Algorytmy geometryczne

Anna Gut, 28.10.2020r., grupa nr 7 (czwartek 16:15 A)

Sprawozdanie z ćwiczenia nr 2 – wyznaczanie otoczki wypukłej

1.Wprowadzenie

1.1.Cel ćwiczenia

Celem ćwiczenia było:

- implementacja algorytmów (Grahama oraz Jarvisa) służących do znajdowania otoczki wypukłej dla zadanego zbioru punktów
- wygenerowanie na płaszczyźnie zbiorów punktów spełniających konkretne założenia początkowe (ilość punktów, ich układ na płaszczyźnie) oraz poddanie ich działaniu powyższych algorytmów
- wizualizacja przebiegu algorytmów dla zadanych zbiorów przy pomocy narzędzia graficznego oraz z wykorzystaniem bibliotek: Numpy oraz Matplotlib
- przeprowadzenie analizy czasu działania algorytmów w zależności od rozkładu punktów oraz ich ilości oraz porównanie otrzymanych wyników

1.2.Wstęp teoretyczny

Otoczką wypukłą CH(P) danego zbioru punktów P nazywamy najmniejszy wypukły wielokąt, taki, że każdy punkt ze zbioru P leży we wnętrzu lub na brzegu tego wielokąta. W tym ćwiczeniu wykorzystywane są dwa znane algorytmy służące do znajdowania otoczki wypukłej dla zbioru punktów:

1) Algorytm Grahama

   -znajdujemy w zbiorze S punkt p_0 najmniejszej współrzędnej y, a jeżeli jest więcej niż jeden taki punkt, to wybieramy spośród nich ten, który posiada najmniejszą współrzędną x

   -pozostałe punkty (p_1,..,p_n) sortujemy ze względu na kąt, jaki tworzy wektor (p_0,p_i) z dodatnim kierunkiem osi x. Jeżeli więcej niż jeden punkt tworzy taki sam kąt, usuwamy wszystkie punkty z wyjątkiem najbardziej oddalonego od p_0

   -tworzymy pusty stos, a następnie wkładamy do niego punkty p_0,p_1,p_2

   -przechodzimy po kolejnych punktach (p_3,..,p_m) i sprawdzamy, po której stronie prostej przechodzącej przed dwa punkty, znajdujące się na wierzchu stosu, leży ten punkt - jeżeli leży po lewej stronie, to umieszczamy go na stosie, a jeżeli po prawej, to usuwamy ze stosu ostatnio dodany do niego punkt

2) Algorytm Jarvisa

-znajdujemy w zbiorze S punkt p_0 o najmniejszej współrzędnej y, a jeżeli jest więcej niż jeden taki punkt, to wybieramy spośród nich ten, który posiada najmniejszą współrzędną x

-znajdujemy punkt, który tworzy najmniejszy kąt (liczony przeciwnie do ruchu wskazówek zegara) z ostatnią krawędzi otoczki, a jeżeli jest więcej takich punktów, to wybieramy ten, który jest najbardziej oddalony od punktu ostatnio dodanego do otoczki. Krok ten powtarzamy aż do momentu, w którym wybranym punktem będzie p_0

2.Uwagi techniczne dotyczące sprzętu

Ćwiczenie zostało przeprowadzone z wykorzystaniem sprzętu o następujących parametrach:

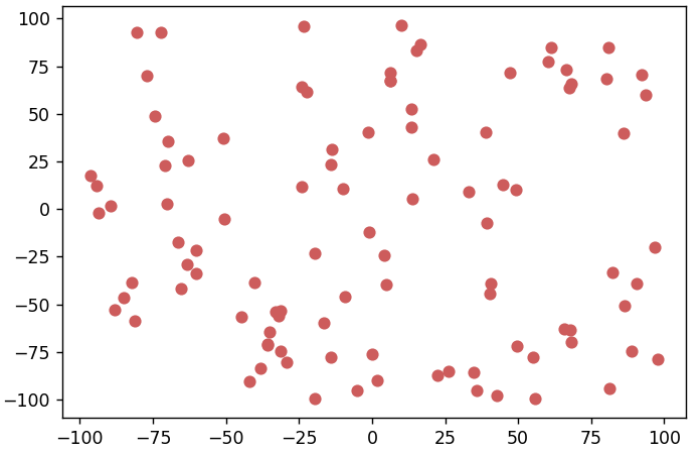
- oprogramowanie Windows 8.1 Pro
- procesor Intel i5
- pamięć RAM 8 GB
- 64-bitowy system operacyjny
- zainstalowane narzędzie graficzne oparte o ProjektJupyter

3.Przebieg ćwiczenia

Przy wykorzystaniu dostępnych w języku Python bibliotek: random oraz math wygenerowane zostały 4 zbiory punktów:

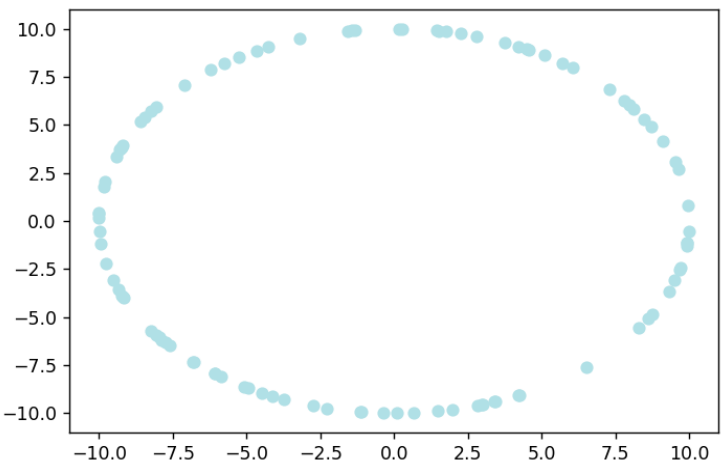
Rysunek nr 1

Zbiór A – 100 losowo wygenerowanych punktów z przedziału [-100,100]



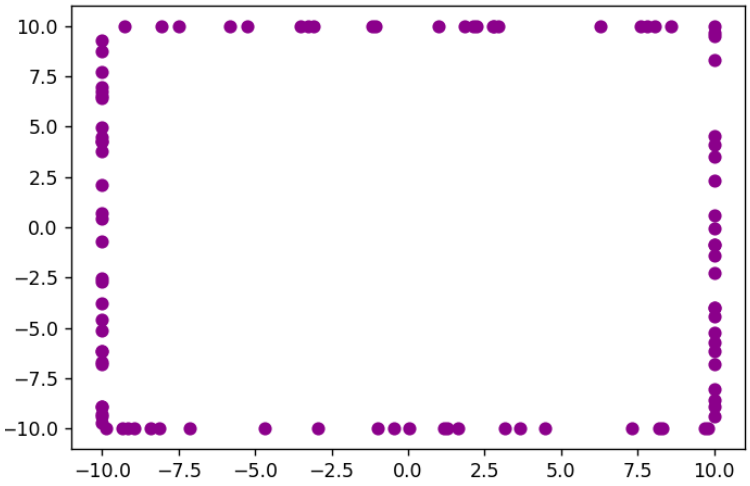
Rysunek nr 2

Zbiór B – 100 losowo wygenerowanych punktów leżących na okręgu o środku w punkcie (0,0) i promieniu R=10



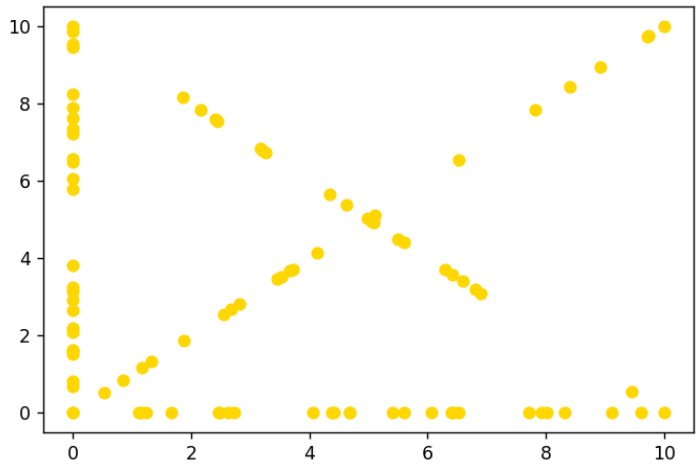
Rysunek nr 3

Zbiór C - 100 losowo wygenerowanych punktów leżących na bokach prostokąta o wierzchołkach (-10,-10),(-10,10), (10,-10),(10,10)



Rysunek nr 4

Zbiór D - zawierający wierzchołki kwadratu (0,0), (10,0), (10,0),(10,10),(0,10) oraz punkty wygenerowane losowo w sposób następujący: po 25 punktów na dwóch bokach leżących na osiach i po 10 punktów na przekątnych.



Następnie zaimplementowane zostały algorytmy służące do wyznaczania otoczki wypukłej (algorytm Grahama i algorytm Jarvisa). Każdy z wygenerowanych wcześniej zbiorów został poddany działaniu powyższych algorytmów, a wynik w postaci zbioru punktów tworzących otoczkę wypukłą wpisywany został do pliku o nazwie *otoczka.json.* Przy pomocy narzędzia graficznego przygotowana została również wizualizacja z przebiegu działania algorytmu. Dla każdego zbioru przedstawiony został czas działania algorytmów oraz dokonana została analiza porównawcza czasu działania w zależności od ilości punktów w zbiorze (dla zbiorów o rozkładzie punktów takim, jak w zbiorach a), c) i d) badany był zakres 1000-30000 punktów w zbiorze, a dla zbioru o rozkładzie punktów jak w zbiorze b) – zakres 500-7000 punktów).

4.Klasyfikacja użytych metod

Podczas wykonywania ćwiczenia zostały zaimplementowane następujące metody:

1) Generowanie zbiorów punktów:

   Wykorzystane biblioteki: *math ,random*

   - **generate_random** – generowanie zbioru punktów z danego przedziału
   - **generate_on_circle** – generowanie zbioru punktów leżących na okręgu o danym środku oraz promieniu
   - **generate_on_rectangle** – generowanie zbioru punktów leżących na prostokącie o danych współrzędnych
   - **generate_on_square** – generowanie zbioru, do których należą dane współrzędne wierzchołków kwadratu oraz punkty należące do jego przekątnych oraz boków leżących na osiach

2) Metody pozwalające na opisanie wzajemnego położenia punktów

   - **scalar_prod** – wyznaczanie iloczynu skalarnego dwóch wektorów
   - **length** – wyznaczanie długości wektora
   - **getvec** – tworzenie wektora na podstawie danych punktów
   - **det2x2** – obliczanie wyznacznika macierzy kwadratowej 2x2 oraz klasyfikowanie wzajemnego położenia punktów ze względu na jego wartość
   - **findlowest** – wyznaczanie spośród dwóch punktów tego, który posiada mniejszą współrzędną y. W przypadku, gdy współrzędne te dla obydwu punktów są równe, wyznaczany jest punkt o mniejszej współrzędnej x
   - **minimumangle**– wyznaczanie punktu, który tworzy mniejszy kąt z ostatnio dodaną krawędzią otoczki. W przypadku punktów o tym samym kącie, wyznaczany jest ten, który znajduje się dalej od punktu ostatnio dodanego do otoczki
   - **compare_angles** – wyznaczanie punktu, do którego wektor poprowadzony z punktu o najmniejszej współrzędnej y, tworzy z dodatnim kierunkiem osi OX najmniejszy kąt. Jeżeli kąty dla obydwu punktów są równe – wyznaczamy ten z nich, który jest bardziej odległy od najniżej położonego punktu

3) Metody znajdujące otoczkę wypukłą

   Wykorzystane biblioteki: *time, functools*

   - **Graham** – algorytm Grahama do znajdywania otoczki wypukłej
   - **Jarvis** – algorytm Jarvisa do znajdywania otoczki wypukłej

We wszystkich wykorzystanych metodach została przyjęta tolerancja dla zera równa $e=10^{-12}$.

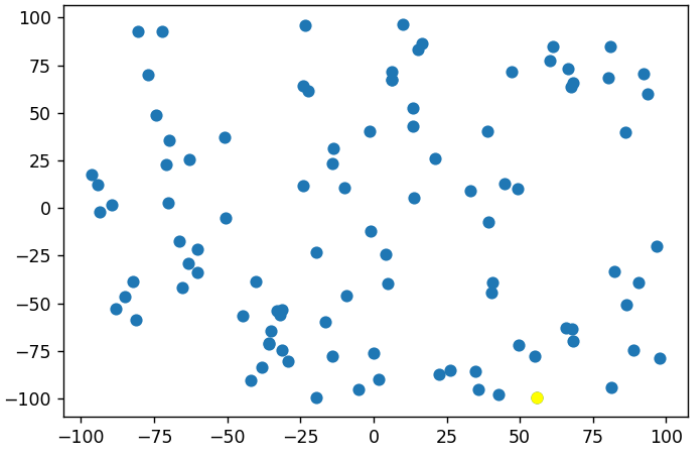
5.Wizualizacja graficzna przebiegu algorytmów

5.1.Algorytm Grahama

Poniżej przedstawiony jest skrócony przebieg działania algorytmu Grahama dla zbioru a)

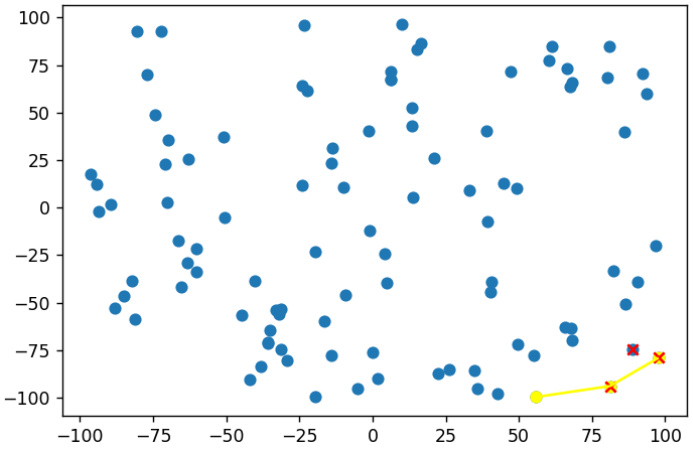
Rysunek nr 5.

Wyróżniony na żółto punkt o najmniejszej współrzędnej y



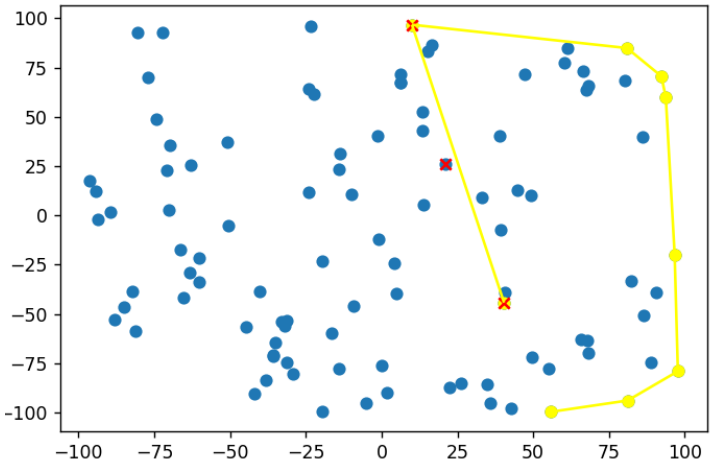
Rysunek nr 6.

Wyróżniona na żółto obecna otoczka wypukła, a na czerwono obecnie rozpatrywane punkty

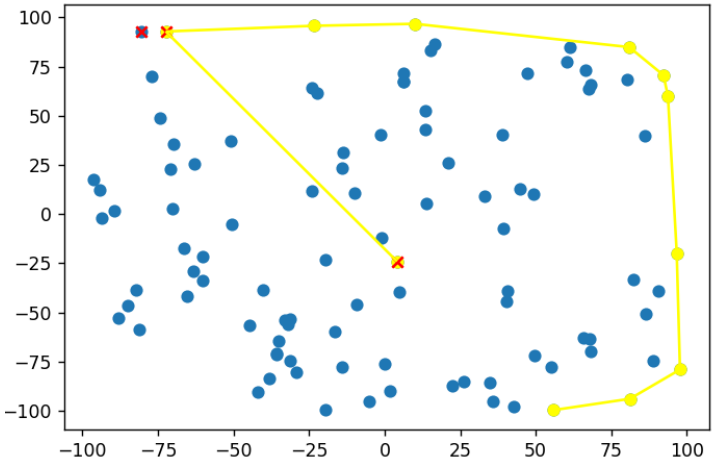


Rysunki 7-10 prezentują dalszy skrócony przebieg działania algorytmu przy zachowaniu powyższych oznaczeń

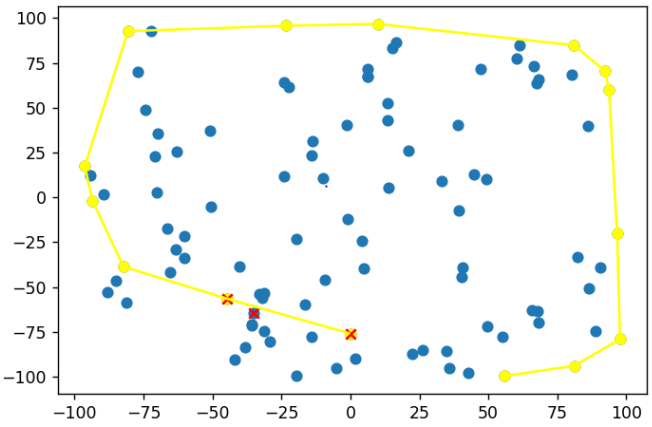
Rysunek nr 7.



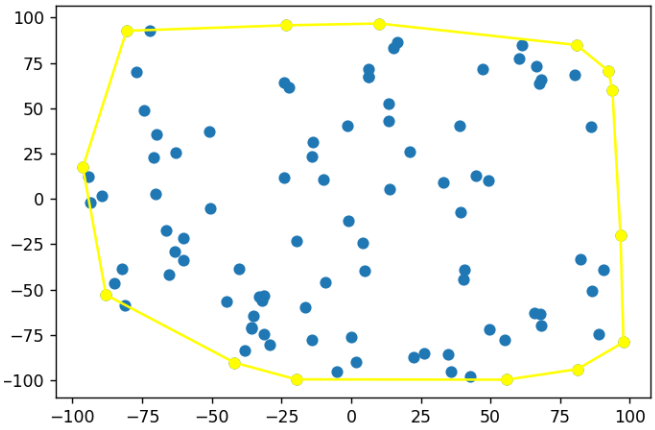
Rysunek nr 8.



Rysunek nr 9.



Rysunek nr 10. Wyznaczona otoczka wypukła

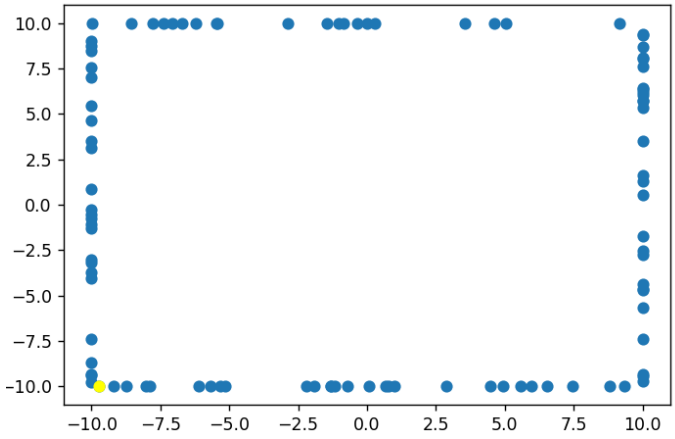


5.2.Algorytm Jarvisa

Poniżej przedstawiony jest przebieg działania algorytmu Jarvisa dla zbioru c)

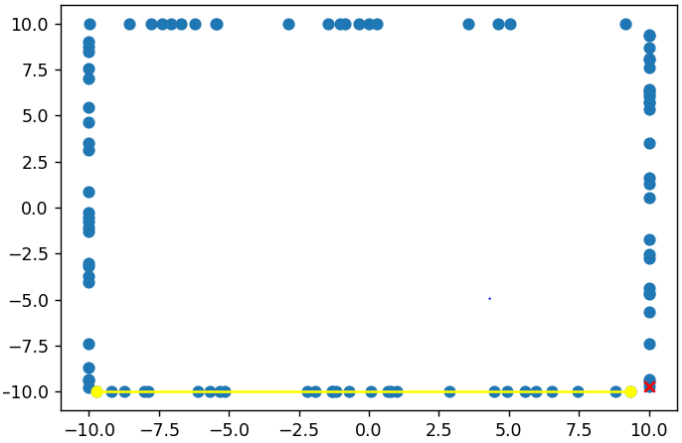
Rysunek nr 11.

Wyróżniony na żółto punkt o najmniejszej współrzędnej y



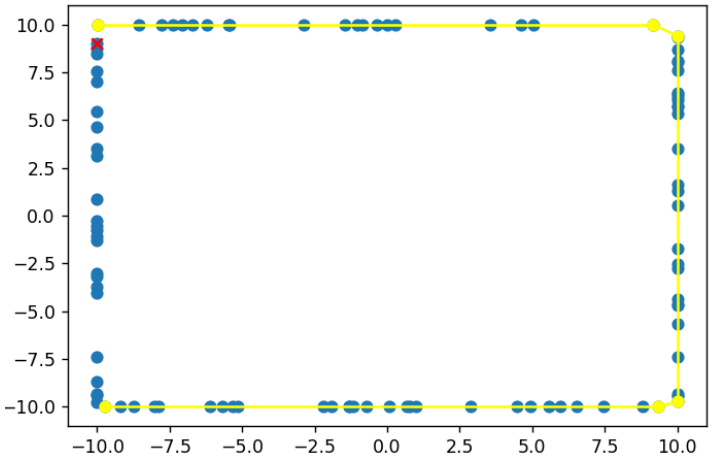
Rysunek nr 12.

Wyróżniona na żółto obecna otoczka wypukła, a na czerwono obecnie rozpatrywany punkt

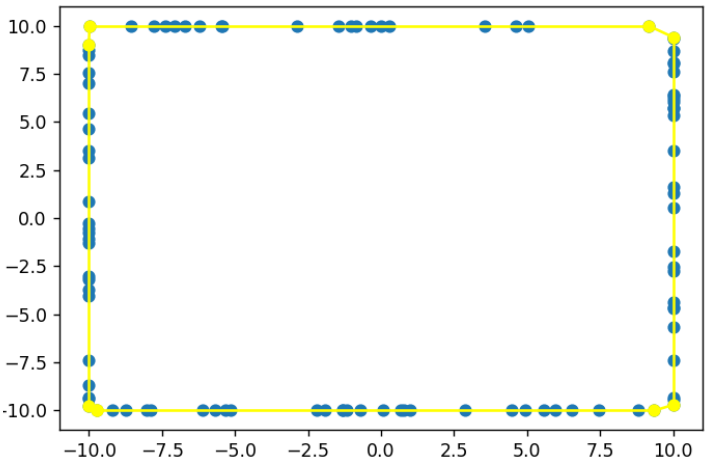


Rysunki 13-14 prezentują dalszy skrócony przebieg działania algorytmu przy zachowaniu powyższych oznaczeń

Rysunek nr 13.

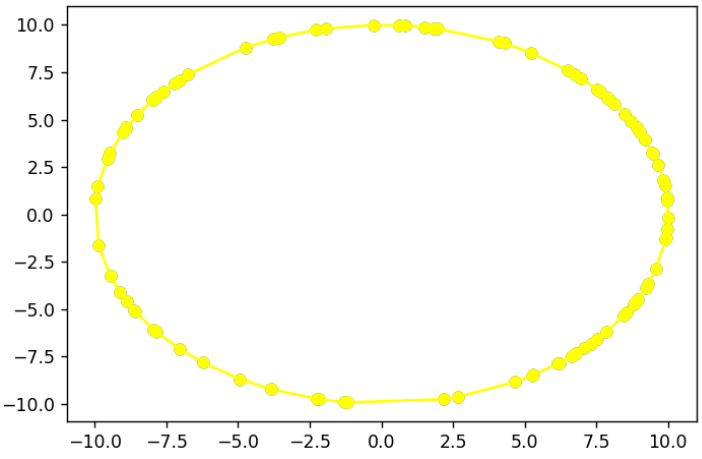


Rysunek nr 14. Wyznaczona otoczka wypukła

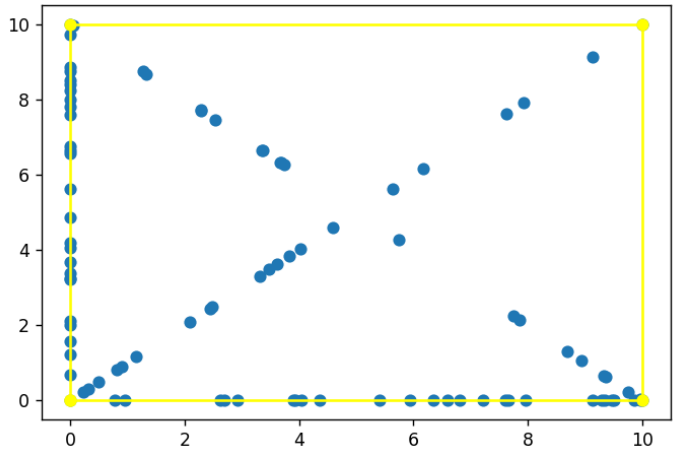


Poniższe rysunki prezentują końcowy rezultat działania algorytmów uzyskany na zbiorach b) oraz d)

Rysunek nr 15. Wyznaczona otoczka wypukła dla zbioru b)



Rysunek nr 16. Wyznaczona otoczka wypukła dla zbioru d)



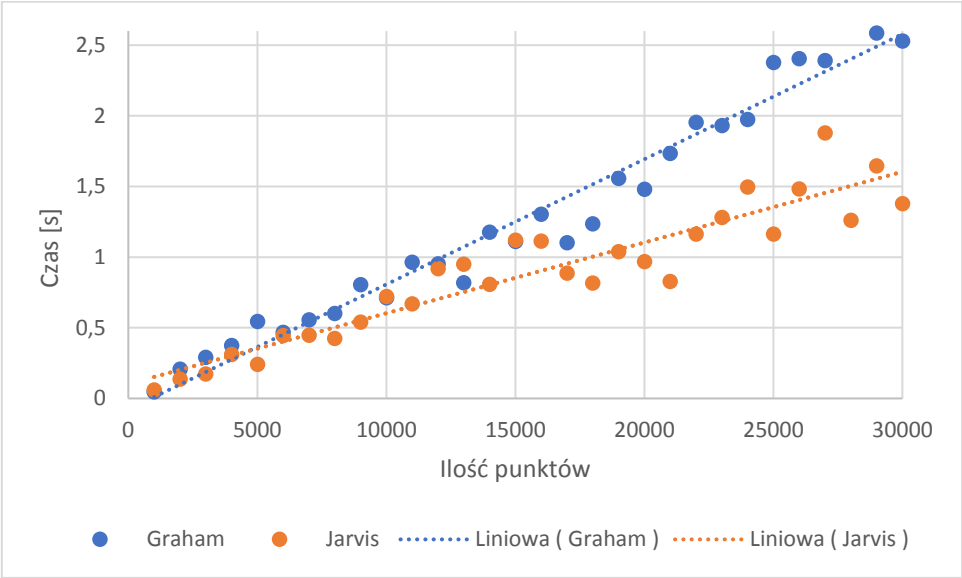
6. Porównanie czasu działania algorytmów

W poniższej tabeli zestawione zostały czasy działania algorytmów Jarvisa i Grahama na wygenerowanych zbiorach punktów. Wartości podane są w sekundach.

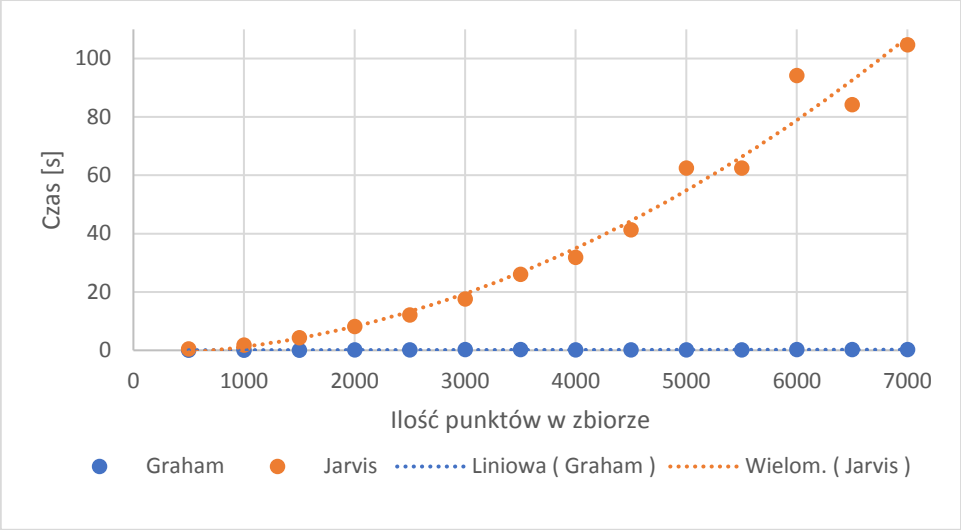
	Zbiór a)	Zbiór b)	Zbiór c)	Zbiór d)
Graham	0.002999305725	0.003001451492	0.005003452301	0.003000974655
Jarvis	0.005503177643	0.038027524948	0.003982305527	0.002000093460

Dla każdego z czterech sposobów rozkładu punktów na płaszczyźnie zbadany został czas działania algorytmów w zależności od ilości punktów w zbiorze. Otrzymane wyniki dla poszczególnych zbiorów przedstawione są na poniższych wykresach. Do wykresów została dopasowana linia trendu pozwalająca na oszacowanie złożoności czasowej algorytmów dla danego układu punktów.

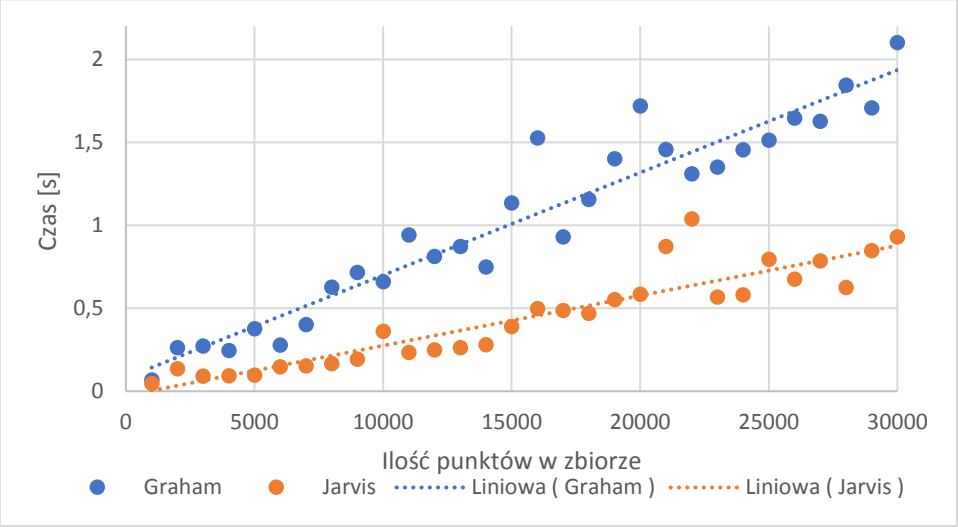
Wykres 1. Losowe punkty z przedziału [-100,100]



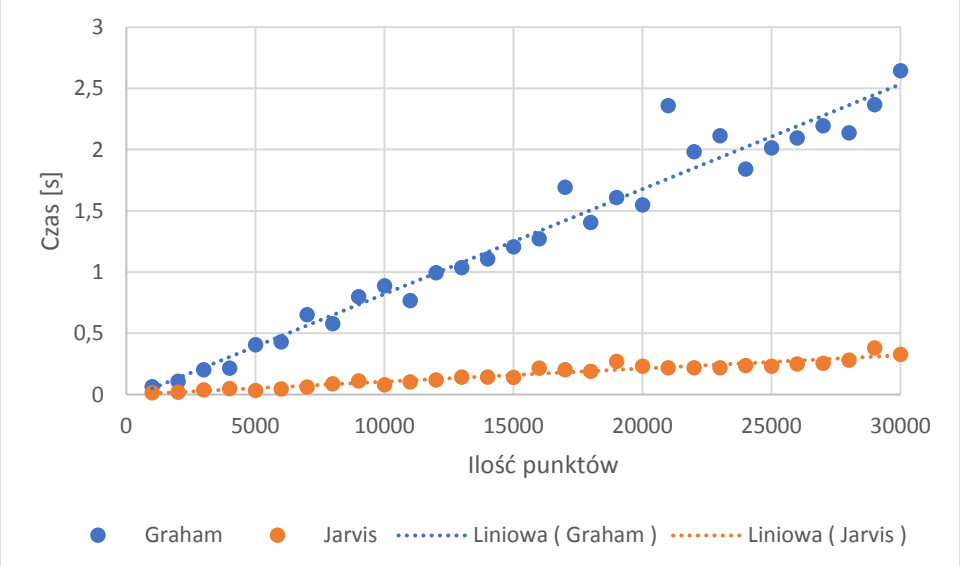
Wykres 2.Losowe punkty leżące na okręgu



Wykres 3. Losowe punkty leżące na prostokącie



Wykres 4. Losowe punkty leżące na dwóch bokach kwadratu oraz na jego przekątnych



7.Podsumowanie i wnioski

- analizując uzyskane wyniki, można stwierdzić, że zaimplementowane w ćwiczeniu algorytmy zadziałały poprawnie dla wszystkich wygenerowanych zbiorów – otrzymane zbiory wynikowe składały się z punktów, które dla danego zestawu współrzędnych faktycznie tworzą otoczkę wypukłą

- dzięki różnorodności we wzajemnym położeniu punktów dla poszczególnych zbiorów, możliwe było dosyć dokładne zweryfikowanie poprawności algorytmów. Korzystając z utworzonej wizualizacji procesu znajdowania otoczki wypukłej, możliwe jest zabserwowanie, jak bardzo różni się ilość wykonanych kroków, oraz liczba rozpatrzanych punktów w zależności od dobranego zbioru wejściowego – najwięcej porównań dokonujemy dla zbiorów b) oraz c), a najmniej dla zbioru d)

- zbiory b) oraz c) są zbiorami, które mogły stwarzać najwięcej problemów w działaniu algorytmów. Zbiór b) składa się z czterech podzbiorów wśród których punkty są względem siebie współliniowe – mogły więc wystąpić trudności przy eliminowaniu ich ze zbioru wynikowego, spowodowane niewystarczającą dokładnością obliczeń. Zbiór c) składa się z punktów, które w całości wchodzą w skład otoczki wypukłej – problemy mogły być spowodowane bardzo dużą ilością punktów do rozpatrzenia oraz niewielkimi różnicami w kątach pomiędzy kolejnymi zbiorami punktów

- dla wygenerowanych w ćwiczeniu zbiorów, algorytm Jarvisa okazał się działać niemal 2 razy szybciej od algorytmu Grahama. Dla obydwu algorytmów, zbiorem, dla którego najkrócej trwało znajdywanie otoczki wypukłej był zbiór d), a najdłuższy proces obserwowaliśmy dla zbiorów b) oraz c).

- analizując wykresy zależności czasu trwania algorytmów od liczby punktów w zbiorze, możemy zaoberwować, że dla dużej ilości punktów, najdłużej trwa znajdowanie otoczki wypukłej dla punktów należących do okręgu, a najkrócej dla punktów leżących na dwóch bokach oraz przekątnych kwadratu

- do wyznacznia otoczki w zbiorach, w których rozkład punktów jest analogiczny do zbiorów a), c) oraz d), dla każdej ilości punktów w zbiorze, lepiej sprawdza się algorytm Jarvisa. Dla punktów należących do okręgu, szybciej działającym okazuje się być algorytm Grahama

- dzięki wyznaczonej lini trendu, możemy ocenić, że złożoność czasowa algorytmu Jarvisa dla zbioru punktów leżących na okręgu jest zbliżona do $O(n^2)$. W pozostałych przybadkach w badanym zakresie punktów w zbiorze, dla obu algorytmów obserwujemy złożoność zbliżoną do liniowej.