

# Algorytmy Zaawansowane - POLE

Piotr Izert, Łukasz Dragan

24 marca 2016

## Spis treści

<b>1</b>	<b>Przedstawienie problemu</b>	<b>3</b>
1.1	Treść zadania . . . . .	3
<b>2</b>	<b>Opis rozwiązania</b>	<b>3</b>
2.1	Pole wielokąta . . . . .	3
2.2	Zawieranie punktu w wielokącie . . . . .	3
2.3	Sprawdzanie, czy wielokąt jest prosty . . . . .	5
<b>3</b>	<b>Analiza poprawności</b>	<b>5</b>
3.1	Pole wielokąta . . . . .	5
3.2	Zawieranie punktu w wielokącie . . . . .	6
3.3	Sprawdzanie, czy wielokąt jest prosty . . . . .	6
<b>4</b>	<b>Opis wejścia/wyjścia</b>	<b>6</b>
4.1	Wejście . . . . .	6
4.2	Wyjście . . . . .	6

# 1 Przedstawienie problemu

## 1.1 Treść zadania

*Zaprojektować i zaimplementować algorytm, który w czasie liniowym względem  $n$  oblicza pole  $n$ -wierzchołkowego prostego wielokąta oraz sprawdza, czy podany punkt leży wewnątrz tego wielokąta. Program powinien zawierać procedurę sprawdzającą, czy dany wielokąt jest prosty.*

# 2 Opis rozwiązania

## 2.1 Pole wielokąta

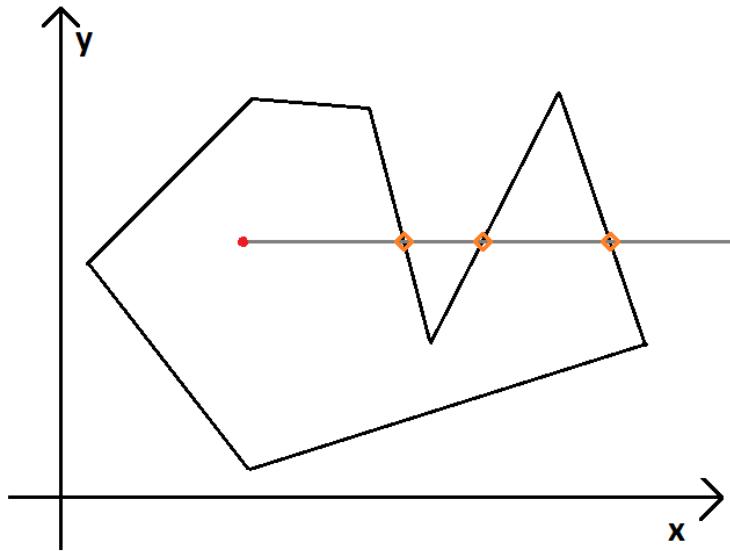
W celu obliczenia pola powierzchni wielokąta prostego stosujemy algorytm wykorzystujący tzw. wzór trapezowy Gaussa  $S = \frac{1}{2} \sum_{i=1}^n (x_i + x_{i+1}) * (y_{i+1} - y_i)$ , gdzie  $S$  to pole powierzchni wielokąta,  $x_i, y_i$  dla  $i = 1 \dots n$  to współrzędne kolejnych wierzchołków wielokąta, a  $n$  to liczba wierzchołków wielokąta. Zakładamy, że  $x_{n+1} = x_1$  oraz  $y_{n+1} = y_1$ .

### Algorytm

1.  $area = 0$
2.  $j = n$
3. dla  $i = 1$  do  $n$  wykonaj:
4.      $area = area + (x_j + x_i) * (y_j - y_i)$
5.      $j = i$
6.  $area = area/2$
7. RETURN  $area$

## 2.2 Zawieranie punktu w wielokącie

Sprawdzenie, czy punkt zawiera się w wielokącie jest realizowane przez zliczenie przecięć półprostej zaczynającej się w badanym punkcie z bokami wielokąta. Jeżeli liczba przecięć jest nieparzysta, to znaczy, że punkt leży wewnątrz wielokąta. W przeciwnym razie punkt leży na zewnątrz wielokąta.



Rysunek 1: Sprawdzanie zawierania się punktu w wielokącie

W zaimplementowanym algorytmie półprosta prowadzona jest równolegle do osi  $X$  w kierunku rosnących wartości. Oznaczmy przez  $x_i, y_i$  dla  $i = 1 \dots n$  współrzędne kolejnych wierzchołków wielokąta, a przez  $X$  i  $Y$  współrzędne badanego punktu.

#### Algorytm

1. inside = false
2. dla  $i = 1$  do  $n$  wykonaj:
  3. sprawdź, czy krawędź  $i, i + 1$  przecina prostą  $y = Y$
  4. jeżeli tak, to sprawdź, czy współrzędna  $x$  punktu przecięcia jest większa od  $X$
  5. jeżeli tak, to inside = !inside

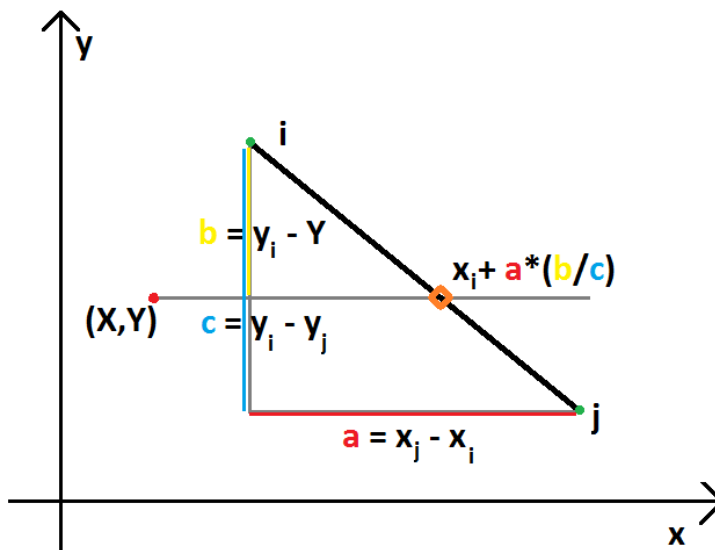
Sprawdzenie z punktu 3. realizowane jest w następujący sposób - sprawdzane jest, czy współrzędne  $y$  obydwu końców krawędzi leżą na różnych półpłaszczyznach wyznaczanych przez prostą - warunek:

$$(y_i > Y) \neq (y_{i+1} > Y)$$

Sprawdzenie z punktu 4. realizowane jest przez znalezienie punktu przecięcia krawędzi z prostą i porównanie go do wartości  $X$ :

$$X < x_i + (x_j - x_i) \frac{y_i - Y}{y_i - y_j}$$

Przykład zasady obliczania punktu przecięcia został przedstawiony na rysunku 2.



Rysunek 2: Obliczanie punktu przecięcia krawędzi z prostą

## 2.3 Sprawdzanie, czy wielokąt jest prosty

# 3 Analiza poprawności

## 3.1 Pole wielokąta

### Poprawność

W pierwszej iteracji pętli z kroku 3.  $area = (x_n + x_1) * (y_n - y_1) (= (x_n + x_{n+1}) * (y_n - y_{n+1}))$ . W kolejnych iteracjach  $j$  jest zawsze o 1 mniejsze od  $i$ , stąd do  $area$  dodawana jest wartość  $(x_j + x_{j+1}) * (y_j - y_{j+1})$  dla  $j = 1 \dots n - 1$ . Stąd ostatecznie  $area = (x_n + x_{n+1}) * (y_n - y_{n+1}) + (x_1 + x_2) * (y_1 - y_2) + \dots + (x_{n-1} + x_n) * (y_{n-1} - y_n) = \sum_{i=1}^n (x_i + x_{i+1}) * (y_{i+1} - y_i)$ . Po podzieleniu  $area$  przez 2 otrzymujemy wzór Gaussa na pole powierzchni wielokąta.

### Interpretacja geometryczna

Sumę we wzorze na pole wielokąta można interpretować jako sumę pól trapezów. Zostało to zilustrowane na rysunku ??.

### **Złożoność czasowa**

Algorytm działa w czasie  $O(n)$ , gdyż główna pętla algorytmu wykonuje dokładnie  $n$  kroków.

## **3.2 Zawieranie punktu w wielokącie**

## **3.3 Sprawdzanie, czy wielokąt jest prosty**

# **4 Opis wejścia/wyjścia**

## **4.1 Wejście**

Program domyślnie jako wejście przyjmuje zawartość pliku „in.txt”, który powinien zawierać w kolejnych liniach:

1. Dane w postaci  $x_1 y_1 \dots x_n y_n$ , gdzie  $(x_i, y_i) \in \mathbb{R}^2$  dla  $i = 1, 2, \dots, n$  to współrzędne kolejnych punktów a  $n$  to liczba wierzchołków wielokąta.
2. Dane postaci  $x y$ , gdzie  $(x, y) \in \mathbb{R}^2$  będące współrzędnymi punktu, którego zawieranie w wielokącie ma zostać sprawdzone.

### **Przykładowe wejście**

```
344,8 91,2 68,8 121,6 352,8 218,4 448 114,4
288,8 136
```

## **4.2 Wyjście**

Rezultat działania programu zapisywany jest w pliku „out.txt” w postaci  $S \text{ } Ans$  gdzie  $S$  to pole powierzchni wielokąta a  $Ans \in \{„TAK”, „NIE”\}$  to odpowiedź na pytanie, czy dany punkt jest zawarty w wielokącie. W przypadku, gdy dany wielokąt nie jest prosty, rezultatem działania programu jest NOT SIMPLE. Jeżeli dane podane na wejściu są niepoprawne, program zapisze do pliku BAD INPUT.

### **Przykładowe wyjście**

```
1243,33 TAK
```