Algorytmy geometryczne

Sprawozdanie z ćwiczeń 2.

Joanna Kulig, grupa 5

Dane urządzenia, na którym wykonano ćwiczenie:

Procesor: 1,4GHz Czterordzeniowy procesor Intel Core i5
Komputer z systemem macOS Monterey
Środowisko: jupyter notebook
Do rozwiązania ćwiczenia wykorzystano język Python wraz z bibliotekami numpy, matplotlib oraz random.

1. Cel ćwiczenia.

Ćwiczenie polegało na implementacji algorytmu Grahama oraz algorytmu Jarvisa wyznaczających otoczkę wypukłą. Następnie należało sprawdzić ich działanie oraz czas działania dla różnych zestawów danych.

2. Wygenerowanie zbiorów punktów.

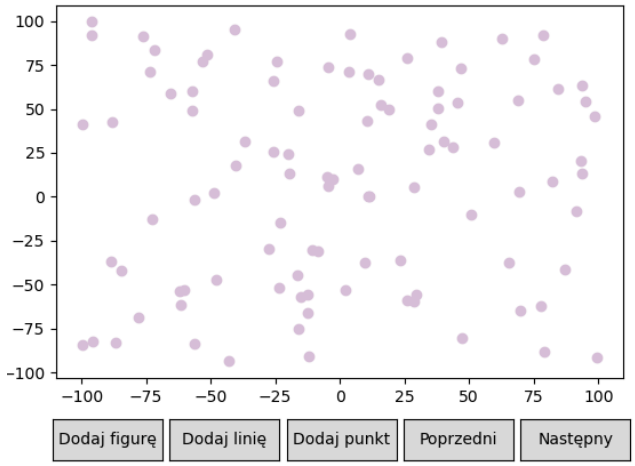
Losowe wygenerowanie punktów wykonano przy wykorzystaniu funkcji random.uniform(), random.rand() oraz funkcji trygonometrycznych:

1. **Zestaw danych a**: 100 losowych punktów o współrzędnych z przedziału [-100, 100].
2. **Zestaw danych b**: 100 losowo wygenerowanych punktów leżących na okręgu o środku (0,0) i promieniu R=10.
3. **Zestaw danych c**: 100 losowo wygenerowanych punktów leżących na bokach prostokąta o wierzchołkach (-10, 10), (-10,-10), (10,-10), (10,10).
4. **Zestaw danych d**: 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.

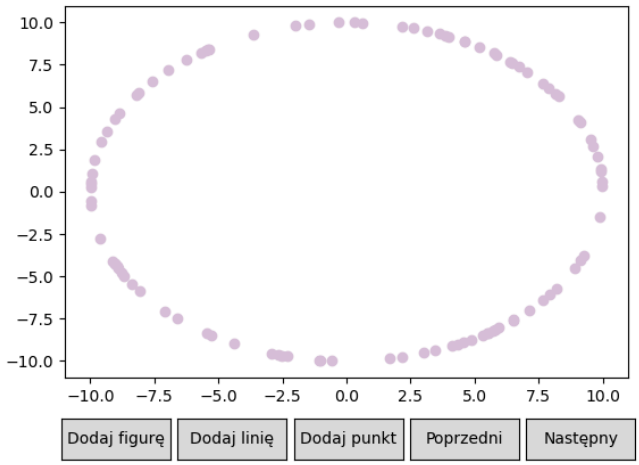
W celu porównania czasów działania zaimplementowanych algorytmów Jarvisa i Grahama stworzono dodatkowe zestawy punktów o takich samych parametrach jak powyższe, ale z większą liczbą punktów.

3. Wizualizacja zestawów danych.

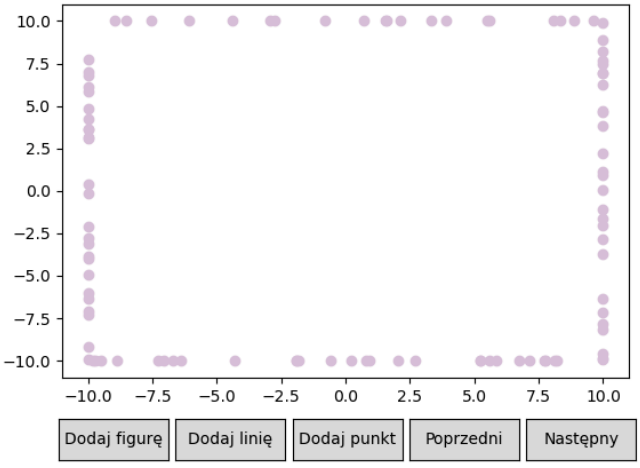
Do wizualizacji graficznej otrzymanych zestawów danych wykorzystano bibliotekę matplotlib języka Python.



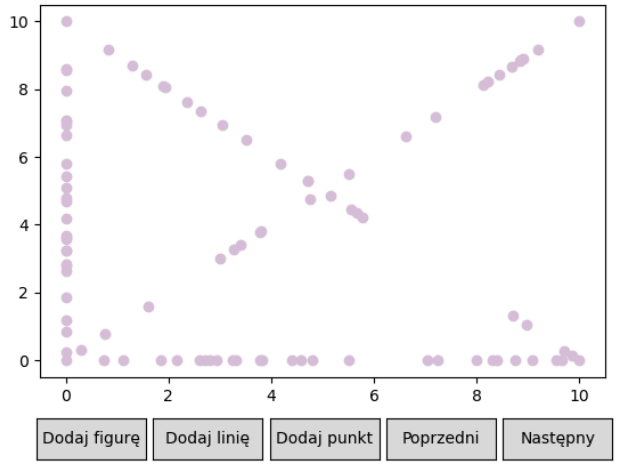
Wykres 1.1 Zestaw danych a



Wykres 1.2 Zestaw danych b



Wykres 1.3 Zestaw danych c



Wykres 1.4 Zestaw danych d

4. Tolerancja dla zera i sposób obliczania wyznacznika.

W obliczeniach przyjęto tolerancje dla zera 10^{-12}. Wykorzystano wyznacznik 3x3 własnej implementacji.

5. Opisy algorytmów.

Algorytm orient

Funkcja orient(a, b, c) korzystając z wyznacznika wskazuje, po której stronie leży punkt c względem prostej wyznaczonej przez punkty a i b lub czy jest on współliniowy.

Wykorzystana jest w algorytmach wyznaczania otoczki wypukłej w celu ustanowienia odpowiedniej kolejności punktów ze względu na kąt, jaki tworzy wektor punktu, z którym porównujemy i pozostałe punkty z dodatnim kierunkiem osi x.

Algorytm Grahama

W zbiorze S wybrany zostaje punkt o najmniejszej współrzędnej y. Jeżeli istnieje więcej takich punktów, to algorytm wybiera taki, który ma także najmniejszą współrzędną x.

W kolejnym kroku punkty zostają posortowane (korzystając z algorytmu Quick Sort własnej implementacji). Algorytm sortowania zmodyfikowano w taki sposób, aby punkty były sortowane ze względu na kąt, jaki tworzy wektor punktu początkowego z punktem rozważanym względem dodatniego kierunku osi x - wykorzystano do tego funkcję orient. Jeśli rozpatrywane punkty tworzą ten sam kąt, to jako pierwsze zostaną wybrane punkty najbardziej oddalone od punktu początkowego.

Następnie algorytm w pętli sprawdza orientację aktualnie rozpatrywanego punktu względem ostatnio rozważanych punktów. Jeśli jest on z nimi współliniowy to usuwa poprzedni punkt i dodaje aktualnie rozpatrywany punkt do otoczki. Gdy leży po prawej stronie to dodaje go do otoczki, a gdy leży po lewej to usuwa ostatni punkt.

Na koniec algorytm weryfikuje, czy ostatni dodany do otoczki punkt nie jest współliniowy względem punktu będącego przed nim oraz punktu początkowego. Jeśli tak, to zostaje on usunięty.

Złożoność obliczeniowa algorytmu: O(nlogn), gdzie n to liczba punktów w zbiorze.

Algorytm Jarvisa (owijanie prezentu)

Podobnie jak w algorytmie Grahama, w zbiorze S wybrany zostaje punkt początkowy o najmniejszej współrzędnej y. Jeżeli istnieje więcej takich punktów, to algorytm wybiera taki, który ma także najmniejszą współrzędną x.

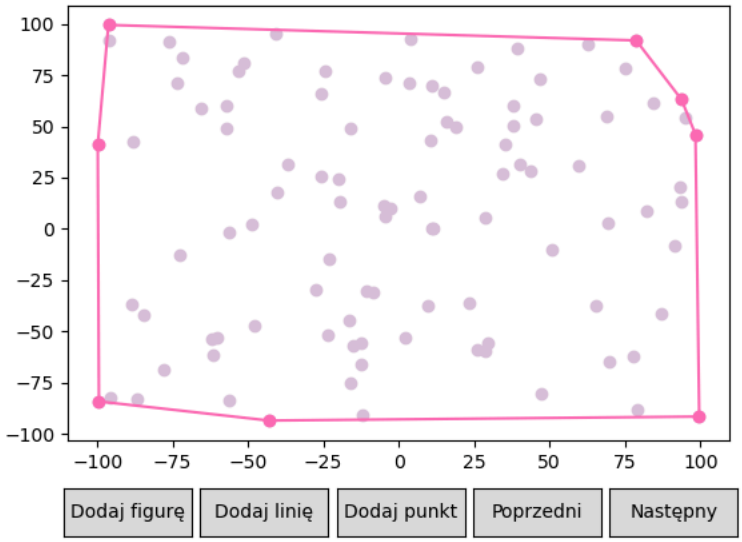
W kolejnym kroku algorytm w pętli szuka punktu p, dla którego pozostałe punkty leżą po lewej stronie odcinka, który jest wyznaczony przez ostatni punkt w otoczce i punkt p (wykorzystana zostaje funkcja orient). W przypadku znalezienia punktów współliniowych, wybrany zostaje ten, który jest najbardziej oddalony od ostatniego punktu w otoczce. Znaleziony punkt jest dodawany do otoczki. Pętla zostaje przerwana dopiero w momencie dotarcia do punktu początkowego.

Złożoność obliczeniowa algorytmu: $O(n^2)$, gdzie n to liczba punktów w zbiorze. Gdy liczba wierzchołków w otoczce zostanie ograniczona do stałej k, złożoność algorytmu jest rzędu O(nk).

6. Otrzymane wyniki.

Algorytmy sprawdzono na tych samych zestawach danych. Otrzymane punkty należące do otoczki wypukłej okazały się poprawne.

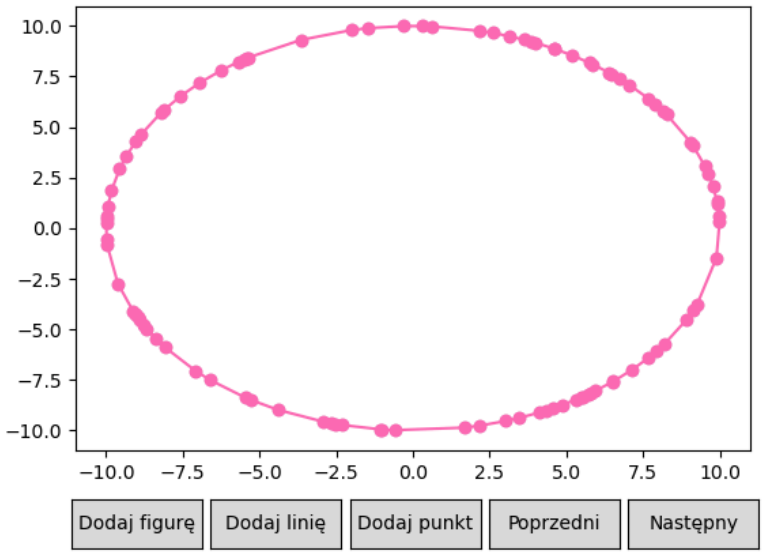
Zestaw danych a



Wykres 2.1 Otoczka wypukła dla zestawu danych a

Wszystkie punkty zostały poprawnie sklasyfikowane. Algorytm Jarvisa w tym przypadku wypadł gorzej pod względem czasu działania.

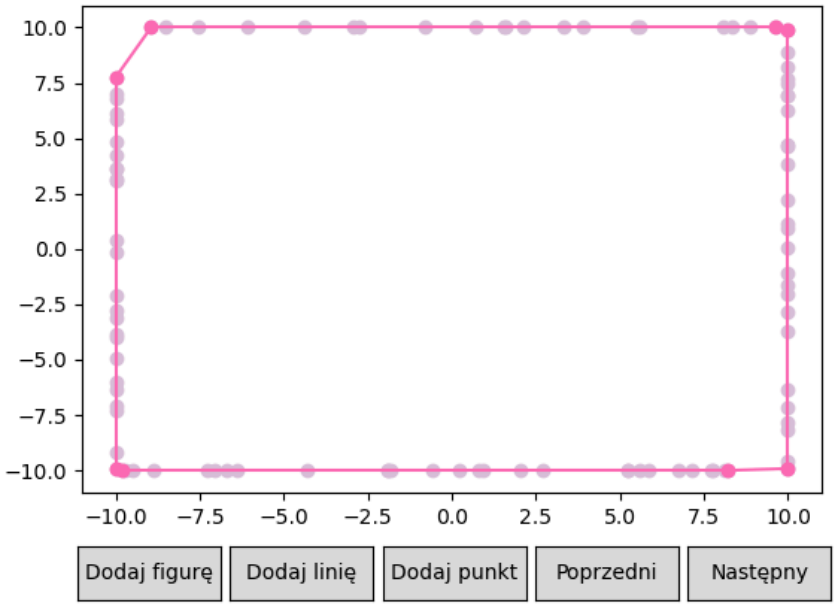
Zestaw danych b



Wykres 2.2 Otoczka wypukła dla zestawu danych b

W zestawie danych b wszystkie punkty zostały dodane do otoczki. Zestaw danych b pokazał przewagę algorytmu Grahama nad algorytmem Jarvisa. Algorytm Grahama wyznaczył tę samą otoczkę, ale zrobił to znacznie szybciej. Jest to spowodowane złożonością obliczeniową algorytmu Jarvisa, który jest zależny od liczby punktów w otoczce. W tym przypadku jego złożoność obliczeniowa to $O(n^2)$, natomiast złożoność algorytmu Grahama to O(nlogn).

Zestaw danych c

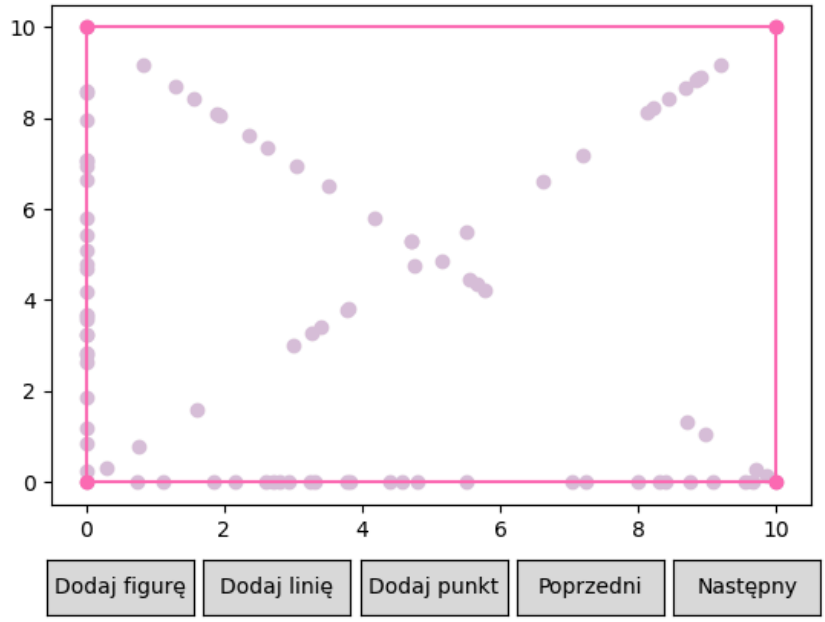


Wykres 2.3 Otoczka wypukła dla zestawu danych c

Zgodnie z oczekiwaniami, algorytmy wyznaczyły skrajne punkty dla każdego z boków prostokąta - dlatego otoczka zawiera osiem punktów. Istnieje możliwość, że w zestawie danych c wylosowałyby się punkty leżące idealnie na rogach prostokąta. Wtedy otoczka mogłaby mieć od czterech do siedmiu punktów.

Taki przypadek jest jednak mniej prawdopodobny. Zestaw danych c idealnie sprawdza, czy punkty są dobrze klasyfikowane pod względem współliniowości. Ze względu na ograniczoną liczbę punktów w otoczce algorytm Jarvisa w tym przypadku jest szybszy i działa w złożoności liniowej (k=8, O(nk) ≈ O(8n) ≈ O(n)).

Zestaw danych d



Wykres 2.4 Otoczka wypukła dla zestawu danych d

Zestaw danych d również pokazuje, że czasem algorytm Jarvisa jest lepszy niż algorytm Grahama. Dzieje się tak, ponieważ w tym przypadku otoczka zawsze powinna zawierać cztery punkty, co dla algorytmu Jarvisa (k=4, O(nk) ≈ O(4n) ≈ O(n)) daje złożoność liniową. Zestaw danych d również sprawdza prawidłowość klasyfikacji współliniowości.

7. Porównanie czasu działania algorytmów (bez usunięcia flag)

Aby uzyskać rzeczywiste czasy działania algorytmów Jarvisa i Grahama zrezygnowano z wizualizacji. W algorytmach nie usunięto jednak flag, które oznaczały, czy dany zbiór ma zostać zwizualizowany. Wykorzystano zestawy danych z tymi samymi parametrami, ale zwiększoną liczbą punktów. Analiza czasowa została przeprowadzona dla następującej liczby punktów: 500, 1000, 2000, 3000, 4000, 5000.

Zestaw danych	Algorytm	500 punktów	1000 punktów	2000 punktów	3000 punktów	4000 punktów	5000 punktów
Zestaw a	Graham	0.015569	0.030085	0.059325	0.089636	0.116910	0.148594
	Jarvis	0.018026	0.050430	0.078948	0.139282	0.191428	0.241560
Zestaw b	Graham	0.014340	0.029092	0.054250	0.096868	0.116022	0.141014
	Jarvis	0.564856	2.201598	8.989232	19.04342	34.37821	55.94336
Zestaw c	Graham	0.013313	0.022907	0.052214	0.071526	0.099213	0.122749
	Jarvis	0.008059	0.014140	0.025950	0.039366	0.052459	0.065321
Zestaw d	Graham	0.024283	0.044025	0.092348	0.145961	0.190836	0.254321
	Jarvis	0.004642	0.008207	0.016463	0.024959	0.032610	0.040548

Tabela 1. Czas działania (w sekundach) algorytmów Grahama i Jarvisa dla zestawów danych a, b, c i d w zależności od liczby punktów przed usunięciem flag.

Dla zestawu danych a algorytm Grahama okazał się być szybszy od algorytmu Jarvisa.

W zestawie danych b algorytm Jarvisa osiągnął złożoność kwadratową. W tym przypadku algorytm Grahama ma znaczną przewagę nad algorytmem Jarvisa ze względu na czas działania.

W zestawach danych c i d szybszy okazał się algorytm Jarvisa. Są to zestawy danych, które mają ograniczoną liczbę punktów w otoczce. W tym przypadku algorytm Jarvisa ma złożoność liniową, dzięki czemu działa szybciej od algorytmu Grahama.

8. Porównanie czasu działania algorytmów (po usunięciu flag)

Zestaw danych	Algorytm	500 punktów	1000 punktów	2000 punktów	3000 punktów	4000 punktów	5000 punktów
Zestaw a	Graham	0.015356	0.029766	0.059291	0.087544	0.116297	0.145320
	Jarvis	0.017923	0.050173	0.077291	0.134711	0.185992	0.238012
Zestaw b	Graham	0.014191	0.029654	0.053250	0.095969	0.117145	0.139400
	Jarvis	0.556588	2.147906	8.768624	19.57710	34.61221	55.81614
Zestaw c	Graham	0.013500	0.024854	0.052233	0.071055	0.098870	0.125039
	Jarvis	0.007842	0.013477	0.026077	0.039031	0.051714	0.066173
Zestaw d	Graham	0.023842	0.036541	0.090708	0.144783	0.190527	0.266510
	Jarvis	0.004401	0.008157	0.016094	0.024229	0.032757	0.042930

Tabela 2. Czas działania (w sekundach) algorytmów Grahama i Jarvisa dla zestawów danych a, b, c i d w zależności od liczby punktów po usunięciu flag.

Przeprowadzono takie same pomiary, przy wykorzystaniu tych samych, wcześniej wygenerowanych zbiorów. Podobnie jak wcześniej zrezygnowano z wizualizacji danych. Dodatkowo

usunięto w algorytmach flagi, które oznaczały, czy dany zbiór ma zostać zwizualizowany.

W większości przypadków zaobserwowano różnice w czasie. Algorytmy bez flag działają szybciej, ponieważ nie wykonują zbędnych porównań wiele razy.

9. Wnioski

Zarówno algorytm Grahama jak i algorytm Jarvisa poprawnie wyznaczają otoczkę wypukłą.

Czas działania algorytmu Grahama dla różnych zestawów danych zawierających taką samą liczbę punktów jest podobny, co wynika z jego złożoności obliczeniowej, która nie zależy od liczby punktów w otoczce wypukłej.

Dla algorytmu Jarvisa otrzymujemy ogromne różnice w czasie dla różnych zestawów danych. Jest to spowodowane jego złożonością obliczeniową, która zależy od liczby punktów w otoczce wypukłej. Prezentuje to idealnie zestaw danych b, w którym algorytm Jarvisa osiąga złożoność kwadratową, ponieważ otoczka wypukła zawiera wszystkie punkty należące do zestawu danych. Natomiast w zestawach danych c i d algorytm Jarvisa działa bardzo szybko, ponieważ jego złożoność obliczeniowa w

tym przypadku jest liniowa ze względu na ograniczoną ilość punktów w otoczce (4-8 punktów).

Algorytm Jarvisa warto stosować jedynie w przypadku, gdy jesteśmy świadomi, że w otoczce znajdzie się niewiele punktów. W pozostałych przypadkach bezpieczniej jest zastosować algorytm Grahama, ponieważ czas jego wykonywania nie zależy od ilości punktów w otoczce i możemy przewidzieć czas jego wykonania.