Algorytmy geometryczne

Sprawozdanie 2 | Otoczka wypukła

Paweł Fornagiel | Informatyka rok II | Grupa 1

Data Wykonania: 28.10.2024 | Data Oddania: 05.02.2025

1. Opis ćwiczenia i realizacja

1.1. Informacje wstępne

Celem ćwiczenia jest implementacja algorytmów Jarvisa i Grahama, służących wyznaczania otoczki wypukłej dla zadanego zbioru punktów o na płaszczyźnie, wizualizacja ich działania oraz analiza wyników. Na potrzeby realizacji zadania zostały wygenerowane następujące zbiory punktów o losowym, równomiernym rozkładzie:

1. **Zestaw A** - 100 punktów na płaszczyźnie $[-100, 100]^2$
2. **Zestaw B** - 100 punktów leżących na okręgu w układzie współrzędnych o środku $O = (0, 0)$ i promieniu $R = 10$
3. **Zestaw C** - 100 punktów leżących na obwodzie prostokąta o wierzchołkach a, b, c, d o współrzędnych $a = (-10, -10), b = (10, -10), c = (10, 10), d = (-10, 10)$
4. **Zestaw D** - 104 punkty leżące na obwodzie oraz przekątnych prostokąta zdefiniowanego na punktach identycznych względem *Zestawu C*, o następującym rozkładzie:
   - 25 punktów leżących na każdym z boków $|da|$ i $|ab|$
   - 20 punktów leżących na każdej z przekątnych $|ac|$ i $|bd|$
   - 4 punkty leżące na wierzchołkach a, b, c, d

1.2. Generowanie punktów

Równomierny rozkład punktów został zrealizowany za pomocą metody `random.uniform` biblioteki `numpy`, generującej liczby typu `float64` z zadanego zakresu domkniętego. Aby uzyskać powtarzalność wyników ustawiono `random.seed` biblioteki `numpy` na stałą wartość.

Punkty położone na okręgu w *Zestawie B* zostały wygenerowane używając funkcji trygonometrycznych korzystając z parametrycznego równania okręgu promieniu R i środku w punkcie O, gdzie t to parametr rzeczywisty:

$$\begin{cases} y = R \sin t + O_y \\ x = R \cos t + O_x \end{cases}$$

Jednostajny rozkład *Zestawu C* został uzyskany poprzez wygenerowanie liczb, służących jako parametr, o jednostajnym rozkładzie z przedziału $[0, 80]$ odpowiadającemu całkowitej długości obwodu prostokąta. Następnie, położenie punktu na danym boku prostokąta została ustalona na podstawie położenia punktu na odcinku, przyporządkowując odpowiednio:

- $[0, 20)$ - punkty leżące na boku $|ab|$
- $[20, 40)$ - punkty leżące na boku $|bc|$
- $[40, 60)$ - punkty leżące na boku $|cd|$
- $[60, 80]$ - punkty leżące na boku $|da|$

Po ustaleniu przynależności do konkretnego boku, obliczenie pozycji punktu na płaszczyźnie odbyło się poprzez obliczenie wersora $\vec{v}$ odpowiadającego danemu odcinkowi i dodanie do współrzędnych x, y

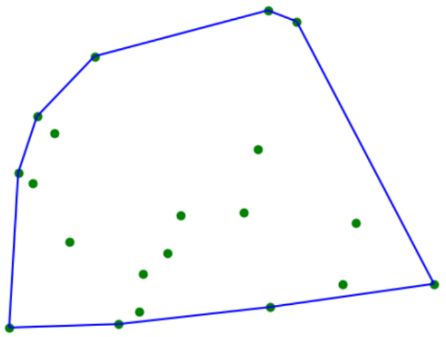
punktu iloczyn odpowiednich składowych wersora $\vec{v}$ i różnicy wartości parametru od początku przedziału w którym się znajduje.

Współrzędne punktów *Zestawu D* zostały obliczone metodą identyczną jak te należące do *Zestawu C*, różniąc się jedynie sposobem generowania parametrów, to jest, dla każdego odcinka został użyty przyporządkowany mu zbiór parametrów. W *Zestawie D* wygenerowano 4 zbiory parametrów o wartościach z przedziałów $[0, \|AB\|]$ ($\|AB\|$ - długość danego odcinka) oraz liczebności odpowiadającej liczbie punktów leżących na tym odcinku.

1.3. Otoczka Wypukła

Otoczka wypukła zbioru Q jest najmniejszym zbiorem wypukłym, który zawiera Q, czyli takim zbiorem $CH(Q)$, że każdy punkt Q znajduje się w $CH(Q)$, a $CH(Q)$ jest wypukły. Zbiór $CH(Q)$ ma tę właściwość, że dla dowolnej pary punktów p,$q \in CH(Q)$ odcinek $|pq|$ również leży w $CH(Q)$.

Rysunek 1 przedstawia przykładową otoczkę wypukłą (kolor niebieski) zdefiniowaną nad zadanym zbiorem punktów (kolor zielony)



Rysunek 1 : Przykład otoczki wypukłej

1.4. Algorytm Grahama

Algorytm Grahama to metoda znajdowania otoczki wypukłej zbioru punktów na płaszczyźnie opierająca się na kątowym sortowaniu punktów.

Po ustaleniu punktu startowego s (punkt o najmniejszej współrzędnej y, a w przypadku kilku takich punktów – także o najmniejszej współrzędnej x), algorytm sortuje wszystkie pozostałe punkty według kątów biegunowych między płaszczyzną $y = s_y$ a prostą, przechodzącą przez punkt rozpatrywany oraz s . Jeżeli grupa punktów jest współliniowa względem danej prostej, z rozpatrywanego zbioru punktów współliniowych usuwane są wszystkie oprócz najbardziej oddalonego od s. Teoretyczna złożoność czasowa algorytmu to $O(n \log n)$, gdzie n to liczba punktów z rozpatrywanym zbiorze.

Użyta implementacja korzysta z sortowania przez scalanie z komparatorem (sortowanie rosnąco), który rozstrzyga relację pomiędzy punktami a i b w następujący sposób:

- $a < b \rightarrow a$ leży po lewej stronie prostej $|sb|$
- $a = b \rightarrow a$ oraz b są współliniowe względem s
- $a > b \rightarrow a$ leży po prawej stronie prostej $|sb|$

Sprawdzanie po której stronie prostej leży dany punkt wykonywane jest przez obliczenie następującego wyznacznika:

$$\det(s,b,a) = \begin{vmatrix} s_x - a_x & s_y - a_y \\ b_x - a_x & b_y - a_y \end{vmatrix} = (s_x - a_x) \cdot (b_y - a_y) - (s_y - a_y) \cdot (b_x - a_x)$$

W procesie scalania algorytmu sortującego, dla punktów które są współliniowe, pomijany jest punkt o mniejszej odległości od s. Porównanie odległości opiera się na wyznaczenie odległości euklidesowej. Dzięki powyższemu, implementacja sortowania przez scalanie spełnia założenia sortowania kątowego.

Reszta algorytmu opiera się na iteracji przez posortowane punkty, dodając bądź usuwając je ze zbioru wynikowego otoczki, będącej stosem. Jeżeli rozpatrywany punkt tworzy skręt w lewo względem odcinka złożonego z ostatnich dwóch elementów stosu, jest wkładany na stos. Jeśli tworzy zakręt w prawo, punkt jest zdejmowany ze stosu.

Implementacja poszczególnych części algorytmu opisana jest w pliku `ipynb` realizującym laboratorium, sam algorytm realizuje funkcja o nazwie `graham_algorithm`.

1.5. Algorytm Jarvisa

Algorytm Jarvisa buduje otoczkę wypukłą przez wybór punktów „owijających" zbiór w jednym kierunku, aż powróci do punktu początkowego, tworząc zamknięty wielokąt.

Począwszy od punktu początkowego, algorytm wybiera kolejne punkty otoczki wypukłej. W tym celu przegląda każdy inny punkt w zbiorze i sprawdza, który z nich tworzy największy kąt w lewo względem bieżącego punktu otoczki. W przypadku współliniowości wybiera punkt najdalszy.

Realizacja powyższego odbywa się bez użycia funkcji trygonometrycznych. Ustalony zostaje punkt początkowy $s = s_0$ oraz dowolnie wybrany inny punkt q. Następuje iteracja po wszystkich pozostałych punktach $\{p_1, p_2, ..., p_n\}/\{s, q\}$. Jeżeli podczas rozpatrywania punktu p_i, punkt q leży po lewej stronie prostej $|sp_i|$ lub punkt p_i jest współliniowy z punktem q względem s i jednocześnie odległość $\|p_i s\|$ jest większa niż odległość $\|qs\|$, następuje przypisanie $q = p_i$. Po zakończeniu pętli q jest dodawany do zbioru wynikowego otoczki i staje się nowym punktem początkowym, przypisując $s = q$. Powyższe kroki wykonywane są dopóki ponownie nie nastąpi przypisanie $s = s_0$. Teoretyczna złożoność czasowa algorytmu to $O(nk)$, gdzie n jest liczba punktów zbiorze, a k - liczbą punktów należących do otoczki.

Kwalifikacja pozycji punktów względem prostej oraz wyznaczenie punktu początkowego s_0 są wykonywane identycznie jak we wcześniej opisanym *Algorytmie Grahama*. Implementacja poszczególnych części algorytmu opisana jest w pliku `ipynb` realizującym laboratorium, sam algorytm realizuje funkcja o nazwie `jarvis_algorithm`.

2. Dane techniczne

Zadanie zostało przeprowadzone z użyciem narzędzi o następujących parametrach:

- Komputer HP EliteBook 840 G6:
  - System operacyjny: Windows 11 x64
  - Procesor Intel(R) Core(TM) i5-8365U CPU 1.60GHz 1.90 GHz
  - Pamięć RAM: 8GB
- Środowisko: Jupyter Notebook
- Język: Python 3.9.20
- Biblioteki języka: Numpy, Seaborn, Pandas, Matplotlib

3. Wizualizacja działania algorytmów

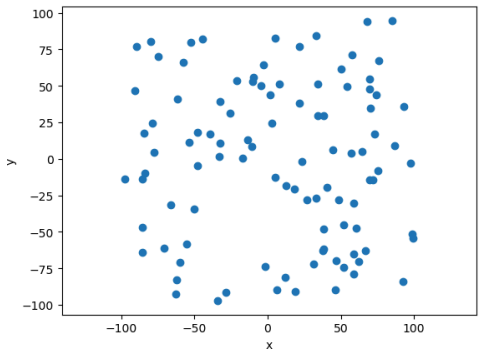
Wizualizacja działania obydwu algorytmów krok po kroku została utworzona z pomocą wizualizatora udostępnionego przez *Koło Naukowe BIT* bazującego na bibliotece `matplotlib`.

Dla każdego punktu na wykresach przyjęto następujące konwencje kolorystyczne:

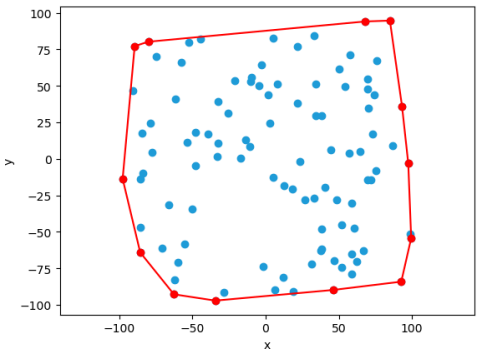
- Niebieski - punkty należące do danego zbioru nie będące częścią otoczki wypukłej
- Czerwony - punkty będące częścią otoczki wypukłej.
- Zielony - punkty aktualnie porównywane podczas wizualizacji algorytmu krok po kroku.

Wykresy Wykres 1, Wykres 3, Wykres 5 oraz Wykres 7 przedstawiają punkty na płaszczyźnie wygenerowane kolejnymi metodami opisanymi w Sekcja 1.2. Wykresy Wykres 2, Wykres 4, Wykres 6, Wykres 8 przedstawiają otoczki wypukłe dla odpowiednich wyżej wymienionych zbiorów wygenerowane za pomocą algorytmu Grahama. Otoczki generowane za pomocą algorytmu Jarvisa dawały wyniki identyczne, więc ich wykresy nie zostały uwzględnione.

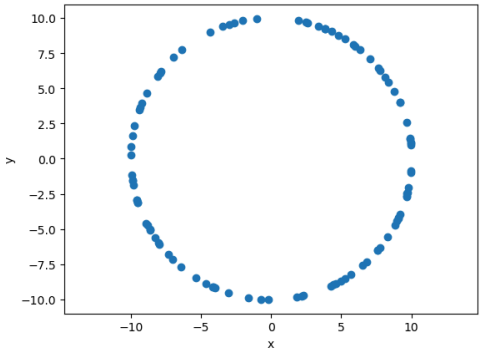
Wizualizacje, przedstawiające działanie algorytmów krok po kroku, znajdują się w plikach `GIF` zapisanych w katalogu `data` oraz dostępne są w środowisku `Jupyter Notebook` zawierającym kod laboratorium.



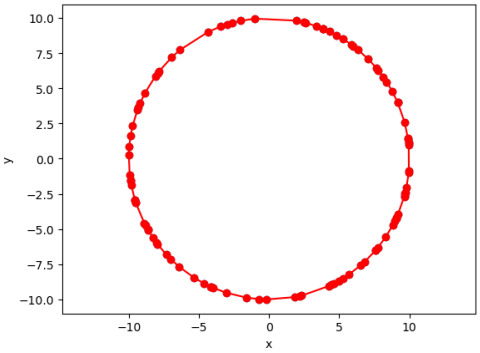
Wykres 1 : Wizualizacja zbioru A



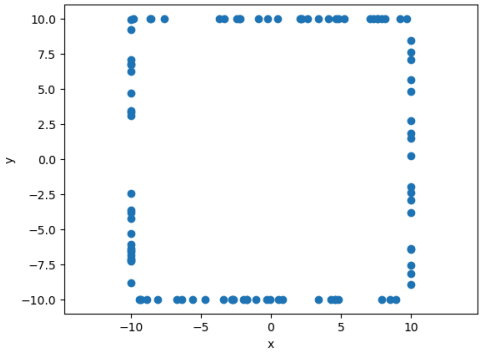
Wykres 2 : Wizualizacja otoczki wypukłej dla zbioru A



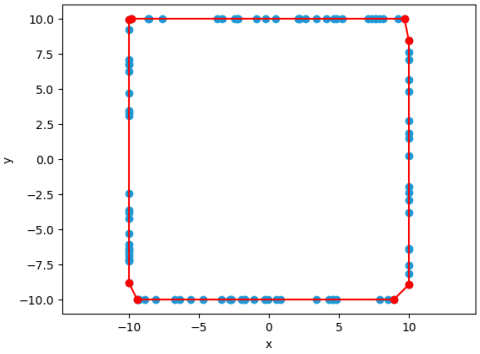
Wykres 3 : Wizualizacja zbioru B



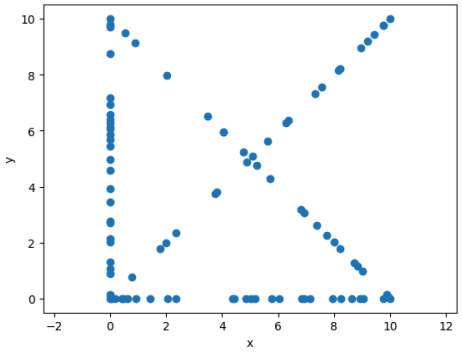
Wykres 4 : Wizualizacja otoczki wypukłej dla zbioru B



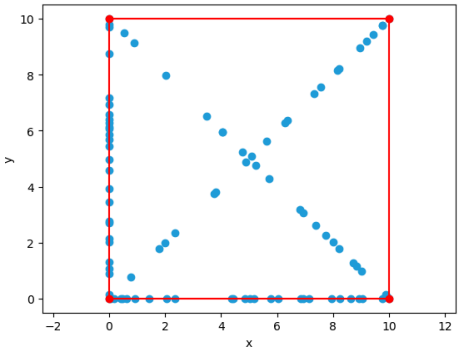
Wykres 5 : Wizualizacja zbioru C



Wykres 6 : Wizualizacja otoczki wypukłej dla zbioru C



Wykres 7 : Wizualizacja zbioru D



Wykres 8 : Wizualizacja otoczki wypukłej dla zbioru D

4. Analiza wyników działania algorytmów

4.1. Analiza niezmodyfikowanych zestawów danych

Dla wszystkich testowanych zbiorów ujętych w Sekcja 1.2 obydwa algorytmy zwracały identyczny zbiór punktów należących do otoczki. Liczba punktów otoczki wypukłej dla poszczególnych zbiorów przedstawiona jest w Tabela 1.

Zbiór punktów	Liczba Punktów Otoczki	
	Algorytm Grahama	Algorytm Jarvisa
Zestaw A	12	12
Zestaw B	100	100
Zestaw C	8	8
Zestaw D	4	4

Tabela 1 : Liczba punktów należących do otoczki po wyznaczeniu daną metodą

Na wykresie *Zbioru A* (Wykres 2) widoczne jest, że otoczka jest poprawna, tworzy wielokąt wypukły i zawarte są w niej wszystkie pozostałe punkty zbioru.

Wszystkie punkty *Zestawu B* (Wykres 4) zostały zakwalifikowane, jako przynależące do otoczki, co w przypadku punktów wygenerowanych za pomocą równania okręgu, jest poprawnym wynikiem.

Wizualizacja *Zestawu C* oraz *Zestawu D* (odpowiednio Wykres 6, Wykres 8) potwierdza poprawne rozpatrywanie punktów współliniowych w algorytmach. W *Zestawie D* do otoczki należą jedynie 4 punkty będące wierzchołkami kwadratu: a, b, c, d, co jest oczekiwanym zachowaniem.

4.2. Analiza zmodyfikowanych zestawów danych

Zbiory A, B, C, D zostały dodatkowo zmodyfikowane względem parametrów takich jak liczba oraz zakres punktów, aby dokonać testów czasowych w celu porównania działania algorytmów. Wyniki znajdują się w poniższych tabelach (Tabela 2, Tabela 3, Tabela 4, Tabela 5).

Punkty zbiorów opisane w Tabela 2 (zmodyfikowany *Zestaw A*) dla zadanej liczby punktów n zostały wygenerowane jednostajnie w przedziale $[-n, n]$. Dla losowego rozkładu punktów różnice pomiędzy czasem działania dla danych testowych są nieznaczne i odpowiadają ich teoretycznym złożonościom czasowym.

Punkty zbiorów opisane w Tabela 3 (zmodyfikowany *Zestaw B*) dla zadanej liczby punktów n zostały wygenerowane jednostajnie używając równania okręgu o środku $O = (0, 0)$ oraz promieniu $R = n$. Przez to, że wszystkie punkty w tych zbiorach należą do otoczki wypukłej, algorytm Jarvisa jest istotnie wolniejszy od algorytmu Grahama, przez osiąganie złożoności rzędu $O(n^2)$.

Punkty zbiorów opisane w Tabela 4 (zmodyfikowany *Zestaw C*) dla zadanej liczby punktów n zostały wygenerowane jednostajnie na obwodzie prostokąta o wierzchołkach a, b, c, d o współrzędnych $a = \left(-\frac{n}{10}, -\frac{n}{10}\right), b = \left(\frac{n}{10}, -\frac{n}{10}\right), c = \left(\frac{n}{10}, \frac{n}{10}\right), d = \left(-\frac{n}{10}, \frac{n}{10}\right)$. Algorytm Jarvisa wykazuje się w tych przypadkach czasem działania rzędu dwukrotnie niższego względem algorytmu Grahama, poprzez dużo mniejszą względem n liczby punktów przynależących do otoczki, co sprawia, że k zawarte w teoretycznej złożoności $O(n \cdot k)$ algorytmu można potraktować, jako stałą, co daje $O(n \cdot \text{const}) = O(n)$. W porównaniu z czasem działania algorytmu Grahama rzędu $O(n \log n)$ jest to znacząca różnica, co potwierdzają wyniki testów czasowych.

Punkty zbiorów opisane w Tabela 5 (zmodyfikowany *Zestaw D*) dla zadanej liczby punktów $n_{\text{przekątna}}$ oraz $n_{\text{obwód}}$ zostały wygenerowane jednostajnie na obwodzie i przekątnych prostokąta o wierzchołkach a, b, c, d o współrzędnych $a = \left(-\frac{4n_{\text{obwód}}}{10}, -\frac{4n_{\text{obwód}}}{10}\right), b = \left(\frac{4n_{\text{obwód}}}{10}, -\frac{4n_{\text{obwód}}}{10}\right), c = \left(\frac{4n_{\text{obwód}}}{10}, \frac{4n_{\text{obwód}}}{10}\right), d = \left(-\frac{4n_{\text{obwód}}}{10}, \frac{4n_{\text{obwód}}}{10}\right)$.

Rozkład punktów prezentuje się w następujący sposób:

- $n_{\text{obwód}}$ punktów leżących na każdym z boków $|da|$ i $|ab|$
- $n_{\text{przekątna}}$ punktów leżących na każdej z przekątnych $|ac|$ i $|bd|$
- 4 punkty leżące na wierzchołkach a, b, c, d

Algorytm Jarvisa wykazuje się tutaj ponownie czasami rzędu dwukrotnie niższego względem algorytmu Grahama, poprzez fakt, że wartość parametru k wynosi $k = 4$, co sprawia, że złożoność algorytmu Jarvisa wynosi $O(4 \cdot n) = O(n)$.

Zmodyfikowany zbiór A		
Liczba punktów	**Czas działania [s]**	
	Algorytm Grahama	**Algorytm Jarvisa**
100	0.001417	0.000941
500	0.005104	0.006069
1000	0.014185	0.012577
5000	0.068664	0.078718
10000	0.137890	0.128786

Tabela 2 : Porównanie czasu działania algorytmów dla *Zbioru A*

Zmodyfikowany zbiór B		
Liczba punktów	**Czas działania [s]**	
	Algorytm Grahama	**Algorytm Jarvisa**
100	0.000761	0.007921
500	0.004169	0.194682
1000	0.009140	0.691579
5000	0.058993	18.95591
10000	0.115255	72.50078

Tabela 3 : Porównanie czasu działania algorytmów dla *Zbioru B*

Zmodyfikowany zbiór C		
Liczba punktów	**Czas działania [s]**	
	Algorytm Grahama	**Algorytm Jarvisa**
100	0.000585	0.000357
500	0.003076	0.001778
1000	0.006788	0.003682
5000	0.038673	0.017918
10000	0.082529	0.036022

Tabela 4 : Porównanie czasu działania algorytmów dla *Zbioru C*

.

Zmodyfikowany zbiór D			
Liczba punktów na obwodzie	**Liczba punktów na przekątnych**	**Czas działania [s]**	
		Algorytm Grahama	**Algorytm Jarvisa**
25	25	0.000485	0.000373
125	125	0.002265	0.001813
250	250	0.010692	0.003523
1250	1250	0.025113	0.017138
2500	2500	0.051468	0.034523

Tabela 5 : Porównanie czasu działania algorytmów dla *Zbioru D*

5. Wnioski

- **Wizualizacja i interpretacja wyników**
  Zebrane dane i wizualizacje dla każdego ze zbiorów potwierdzają, że algorytmy działają poprawnie, wyznaczając minimalne otoczki wypukłe, w których zawarte są wszystkie punkty zbiorów. Wyniki te ułatwiają wizualną weryfikację poprawności algorytmów, zwłaszcza przy interpretacji otoczek dla punktów współliniowych, jak w *Zestawie D*.
- **Złożoność obliczeniowa i wydajność**
  Analiza czasów działania wykazała, że algorytm Grahama, o złożoności teoretycznej $O(n \log n)$, jest bardziej efektywny przy dużych zbiorach losowych, takich jak *Zestaw A*, w którym punkty są równomiernie rozmieszczone. Algorytm Jarvisa, ze złożonością $O(nk)$, okazał się natomiast bardziej wydajny dla przypadków z dużą liczbą współliniowych punktów względem punktu startowego (np. *Zestaw C* i *Zestaw D*), gdzie można powiedzieć, że złożoność sprowadza się do $O(n)$, ponieważ liczba punktów należących do zbioru otoczki jest istotnie mniejsza niż całkowita liczba punktów.
- **Zależność od zadanego zbioru**
  Wyniki czasowe działania algorytmów zmieniają się znacząco w zależności od rozkładu punktów. W przypadku punktów tworzących okrąg (*Zestaw B*), algorytm Jarvisa jest zdecydowanie mniej efektywny niż algorytm Grahama, co wynika z faktu, że każdy punkt należy do otoczki, przez co algorytm Jarvisa ma złożoność kwadratową $O(n^2)$ (Tabela 3). Z kolei algorytm Grahama działa znacząco wolniej od algorytmu Jarvisa w przypadku, gdy liczba punktów należących do zbioru otoczki jest mała (Tabela 4,Tabela 5)