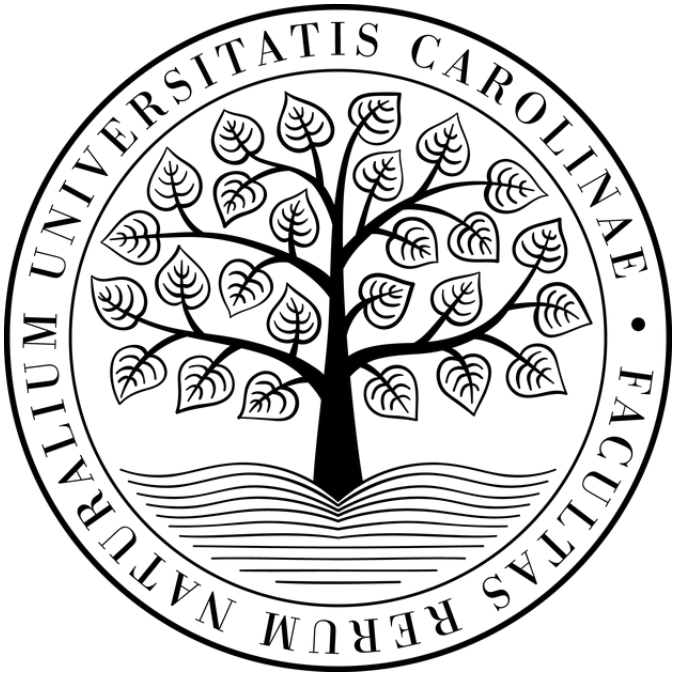
Přírodovědecká fakulta
Univerzita Karlova



Algoritmy počítačové kartografie

Úkol č. 1: Point Location Problem

Anna Brázdová a Petra Pajmová

2.N-GKDPZ

Praha 2024

Zadání

Úloha č. 1: Geometrické vyhledávaní bodu

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

Výstup: $P_i, q \in P_i$.

Nad polygonovou mapou implementujete Ray Crossing Algorithm pro geometrické vyhledávání 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

V rámci tohoto programu byly řešeny následující bonusové úlohy:

Krok	**hodnocení**
Detekce polohy bodu rozšiřující stavy uvnitř, vně polygonu.	10 b.
Analýza polohy bodu (uvnitř/vně) metodou Winding Number Algorithm.	*+10 b.*
Ošetření singulárního případu u Winding Number Algorithm: bod leží na hraně polygonu.	*+ 5 b.*
Ošetření singulárního případu u Ray Crossing Algorithm: bod leží na hraně polygonu.	*+ 5 b.*
Ošetření singulárního případu obou algoritmů: bod je totožný s vrcholem jednoho či více polygonů.	*+ 2 b.*
Zvýraznění všech polygonů pro oba výše uvedené singulární případy.	*+ 3 b.*
Celkem	**35 b.**

Popis a rozbor problému

Point Location Problem

Problém point in polygon, známý také jako Point Location Problem se zabývá určením prostorového vztahu mezi daným bodem a polygonem. Tento problém je vyskytuje v široké škále počítačových úloh, které využívají geometrické struktury, jako je například počítačová grafika, robotika, databáze nebo geografické informační systémy.

Problém lze řešit buď lokálně anebo globálně. Při lokální proceduře se snažíme zjistit, zda dotazovaný bod leží uvnitř, vně nebo na hranici dotazovaného mnohoúhelníku definovaného množinou vrcholů. Při globální proceduře se snažíme zjistit polohu bodu vůči množině mnohoúhelníků, tedy zda bod leží uvnitř jednoho z mnohoúhelníků, vně všech mnohoúhelníku, nebo zda inciduje s více mnohoúhelníky (bod leží na hraně/uzlu dvou a více mnohoúhelníků).

K řešení tohoto problému může být aplikována řada algoritmů. Jednotlivé varianty problému vyplývají z různých rozměrů a typů polygonů. U konvexních polygonů nejsou úhly mezi spojenými vrcholy větší než 180°, naopak nekonvexní polygony mají alespoň jeden úhel, který tuto hodnotu přesahuje. Pro konvexní polygony lze využít algoritmus Ray Crossing Method nebo Half-Plane Test. V geoinformatice a kartografii se častěji setkáme s nekonvexními útvary, jako jsou například hranice obcí, okresů, krajů nebo států, pro které je vhodné použít například algoritmy Winding Number nebo Ray Crossing

Winding Number Method

Základním konceptem algoritmu Winding Number (tzv. metoda ovíjení) je výpočet počtu ovinutí takovým způsobem, kdy se spočte počet protnutí paprsku vycházejícího z bodu q s hranami polygonu P, když se otáčí kolem bodu proti směru hodinových ručiček. Výsledná hodnota je součet všech úhlů, které svírá pozorovaný bod q s danými vrcholy polygonu P.

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

V případě, že je hodnota Winding Number nenulová, bod q leží uvnitř polygonu P. Pokud je číslo otočení rovno nule, bod q leží mimo polygon P.

$$d = \begin{vmatrix} q_x & q_y \\ v_x & v_y \end{vmatrix} = \begin{cases} > 0, & q \in \sigma_l \\ = 0, & q \in p \\ < 0, & q \in \sigma_r \end{cases}$$

Pro výpočet polohy bodu q vůči přímce $p \approx \overleftrightarrow{P_i P_{i+1}}$ je potřeba nejprve spočítat směrové vektory q a v, kdy

u je směrový vektor přímky a vektor v spojuje bod q a první bod hrany p_i.

$$\vec{p} = (x_{i+1} - x_i, y_{i+1} - y_i)$$
$$\vec{v} = (x_q - x_i, y_q - y_i)$$

Přímka p pak dělí rovinu σ na levou a pravou polorovinu (σ_l a σ_r). Výsledek Ω je znaménkově závislý na směru pohybu. V případě, že bod q leží v pravé polorovině, je úhel orientován ve směru hodinových ručiček a úhly se od sebe odčítají. U levé poloroviny je úhel orientován proti směru hodinových ručiček, při výpočtu se tedy úhly přičítají a hodnota Ω vyjde kladně. Výsledná hodnota Ω je uváděna v počtech oběhů, tedy v násobcích 2π.

$$d = \begin{vmatrix} q_x & q_y \\ v_x & v_y \end{vmatrix} = \begin{cases} > 0, & q \in \sigma_l \\ = 0, & q \in p \\ < 0, & q \in \sigma_r \end{cases}$$

Výhodou algoritmu je lepší ošetření singulárních případů oproti algoritmu Ray Crossing, nevýhodou je pak jeho vyšší časová náročnost a nutnost předzpracování dat.

Ray Crossing Algorithm

Zkoumaným bodem q je vedena polopřímka r (paprsek, tzv. Ray). Pro řešení úlohy je určující počet průsečíků polopřímky r vedené z bodu q s hranami polygonu P. Algoritmus je přitom invariantní vůči směru vedení polopřímky. Počet průsečíků polopřímky s ranami polygonu P je značen písmenem k. Pokud je hodnota k po vydělení dvěma rovna 1, zkoumaný bod leží uvnitř polygonu P. Pokud je ovšem hodnota po dělení rovna 0, bod leží mimo polygon P.

$$k\%2 = \begin{cases} 0, & q \notin P \\ 1, & q \in P \end{cases}$$

Tato základní podoba algoritmu Ray Crossing není uzpůsobena pro všechny situace, které mohou v prostoru nastat. Varianty tohoto algoritmu proto řeší také singulární případy vztahu polopřímky r a polygonu P.

Problematické situace a jejich rozbor

Metoda Winding Number

Výše popsaný algoritmus je schopný řešit pouze úlohu, kdy je bod jednoznačně uvnitř, nebo vně polygonu. Analyzovaný bod se ale může nacházet na hraně polygonu, nebo na jedno z jeho vrcholů.

Bod bude ležet na hraně polygonu právě tehdy když,

$$det = 0.$$

Tato podmínka ovšem samotná nestačí, a pro musí také platit, že

$$\omega = \pi.$$

Pokud bude bod ležet ve vrcholu polygonu, musí platit podmínka

$$q = p_i.$$

Ray Crossing Algorithm

Protože polopřímku p lze vést pod libovolným úhlem, může nastat situace, kdy polopřímka prochází vrcholem polygonu P, nebo je totožná s jeho hranou. Možným řešením je variace algoritmu Ray Crossing s redukcí ke q, která se skládá ze dvou kroků: Přepočítání souřadnic vrcholů polygonu P na základě posunutí počátku souřadnicového systému do bodu q a rozdělením paprsku na dvě polopřímky r_l a r_r s opačnou orientací (levostrannou a pravostrannou).

Ověření situace, kdy bod q je totožný s jedním z vrcholů polygonu P, probíhá pomocí porovnání souřadnic bodu q s každým vrcholem polygonu P podle nového souřadnicového systému. Pokud souřadnice vrcholu polygonu P a bodu q jsou totožné, bod q se nachází na vrcholu polygonu P. Pro ověření situace, zda bod q leží na hraně polygonu P, je určují počet průsečíků protilehlých polopřímek. V případě, že se počet průsečíků k_l a k_r s polopřímkami r_l a r_r nerovná, bod q se nachází na hraně polygonu P.

Popisy algoritmů formálním jazykem

Pseudokód metody Winding Number

Algorithm 1 *Winding Number*

inicialiizuj $\Omega = 0$, délku polygonu n a toleranci ϵ
pro všechny vrcholy polygonu:
    zkontroluj, zda bod q je totožný s nějakým vrcholem
    spočti vektory u a v
    spočti determinant z vektorů u a v
    spočti skalární součin vektorů u a v
    spočti vzdálenost mezi bodem q k hraně tvořenou vrcholy p_i a p_{i+1}
    pokud vzdálenost = 0:
        $q \in \partial P$
    spočti úhel ω
    pokud determinant > 0:
        $\Omega + \omega$
    jinak pokud determinant < 0:
        $\Omega - \omega$
    pokud determinant $= 0$ a zároveň je $\Omega = \pi$ ¡ tolerance:
        $q \in \partial P$
pokud $|\Omega| - 2\pi < tolerance$:
    $q \in P$
jinak $q \notin P$

Pseudokód metody Ray Crossing

Algorithm 2 *Ray Crossing*

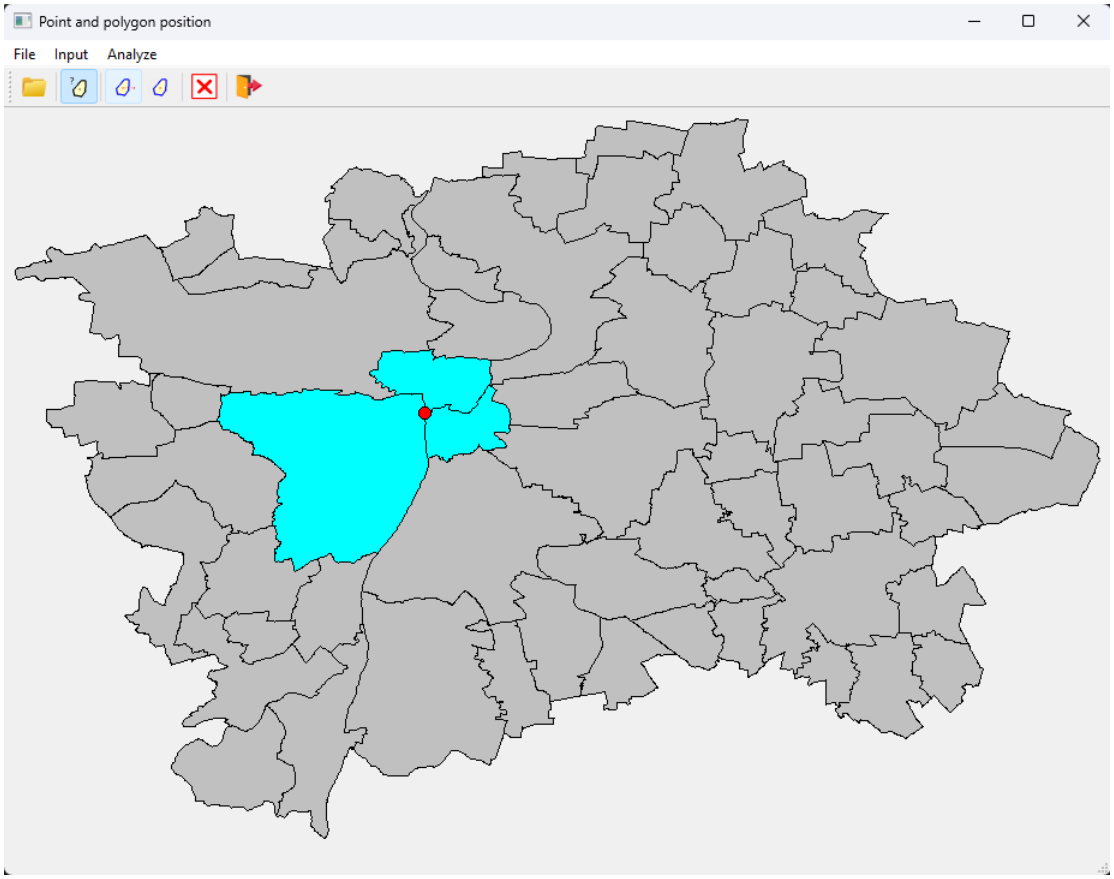
inicializuj počet levostranných a pravostranných průsečíků k_l a k_r na 0 a počet vrcholů n
pro všechny vrcholy:
    vypočti redukované souřadnice x_{ir}, y_{ir}, x_{i+1r}, y_{i+1r}
    pokud $x_{ir} = 0$ a $y_{ir} = 0$
        q je totožný s vrcholem
    vypočti x souřadnice průsečíku x_m
    pokud je $(y_i < 0) \neq (y_{i+1} < 0)$
        pokud $x_m < 0$:
            zvyš počet průsečíků v lévé polorovině k_l
    pokud $(y_i > 0) \neq (y_{i+1} > 0)$
        pokud $x_m > 0$:
            zvyš počet průsečíků v pravé polorovině k_r
pokud $(k_l\%2) \neq (k_r\%2)$
    $q \in \partial P$
pokud $k_r\%2 = 1$
    $q \in P$
jinak $p \notin P$

Vstupní data a výstupní data

Vstupními daty pro analýzu a testování problematiky Point Location jsou městské části Prahy ve formátu shapefile, které jsou volně ke stažení na webových stránkách Geoportál Praha.

Výstupem je aplikace vytvořená v rozhraní QT, která umožňuje grafické znázornění analýzy. Aplikace umožňuje načtení dat (polygonů) a výběr bodu zvoleného uživatelem. Výsledkem je zvýraznění polygonů a zobrazení okna s informací o vzájemném vztahu polygonu a dotazovaného bodu.

Printscreen vytvořené aplikace



Dokumentace

Třída Algorithms

Třída Algorithms se nachází ve scriptu *algorithms.py* a obsahuje definice metod *rayCrossing* a *windingNumber*. Funkce *rayCrossing* analyzuje vzájemnou polohu bodu a polygonu pomocí algoritmu Ray Crossing a vrací informaci o tom, zda se bod nachází uvnitř, vně, na hraně, nebo ve vrcholu polygonu. Funkce *windingNumber* provádí totožnou analýzu, ovšem pomocí algoritmu Winding Number.

Třída Draw

Třída Draw se nachází ve scriptu *draw.py* a obsahuje následující parametry: *self.q* – typ QPointF s počátečními souřadnicemi mimo pracovní okno, *self.add-vertex* - vykreslení bodu, *self.pol* – typ QPolygonF, *self.polygons* – seznam polygonů, *self.features* – ukládání objektů z načteného shapefilu (iniciován jako None), *self.min-max* – ukládání minimálních a maximálních souřadnic polygonů, *self.result* – seznam výsledků pro každé spuštění analýzy.

Třída Draw dále obsahuje následují metody: *loadData* – načtení vstupních dat ve formátu shapefile, *resizeData* – úprava souřadnic bodů jednotlivých polygonů pro zobrazení uvnitř okna aplikace, *mousePressEvent* – změna polohy bodu q podle pozice myši při kliknutí, *paintEvent* – vykreslení polygonů a bodů, *setResult* – vracení všech polygonů, které odpovídají analyzovanému bodu, *switchDrawing* – přidaní, nebo odstranění bodu, *getQ* – vrácení analyzovaného bodu, *getPol* – vrácení analyzovaného polygonu, *getPols* – vrácení analyzovaných polygonů (pokud analyzovaný bod odpovídá více polygonům), *clearData* – obnoví prázdné pracovní okno.

Třída MainForm

Třída MainForm se nachází ve skriptu $MainFrom.py$ a slouží ke spuštění uživatelského rozhraní aplikace a zajišťuje propojení jednotlivých algoritmů definovaných v ostatních třídách.

Závěr

V rámci této práce byla řešena úloha Point Location Problem pomocí dvou zvolených algoritmů, kterými byly Winding Number Algorithm a Ray Crossing Algorithm. Algoritmy dovedly úspěšně analyzovat polohu bodu vůči polygonům včetně speciálních případů (singularit), kdy daný bod ležel na hraně polygonu, nebo byl totožný s jedním z jeho vrcholů.

Výstupem této práce je aplikace, která umožňuje jejímu uživateli vytvářet situace, ke kterým může ve smyslu polohy bodu a polygonu v prostoru docházet. Pro snadnou interpretaci výsledku analýzy je aplikace také schopna situaci náležitě vizualizovat.

Práce stále nabízí prostor pro další vylepšní. Nabízí se například možnost tvorby vlastních polygonů nebo ošetření problému polygonu s dírou.

Seznam literatury

BAYER, T. (2008): Algoritmy v digitální kartografii. Nakladatelství Karolinum, Praha.

BAYER, T. (2024): Point Location Problem. Přednáška pro předmět Algoritmy počítačové kartografie, Katedra aplikované geoinformatiky a kartografie. Přírodovědecká fakulta UK (cit. 15. 3. 2024). Dostupné z: https://web.natur.cuni.cz/ bayertom/index.php/teaching/algoritmy-pocitacove-kartografie

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. IFACPapersOnLine, 51, 1, 548-553.