

Univerzita Karlova
Přírodovědecká fakulta



Algoritmy počítačové kartografie

Technická zpráva k první semestrální úloze

Geometrické vyhledávání bodu

David Šklíba a Filip Zadražil
1. N-GKDPZ
Praha 2022

1 Zadání a údaje o bonusových úlohách

Vstup: Souvislá polygonová mapa n polygonů $\{P_1, \dots, P_n\}$, analyzovaný bod q .

Výstup: P_i , $q \in P_i$.

Nad polygonovou mapou implementujete Winding Number Algorithm pro geometrické vyhledání incidujícího polygonu obsahujícího 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 reprezentaci jednotlivých polygonů použijte špagetový model.

Hodnocení:

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

2 Popis a rozbor problému

2.1 Point Location Problem

V rámci známého jednoduchého obecného problému *Point Location Problem* bývá zpravidla hledána poloha bodu ve vztahu k množině bodů, které tvoří m mnohoúhelníků (polygonů). V mnoha vědeckých a technických aplikacích je totiž důležité vědět, zda zadaný bod q leží uvnitř polygonu P či nikoli. *Point Location Problem* je klasickým problémem v počítačové grafice a výpočetní geometrii a je řešen od samotného vzniku těchto vědních disciplín. Jeho užití nalezneme například v GIS, kdy velmi často hledáme polohu bodu vzhledem k ostatním objektům v prostoru. Nalezení příslušného mnohoúhelníku je proto mnohdy velmi důležité.

V rámci tohoto problému bylo vyvinuto již velké množství algoritmů, které byly dokonce úspěšně implementovány v praxi. Hlavním problémem mnohých algoritmů jsou však podmínky singularity, které jsou ne vždy vhodně ošetřeny. Dalším nešvarem některých algoritmů je jejich velká časová náročnost (Kumar, Bangi 2018).

V rámci této úlohy byl vytvořen program v uživatelském rozhraní *Qt*, který hledá polohu bodu vůči polygonům pomocí dvou vybraných hojně užívaných metod - *Winding Number* a *Ray Crossing*. I vzhledem k využitelnosti tohoto programu byl zdrojový kód napsán tak, aby byly obě metody implementovatelné k nekonvexním mnohoúhelníkům, jež se v reálném světě vyskytují mnohem častěji než konvexní.

2.2 Metoda Winding Number

Tato metoda je založena na tom, že z pozorovaného bodu označovaného písmenem q jsou postupně ke všem vrcholům mnohoúhelníku P počítány úhly ω_i , jejichž součet označovaný řeckým písmenem Ω udává polohu bodu vůči mnohoúhelníku.

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

Tato hodnota je uváděna v počtech oběhů čili v násobcích 2π . Výsledná hodnota je závislá na směru pohybu. Pokud procházíme jednotlivé vrcholy ve směru hodinových ručiček, vyjde nám výsledná hodnota omega záporná.

Pokud se však jedná o záporný násobek 2π beze zbytku, je hledaný bod uvnitř polygonu, stejně jako kdyby byla hodnota kladná. I proto využívá sestrojený algoritmus absolutní hodnotu omega.

Obecně lze winding number definovat také jako obrysový integrál v komplexní rovině (Alciatore, Miranda 1995):

$$\omega = \frac{1}{2\pi i} \int_P \frac{1}{z} \quad \text{where } z = x + iy$$

Mezi výhody tohoto algoritmu patří lepší ošetření singulárních případů (na rozdíl od Ray Crossing metody), avšak jedná se o metodu pomalejší a problém nastává v případě, že je hledaný bod zároveň jedním z vrcholů mnohoúhelníku. Špatný výsledek lze však dostat i z tzv. chyby ze zaokrouhlení. V odborné literatuře se pak s touto metodou můžeme setkat i pod názvem *Sum of Angles Method* (Huang, Shih 1997).

2.3 Metoda Ray Crossing

Princip této metody tkví ve vedení polopřímky r (paprsek, tj. ray) procházející zkoumaným bodem q . Jistou výhodou je invariance vůči směru tohoto paprsku. Metoda funguje stejně, když je paprsek veden pod libovolným úhlem. Nevýhodou této metody však jsou singularity.

Polopřímka r tedy na své délce protíná určitý počet hran v zadaném směru. Tato hodnota je označována písmenem k . O této hodnotě platí, že pokud je dělitelná dvěma beze zbytku, pak bod leží mimo polygon. V případě, že je zbytek po dělení roven 1, pak se bod q nachází uvnitř polygonu.

Jedním z využívaných způsobů, jak ošetřit některé singularity, je varianta metody s redukcí ke q . Při ní dojde k vytvoření lokálního souřadnicového systému (q, x', y') . Počátek se tedy promítne do bodu q a zároveň vzniknou dvě nové osy x' a y' . Tato metoda bývá v odborné literatuře nazývána jako *Axis Crossing Method* (Kumar, Bangi 2018).

Hlavní nevýhodou této varianty je však neschopnost algoritmu detekovat stav, kdy zkoumaný bod leží na hraně polygonu P . K odhalení této skutečnosti využíváme speciální úpravu algoritmu, která pracuje se dvěma paprsky r_1 a r_2 , které mají opačnou orientaci - r_1 je levostranný a r_2 pravostranný. Pokud se počet průsečíků u paprsků r_1 a r_2 různí, pak bod q leží na hraně polygonu P .

V případech, kdy paprsek prochází bodem q a zároveň hranou/hranami zkoumaného polygonu, je aplikována upravená varianta metody *Ray Crossing*. V takovém případě dojde k tomu, že paprsek rozdělí rovinu na dvě poloroviny. Poté dochází k inkrementaci počtu průsečíků v závislosti na tom, kde se hrana vůči paprsku nachází. Existují dvě varianty aplikace. Tou první je, když se průsečík r a hrana/y mezi body p_{i-1} a p_i nachází v obou polorovinách nebo jen v té horní. Pak inkrementujeme k o příslušný počet průsečíků. Inkrementujeme však i v případě druhé varianty, která nastane, když je hrana p_{i-1}, p_i v obou vzniklých polorovinách nebo pouze v té dolní. Základním pravidlem zavedení této upravené metody do praxe však je, že se obě zmíněné varianty nesmí kombinovat (Bayer 2022).

3 Popis algoritmů formálním jazykem

V této části úlohy jsou přiloženy popisy algoritmů jednotlivých metod, které jsou popsány formálním jazykem. Oba vysvětlují, jak daný kód pracuje s vloženými daty a analyzuje polohu bodu vůči polygonům.

3.1 Winding Number

Algorithm 1 Winding Number

Pro každý polygon:

Nastavení Ω na hodnotu 0.

Pro každou hranu:

Výpočet vektoru u hrany.

Výpočet vektoru od prvního bodu hrany k bodu q .

Výpočet determinantu matice tvořené oběma vektory.

Pokud je determinant záporný:

Bod q leží v pravé polorovině.

Pokud je determinant kladný:

Bod q leží v levé polorovině.

Pokud je determinant roven nule:

Bod q leží ve směru hrany.

Výpočet úhlu α mezi vektory.

Pokud bod q leží v levé polorovině:

Přičtení úhlu α k hodnotě Ω .

Pokud bod q leží v pravé polorovině:

Odečtení úhlu α od hodnoty Ω .

Pokud bod q leží ve směru hrany:

Výpočet součinu rozdílů souřadnic bodu q od souřadnic krajních bodů hrany.

Pokud jsou oba součiny záporné:

Bod q leží na hranici polygonu.

Pokud je $|\Omega|$ rovna 2π :

Bod q leží uvnitř polygonu.

Pokud je $|\Omega|$ menší než 2π :

Bod q leží vně polygonu.

3.2 Ray Crossing

Algorithm 2 Ray Crossing

Pro každý polygon:

Pro každou hranu tvořenou body p_1 a p_2 :

Bod p_1 má nižší hodnotu souřadnice y .

Paprsek vysíláme od bodu q doprava.

Pokud bod q leží nad hranou nebo pod hranou nebo napravo od hrany:

Paprsek neprochází hranou.

Pokud bod q leží vlevo od hrany:

Paprsek prochází hranou.

Pro další případy:

Výpočet směrnice hrany od bodu p_1 do bodu q .

Výpočet směrnice od bodu p_1 do bodu q .

Pokud se sobě směrnice rovnají:

Bod q leží na hraně.

Pokud je směrnice k bodu q větší:

Paprsek prochází hranou.

Pokud je směrnice k bodu q menší:

Paprsek neprochází hranou.

Pokud je počet protnutých hran lichý:

Bod leží uvnitř polygonu.

Pokud je počet protnutých hran sudý:

Bod leží vně polygonu.

4 Problematické situace

Problematická se ukázala být situace u metody Winding Number, při níž bod q leží ve směru hrany polygonu. Ošetření této singularity bylo učiněno tak, že byly vypočítány součiny rozdílů souřadnic bodu q od souřadnic krajních bodů hrany. V případě, že oba součiny vyšly záporné, bylo rozhodnuto, že bod q leží na hranici polygonu (přesný popis postupu v kódu, viz kapitola *Popis algoritmů formálním jazykem, Algorithm 1*). U metody *Ray Crossing* žádná speciální problematická situace řešena nebyla.

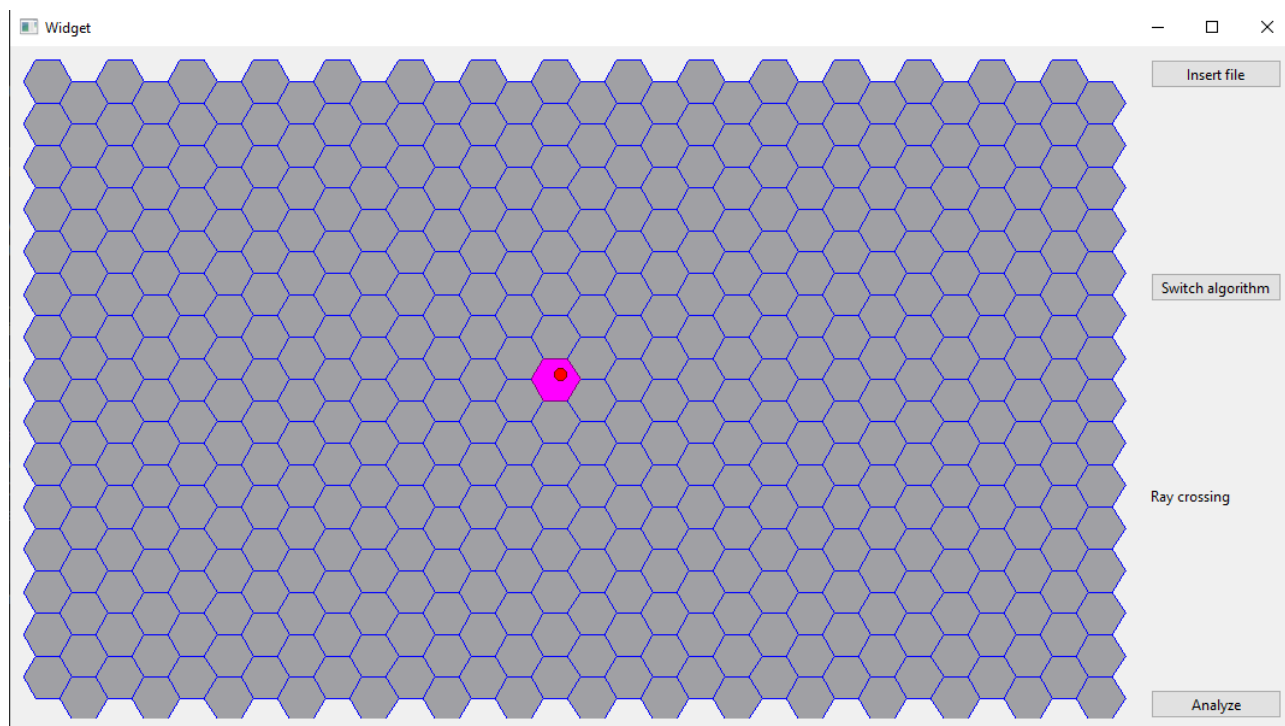
5 Vstupní a výstupní data

Data, která vstupují do programu jsou polygony uložené ve formátu *.shp* (shapefile). Lze analyzovat polygony konvexní i nekonvexní, ale podmínkou je, že musí být v souřadnicovém systému S-JTSK. Pro ukázkou je jako příloha k úloze připojen soubor s názvem *honeycomb.shp*, který obsahuje konvexní polygony uspořádané ve tvaru včelí plástve.

Vytvořený program pak nemá žádná výstupní data, která by byla k dispozici jako samostatný soubor. Proto za výstupní data lze označit maximálně grafické znázornění incidujícího polygonu.

6 Ukázka vytvořené aplikace

Na Obrázku 1 lze vidět výsledný *Widget*, neboli grafický výstup vytvořeného kódu. Hlavní část tvoří okno, v němž může uživatel kliknutím stanovit polohu zkoumaného bodu. V pravé části se pak nachází tři tlačítka. Prvním z nich je *Insert File*, pomocí něž můžeme nahrát soubor s polygony. Prostřední *Switch algorithm* slouží k překliknutí na vybraný algoritmus. V nabídce jsou dva - *Winding Number* a *Ray Crossing*. Tlačítko *Analyze* v pravém dolním rohu pak umožňuje uživateli zahájit proces analýzy, při němž dojde k vypočítání zvoleného algoritmu a následnému určení příslušnosti nakliknutého bodu k některému z polygonů (v případě, že se bod nachází uvnitř některého z nich).



Obrázek 1: Ukázka aplikace vyhledání polohy bodu metodou Ray Crossing

7 Dokumentace

Skript *MainForm.py*, přes který se program spouští, obsahuje třídu uživatelského rozhraní *Ui_MainForm* navrženého pomocí SW *QTCreator*. Pro jednotlivá tlačítka má třída definované metody odkazující na skripty *draw.py* a *algorithms.py*. Skript *draw.py* obsahuje třídu *Draw*, která byla navržena pro vizualizaci objektů ve widgetu uživatelského rozhraní.

Parametry třídy jsou:

- objekt *self.q* typu *QPoint* jehož poloha se bude vyšetřovat,
- seznam *self.polygons* objektů typu *QPolygon* vstupních polygonů,
- seznam *self.res_pol* obsahující polygony ke zvýraznění,
- seznam *self.coordinates* obsahující seznamy souřadnic vstupních polygonů,

- seznam *self.extent* obsahující souřadnice rozsahu polygonů,
- seznam *self.canvas_extent* obsahující šířku a výšku widgetu v pixelech.

Metodami třídy jsou:

- *insertFile* - načítá vstupní soubor ve formátu *.shp*, ukládá jeho souřadnice do seznamu *self.coordinates* a hledá souřadnicové extrémy, které ukládá do seznamu *self.extent*
- *rescaleData* - škáluje souřadnice polygonů tak, aby se celý soubor vešel do okna widgetu
- *mousePressEvent* - upravuje souřadnice objektu *self.q* kliknutím do prostoru widgetu
- *paintEvent* - vykresluje jednotlivé objekty do okna widgetu
- *getPoint* - vrací objekt *QPoint*
- *getPolygons* - vrací seznam objektů typu *QPolygon* vstupních polygonů
- *setResPol* - přidává polygon do seznamu *self.res_pol*
- *clearResPol* - vyprazdňuje seznam *self.res_pol*

Skript *algorithms.py* obsahuje třídu *Algorithms*, která byla vytvořena pro implementaci Point Location Problem algoritmů.

Parametrem třídy je:

- Booleovská proměnná *self.winding_num*, která určuje, zdali program momentálně využívá algoritmus Winding Number.

Metodami třídy jsou:

- *getPointAndLinePosition* – určuje vzájemnou polohu bodu a linie pomocí poloroviny
- *get2LinesAngle* – počítá velikost úhlu svíraného mezi dvěma úsečkami
- *getPositionPointAndPolygon* – určuje vzájemnou polohu bodu a polygonu metodou winding number
- *detectIntersection* – zjišťuje, zdali paprsek vyslaný z bodu doprava prochází úsečkou
- *rayCasting* - určuje vzájemnou polohu bodu a polygonu metodou ray crossing/ray casting
- *isWindingNumber* – vrací hodnotu parametru *self.winding_num*
- *setSource* – provádí negaci parametru *self.winding_num*

8 Závěr

Cílem této úlohy bylo sestavit program pomocí uživatelského rozhraní *Qt*, který by analyzoval polohu vybraného bodu vůči zvoleným polygonům, a to pomocí základních metod *Winding Number* a *Ray Crossing*. Tento program se podařilo úspěšně sestavit. Ke zdokonalení základního programu vedlo i splnění některých bonusových úloh.

Přesto by však bylo stále možné současný program ještě vylepšit. Chybí zde například ošetření simplexů u metody *Ray Crossing*. Konkrétně jde o ošetření případu, kdy bod leží na hraně polygonu. Výhodou by bylo i zvýraznění všech polygonů, kterých se tato singularita týká.

V celkovém kontextu se však podařilo sestavit funkční program, který poskytuje uživateli relativně uspokojivé výsledky. Zároveň je jednoduchý na používání i pro laickou veřejnost a snadno se v něm orientuje.

9 Seznam literatury

Zdrojový kód programu vznikl z velké části na základě informací z přednášky a cvičení vedených doc. Bayerem (2022). Informace byly čerpány z prezenčních lekcí a zároveň z prezentace dostupné na odkazu https://web.natur.cuni.cz/bayertom/images/courses/Adk/adk3_new.pdf (cit. 7. 3. 2022)

Alciatore, D., Miranda, R. (1995): A Winding Number and Point-in-Polygon Algorithm. Glaxo Virtual Anatomy Project Research Report.

Huang, C., W., Shih, T., Y. (1997): On the complexity of point-in-polygon algorithms. Computers & Geosciences, 23, 1, 109-118.

Kumar, B. N., Bangi, M. (2018): An Extension to Winding Number and Point-in-Polygon Algorithm. IFAC-PapersOnLine, 51, 1, 548-553.