Kacper Garus, 11.11.2023, ćwiczenie nr.2 Implementacja podstawowych predykatów geometrycznych

Dane techniczne:

Język - python, translator - Visual Studio Code, procesor -Intel(R) Core (TM) i7-8565U CPU @ 1.80GHz, system operacyjny - Windows 10 Home 64bit

Realizacja ćwiczenia:

Na początku stworzyłem funkcje do generowania losowych (używałem funkcji z biblioteki random) punktów na płaszczyźnie za pomocą których stworzyłem zbiory: points_a-100 losowo wygenerowanych punktów o współrzędnych z przedziału [-100, 100], points_b -100 losowo wygenerowanych punktów leżących na okręgu o środku (0,0) i promieniu R=10, points_c - 100 losowo wygenerowanych punktów leżących na bokach prostokąta o wierzchołkach (-10, 10), (-10,-10), (10,-10), (10,10), points_d-zbiór zawierający wierzchołki kwadratu (0, 0), (10, 0), (10, 10), (0, 10) oraz punkty wygenerowane losowo w sposób następujący: po 50 punktów na dwóch bokach kwadratu leżących na osiach i po 50 punktów na przekątnych kwadratu. Następnie za pomocą algorytmu Grahama oraz algorytmu Jarvisa tworzyłem dla tych zbiorów oraz, dla tych zbiorów z innymi parametrami, otoczkę wypukłą. Stworzyłem również wizualizacje działania algorytmów, lecz niestety nie udało mi się zrobić dokładnej wizualizacji algorytmu Jarvisa, w której jest ukazany każdy badany punkt, ponieważ pojawiał się error przy próbie kompilacji funkcji vis.show_gif().

Wyniki i analiza:

Zbiory podstawowe:

```
Testowanie czasu działania dla points_a:
Algorytm Grahama - Czas wykonania: 0.0003807544708251953 sekundy
Algorytm Jarvisa - Czas wykonania: 0.0005114078521728516 sekundy
Dla zbioru points_a szybszy był algorytm Grahama
----------------------------------------
Testowanie czasu działania dla points_b:
Algorytm Grahama - Czas wykonania: 0.0003345012664794922 sekundy
Algorytm Jarvisa - Czas wykonania: 0.006246805191040039 sekundy
Dla zbioru points_b szybszy był algorytm Grahama
----------------------------------------
Testowanie czasu działania dla points_c:
Algorytm Grahama - Czas wykonania: 0.0020275115966796875 sekundy
Algorytm Jarvisa - Czas wykonania: 0.0015077590942382812 sekundy
Dla zbioru points_c szybszy był algorytm Jarvisa
----------------------------------------
Testowanie czasu działania dla points_d:
Algorytm Grahama - Czas wykonania: 0.0016171932220458984 sekundy
Algorytm Jarvisa - Czas wykonania: 0.001322031021118164 sekundy
Dla zbioru points_d szybszy był algorytm Jarvisa
```

Zbiory zmodyfikowane1: points_a2-1000 punktów z zakresu [-100,100]^2, points_b2-100 punktów na okręgu o promieniu R=10 i środku w (10,10), points_c2-1000 punktów na prostokącie o takich samych wierzchołkach jak points_c, points_d2-1000 punktów na przekątnych i 50 na osiach.

```
Testowanie czasu działania dla points_a2:
Algorytm Grahama - Czas wykonania: 0.010236263275146484 sekundy
Algorytm Jarvisa - Czas wykonania: 0.02111959457397461 sekundy
Dla zbioru points_a2 szybszy był algorytm Grahama
----------------------------------------
Testowanie czasu działania dla points_b2:
Algorytm Grahama - Czas wykonania: 0.011090517044067383 sekundy
Algorytm Jarvisa - Czas wykonania: 0.48882055282592773 sekundy
Dla zbioru points_b2 szybszy był algorytm Grahama
----------------------------------------
Testowanie czasu działania dla points_c2:
Algorytm Grahama - Czas wykonania: 0.002742290496826172 sekundy
Algorytm Jarvisa - Czas wykonania: 0.005441427230834961 sekundy
Dla zbioru points_c2 szybszy był algorytm Grahama
----------------------------------------
Testowanie czasu działania dla points_d2:
Algorytm Grahama - Czas wykonania: 0.0035223960876464844 sekundy
Algorytm Jarvisa - Czas wykonania: 0.0022306442260742188 sekundy
Dla zbioru points_d2 szybszy był algorytm Jarvisa
```

Zbiory zmodyfikowane 2: zestawy jak wcześniej, lecz w każdym zbiorze punktów jest 10 razy więcej

```
Testowanie czasu działania dla points_a3:
Algorytm Grahama - Czas wykonania: 0.10866093635559082 sekundy
Algorytm Jarvisa - Czas wykonania: 0.1257791519165039 sekundy
Dla zbioru points_a3 szybszy był algorytm Grahama
----------------------------------------
Testowanie czasu działania dla points_b3:
Algorytm Grahama - Czas wykonania: 0.03548455238342285 sekundy
Algorytm Jarvisa - Czas wykonania: 48.06997990608215 sekundy
Dla zbioru points_b3 szybszy był algorytm Grahama
----------------------------------------
Testowanie czasu działania dla points_c3:
Algorytm Grahama - Czas wykonania: 0.03220820426940918 sekundy
Algorytm Jarvisa - Czas wykonania: 0.042890071868896484 sekundy
Dla zbioru points_c3 szybszy był algorytm Grahama
----------------------------------------
Testowanie czasu działania dla points_d3:
Algorytm Grahama - Czas wykonania: 0.04137110710144043 sekundy
Algorytm Jarvisa - Czas wykonania: 0.026918888092041016 sekundy
Dla zbioru points_d3 szybszy był algorytm Jarvisa
```

Możemy zauważyć że dla danych na okręgu, w których jest dużo punktów należących do otoczki wypukłej szybszy będzie algorytm Grahama, natomiast dla punktów z mniejszą ilością punktów należących do otoczki szybszy powinien być algorytm Jarvisa.

Wnioski:

Moje algorytmy Grahama i Jarvisa działały poprawnie, przeszły wszystkie testy od kn bit. Według mnie zaproponowano takie a nie inne zbiory punktów aby lepiej zrozumieć działanie tych algorytmów i ich słabsze i mocniejsze strony przy danych ułożeniach danych np. algorytm Jarvisa nie nadaje się do większych zbiorów danych ułożonych na okręgu.

Wizualizacje:

Wszystkie potrzebne wizualizacje znajdują się w pliku garus_kod_1.ipynb