České vysoké učení technocké v Praze
Fakulta stavební

Algoritmy v digitální kartografii

Množinové operace s polygony

Bc. Robin Pflug
Bc. Tomáš Klemsa

Obsah

1 Zadání úlohy

Vstup: *množina* $P = (P_1, ..., P_n)$.
Výstup: *množina polygonů* $P' = (P'_1, ..., P'_m)$

S využitím algoritmu pro množinové operace s polygony implementujte pro libovolné dva polygony $P_i, P_j \epsilon P$ následující operace:

- Průnik polygonů $P_i \cap P_j$,
- Sjednocení polygonů $P_i \cup P_j$,
- Rozdíl polygonů $P_i \cap \overline{P}_j$ resp $P_j \cap \overline{P}_i$. .

Jako vstupní data použijte existující kartografická data (např. konvertované shape fily) či syntetické data, která budou načítána z textového souboru.

Grafické rozhraní realizujte s využitím frameworku QT.

Při zpracování se snažte postihnout nejčastější singulární případy: společný vrchol, společná část segmentu, společný celý segment čí více společných segmentů. Ošetřete situace, kdy výsledkem není 2D entita, ale 0D či 1D entita.

Pro výše uvedené účely je nutné mít řádně odladěny algoritmy z úlohy 1. Postup ošetřená těchto případů diskutujte v technické zprávě, zamyslete se nad dalšími singularitami, které mohou nastat.

Hodnocení:

Krok	**Hodnocení**
Množinové operace: průnik, sjednocení, rozdíl	20b
Konstrukce offsetu (bufferu)	*+10b*
Výpočet průsečíků segmentů algoritmem Bentley & Ottman	*+8b*
Řešení pro polygony obsahující holes (otvory)	*+6b*
Max celkem:	**44b**

Obrázek 1: Bodové ohodnocení úlohy [zdroj: 1]

2 Údaje o bonusových úlohách

Bonusové úlohy nebyly řešeny. Obsahem této úlohy je pouze povinná část zadání.

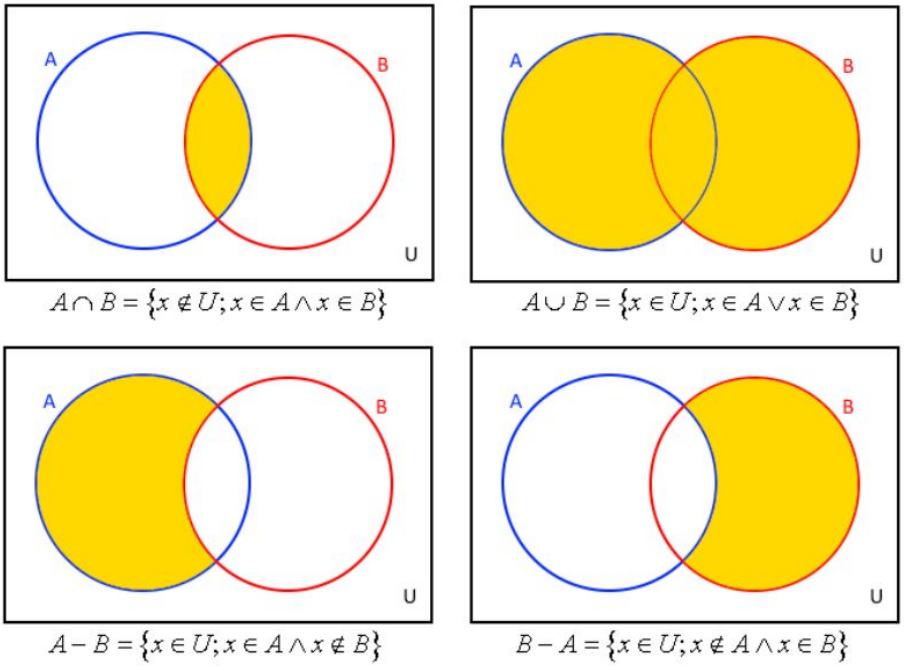
3 Obecná formulace a řešení problému

Obsahem této úlohy je vytvořit aplikaci, která bude schopna provádět čtyři základní množinové operace nad dvěma zadanými polygony: průnik, sjednocení, a rozdíly.

Sjednocení množin: $A \cup B$
Sjednocením množin A a B vznikne nová množina, která bude obsahovat všechny prvky z množiny A a také všechny prvky z množiny B. [zdroj: 2]

Průnik množin: $A \cap B$
Průnikem dvou množin A a B vznikne nová množina, která bude obsahovat prvky, které mají ty dvě množiny společné. [zdroj: 2]

Rozdíl množin: $A \cap \overline{B}; B \cap \overline{A}$
Rozdílem dvou množin A a B chápeme takovou množinu, která bude obsahovat všechny prvky z A a zároveň nebude obsahovat žádný prvek z B. Pro dva zadané polygony je tedy možné vyhodnotit jejich rozdíl dvěma způsoby *(A-B nebo B-A)*. [zdroj: 2]



Obrázek 2: Grafické znázornění jednotlivých množinových operací. [zdroj: 3]

4 Aplikované algoritmy

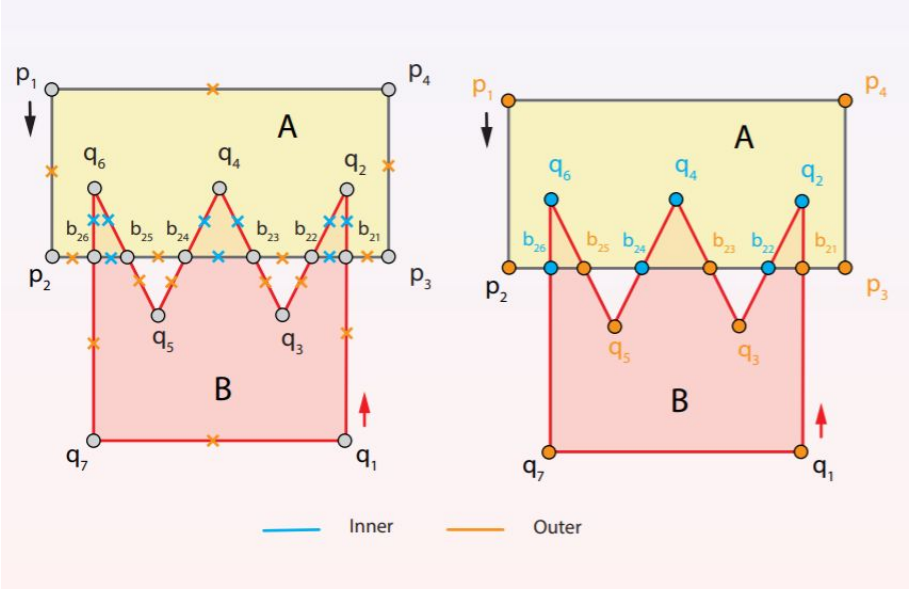
V aplikaci byly použity algoritmy pro vyhodnocení polohy bodu a přímky nebo polygonu, výpočet úhlu dvou vektorů, určení vztahu dvou přímek, výpočet průsečíků dvou polygonů. Aplikace algoritmů funguje pro nekonvexní polygony, kde výsledkem je prázdná množina, bod, přímka, plocha nebo kombinace těchto entit.

4.1 Výpočet průsečíků polygonů

Pro výpočet průsečíků polygonu byl použit naivní algoritmus. Algoritmus postupně prochází všechny segmenty jednoho polygonu a porovnává je se všemi segmenty polygonu druhého. Pro případy, kdy existuje pro aktuální segment jednoho polygonu více průsečíků se segmenty polygonu druhého, není možné přidat mezi počáteční a koncový bod segmentu okamžitě nalezený průsečík. Je nutno nejprve nalézt všechny průsečíky náležící aktuálnímu segmentu a následně je přidávat ve správném pořadí. [zdroj 1]

Pro tento krok je v aplikaci využita mapa. V mapě je jako klíč hodnota α nebo β (*poloha průsečíku na přímce*) a bod průsečíku. Pokud je nalezen nový bod, je seznam souřadnic bodů polygonu aktualizován na základě právě hodnoty klíče mapy. Tento postup zajistí zápis bodů ve správném pořadí.

Poloha hrany vůči druhému polygonu je vyhodnocena na základě Winding number algoritmu a to tak, že je určována poloha středového bodu hrany vůči druhému polygonu.



Obrázek 3: Příklad ohodnocení středů hran dvou polygonů. [zdroj: 1]

4.1.1 Algoritmus Process intersection

1. $if(t >= \epsilon \wedge t <= 1 - \epsilon)$:
2. $\quad i \longleftarrow i + 1$
3. $\quad P \longleftarrow (begin + i, pi)$ // Přidej průsečík na pozici i+1

[zdroj: 1]

4.1.2 Algoritmus pro výpočet průsečíků polygonů

1. $for(i = 0; i < n, i++)$
2. $intersections = map(double, QPointFB)$ // Vytvoření mapy
3. $for(j = 0; j < m, j++)$
4. if $b_{ij} = (p_i, p_{(i+)1\%n} \cap q_j, q_{(j+)1\%m}) \neq 0$ // Existuje průsečík
5. $intersections[\alpha_i] \longleftarrow b_{ij}$ //Přidej do intersections
6. processIntersection (b_{ij}, β, B, j) //Zpracuj první průsečík pro e_j
7. if $(||intersections|| > 0)$ // Nějaké průsečíky jsme nalezli
8. $for \forall m \in intersections$: // Procházej všechny průsečíky v intersections
9. $b \longleftarrow intersections.second$ // Získej 2. hodnotu z páru
10. ProcessIntersection (b, α, A, i) //Zpracuj první průsečík pro e_i

4.1.3 Algoritmus pro polohu středového bodu hrany

1. for $(i = 0; i < n, i++)$:
2. $M = \frac{p_i(x,y)+p_{i+1}(x,y)}{2}$ // Výpočet středu hrany
3. pozice = $positionPointPolygonWinding(\bar{p}, B)$ //Určení polohy M.
4. $p_i[pozice]$ = pozice

4.2 Množinové operace

Výsledný algoritmus množinových operací je založen na kombinaci algoritmů z předešlých kapitol.

4.2.1 Algoritmus selecEdges

1. $for(\forall P[i])$
2. if$(P[i]_{position} = position)$ //Pokud je pozice bodu rovna hledané pozici
3. $e \leftarrow (P[i], P[i+1])$ //Vytvoř danou hranu
4. $edges \leftarrow e$

4.2.2 Algoritmus boolean operations

1. computePolygonIntersection(PA, PB) //Nalezení průsečíků polygonů
2. setPositionAB; //Nastavení hran
3. if (Union)
4. seletEdges(PA, Outer)
5. selectEdges(PB, Outer) //Sjednocení
6. if (Intersect)
7. seletEdges(PA, Inner)
8. selectEdges(PB, Inner) //Průnik
9. if (DiffAB)
10. seletEdges(PA, Outer)
11. selectEdges(PB, Inner) //Rozdíl AB
12. if (Union)
13. seletEdges(PA, Inner)
14. selectEdges(PB, Outer) //Rozdíl BA

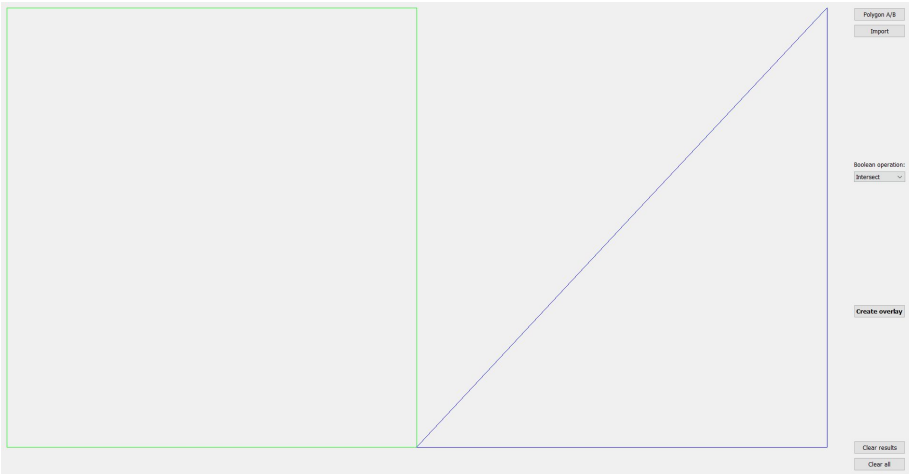
4.2.3 Algoritmus setPositionAB

1. poloha bodů středů hran pro PA,PB
2. poloha bodů středů hran pro PB,PA

5 Problémové vstupní množiny

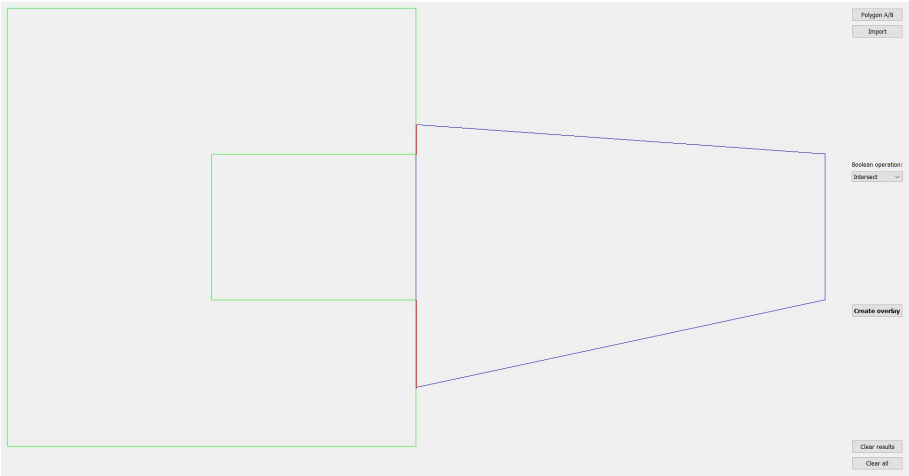
Vstupní množiny vytvářející polygony mohou obsahovat singulární případy. Testované singulární případy jsou situace, kdy mají polygony společný jeden nebo více vrcholů a jednu nebo více hran.

Pro první možnost (společný jeden bod), kdy je požadavek Intersect, vrátí aplikace prázdnou množinu hran. Ostatní množinové operace fungují ze své podstaty klasicky.



Obrázek 4: Příklad výstupu aplikace pro průnik množin se společným vrcholem

Pro případy, kdy je na vstupu množina bodů vytvářející polygony s jednou či více společnými hranami je opět zajímavý požadavek průniku množin. Aplikace navrací seznam hran obsahující právě tyto společné hrany.



Obrázek 5: Příklad výstupu aplikace pro průnik množin se společnými hranami

6 Vstup dat do aplikace

Vstup dat do aplikace lze realizovat dvěma způsoby. Uživatel může graficky vkládat jednotlivé vrcholy polygonů přímo kurzorem myši (se zřetelem na topologickou korektnost). Změna příslušnosti budoucího vrcholu k polygonu lze měnit tlačítkem "Polygon A/B".

Druhá možnost vstupu dat do aplikace je formou importu souřadnic vrcholů polygonu. Importovaný soubor ve formátu txt musí být ve formátu: *číslo Polygonu souřadnice X souřadnice Y* . Pořadí vrcholů polygonu určuje také pořadí vytvořených hran.

```
1 10 20
1 10 30
1 0 30
2 10 8
2 20 10
2 20 20
```

Obrázek 6: Vzor vstupních dat pro nahrávaní ze souboru .txt

7 Výstup aplikace

Grafický výstup je prezentován v Canvasu grafického rozhraní.

Obrázek 7: Příáklad výstupu aplikace pro zadané polygony s vyhodnocením jejich sjednocení

8 Dokumentace

8.1 Třídy

8.1.1 Algorithms

Třída Algorithms obsahuje metody zajišťující výpočty daných algoritmů.

TPointLinePosition getPointLinePosition(QPointFB q, QPointFB p1, QPointFB p2)
Návratová hodnota: *TPointLinePosition*;
Metoda vyhodnocující polohu bodu vůči zadané přímce. Metoda navrací hodnoty *LeftHp*(bod v lévé polorovině), *RightHp*(bod v pravé polorovině),nebo *Colinear*(bod na přímce).

double getAngle2Vectors(QPointFB p1, QPointFB p2, QPointFB p3, QPointFB p4)
Návratová hodnota: *double*;
Metoda navrací hodnotu úhlu mezi dvěma vektory.

TPointPolygonPosition positionPointPolygonWinding(QPointFB q, std::vector< *QPointFB* **> pol)**
Návratová hodnota: *TPointPolygonPosition*;
Metoda určující vzájemnou polohu bodu a polygonu algoritmem Winding number.Metoda navrací hodnoty *Inner*(bod uvnitř), *Outer*(bod vně),nebo *On*(bod na hraně).

T2LinesPosition get2LinesPosition(QPointFB p1, QPointFB p2, QPointFB p3, QPointFB p4, QPointFB pi)
Návratová hodnota: *T2LinesPosition*;
Metoda vyhodnocující vzájemný vztah dvou přímek. Metoda navrací hodnoty *Paralel*(rovnoběžné přímky), *Identical*(přímky se překrývají), *NonIntersected*(přímky nemají společný bod), nebo *Intersected*(přímky mají společný bod).

vector< *Edge* **> booleanOperations(vector<** *QPointFB* **> polygonA,vector<** *QPointFB* **> polygonB, TBooleanOperation operation)**
Návratová hodnota: *vector< Edge >*;
Metoda vrátí vektor hran, které jsou výsledkem zadané boolean operace.

void processIntersection(QPointFB pi, double t, std::vector¡QPointFB¿ polygon, int i)
Návratová hodnota: *void*;
Metoda přidá bod do listu vrcholů polygonů, pokud zatím neexistuje.

void computePolygonIntersection(vector< *QPointFB* **> pa, vector<** *QPointFB* **> pb)**
Návratová hodnota: *void*;
Metoda vypočte průsečíky hran polygonů.

void setPositionsAB(vector< $QPointFB$ > **pa, vector**< $QPointFB$ > **pb)**;
Návratová hodnota: *void*;
Metoda nastaví vztah hran vůči druhému polygonu u obou polygonů.

void setPositions(vector< $QPointFB$ > **pa, vector**< $QPointFB$ > **pb)**;
Návratová hodnota: *void*;
Metoda provádí výpočet vztahu hran polygonu vůči druhému.

void selectEdges(vector< $QPointFB$ > **pol, TPointPolygonPosition position, vector**< $Edge$ > **edges)**
Návratová hodnota: *void*;
Metoda naplní vektor hran na základě požadavku na jejich pozici.

8.1.2 Edge

Třída Edge definuje hranu. Hrana je dána jako počáteční bod s a koncový bod e.

QPointFB getStart()
Návratová hodnota: *QPointFB*;
Metoda vracející počáteční bod linie.

QPointFB getEnd()
Návratová hodnota: *QPointFB*;
Metoda vracející koncový bod linie.

void setStart(QPointFB s)
Návratová hodnota: *void*;
Metoda nastaví počáteční bod linie.

void setEnd(QPointFB e)
Návratová hodnota: *void*;
Metoda nastaví koncový bod linie.

8.1.3 QPointFB

Třída QPointFB definuje bod o souřadnicích x a y. Dalé body uchovávají hodnotu alfa a beta (jejich pozice na přímce polygonů) a také hodnotu polohy vůči přímce (x; y; alpha; beta; position).

double getAlpha()
Návratová hodnota: *double*;
Metoda vracející hodnotu alfa.

double getBeta()
Návratová hodnota: *double*;
Metoda vracející hodnotu beta.

TPointPolygonPosition getPosition()
Návratová hodnota: *TPointPolygonPosition*;
Metoda vracející polohu bodu vůči přímce.

void setAlpha(double alpha)
Návratová hodnota: *void*;
Metoda nastaví pro bod *QPointFB* hodnotu *alfa*.

void setBeta(double beta)
Návratová hodnota: *void*;
Metoda nastaví pro bod *QPointFB* hodnotu *beta*.

void setPosition(TPointPolygonPosition position)
Návratová hodnota: *void*;
Metoda nastaví pro bod *QPointFB* polohu vůči přímce.

9 Testování

Testování probíhalo na syntetických datech vytvořených v textovém editoru. Další způsob zadávání množin bodů pro testování bylo realizováno přímo grafickým vstupem.

9.1 Test 1

První testování bylo provedeno na množině obsahující řádově desítky vrcholů polygonů. Množina byla vytvořena grafickým vstupem. Pro všechny množinové operace dávala aplikace správné výsledky.

Obrázek 8: Union

Obrázek 9: Intersect

Obrázek 10: Difference AB

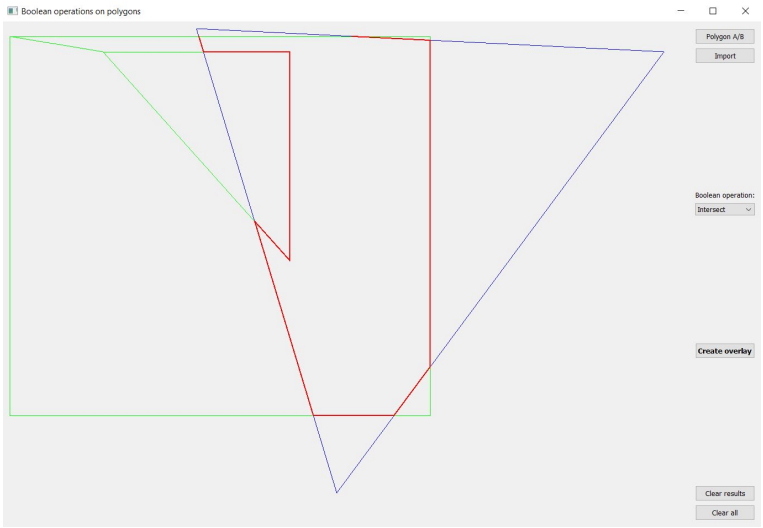
Obrázek 11: DifferenceBA

9.2 Test2

Druhá množina použita pro testování dat byla vytvořena syntetiky v textovém editoru. Množina obsahovala dva polygony se dvěma společnými hranami. Aplikace dávala ve všech případech opět korektní řešení. Příklad výstupu tohoto testu je obsahem kapitoly *5 Problémové vstupní množiny*

9.3 Test3

Třetí množina použita pro testování dat obsahovala opět dva polygony. Jeden polygon ovšem obsahoval "díru". Pro tento polygon nebyla aplikace schopna správně vyhodnotit množinové operace. Možné úskalí je ve formátu, jakým je polygon načítán. Pro polygony nelze v aplikaci definovat prázdnou vnitřní oblast zvlášť. Proto aplikace vytvoří nekorektní linii, spojující obvod polygonu s vnitřní oblastí ("díra").



Obrázek 12: Chybný výsledek pro průnik polygonů.

10 Závěr

Aplikace provádí nad zadanou množinou vrcholů dvou polygonů množinové operace. Pro "běžné"polygony je výsledný výstup aplikace správný a korektní. Problematické situace popsané v kapitole *5* jsou ošetřeny.

11 Náměty pro vylepšení

Vylepšení aplikace by bylo možné vytvořením funkcí množinových operací pro více polygonů najednou.

Vhodné by bylo také odladění situace kdy polygon obsahuje prázdný polygon ("díru"). Tento problematický případ není v úloze vyřešen.

Pro pokrytí většího pole užívání aplikace by také bylo vhodné přidat další funkce jako například konstrukce offsetu.

Jednoduchým vylepšením by bylo také přidání množinových operací pro úsečky importované z textového souboru. V aktuální verzi aplikace se takovýto vstup nenačte. Úsečky zadané grafikou cestou zobrazeny jsou, ovšem v případě například průniku je výsledek prázdný vektor hran.

12 Reference

1. BAYER, Tomáš. Metody konstrukce konvexní obálky [online][cit. 5.11.2019]. Dostupné z: https://web.natur.cuni.cz/ bayertom/images/courses/Adk/adk4.pdf

2. Matematika.cz [online][cit. 11.12.2019]. Dostupné z: matematika.cz

3. MatFyz.jeCool.net [online][cit. 11.12.2019]. Dostupné z: www.matfyz.jecool.net