Dokumentacja

Program znajdujący sumę i iloczyn dwóch wielokątów 2D.

Jan Chyb

1. Założenia programu:

Program przyjmuje na wejściu dwa zbiory prostych tworzących wielokąty. Następnie, korzystając z algorytmu Greinera-Hormanna, znajduje wielokąty będące sumą lub iloczynem dostarczonych figur. Na wyjściu zwraca zbiór list prostych tworzących wielokąty, które z kolei tworzą figury będące wynikiem operacji. Zostają one także wyświetlone.

2. Uruchamianie i obsługa programu:

Program należy uruchomić przy użyciu notebooków Jupytera. W pliku BooleanPolygon.ipynb trzeba uruchomić wszystkie komórki oznaczone nagłówkiem konfiguracja. Następnie, w sekcji “Wprowadzenie Danych”, należy uruchomić pierwszą komórkę. Na wyświetlonym układzie współrzędnych wprowadzamy ręcznie dwa wielokąty. Uruchomienie kodu poniżej spowoduje zaakceptowanie wprowadzonych figur.

Punkty wielokątów można wprowadzić na *dowolny* sposób, nie muszą być one wstawiane w żadnej określonej kolejności. W szczególności dozwolone jest wprowadzanie wielokątów tzw. *samoprzecinających*. Program nie obsługuje jednak przypadku zdegenerowanego, kiedy dwa wielokąty mają wierzchołki o równych współrzędnych.

Aby uruchomić algorytm uruchamiamy komórkę podpisaną “Rozwiązanie - Suma Wielokątów”, lub “Rozwiązanie - Iloczyn Wielokątów”. Wyświetlą się wtedy na niebiesko wprowadzone figury, a na czerwono szukana figura, będąca odpowiednio sumą, lub iloczynem.

Uwaga! Wynikiem operacji na samoprzecinających się wielokątach mogą być samoprzecinające się wielokąty.

3. Graficzne przedstawienie działania algorytmu:

Aby móc zobaczyć kolejne kroki algorytmu, należy w wywoływanej funkcji greinerHormann ustawić argument toDraw na True. Trzeba mieć na uwadzę, że program stanie się wtedy znacznie wolniejszy. Aby przechodzić przez kolejne kroki algorytmu korzysta się z przycisków “Następny” i “Poprzedni”:
Objaśnienie kolejnych scen:

- pierwsza scena na tle wczytanych wielokątów pokazuje znalezione przecięcia;
- następne dwie sceny obrazują klasyfikację wierzchołków, pierwsza pierwszego wczytanego wielokąta, druga drugiego;
    - niebieskimi strzałkami oznaczone są wierzchołki “zewnętrzne”;
    - czerwonymi kropkami oznaczone są wierzchołki “wewnętrzne”;
- kolejne sceny, z wyłączeniem ostatniej, pokazują część 3. algorytmu, czyli budowanie szukanych wielokątów;

Ostatnią sceną jest wynik działania algorytmu. Jeżeli toDraw jest równe False to jest to jedyna scena. Jeżeli chcemy szybko przejść z pierwszej do ostatniej sceny (np. aby zobaczyć od razu wynik algorytmu) wystarczy wcisnąć przycisk “poprzedni”

4. Objaśnienie wybranych procedur:

Procedura *greinerHormann (fig1Lines, fig2Lines, oparation, toDraw)*

- fig1Lines, fig2Lines: listy kolejnych odcinków pierwszego i drugiego wielokąta
- operation: operacja do wykonania na wielokątach, dozwolone jedynie '|' - suma i '&' - iloczyn
- toDraw: bool, specyfikuje, czy rysować kolejne kroki algorytmu, czy nie (zawsze jest rysowany wynik algorytmu)

Zwraca foundPolygons, scenes

- foundPolygons: lista wielokątów składających się na wynik
- scenes: lista scen pokazujących graficznie kolejne kroki algorytmu, jeżeli toDraw jest równe False, to jest to jedynie wynik działania operacji

Działa w czasie O(mn).

Powyższa jak i poniższe procedury realizowane są z użyciem obiektów klasy *Vertex*, które tworzą strukturę wielokąta. W praktyce jest ona dwiema połączonymi listami odsyłaczowymi dwukierunkowymi.

Prodecura *convert (lines)*

- lines: lista współrzędnych odcinków, które połączone po kolei tworzą wielokąt

Zwraca pierwszy element listy Vertex, na którą lines zostało przekształcone.

Procedura *evenOddRule (vertexFig)*

- vertexFig: element struktury Vertex, reprezentującej zamknięty wielokąt. Musi mieć już poprawnie oznaczone pole inside, określające, czy punkt jest wewnątrz drugiego wielokąta

Sprawdza, czy kolejne wierzchołki Vertex są wewnątrz, czy zewnątrz drugiego wielokąta.
Zwraca dwie listy odcinków, sklasyfikowanych w ten sam sposób, w celu graficznego przedstawienia działania algorytmu.

Procedura *findAndAddAllIntersections (vertexFig1, vertexFig2)*

- vertexFig1: pierwszy wierzchołek listy Vertex pierwszego wielokąta
- vertexFig2: pierwszy wierzchołek listy Vertex drugiego wielokąta

Znajduje przecięcia i dodaje wierzchołki Vertex w miejsca przecięć do struktury wierzchołków.
Zwraca listę wierzchołków będących przecięciami, potrzebną do 3. części algorytmu.
Dla każdego boku pierwszego wielokąta bada przecięcie z każdym bokiem drugiego, stąd złożoność O(mn), gdzie m,n to liczby boków wielokąta pierwszego i drugiego.
Po znalezieniu punktów przecięć, są one wstawiane do indywidualnych list punktów każdego wierzchołka bazowego, sortowane quicksortem względem kolejności wystąpienia na odcinku, i po kolei wkładane (O((m+n)log(m+n))).

Procedury *checkIntersection, intersect, getIntersectPoint, insertIntersection*, są funkcjami pomocniczymi do powyższej i realizują znajdowanie i wkładanie wierzchołka do listy w czasie stałym.

Procedura *isInside (point, lines)*

- point: współrzędne punktu, który badamy
- lines: lista odcinków wielokąta, względem którego badamy, czy punkt point jest wewnątrz, czy na zewnątrz

Sprawdza, czy punkt leży wewnątrz danego wielokąta. Realizuje to poprzez policzenie liczby przecięć półprostej równoległej do osi y, rozpoczynającej się w punkcie point, z odcinkami lines. Jeżeli liczba ta jest parzysta to punkt jest na zewnątrz, wpp wewnątrz.

5. Złożoność:

Najbardziej kosztowną częścią algorytmu jest szukanie punktów przecięć odcinków wykonywane w O(mn). Dalsze części są już nie dłuższe niż liniowe.
m - liczba wierzchołków pierwszego wielokąta
n - liczba wierzchołków drugiego wielokąta

6. Bibliografia:

Internet 1: https://www.inf.usi.ch/hormann/papers/Greiner.1998.ECO.pdf

Internet 2: http://davis.wpi.edu/~matt/courses/clipping/