ALGORYTMY GEOMETRYCZNE

Sprawozdanie z ćwiczenia 2.

Bartosz Kucharz
Grupa nr 1

1. Specyfikacja techniczna

Komputer na którym wykonywano obliczenia:

System operacyjny: Windows 10 x64

Procesor: Intel® Core™ i5-5300U CPU @ 2.30Ghz

Pamięć RAM: 8GB

Obliczenia zostały wykonywane w języku Python 3 z wykorzystaniem bibliotek: numpy, random i matplotlib.

2. Opis ćwiczenia

Ćwiczenie polegało na implementacji algorytmów Grahama oraz Jarvisa, wyznaczających otoczkę wypukłą zbioru punktów. Analizy ich działania na określonych wygenerowanych zestawach. Należało również zwizualizować pracę algorytmów.

3. Generowanie punktów

Wygenerowane zostały następujące zbiory punktów:

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

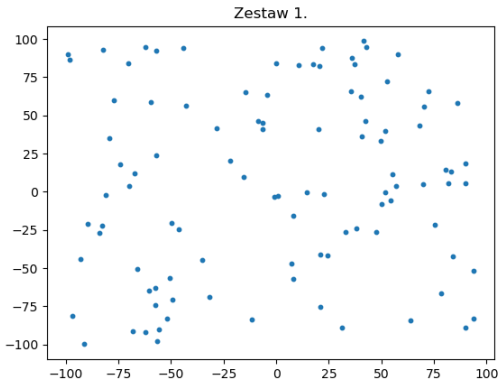
W celu dokonania analizy czasowej algorytmów dodatkowo zostały wygenerowane zbiory o tych samych parametrach, jednak ze zmienionymi liczbami punktów.
Zbiory były reprezentowane jako dwuwymiarowe tablice z biblioteki numpy (numpy arrays). Współrzędne były liczbami typu float (typ float w Pythonie jest podwójnej precyzji).

Do wygenerowania losowych punktów skorzystano z funkcji random.uniform(), która generuje liczby rzeczywiste typu float z podanego zakresu.

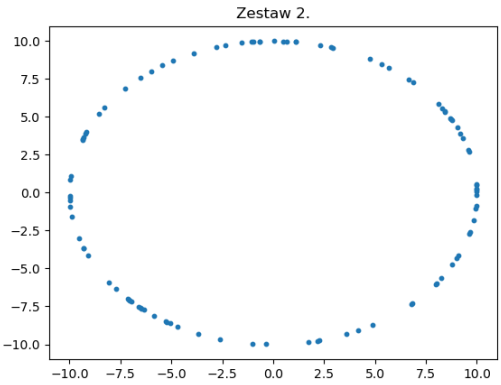
W zestawie 2. dodatkowo skorzystano z metod sin() oraz cos() z biblioteki numpy.

W zestawie 3. najpierw losowano jeden z czterech boków kwadratu, a później zależnie od przypadku losowano punkt na prostej.

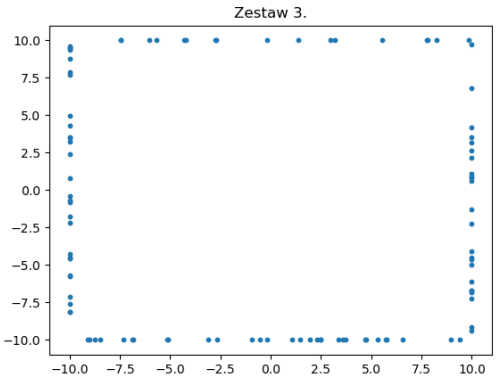
W zestawie 4. Do wygenerowanie punktów na przekątnych użyto postaci ogólnych prostych.



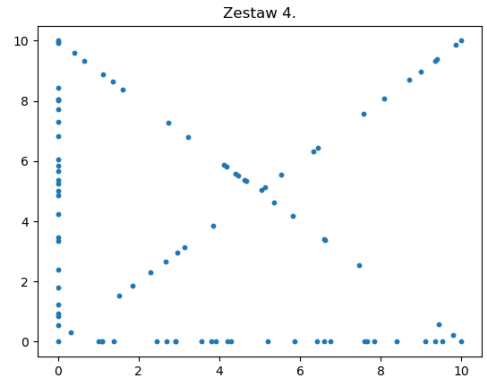
Wykres 1 Zbiór punktów z zestawu 1.



Wykres 2 Zbiór punktów z zestawu 2.



Wykres 3 Zbiór punktów z zestawu 3.



Wykres 4 Zbiór punktów z zestawu 4.

4. Wyznaczanie orientacji punktów

W ćwiczeniu do porównywania położeń punktów korzystano z wyznacznika macierzy 3x3 implementacji własnej. Przy określaniu orientacji punktów zastosowano tolerancję dla zera równą 10^{-12}, ponieważ jest to najmniejsza tolerancja dla której, otoczka zawsze była wyznaczana prawidłowo.

5. Implementacja algorytmu Grahama

Algorytm znajduje punkt p_0 o najmniejszej współrzędnej y (w przypadku znalezienia więcej niż jednego wybiera ten o najmniejszej współrzędnej x). Następnie za pomocą wbudowanej funkcji sorted() sortuje resztę punktów względem kąta, jaki tworzy wektor (p0,p) z dodatnim kierunkiem osi x (jeśli więcej niż jeden punkt tworzy taki sam kąt, za większy w porządku wybiera punkt leżący w większej odległości od punktu p_0). Funkcja sorted() korzysta z metody porównującej własnej implementacji opartej na obliczaniu wyznacznika macierzy wektorów. Następnie przechodzi po posortowanej tablicy usuwając punkty współliniowe w rezultacie zostawiając ten najbardziej oddalony. W głównej pętli algorytmu używa stosu reprezentowanego przez listę pythonowską. Stosując metody append() oraz pop() do dodawania lub usuwania punktów ze stosu. Algorytm ostatecznie zwraca dwuwymiarową tablicę zawierającą punkty otoczki.

6. Implementacja algorytmu Jarvisa

Algorytm znajduje punkt p_0 jak w algorytmie Grahama. Następnie w głównej pętli algorytmu znajduje punkt tworzący największy kąt między ostatnią krawędzią otoczki a tym punktem zgodnie z ruchem wskazówek zegara (używając do tego wyznacznika implementacji własnej).

W przypadku znalezienia punktu o tym samym kącie wybiera ten najbardziej oddalony od ostatniego punktu otoczki. Algorytm kończy się w chwili powrotu do punktu startowego p_0 oraz zwraca punkty otoczki.

7. Wizualizacje

Do wizualizacji wykorzystano bibliotekę matplotlib. W celu zrealizowania wizualizacji w obu implementacjach algorytmów w ich głównych pętlach tworzone są sceny punktów (punkty otoczki na danym etapie algorytmu), które wpływają na czas trwania algorytmów. Jednak tą opcje można wyłączyć podczas analizy czasowej.

8. Wyniki

Oba algorytmy były analizowane na tych samych zbiorach punktów. Oba działały poprawnie i zwracały takie same punkty otoczki.

8.1. Zestaw 1.

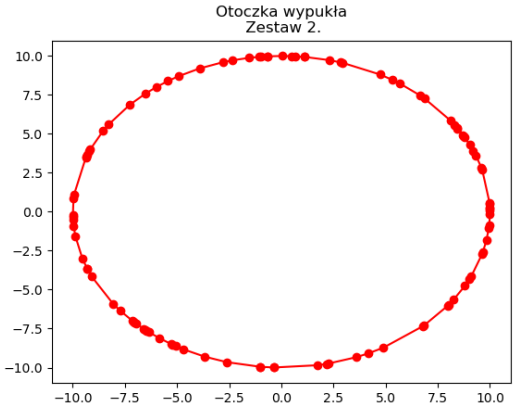
Ten zestaw był prostym przypadkiem. Była w nim nieznaczna ilość punktów współliniowych oraz mała liczba punktów otoczki.

Wykres 5 Otoczka wyznaczona na zestawie 1.

```
Liczba punktów otoczki: 12
Liczba wszystkich punktów: 100
```

8.2. Zestaw 2.

W tym zestawie można zauważyć dużą różnicę w czasie (dokładna analiza zostanie omówiona w dalszej części) między algorytmami Grahama i Jarvisa. Ze względu na dużą liczbę punktów które należą do otoczki złożoność czasowa algorytmu Jarvisa wynosi $O(n^2)$, ponieważ dla każdego punktu z otoczki przechodzi po wszystkich punktach szukając punktu dającego największy kąt.

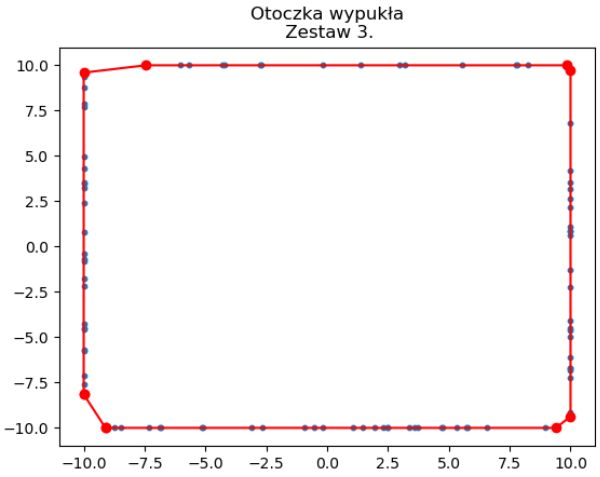


Wykres 6 Otoczka wyznaczona na zestawie 2.

```
Liczba punktów otoczki: 100
Liczba wszystkich punktów: 100
```

8.3. Zestaw 3.

W tym zestawie kluczowe znaczenie miało radzenie sobie algorytmów z punktami współliniowymi. Dodatkowe porównywanie ze względu na odległość oraz eliminacja reszty punktów w algorytmie Grahama rozwiązały ten problem. W rezultacie oba algorytmy poprawnie wyznaczyły otoczkę.

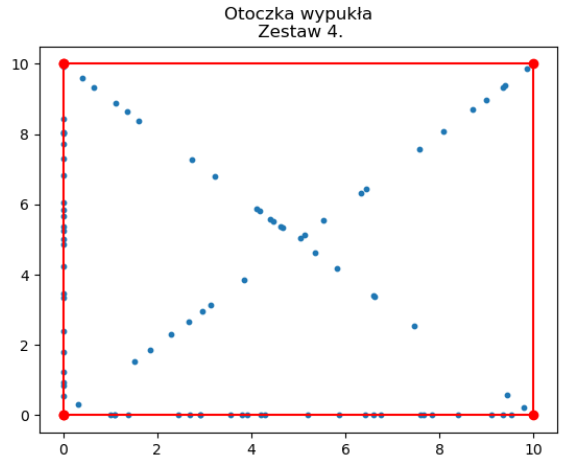


Wykres 7 Otoczka wyznaczona na zestawie 3.

```
Liczba punktów otoczki: 8
Liczba wszystkich punktów: 100
```

8.4. Zestaw 4.

Ostatni zestaw podobnie jak trzeci zawierał dużo punktów współliniowych. Ze względu na małą liczbę punktów otoczki która wynosiła 4. Algorytm Jarvisa poradził sobie z zadaniem szybciej od algorytmu Grahama.



Wykres 8 Otoczka wyznaczona na zestawie 4.

```
Liczba punktów otoczki: 4
Liczba wszystkich punktów: 94
```

9. Analiza czasowa

Na koniec algorytmy zostały przeanalizowane i porównane ze względu na czas wykonywania. W tym celu skorzystano z funkcji time z biblioteki o takiej samej nazwie. Wyłączono tworzenie wizualizacji żeby nie wpływała na wynik pomiaru czasu.
Oba algorytmy zostały uruchomione na czterech zestawach o podanych wcześniej parametrach, jednak tym razem sprawdzano działanie na zwiększonej liczbie punktów. Liczby punktów w testach wynosiły:
100, 1000, 2500, 5000, 7500, 10000, 15000, 20000

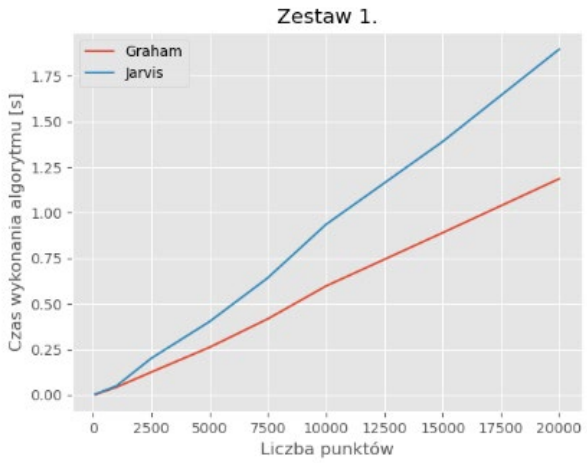
W Zestawie 1. Algorytm Grahama okazał się szybszy od algorytmu Jarvisa, jednak dla danych punktów nie jest to znacząca różnica.

W Zestawie 2. największą różnicę między algorytmami. Ze względu na to że wszystkie lub prawie wszystkie punkty z zestawu należą do otoczki algorytm Jarvisa działał w czasie $O(n^2)$ co dobrze widać na poniższym wykresie zależności czasu od ilości punktów. Ze względu na bardzo duży wzrost czasu wykonywania się algorytmu Jarvisa dla liczby punktów większej od 5000 algorytm ten nie był wykonywany dla tych punktów.

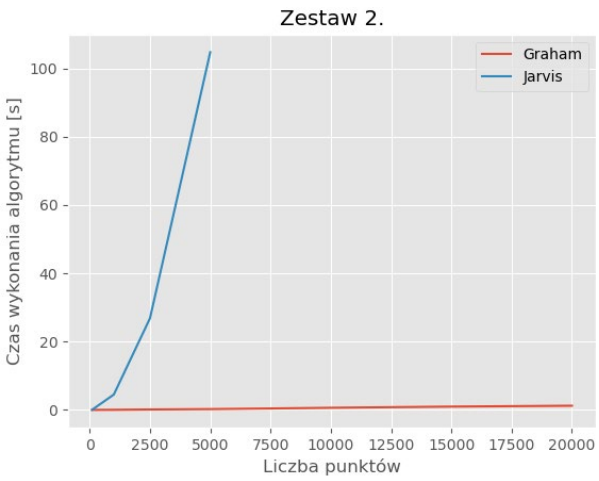
W Zestawie 3. to jednak algorytm Jarvisa szybciej znajdował otoczkę wypukłą, ponieważ tym razem liczba punktów otoczki była bardzo mała.
Podobna sytuacja miała miejsce w Zestawie 4. gdzie otoczka zawsze miała tylko 4 punkty, zatem algorytm Jarvisa jedynie cztery razy musiał wyszukiwać odpowiedni punkt.

		100	1000	2500	5000	7500	10000	15000	20000
Zestaw	Algorytm								
Zestaw 1.	Grahama	0.002998	0.042999	0.125975	0.262004	0.418023	0.597978	0.889995	1.186025
	Jarvisa	0.005	0.050001	0.201973	0.403	0.643004	0.936	1.390022	1.896
Zestaw 2.	Grahama	0.0	0.031215	0.124998	0.23443	0.41587	0.617979	0.965699	1.222087
	Jarvisa	0.046858	4.495156	26.830878	104.750332	None	None	None	None
Zestaw 3.	Grahama	0.003975	0.045998	0.130026	0.275978	0.506999	0.613998	0.965003	1.334996
	Jarvisa	0.003996	0.037999	0.092998	0.16802	0.268003	0.346	0.5	0.675973
Zestaw 4.	Grahama	0.00299	0.045003	0.138002	0.306997	0.516001	0.681998	1.125006	1.450994
	Jarvisa	0.001	0.016999	0.04	0.080027	0.127977	0.168002	0.26102	0.332973

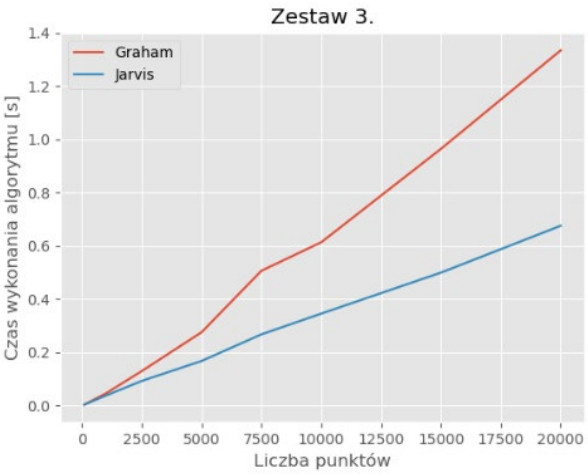
Tabela 1 Czasy wykonywania się algorytmów dla poszczególnych zestawów punktów w zależności od liczby punktów w zestawie wyrażone w sekundach



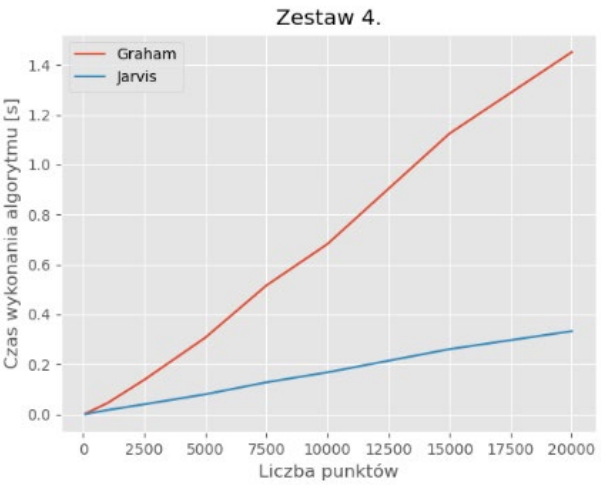
Wykres 9 Porównanie czasów algorytmów w zależności od liczby punktów dla zestawu 1.



Wykres 10 Porównanie czasów algorytmów w zależności od liczby punktów dla zestawu 2.



Wykres 11 Porównanie czasów algorytmów w zależności od liczby punktów dla zestawu 3.



Wykres 12 Porównanie czasów algorytmów w zależności od liczby punktów dla zestawu 4.

10. Wnioski

Na podstawie przeprowadzonych obliczeń można stwierdzić że oba algorytmy prawidłowo wyznaczają otoczkę wypukłą dla dowolnych zbiorów punktów. Algorytm Grahama w podobnym czasie wyznacza otoczkę dla wszystkich zestawów mających podobną liczbę punktów. Natomiast czas wykonywania algorytmu Jarvisa różni się w zależności od zbioru danych na którym działają algorytmy. Wynika to ze złożoności czasowej algorytmu Jarvisa która wynosi $O(nh)$ gdzie h jest liczbą punktów otoczki. W Zestawie 2. gdzie prawie wszystkie punkty zestawu należą do otoczki widać, że złożoność zaczyna być rzędu $O(n^2)$. Jednak w zestawach 3. i 4. Przy małej liczbie punktów otoczki Jarvis ma złożoność praktycznie liniową.

Zatem algorytm Grahama ma szerokie zastosowanie do wszystkich rodzajów zbiorów punktów, ponieważ jego złożoność czasowa nie zależy od zestawu danych. Jednakże w przypadku danych z dużą liczbą wszystkich punktów i względnie małą liczbą punktów otoczki warto stosować algorytm Jarvisa.