Algorytmy Zaawansowane - POLE

Piotr Izert, Łukasz Dragan

29 marca 2016

Spis treści

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

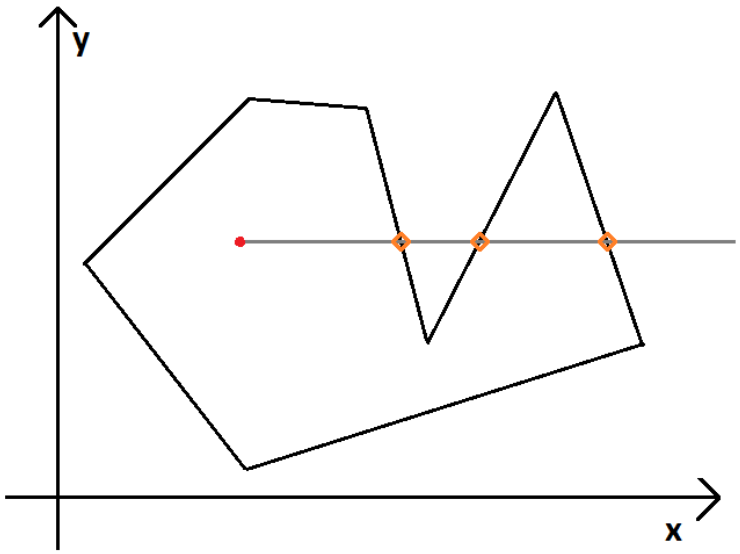
W celu obliczenia pola powierzchni wielokąta prostego stosujemy algorytm wykorzystujący tzw. wzór trapezowy Gaussa $S = \frac{1}{2}\sum_{i=1}^{n}(x_i + x_{i+1}) * (y_{i+1} - y_i)$, gdzie S to pole powierzchni wielokąta, x_i, y_i dla $i = 1...n$ to współrzędne kolejnych wierzchołków wielokąta, a n to 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. $\quad area = area + (x_j + x_i) * (y_j - y_i)$
5. $\quad j = i$
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* $i = 1...n$ 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

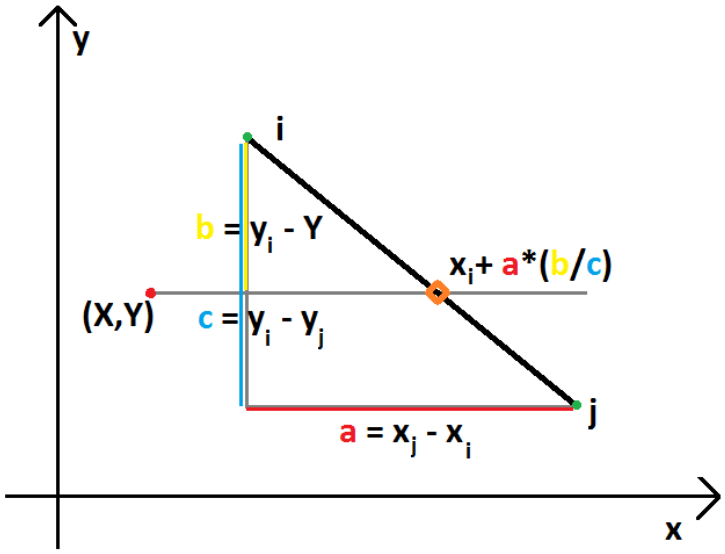
Sprawdzenie z punktu 3. realizowane jest w następujący sposób - sprawdzane jest, czy współrzędne y obydwu końców krawędzi leżą na różnych półpłaszczyznach wyznaczanych przez prostą - warunek:

$$(y_i > Y)! = (y_{i+1} > Y)$$

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

Algorytmy prezentowane w niniejszym mają zastosowanie jedynie, gdy dany wielokąt jest prosty.

Wielokąt prosty: wielokąt, którego boki tworzą zamkniętą łamaną, a dwa jego boki mają punkt wspólny tylko, gdy są sąsiadami. (Wikipedia)

Algorytm
W celu sprawdzenia, czy wielokąt jest prosty dokonujemy sprawdzenia:

1. czy istnieje przecięcie pomiędzy dowolnymi dwiema krawędziami e_i i e_j dla $|i-j| > 1,\ i,j = 1,...,n+1$ danego wielokąta. Przyjmujemy, iż $e_1 = e_{n+1},\ n$ - liczba krawędzi wielokąta.
2. W przypadku $|i-j| = 1$ (kolejne krawędzie wielokąta): czy jedna z krawędzi zawiera się w drugiej.

Algorytm działa na zasadzie sprawdzenia każdej możliwej pary krawędzi $e_i,\ e_j\ dla\ j > i, i, j = 1, ..., n+1$. Jeżeli dla danej pary spełniony jest dowolny z warunków (1,2), wielokąt nie 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 $j = 1...n-1$. Stąd ostatecznie $area = (x_n + x_{n+1}) * (y_n - y_{n+1}) + (x_1 + x_2) * (y_1 - y_2) + ... + (x_{n-1} + x_n) * (y_{n-1} - y_n) = \sum_{i=1}^{n} (x_i + x_{i+1}) * (y_{i+1} - y_i)$. 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

Złożoność czasowa
Algorytm działa w czasie $O(n^2)$, gdyż dla krawędzi i wykonuje sprawdzenie warunków $n-i$ razy. Liczba wywołań funkcji sprawdzenia warunków $S = \sum_{i=1}^{n} (n-i) = \frac{n(n-1)}{2}$

4 Opis wejścia/wyjścia

4.1 Wejście

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

1. Dane postaci x_1 y_1 ... x_n y_n, gdzie $(x_i, y_i) \in \Re^2$ dla $i = 1, 2, ..., n$ to współrzędne kolejnych punktów a n to liczba wierzchołków wielokąta.
2. Dane postaci x y, gdzie $(x, y) \in \Re^2$ 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 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
```