Marceli Jędryka

Algorytmy Geometryczne – Laboratorium 2

Środowisko oraz sprzęt:

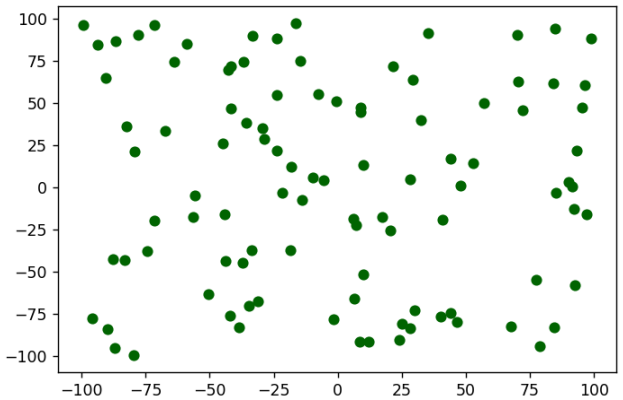
→ Wszystkie ćwiczenia zostały wykonane w Jupyter Notebook przy użyciu języka python oraz bibliotek Numpy i Matplotlip.

→ Obliczenia przeprowadzone na systemie operacyjnym Windows 10 x64 z procesorem Intel Core i5-10210U CPU 2.11 GHz.

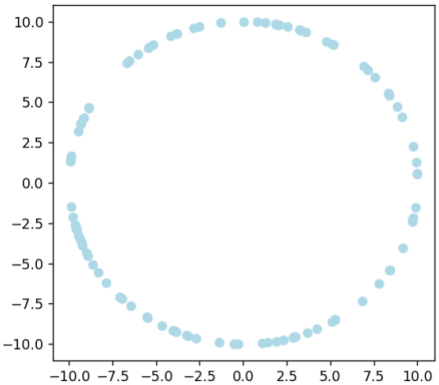
1.Generowanie zbiorów punktów o losowych współrzędnych spełniających następujące warunki:

a) zawierający 100 losowo wygenerowanych punktów o współrzędnych z przedziału [-100, 100],
b) zawierający 100 losowo wygenerowanych punktów leżących na okręgu o środku (0,0) i promieniu R=10,
c) zawierający 100 losowo wygenerowanych punktów leżących na bokach prostokąta o wierzchołkach (-10, 10), (-10,-10), (10,-10), (10,10)
d) zawierający wierzchołki kwadratu (0, 0), (10, 0), (10, 10), (0, 10) oraz punkty wygenerowane losowo w sposób następujący: po 25 punktów na dwóch bokach kwadratu leżących na osiach i po 20 punktów na przekątnych kwadratu.

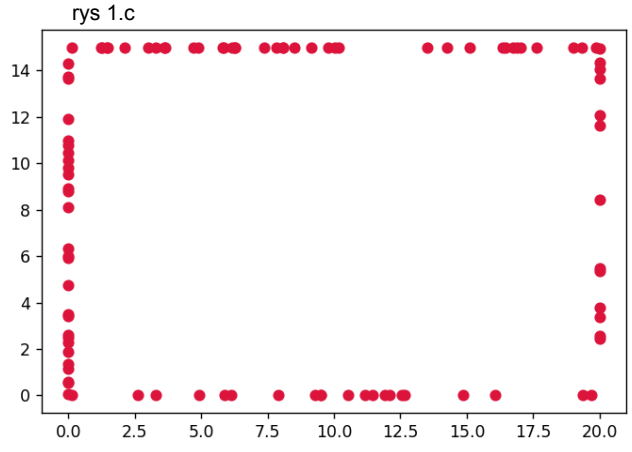
rys 1.a



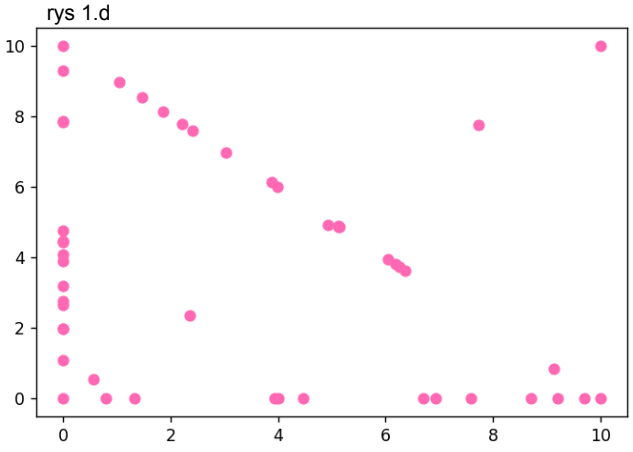
rys 1.b



rys 1.c



rys 1.d



Powyższe zbiory zostały zaimplementowane zgodnie z instrukcjami z zadań 1 oraz 3

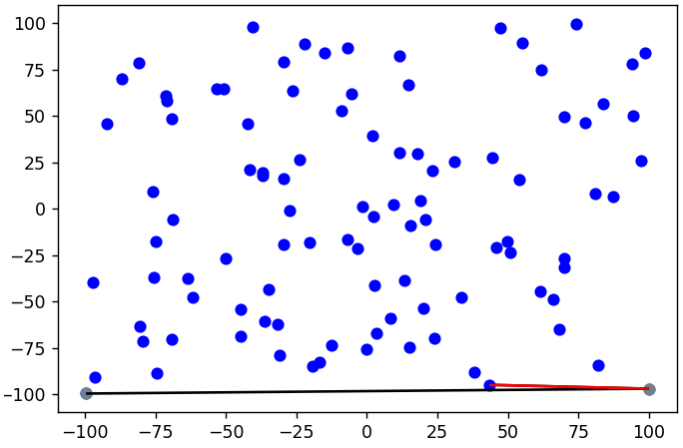
2. Implementacja Algorytmów Grahama i Jarvisa w celu wyznaczenia otoczki wypukłej dla zadanych zbiorów punktów.

Algorytm Grahama :

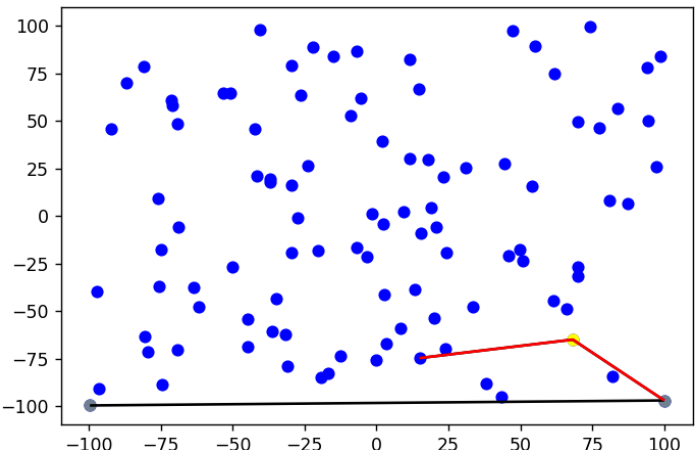
- znajduje punkt o najmniejszej współrzędnej y (punkt startowy) (O(n))
- sortuje pozostałe punkty w zbiorze względem punktu startowego (O(nlogn))
- używając stosu sprawdza potencjalne punkty należące do otoczki i odrzuca te niewłaściwe (O(n))

Złożoność obliczeniowa : O(n) + O(nlogn) + O(n) = O(nlogn)

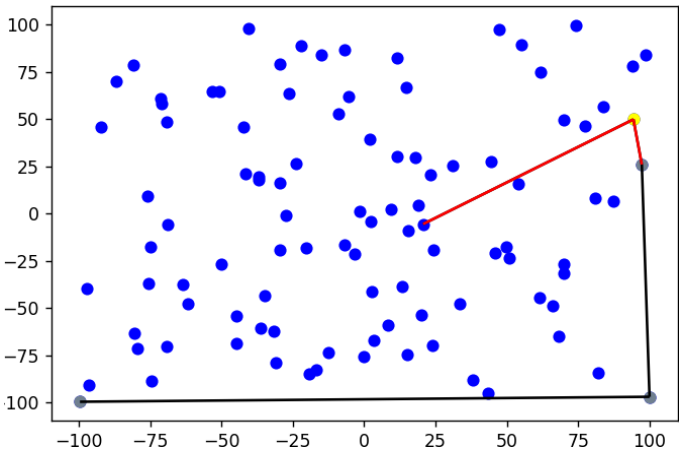
rys 2.1



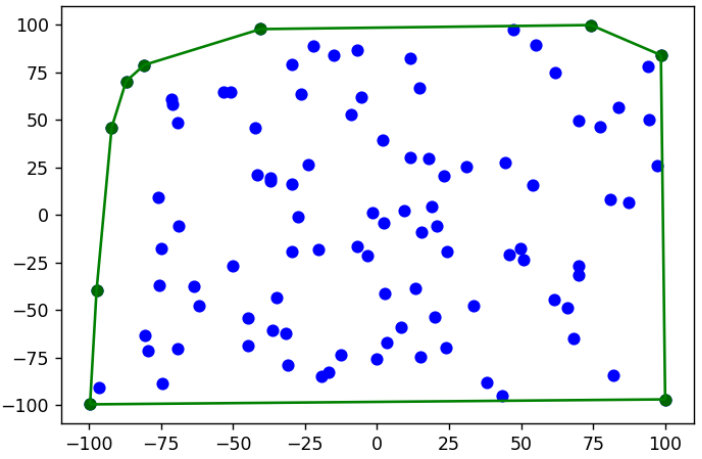
rys 2.2



rys 2.3



rys 2.4



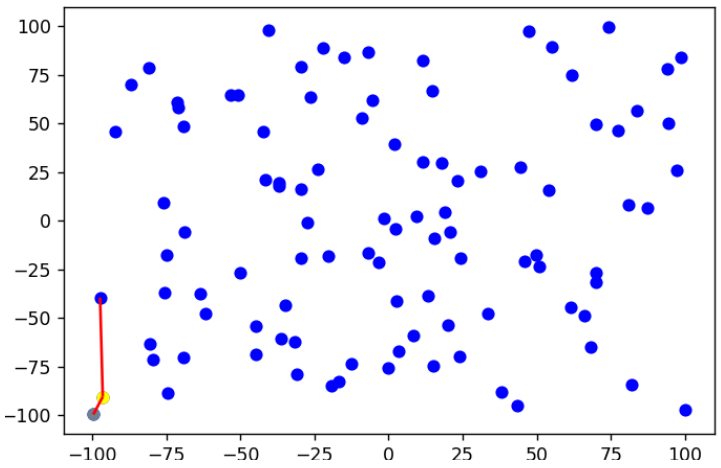
Powyższe grafiki przedstawiają proces powstawania otoczki przy użyciu algorytmu Grahama

Algorytm Jarvisa (Gift wrapping algorithm) :

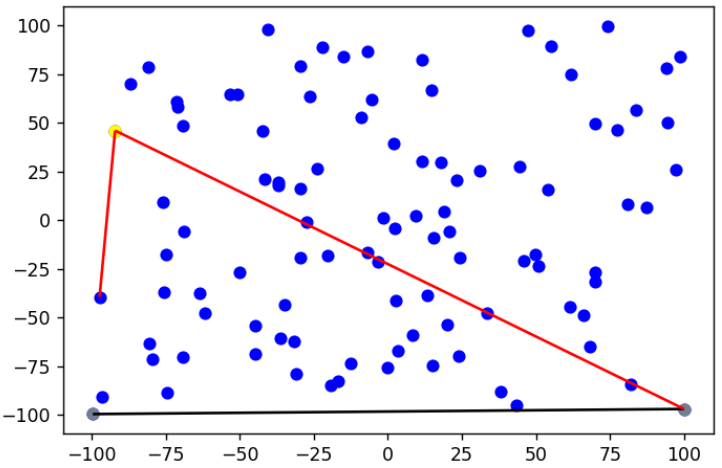
- wyznaczenie punktu o najmniejszej wartości współrzędnej x (punkt startowy) (O(n))
- iterując po wszystkich punktach znajduje punkt tworzący największy kąt z ostatnim punktem dodanym do otoczki. Proces powtarza się dopóki rozpatrywany punkt nie jest punktem startowym. (O(n^2))

Złożoność obliczeniowa : O(n) + O(n^2) = O(n^2)

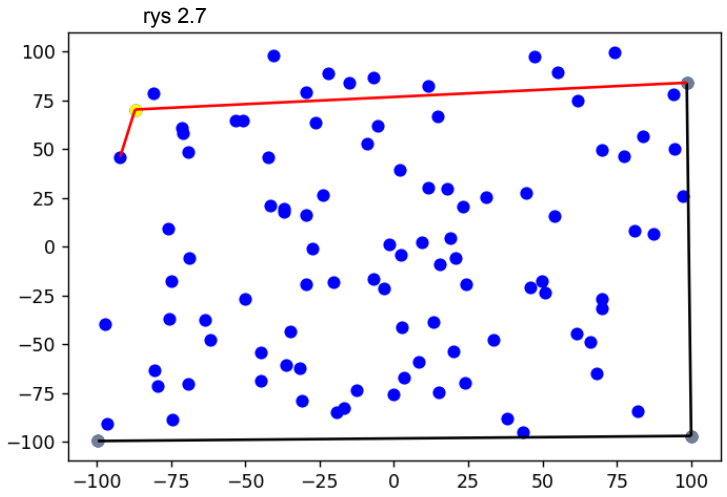
rys 2.5



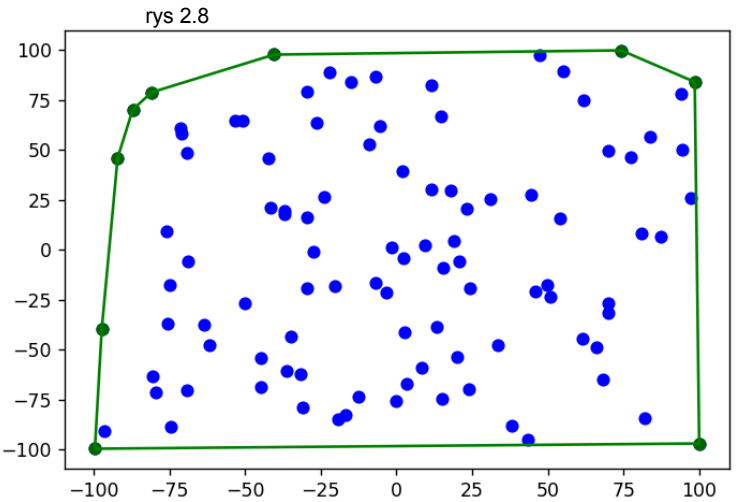
rys 2.6



rys 2.7



rys 2.8



Powyższe grafiki przedstawiają proces powstawania otoczki przy użyciu algorytmu Jarvisa

Liczenie wyznacznika macierzy w celu ustalenia położenia punktów w obu algorytmach było liczone z dokładnością epislon = 10^(-8)

3. Porównanie czasów działania obu algorytmów w zależności od liczby punktów w poszczególnych zbiorach.

Zbiór a)

Ilość Punktów	Algorytm Grahama	Algorytm Jarvisa
100	0,01s	0,01s
1000	0,08s	1,79s
10^4	1,72s	2,10s
10^5	33,86s	7,60s

tab 3.1a (z wizualizacją)

Ilość Punktów	Algorytm Grahama	Algorytm Jarvisa
100	0,00s	0,00s
1000	0,02s	0,02s
10^4	0,14s	0,22s
10^5	1,78s	3,02s

tab 3.1b (bez wizualizacji)

Zbiór b)

Ilość Punktów	Algorytm Grahama	Algorytm Jarvisa
100	0,09s	0,20s
1000	8,55s	24,61s
10^4	-	-
10^5	-	-

tab 3.2a (z wizualizacją)

Ilość Punktów	Algorytm Grahama	Algorytm Jarvisa
100	0,00s	0,01s
1000	0,02s	1,24s
10^4	0,25s	-
10^5	3,41s	-

tab 3.2b (bez wizualizacji)

Zbiór c)

Ilość Punktów	Algorytm Grahama	Algorytm Jarvisa
100	0,01s	0,01s
1000	0,10s	0,04s
10^4	2,37s	0,45s
10^5	-	-

tab 3.3a (z wizualizacją)

Ilość Punktów	Algorytm Grahama	Algorytm Jarvisa
100	0,00s	0,00s
1000	0,05	0,00s
10^4	1,66s	0,1s
10^5	-	-

tab 3.3b (bez wizualizacji)

Zbiór d)

Ilość Punktów	Algorytm Grahama	Algorytm Jarvisa
Przekątne : 20 Boki : 25	0,00s	0,00s
Przekątne : 200 Boki : 250	0,04s	0,01s
Przekątne : 2000 Boki : 2500	1,94s	0,05s
Przekątne : 20000 Boki : 25000	-	0,79s

tab 3.4a (z wizualizacją)

Ilość Punktów	Algorytm Grahama	Algorytm Jarvisa
Przekątne : 20 Boki : 25	0,00s	0,00s
Przekątne : 200 Boki : 250	0,05	0,00s
Przekątne : 2000 Boki : 2500	1,66s	0,02s
Przekątne : 20000 Boki : 25000	114,31s	0,20s

tab 3.4b (bez wizualizacji)

4. Wnioski.

Analizując wyniki czasowe obu algorytmów dla poszczególnych przypadków możemy stwierdzić, że w sytuacji, gdy zbiory zawierają niewielką liczbę punktów współliniowych lub ich całkowity brak (zbiory a i b), to algorytm Grahama wypada znacznie lepiej. Sytuacja odwraca się, gdy punkty współliniowe występują w większej liczbie. W takim wypadku algorytm Jarvisa jest o wiele szybszy. Możemy również zaobserwować, że czasy działania algorytmów z wizualizacją są zdecydowanie wolniejsze niż bez niej. Przy dużym zagęszczeniu punktów algorytmy zwracały błędny wynik. Rozwiązaniem problemu było wyznaczenie dokładności obliczeniowej przy obliczaniu wyznacznika w funkcji find_side (przyjęty epislon wynosił 10^(-8)).

Zaproponowane zbiory sprawdzały zachowanie algorytmów w przypadku różnego rozmieszczenia punktów. W przypadku algorytmu Jarvisa największy problem sprawiał zbiór b, co widać po czasie wykonania się programu, natomiast algorytm Grahama najgorzej radził sobie ze zbiorami c i d. Wynika to ze sposobu implementacji obu programów.