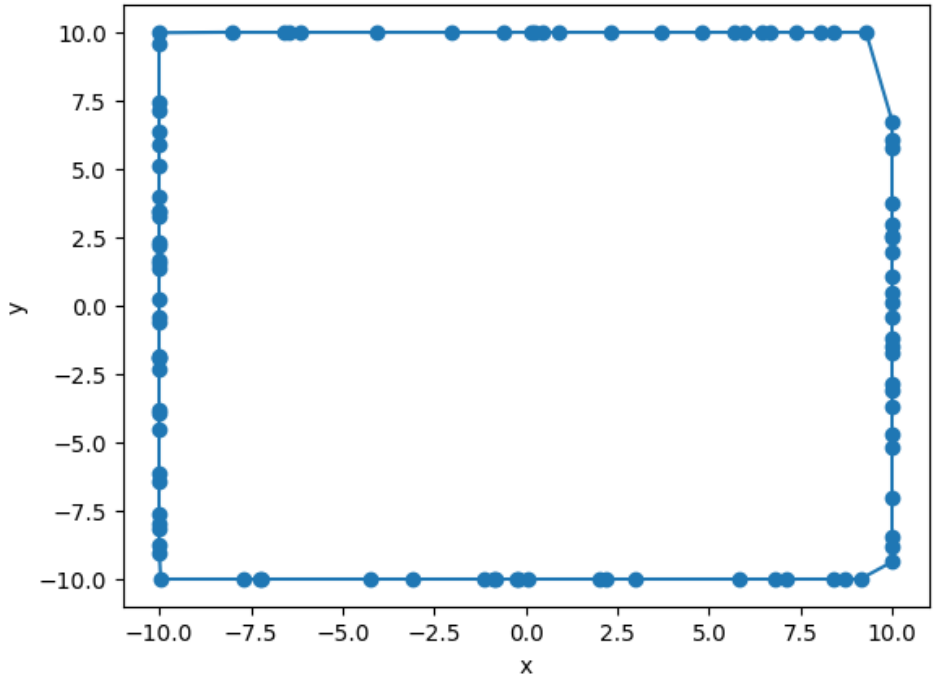
(Wykres 3.1): Otoczka wypukła dla zbioru A

Ilość punktów należących do otoczki: 14



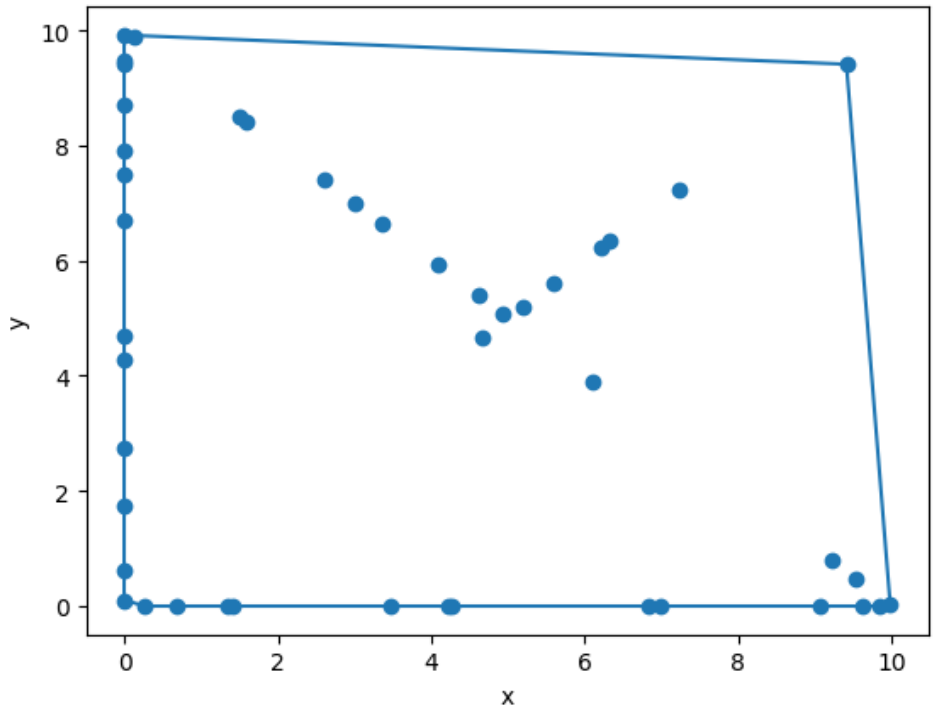
(Wykres 3.2): Otoczka wypukła dla zbioru B

Ilość punktów należących do otoczki: 101



(Wykres 3.3): Otoczka wypukła dla zbioru C

Ilość punktów należących do otoczki: 11



(Wykres 3.4): Otoczka wypukła dla zbioru D

Ilość punktów należących do otoczki: 6

Przedstawiona jest jedna ilustracja dla każdego zbioru bo nie ważne którego algorytmu użyjemy, otoczka będzie wyglądała tak samo.

Porównanie czasu działania algorytmów

Tak jak wspomniałem powyżej, złożoność tych algorytmów się różni, bo:

- Algorytm Grahama – O$(n * logn)$ – złożoność oczekiwana
- Algorytm Jarvisa – O$(n * h)$ – gdzie n to liczba punktów w zbiorze, a h to liczba punktów należących do otoczki, a złożoność pesymistyczna do aż O(n^2)

Wnioskujemy z tego, że algorytm Jarvisa będzie opłacalny tylko w przypadku zbiorów punktów z bardzo małą ilością punktów należących do otoczki. Zatem poniżej zebrałem dane z przykładowych wielkości zbiorów punktów A, B, C i D:

Ilość punktów w zbiorze A	100	1000	2500
Algorytm Grahama	0.00099 [s]	0.00700 [s]	0.01999 [s]
Algorytm Jarvisa	0.01799 [s]	0.23399 [s]	0.89457 [s]

(Tabela 3.1): Czasy algorytmów dla punktów w zbiorze A

Ilość punktów w zbiorze B	100	1000	2500
Algorytm Grahama	0.00099 [s]	0.01400 [s]	0.01900 [s]
Algorytm Jarvisa	0.32700 [s]	15.51527 [s]	88.35496 [s]

(Tabela 3.2): Czasy algorytmów dla punktów w zbiorze B

Ilość punktów w zbiorze C	100	1000	2500
Algorytm Grahama	0.00099 [s]	0.01199 [s]	0.028997 [s]
Algorytm Jarvisa	0.01100 [s]	0.12400 [s]	0.28100 [s]

(Tabela 3.3): Czasy algorytmów dla punktów w zbiorze C

Ilość punktów w zbiorze D (na osiach/na przekątnych	25/20	250/200	750/600
Algorytm Grahama	0.0 [s]	0.00299 [s]	0.01099
Algorytm Jarvisa	0.00399 [s]	0.02999 [s]	0.11300

(Tabela 3.4): Czasy algorytmów dla punktów w zbiorze D

4. Podsumowanie i wnioski

Na podstawie przeprowadzonych testów na zestawach punktów A, B, C i D, oba algorytmy poprawnie i efektywnie wyznaczały otoczkę wypukłą. Pewne trudności pojawiły się przy obliczaniu kąta względem punktu odniesienia i osi X, co było związane z ograniczoną precyzją liczb. Dodatkowo wyzwaniem było usuwanie „duplikatów", czyli współliniowych punktów, które w zasadzie można pominąć, ponieważ interesują nas jedynie te najbardziej wysunięte na prawo i lewo.

Po przeanalizowaniu czasów działania algorytmów dla różnych zbiorów punktów, zauważamy, że algorytm Jarvisa jest w każdym przypadku wolniejszy. W sytuacjach, gdy liczba punktów na otoczce była mała, obydwa algorytmy działały w miarę zbliżonymi prędkościami. Jednak największe różnice wystąpiły w przypadku zbioru B, który składał się z punktów na okręgu, gdzie wszystkie punkty tworzyły otoczkę. W tym scenariuszu algorytm Jarvisa okazał się niezwykle nieefektywny, a czas jego działania przekroczył 100-krotność czasu obliczeń algorytmu Grahama.

Sądzę, że wybrane zbiory punktów zaproponowano właśnie dlatego, że stanowią przypadki skrajne, które mogą sprawiać trudności algorytmom. Dzięki temu można zaobserwować ich działanie w wymagających sytuacjach. Na przykład w zbiorze B wszystkie punkty należą do otoczki, co bywa problematyczne dla algorytmu Jarvisa. Natomiast w zbiorach C i D występuje wiele współliniowych punktów, które program musi traktować ostrożnie, aby nie uwzględniać ich wielokrotnie.