Univerzita Karlova
Přírodovědecká fakulta

Algoritmy počítačové kartografie

Úloha 1: Geometrické vyhledávání bodu

Adam Kulich, Markéta Růžičková, Eliška Sieglová

N-GKDPZ

Praha 2023

1 Zadání

Vstup: *Souvislá polygonová mapa n polygonů* $\{P_1,\ldots,P_n\}$, *analyzovaný bod q.*
Výstup: $P_i,\ q \in P_i$.
Nad polygonovou mapou implementujete Ray Crossing Algorithm pro geometrické vyhledání incidujícího polygonu obsahující zadaný bod q.
Nalezený polygon graficky zvýrazněte vhodným způsobem (např. vyplněním, šrafováním, blikáním). Grafické rozhraní vytvořte s využitím frameworku QT.
Pro generování nekonvexních polygonů můžete navrhnout vlastní algoritmus či použít existující geografická data (např. mapa evropských států).
Polygony budou načítány z textového souboru ve Vámi zvoleném formátu. Pro datovou repretentaci jednotlivých polygonů použijte špagetový model.

Hodnocení:

Krok	**Hodnocení**
Detekce polohy bodu rozlišující stavy uvnitř, vně, na hranici polygonu.	10b
Analýza polohy bod (uvnitř/vně) metodou Winding Number	*+5b*
Ošetření singulárního případu u Ray Algorithm: bod leží na hraně polygonu.	*+5b*
Ošetření singulárního případu u obou algoritmů: bod je totožný s vrcholem jednoho či více polygonů.	*+2b*
Zvýraznění všech polygonů pro oba výše uvedené singulární případy.	*+3b*
Max celkem:	**25b**

Čas zpracování: 1 týden.

2 Údaje o bonusových úlohách + body

Algorithm 1 Body

if povinná úloha && všechny bonusové úlohy && psané v LaTeXu **then**
    30 bodů?
end if

3 Popis a rozbor problému + vzorce

3.1 Úvod do problému

V této úloze se zaměřujeme na vyřešení problém zvaného *Point in Polygon* či *Point Location Problem*, tedy zjišťujeme, zda se zadaný bod nachází uvnitř, na hraně, nebo vně polygonu. S tímto problémem se často setkáme ve výpočetní geometrii a zejména v mnoha geoinformačních úlohách potřebujeme naleznout jeho řešení. Jedním příkladem může být, že máme zadaný bod, a zadané polygony regionů (např. států), a snažíme se přijít na to, ve kterém státě bod leží.

Princip řešení problému *Point in Polygon* se skládá ze dvou procedur: lokální a globální. Lokální procedura se zabývá polohou řešeného bodu q vůči jednomu konkrétnímu polygonu, a jejím výsledkem je, že bod leží uvnitř, vně, či na hraně tohoto polygonu. Do globální procedury poté vstupují všechny polygony v datasetu, a jejím výstupem je, že bod leží uvnitř jednoho mnohoúhelníku, vně všech polygonů, či na hraně/v uzlu dvou či více polygonů.

Techniky řešení tohoto problému se dělí na ty, které problém řeší u konvexních polygonů, a ty, které problém řeší u nekonvexních polygonů. Pro konvexní polygony se úloha Point in Polygon řeší algoritmy Ray Crossing Method (paprsková metoda, více v další části), či Half-Plane Test.

Half-Plane test porovnává řešený bod q s hranou polygonu, kdy řeší, zda se pokaždé nachází ve stejné polorovině. Pokud se pokaždé bod nachází ve stejné polorovině, pak leží uvnitř polygonu. Toto se dá aplikovat například na triangulační síť.

Řešení úlohy u konvexních polygonů je značně jednodušší, nicméně v praxi se s setkáme spíše s polygony nekonvexními, kterými jsme se zabývali i v této úloze. Pro nalezení řešení pro nekonvexní mnohoúhelníky se využívají například algoritmy Ray Algorithm (paprskový algoritmus), či Winding Number Algorithm (česky metoda ovíjení).

3.2 Ray Algorithm

V případě paprskového algoritmu se ze zadaného bodu q vede polopřímka r (analogie paprsku, anglicky ray), často rovnoběžná s jednou z hlavních os, a počítá se, kolikrát protne hranu polygonu. Tyto průsečíky značíme písmenem k. Pokud je počet průsečíků sudý, bod leží vně polygonu, pokud je počet průsečíků lichý, bod leží uvnitř.

$$k \begin{cases} 1 & q \in P, \\ 0 & q \notin P. \end{cases} \tag{1}$$

Tento algoritmus je o poznání rychlejší než Winding Number (ten bude vysvětlen později), ale jeho problémem jsou případy, kdy bod leží buď na hraně, či ve vertextu polygonu. Tyto singularity se v kódu musí ošetřit zvlášť.
První zmínka o tomto algoritmu je z roku 1962 (Shimrat, 1962), a dnes je stále jedním z nejpoužívanějších algoritmů pro řešení problému point in polygon.

3.3 Winding Number

Metoda Winding Number (česky metoda ovíjení) spočívá ve spočtení počtu ovinutí polygonu okolo bodu pomocí sčítání úhlů. Hodnota winding number (značené Ω) je suma všech rotací měřená proti směru hodinových ručiček a udává vztah bodu a polygonu. Hodnoty Ω mohou nabývat hodnot násobků 2π, nebo 0. V případě, že je hodnota winding number nenulová, bod leží uvnitř polygonu.

$$\Omega(q,P) = \begin{cases} 1 & q \in P, \\ 0 & q \notin P. \end{cases} \tag{2}$$

Úhel Ω je počítán na základě úhlu mezi dvěma směrovými vektory $\vec{u}$ a $\vec{v}$, kdy $\vec{u}$ spojuje dva body hrany, a $\vec{v}$ spojuje bod q s prvním bodem hrany.

$$\vec{u} = (x_{p_{i+1}} - x_{p_i}, y_{p_{i+1}} - y_{p_i}) \tag{3}$$

$$\vec{v} = (x_q - x_{p_i}, y_q - y_{p_i}) \tag{4}$$

Hodnota Ω je závislá i na směru rotace. Úhly orientované proti směru hodinových ručiček jsou počítány záporně, úhly orientované po směru hodinových ručiček zase kladně. Toho se prakticky docílí tak, že přímka $p(p_i, p_{i+1})$, tedy přímka procházející hranou polygonu, dělí rovinu σ na poloroviny σ_l (levá polorovina) a σ_r. Pokud bod q leží v polorovině σ_l, hodnota se přičítá, pokud v polorovině σ_r, hodnota se odčítá. Proto když procházíme body proti směru hodinových ručiček, Ω nám vyjde jako kladné číslo.

$$\Omega(q,P) = \frac{1}{2\pi} \sum_{i=1}^{n} \omega(p_i, q, p_{i+1}) \tag{5}$$

Nevýhodou tohoto algoritmu je jeho složitost zaprvé kvůli nutnosti předzpracování dat, a zadruhé kvůli časové náročnosti opakovaného výpočtu úhlu. Také kvůli práci s desetinnými čísly (při výpočtu úhlů) je možné dojít k chybě kvůli zaokrouhlování. Také je potřeba ošetřit některé singulární případy, například když bod leží ve vertexu polygonu, nebo když jej počítáme nad nejednoduchými polygony.

4 Popisy algoritmů formálním jazykem

V této kapitole jsou vloženy pseudokódy zpracovávaných algoritmů.

Algorithm 2 Ray Algorithm

```
for každý polygon do
    inicializace k = 0
    posun souřadnicového systému do bodu q
    for každá hrana polygonu do
        if y bodu q leží mezi y-souřadnicemi dvou bodů hrany then
            if osa x protíná hranu then
                k = k + 1
            end if
        end if
    end for
    if počet hran není dělitelný dvěma then
        bod leží uvnitř polygonu
    else if počet hran je dělitelný dvěma then
        bod leží vně polygonu
    end if
end for
```

Algorithm 3 Ray Algorithm: ošetření singularit

```
for každý polygon do
    inicializace kr = 0 a kl = 0                ▷ průsečíky napravo a nalevo od bodu q
    posun souřadnicového systému do bodu q
    for každá hrana polygonu do
        if jeden z bodů hrany je totožný s q then
            bod leží na vertexu polygonu
        end if
        if y bodu q leží mezi y-souřadnicemi dvou bodů hrany then
            spočti průsečík k
            if průsečík k je napravo od bodu q then
                kr = kr + 1
            else if průsečík k je nalevo od bodu q then
                kl = kl + 1
            else if průsečík je v bodě q then
                bod leží na hraně
            end if
        end if
    end for
    if počet hran není dělitelný dvěma then
        bod leží uvnitř polygonu
    else if počet hran je dělitelný dvěma then
        bod leží vně polygonu
    end if
end for
```

Algorithm 4 Winding Number Algorithm

```
for každý polygon do
    inicializace Ω = 0
    for každá hrana polygonu do                ▷ hranou je myšlená dvojice bodů p_i, p_{i+1}
        urči úhel ω svíraný vektory u = (x_{p_{i+1}} − x_{p_i}, y_{p_{i+1}} − y_{p_i}) a v = (x_q − x_{p_i}, y_q − y_{p_i})
        if q leží v σ_l then
            Ω = Ω + ω
        else
            bod leží v pravé polorovině, Ω = Ω − ω
        end if
        if |Ω| > 2π then
            bod leží v polygonu
        else
            bod neleží v polygonu
        end if
    end for
end for
```

5 Problematické situace a jejich rozbor + ošetření v kódu

5.1 Třída Algorithms

V této třídě jsou definovány metody getPolygonPositionR a getPolygonPositionW. Funkce getPolygonPositionR vrací zda se bod nachází vně/uvnitř/na hraně/ve vertexu polygonu pomocí paprskového algoritmu, funkce getPolygonPositionW pak to samé s využitím metody ovíjení.

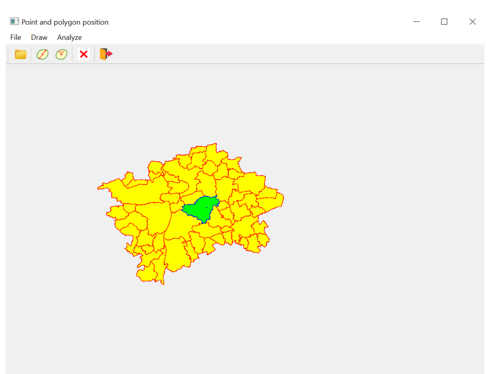
6 Vstupní data, formát vstupních dat, popis

Vstupní data jsou data pražských městských částí ve formátu shapefile.

7 Výstupní data, formát výstupních dat, popis

Výstupem programu je vykreslení všech polygonů v dataset a bodu, který určujeme. Program v grafickém rozhraní QT zvýrazní polygon, ve kterém se bod nachází.

8 Printscreen vytvořené aplikace



9 Dokumentace: popis tříd, datových položek, jednotlivých metod

10 Závěr, možné ši neřešené úlohy, námět na vylepšení

V rámci první úlohy z předmětu Algoritmy počítačové kartografie byly řešen problém Point in Polygon pomocí paprskového algoritmu a metody ovíjení. Kód byl psán v jazyce Python v prostředí PyCharm s využitím grafického rozhraní QT. V rámci algoritmů byly ošetřeny singulární případy, kdy bod leží na hraně polygonu či na jednom z vertexů.

11 Seznam literatury

Bayer, Tomáš. 2023. "Point Location Problem." Praha, February 23. `https://web.natur.cuni.cz/~bayertom/images/courses/Adk/adk3_new.pdf`.

Heckbert, Paul S. 1994. Graphics Gems IV. Morgan Kaufmann.

Huang, Chong-Wei, and Tian-Yuan Shih. 1997. "On the Complexity of Point-in-Polygon Algorithms." Computers & Geosciences 23 (1): 109–18. `https://doi.org/10.1016/S0098-3004(96)00071-4`.

Ye, Yuan, Fan Guangrui, and Ou Shiqi. 2013. "An Algorithm for Judging Points Inside or Outside a Polygon." In 2013 Seventh International Conference on Image and Graphics, 690–93. `https://doi.org/10.1109/ICIG.2013.140`.