Sprawozdanie z ćwiczenia drugiego

Algorytmy Geometryczne

Iwo Szczepaniak, Windows 11 – 4,2GHz

Jupyter Notebook, VS Code Python 3.9.12

1) Wstęp

Celem ćwiczenia jest wyznaczenie otoczki wypukłej(najmniejszego zbioru wypukłego zawierającego podzbiór wszystkich punktów na płaszczyźnie) dla konkretnego zbioru punktów. Do tego celu, pierwszym zadaniem jest wygenerowanie zadanych punktów i ich wizualizacja, a następnie umożliwienie zmiany ilości punktów oraz ich zakresu w zależności od życzenia użytkownika. Operując na tak wyznaczonych punktach, naszym zadaniem jest wyznaczyć otoczkę wypukłą. Do jej generowania służą dwa algorytmy – Grahama oraz Jarvisa.

2) Opis algorytmów

- Algorytm Grahama:

Złożoność obliczeniowa :
n – szukanie minimum
nlogn – sortowanie
n – przejście po punktach sprawdzając ich położenia względem siebie
Prognozowana złożoność: słabe nlogn

1. Wybierz *punkt startowy* o najmniejszej współrzędnej x; jeśli kilka punktów ma tę samą współrzędną y, wybierz spośród nich ten o największej współrzędnej y.
2. Posortuj punkty względem:
    - Orientacji względem *punktu startowego* -> funkcja orient(a,b,c) z poprzedniego laboratorium
    - Gdy „kąt” równy, względem odległości punktu od punktu startowego
3. Przeglądaj listę posortowanych punktów poczynając od *punktu startowego*:
    - Od bieżącej pozycji weź trzy kolejne punkty
    - Jeśli punkt trzeci punkt leży po prawej stronie od prostej na której leżą pierwsze dwa punkty, to może należeć do otoczki
    - Jeśli punkt B leży na lewo od prostej na której leżą pierwsze dwa punkty, to znaczy, że nie należy do otoczki. Wtedy usuń ten punkt ze stosu i cofnij się o jedną pozycję.
    - Powtarzaj aż dotrzesz z powrotem do punktu startowego

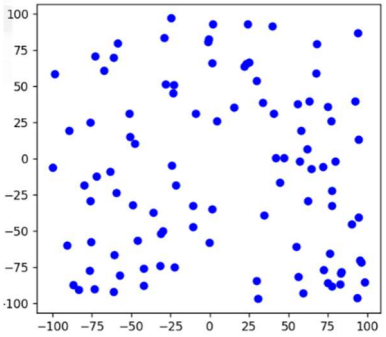
-Algorytm Jarvisa:

Złożoność obliczeniowa:
nh – n szukań dla h punktów otoczki(pesymistycznie n^2)
Prognozowana złożoność: wyniki pomiędzy n a n^2, bliskie n^2

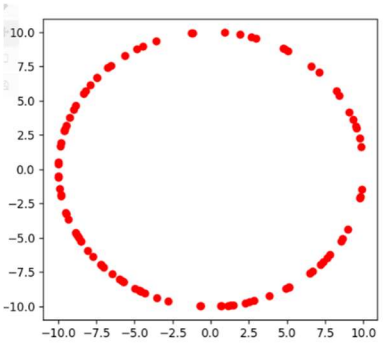
1. Wybierz *punkt startowy* tak samo jak w algorytmie Grahama.
2. Przejrzyj wszystkie wierzchołki i dodaj na stos ten który jest najbardziej na lewo od ostatniego wierzchołka na stosie.
3. Powtarzaj aż dotrzesz z powrotem do punktu startowego.

3) Generowanie i wizualizacja punktów

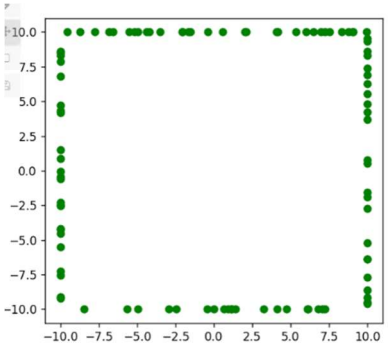
Zadaniem było przygotowanie:
a) zawierający 100 losowo wygenerowanych punktów o współrzędnych z przedziału [-100, 100],
b) zawierający 100 losowo wygenerowanych punktów leżących na okręgu o środku (0,0) i promieniu R=10,
c) zawierający 100 losowo wygenerowanych punktów leżących na bokach prostokąta o 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.



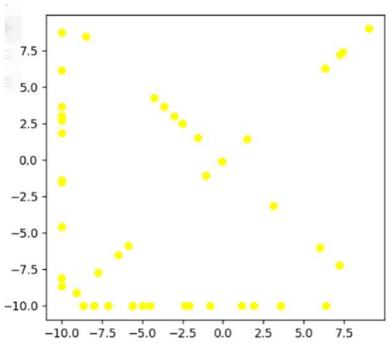
Wizualizacja zadania a)



Wizualizacja zadania b)



Wizualizacja zadania c)



Wizualizacja zadania d)

Implementacja pozwala też na zmianę ilości generowanych punktów oraz wielkości figur na których są generowane punkty – okręgu, kwadratu i wycinka kwadratu wraz z przekątną– lub zakresu generowanych punktów

4) Wizualizacja działania algorytmów

Pełna wizualizacja w Jupyter Notebook.

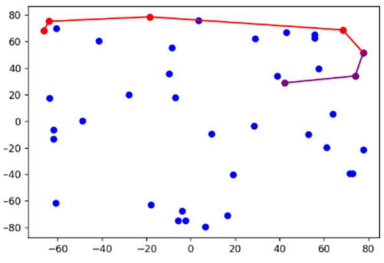
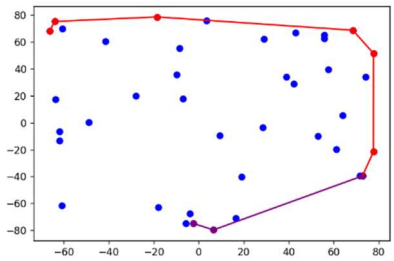
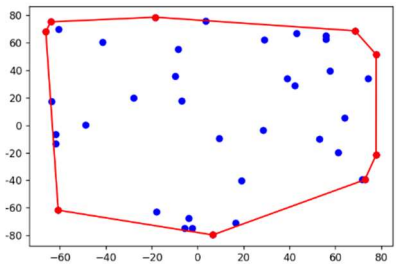
Do wizualizacji działania programu wykorzystano, zgodnie z poleceniem, trzy kolory. Czerwony odpowiada za punkty na otoczce, fioletowy za punkty aktualnie przetwarzane, a niebieski za pozostałe.
Aby zwizualizować konkretny podpunkt należy wprowadzić odpowiadającą mu literę.

Poniżej umieszczono jedynie niektóre interesujące wizualizacje w trakcie działania algorytmów.

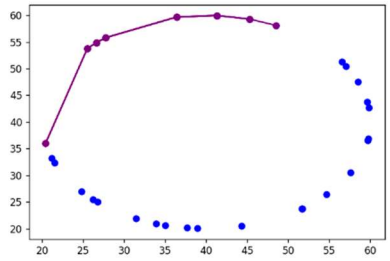
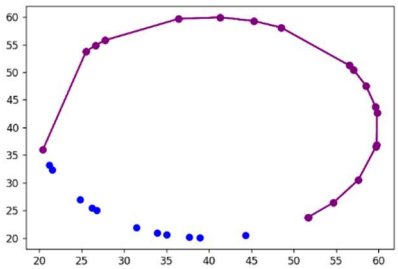
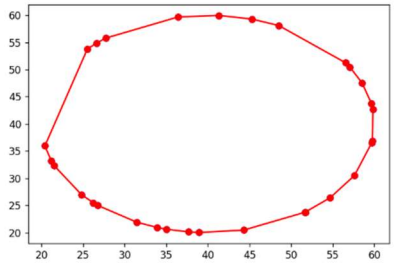
Słowniczek wizualizacji:

n – ilość punktów
r – promień okręgu
S – środek okręgu
downLeft – lewy dolny róg prostokąta
upRight – prawy górny róg prostokąta
a – długość boku w kwadracie w podpunkcie d
m – ilość punktów na przekątnej w podpunkcie d

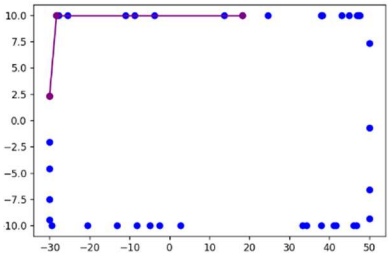
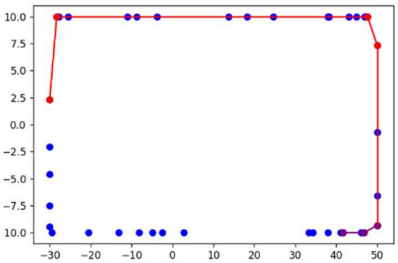
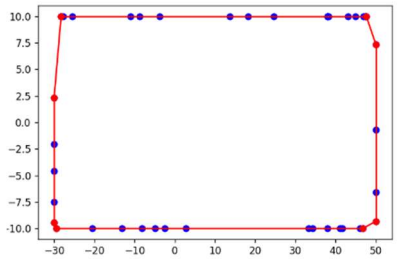
- Algorytm Grahama

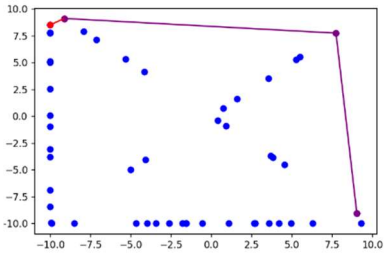
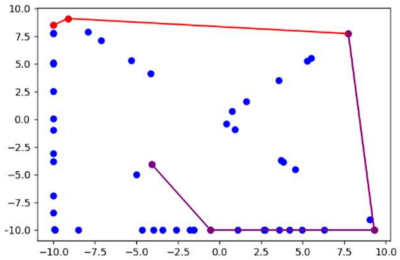
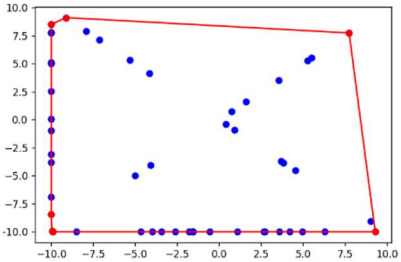
*Podpunkt a) z parametrami: n = 40, zakres [-80,80]

*Podpunkt b) z parametrami: n = 30, r = 20, S = (40,40)

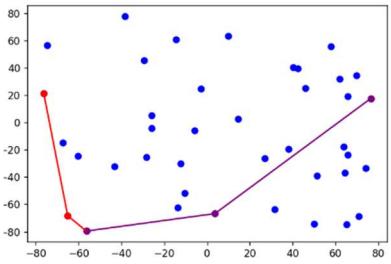
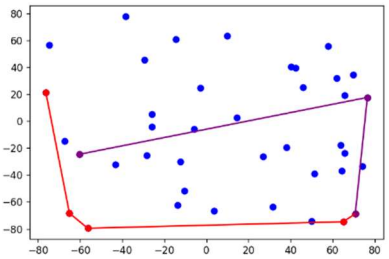
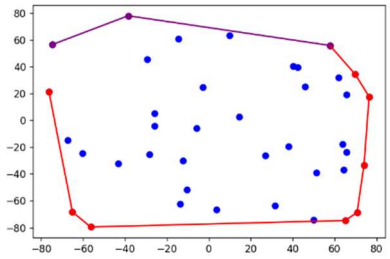




*Podpunkt c) z parametrami: n = 40, downLeft = (-30,-10), upRight = (50,10)

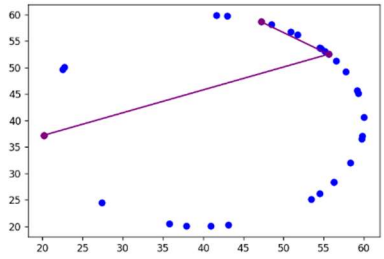
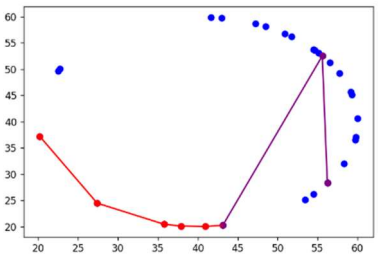
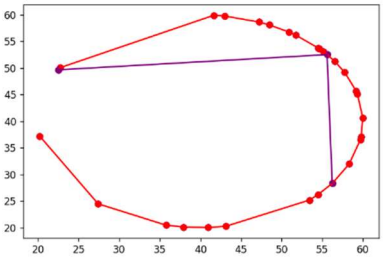




*Podpunkt d) z parametrami: n = 30, m =20, downLeft = (-10,-10), a = 20

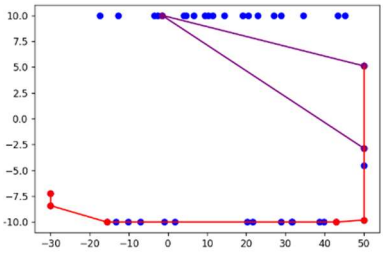
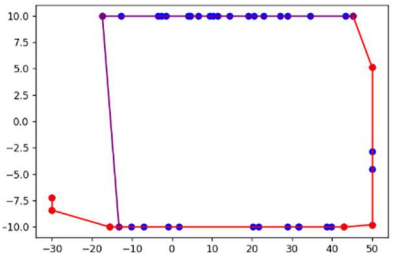
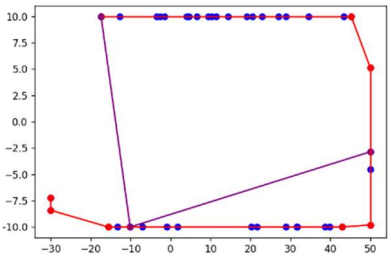
- Algorytm Jarvisa

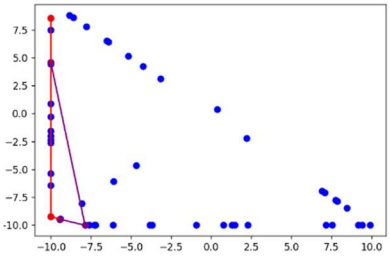
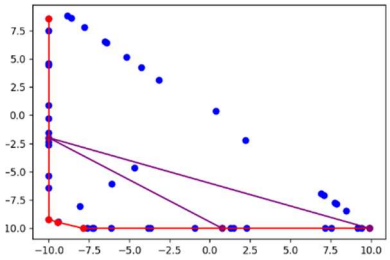
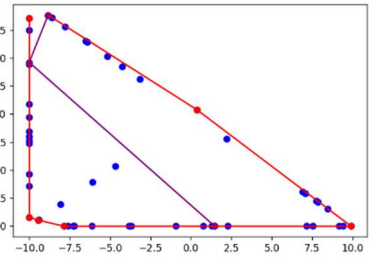
*Podpunkt a) z parametrami: n = 40, zakres [-80,80]

*Podpunkt b) z parametrami: n = 30, r = 20, S = (40,40)

*Podpunkt c) z parametrami: n = 40, downLeft = (-30,-10), upRight = (50,10)

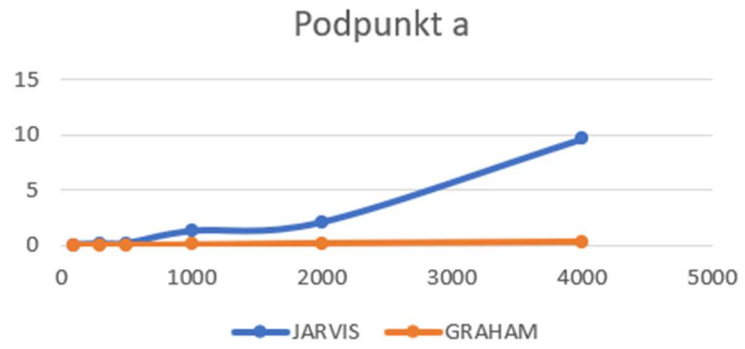




*Podpunkt d) z parametrami: n = 30, m =20, downLeft = (-10,-10), a = 20

5) Czasochłonność ze względu na ilość punktów:

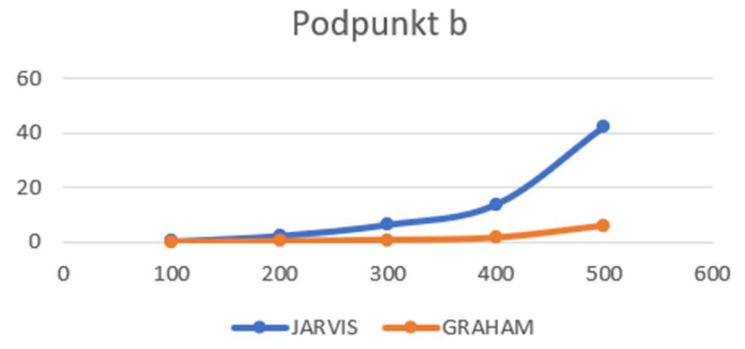
Podpunkt a)

n	JARVIS [s]	GRAHAM [s]
100	0.02352047	0.005017996
300	0.098564863	0.008991718
500	0.128087521	0.018006086
1000	1.25450182	0.043254375
2000	2.045077801	0.177389622
4000	9.666536808	0.33026433



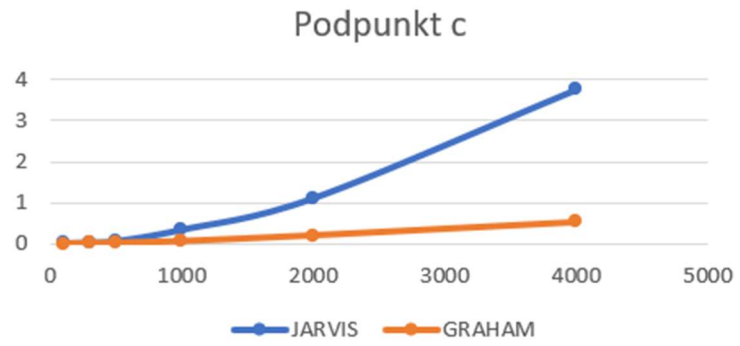
Podpunkt b)

n	JARVIS [s]	GRAHAM [s]
100	0.192240477	0.0504632
200	2.231305361	0.24768424
300	6.34507966	0.719551086
400	13.7860055	1.574402332
500	42.40614486	6.080311537



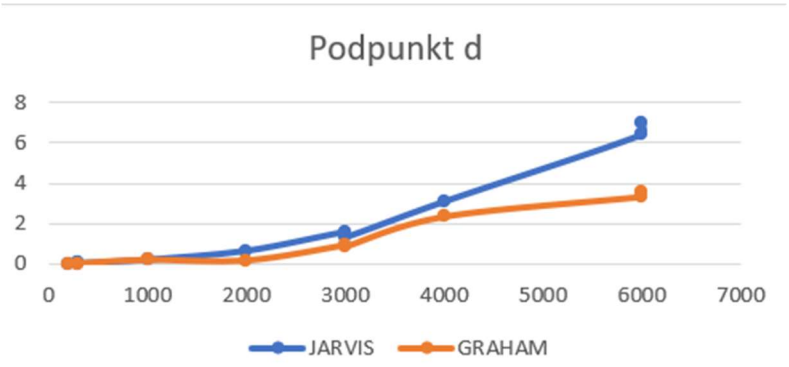
Podpunkt c)

n	JARVIS [s]	GRAHAM [s]
100	0.015676737	0.005457401
300	0.031009436	0.012002468
500	0.064010859	0.02415061
1000	0.344103336	0.06397748
2000	1.113039494	0.205542088
4000	3.769374609	0.531229258



Podpunkt d)

n+m	JARVIS [s]	GRAHAM [s]
100+100	0.013278008	0.009525299
100+200	0.025690079	0.010002136
200+100	0.024696589	0.014503241
500+500	0.203298807	0.190287113
1000+1000	0.639288664	0.156689405
1000+2000	1.57102704	0.922906399
2000+1000	1.308087111	0.846460342
2000+2000	3.076281786	2.347368002
4000+2000	6.400309324	3.344535828
2000+4000	7.627174616	3.548235416



Wnioski

Na podstawie otrzymanych wykresów można stwierdzić, że czas wykonywania był zgodny z przewidywaniami – Graham około nlogn, Jarvis w pierwszych 3 podpunktach pesymistyczne n^2.

W podpunktach a i c widać znaczącą różnicę między dwoma podejściami, ale w podpunkcie b różnica jest największa. Różnica w czasie wykonywania algorytmów wynosząca 35 s pojawia się już przy generowaniu 500 punktów. Jest to prawdopodobnie spowodowane tym, że w tym podpunkcie wszystkie punkty wygenerowane należą do otoczki(zbiorem punktów jest okrąg) więc współczynnik h jest równy n. W podpunkcie d różnica między algorytmami się znacznie zaciera i to z kolei jest to spowodowane stosunkowo małą ilością punktów wchodzących w skład otoczki na zadanej figurze, co zmniejsza współczynnik h. Mimo to algorytm Grahama nadal był zauważalnie wydajniejszy w tym podpunkcie.

Dla wszystkich zbiorów punktów algorytm Grahama okazał się szybszy od algorytmu Jarvisa. To pokazuje, że nlogn rośnie wolniej niż h i dlatego pierwszy algorytm okazał się skuteczniejszy. Ta przewaga znacząco zależy jednak od rodzaju figury lub zbioru na których generowane są punkty.