VŠB TECHNICKÁ UNIVERZITA OSTRAVA | HORNICKO GEOLOGICKÁ FAKULTA | KATEDRA GEOINFORMATIKY

Programování v GIS 1
8 - Algoritmy nad vektorovými daty

Michal Kačmařík

A924, tel.: 5512

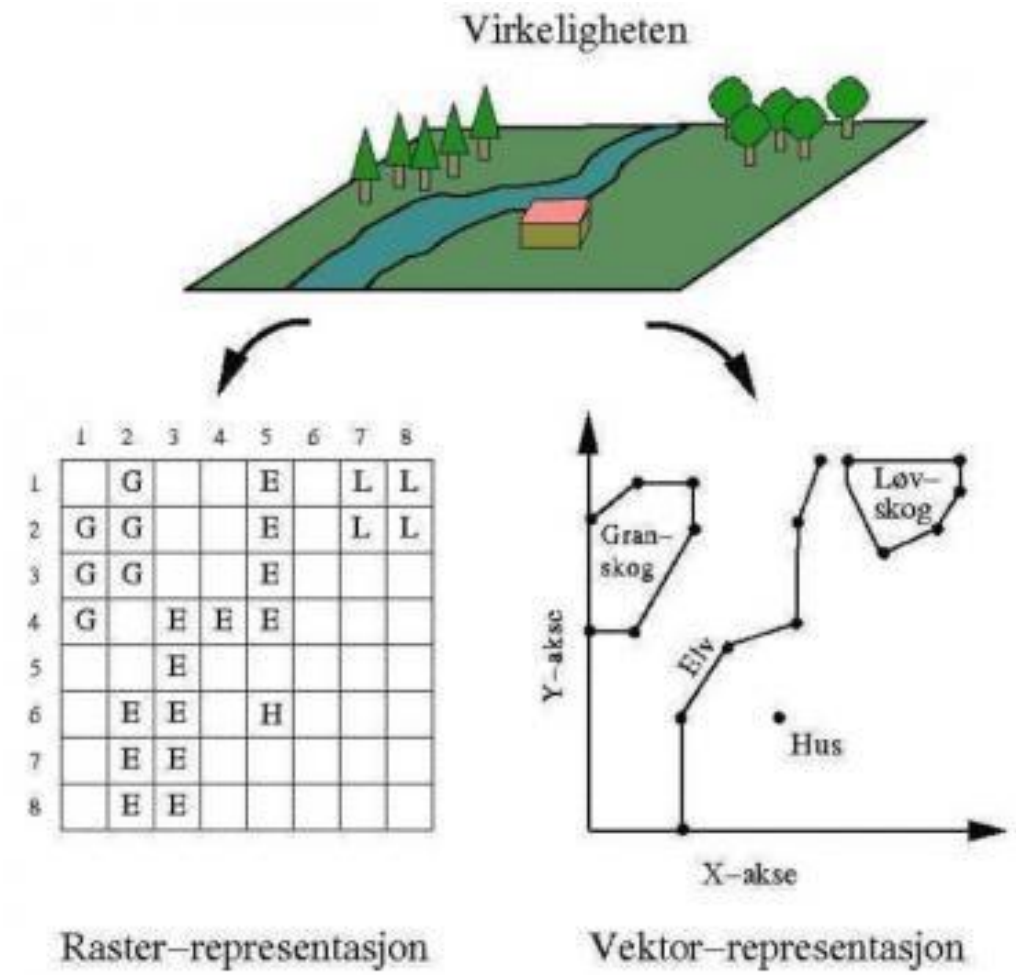
e-mail: michal.kacmarik@vsb.cz

https://www.hgf.vsb.cz/548/cs/

Prostorová data

- Geoprvek – entita definovaná v prostoru
- Znalost jeho
  - identifikace,
  - lokalizace – umístění v prostoru,
  - vlastností – vlastních (atributy), vztahových.
- Reprezentace geoprvku – určuje typ úloh, které s nimi dále bude možné provádět.

Vektor x Raster



Zdroj: http://ndla.no/nb/node/27054

Vektor

- Vektorový model používá pro reprezentaci geodat složky:
  - prostorovou – geometrie,
  - popisnou – atributy (reprezentace pomocí DT),
  - topologickou – v případech topologického rozšíření.

Vektory – geometrická složka

- Tři základní geometrie:
  - bod – 0D,
  - linie (čára) – 1D,
  - polygon (region, area) – 2D.

- Omezíme se na 2D prostor.

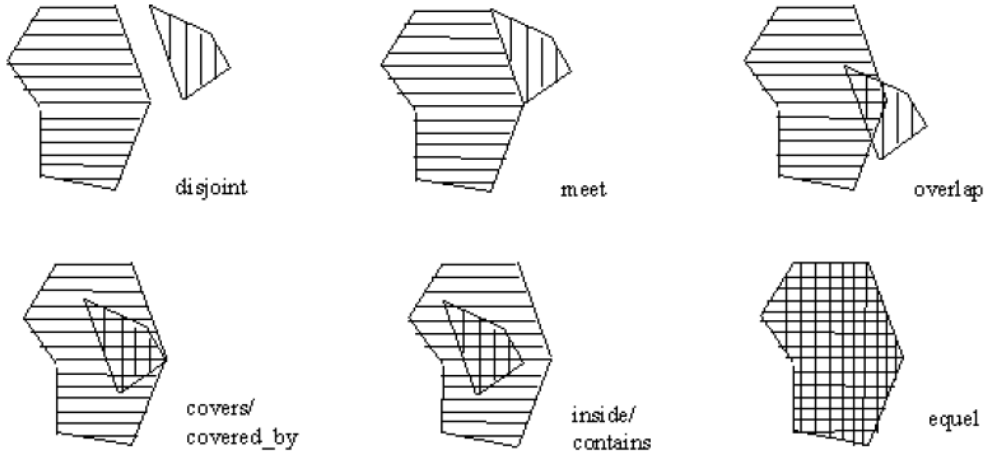
Výpočetní geometrie

- Computational geometry
- Zabývá se řešením geometrických úloh (ty se často objevují v geoinformatice)
- poskytuje nástroje a postupy pro jejich řešení v dimenzích podle charakteru dat (2D/3D).
- Příklady úloh:
  - nalézt cestu do cíle, nejbližší zařízení vybraného typu.
  - nalézt konvexní obálku – nejmenší polygon nad zadanou množinou bodů.
  - výpočet viditelnosti nad DMR.
  - průnik – vyhledat průniky geometrických složek (výběr geoprvků v oblasti).
  - průsečík – vyhledat průsečk geometrií (bodů, linií a polygonů).

Operace nad vektory

Predikátové operace

- equal(), disjoint()
- intersects(), touch(), crosses(),
- within(), contains(), overlaps(), relate()



Analytické funkce

- distance(), buffer(), convexHull(), intersection(),
- union(), difference()

Podrobněji v předmětu Zpracování dat v GIS

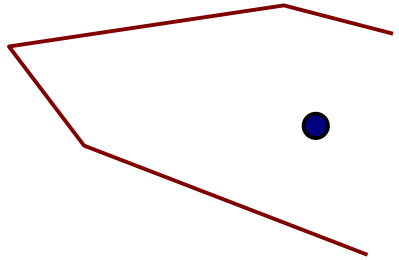
Predikátové operace

- Chápeme je jako Boolenovskou (= binární) funkci.
- Návratová hodnota:
  - 1 (TRUE) – případ, kdy je podmínka splněna,
  - 0 (FALSE) – případ, podmínka není splněna.

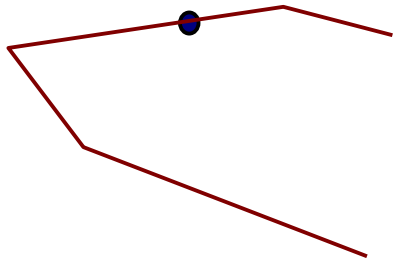
Predikátové operace

Příklad:

disjoint(point, linie)=TRUE

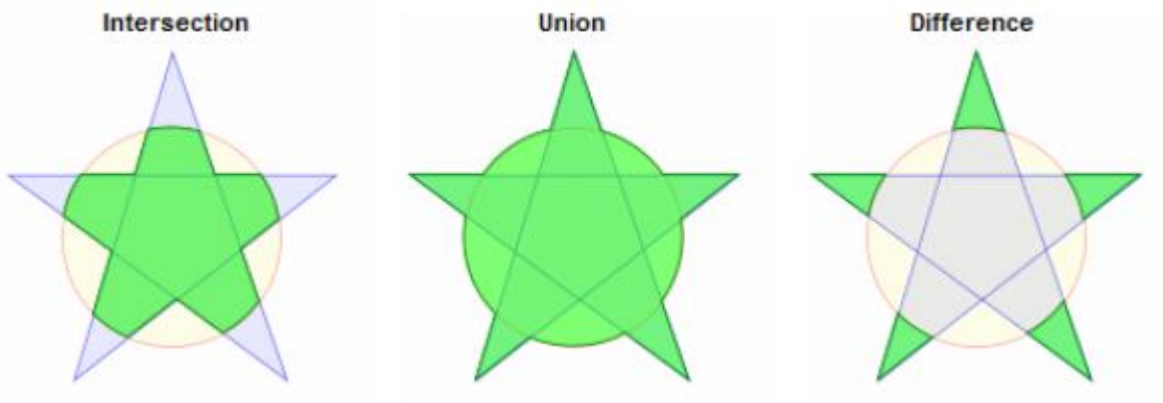


disjoint(point, linie)=FALSE



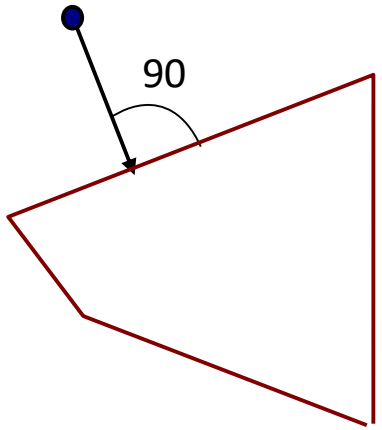
Analytické funkce

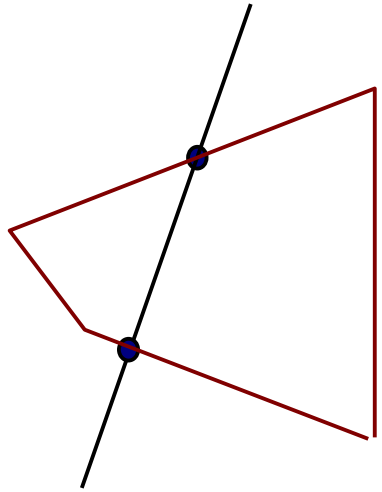
- Vrací hodnotu jako výsledek prostorového vztahu. Ten je dán dvojicí geometrií, které vstupují jako parametry funkce.
  - distance (vzdálenost) – návratová hodnota je číselná, představuje prostor, který odděluje dvě geometrie.
  - intersection (průsečík) – vrací geometrii jako výsledek kombinace dvou geometrií (výsledek záleží na typu vstupních geoprvků – bod, linie, polygon).
  - union (sjednocení) – vrací geometrii vzniklou sjednocením vstupních geoprvků



Analytické funkce

- distance(point, polygon)
- intersection(linie, polygon)





Výpočetní geometrie

- Složitější algoritmy v GIS se skládají z mnoha jednoduchých algoritmů.
- “Jednoduché” algoritmy řeší výpočetní geometrie, kde se zkoumá zlepšení algoritmů z pohledu složitosti.

Příklady operací a jejich algoritmy

- Průsečík přímek
- Vzdálenost bodu od přímky
- Generalizace geometrie linie
- Bod v polygonu
- Plocha polygonu
- Triangulace

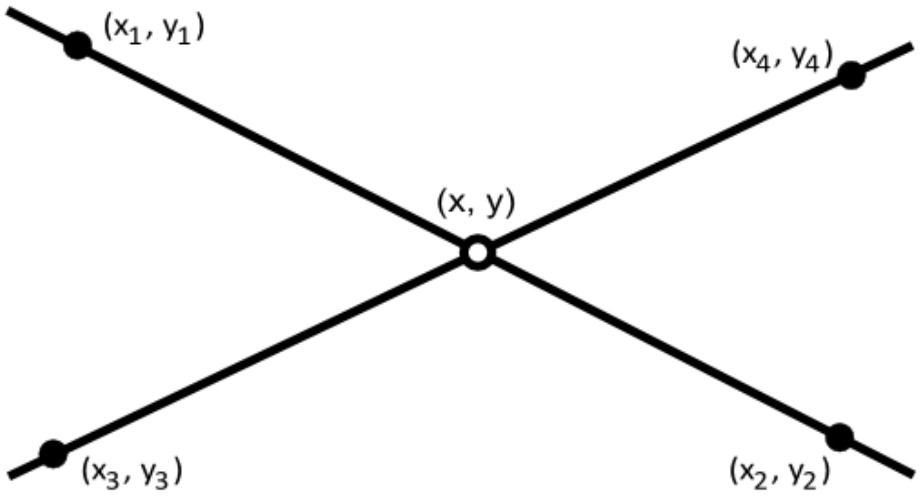
Průsečík linií

Průsečík linií lze považovat za kritickou operaci v GIS. Je použita

- v překryvných operacích,
- při spojování polygonů a linií, i při jejich rozkládání,
- v operacích bod v polygonu,
- při odstraňování mezer mezi polygony.

Průsečík dvou úseček

Zadání: Hledáme bod $P = [x; y]$, který je průsečíkem dvou úseček u a v, kde
u je dána body $P1 = [x_1, y_1]$ a $P2 = [x_2, y_2]$
v je dána body $P3 = [x_3, y_3]$ a $P4 = [x_4, y_4]$



Obrázek převzat z http://en.wikipedia.org/wiki/Line%E2%80%93line_intersection

Průsečík dvou úseček

Rovnice pro přímku: *směrnicová rovnice*
$y = a + bx$
kde a je svislý posun, b je sklon. *b= tg(uhel)*
Máme-li dva body na přímce p dány takto $P1 = [x_1, y_1]$ a $P2 = [x_2, y_2]$,
pak sklon po dosazení hodnot dvou bodů vypočteme podle vzorce

$$b = \frac{y_1 - y_2}{x_1 - x_2}$$

Do rovnice přímky dosadíme b

$$y = a + \frac{y_1 - y_2}{x_1 - x_2} x$$

a hodnotu a získáme dosazením hodnot libovolného zadaného bodu.
Poté zpětně dopočteme hodnotu b.
Tím jsme schopni získat konkrétní hodnoty sklonu a posunu pro dano přímku.

Průsečík dvou úseček - příklad

Postup hledání průsečíku si ukážeme na příkladu s konkrétními hodnotami.
Máme dány dvě úsečky u, v, kde
u je dána body $P1 = [0; 3]$ a $P2 = [3; 0]$
v je dána body $P3 = [3; 2]$ a $P4 = [1; 0]$
Použijeme rovnici přímky
$y = a + bx$
a vypočteme si a a b pro obě úsečky.
Pro u bude platit:

$$b = \frac{3 - 0}{0 - 3} = -1$$

dosadíme pro bod $P1 = [0; 3]$

$$3 = a - 1.0$$

a získáme rovnici pro úsečku u: $y = 3 - x$.
Stejný postup použijeme pro úsečku v a její rovnice bude mít tvar $y = -1 + x$.

Průsečík dvou úseček - příklad

Vypočteme soustavu dvou rovnic

$$y = 3 - x$$

$$y = -1 + x$$

a dostaneme hodnoty pro bod P.

$$3 - x = -1 + x$$

$$4 = 2x$$

$$2 = x$$

dosadíme

$$y = 3 - 2$$

$$y = 1$$

pak $P = [2; 1]$.

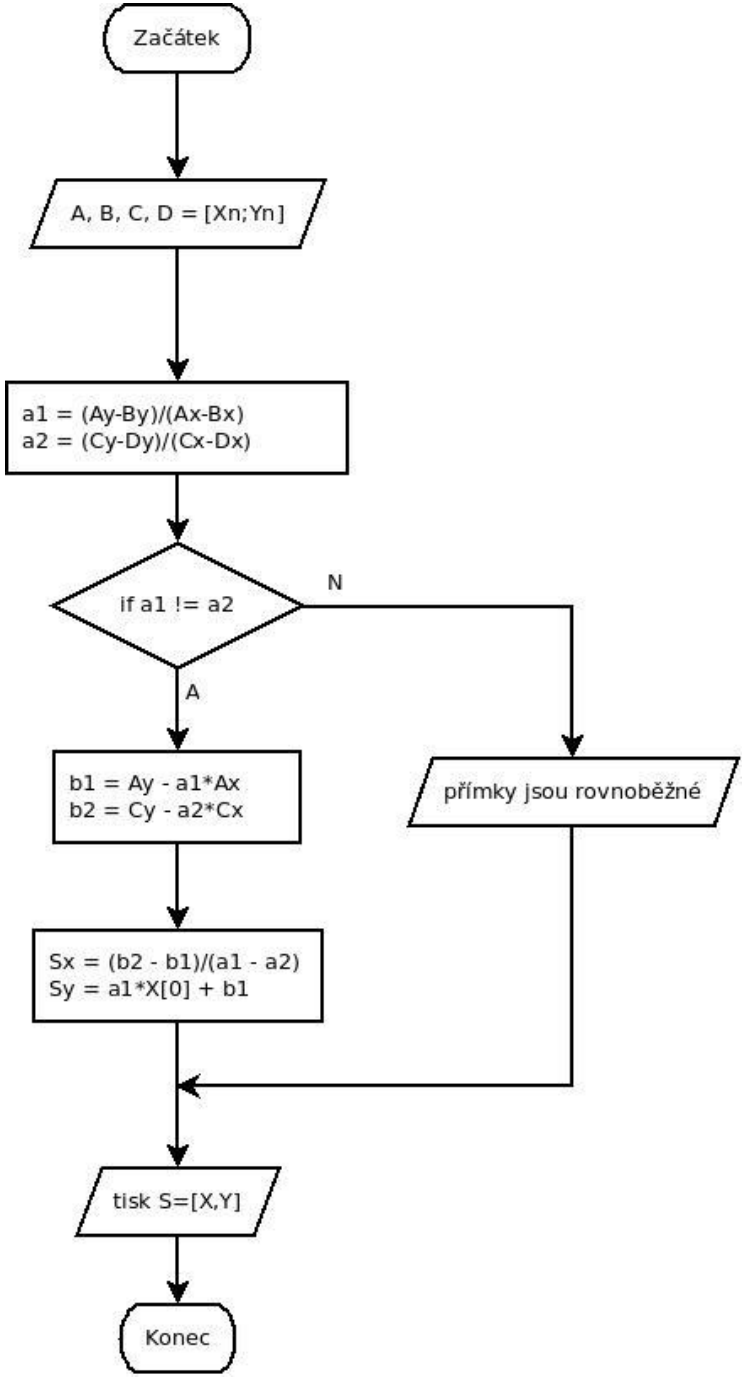
Průsečík úseček – speciální případy

1. Úsečky jsou rovnoběžné:
   - Pokud *b1*==*b2*

2. jedna z úseček je rovnoběžná s osou *x*:
   - tedy *b1*==0 či *b2* ==0 => tvar rovnice *y=a*
   - *P*[*y*] = y-ová souřadnice jednoho z bodů úsečky rovnoběžné s osou *x*

3. jedna z úseček je rovnoběžná s osou *y*:
   - tedy b1 či b2 neexistuje (tg 90 °)
   - *P*[*x*] = x-ová souřadnice jednoho z bodů úsečky rovnoběžné s osou *y*

Průsečík dvou úseček



Průsečík dvou úseček - Python

- Zadání: vytvořte skript umožňující výpočet průsečíku dvou úseček zadaných počátečním a koncovým bodem.
- 1. úsečka: body A, B; 2. úsečka: body C, D

A = [2.0,4.0]
B = [13.0,8.0]
C = [7.0,3.0]
D = [10.0,2.0]

Průsečík dvou úseček - Python

```
prusecik = [-999.0,-999.0] #vytvoreni seznamu o dvou prvcich, kam se pozdeji ulozi vypoctene souradnice pruseciku
status = 0 #promenna status rozhoduje o dalsim postupu ve zpusobu vypoctu pruseciku dle stavajici situace

#vypocet parametru b – směrnice (sklon) primky
try: #funkce try umoznuje vyzkouset konkretni prikaz(y) a urcit pomoci except, co se ma stat, pokud je nelze realizovat
    #v tomto pripade je potreba osetrit situaci, kdy je nektera z usecek rovnobezna s osou y a tudiz parametr b neexistuje
    b1 = (B[1]-A[1])/(B[0]-A[0])
except:
    print ("prvni usecka je rovnobezna s osou y")
    status = 1 #hodnota 1 oznacuje stav, kdy je prvni usecka rovnobezna s osou y

try:
    b2 = (D[1]-C[1])/(D[0]-C[0])
except:
    print ("druha usecka je rovnobezna s osou y")
    if status == 1:
        status = 3 #hodnota 3 oznacuje stav, kdy jsou obe usecky rovnobezne s osou y
    else:
        status = 2 #hodnota 2 oznacuje stav, kdy je druha usecka rovnobezna s osou y
```

Průsečík dvou úseček - Python

```
if status == 1: #podminka resici situaci, kdy je prvni primka rovnobezna s osou y, pak je x-ova souradnice
    #pruseciku rovna x-ove souradnici jednoho z jejich bodu (v nasem pripade bodu A nebo B)
    prusecik[0] = A[0]
    a2 = C[1] - b2 * C[0]
    prusecik[1] = a2 + b2 * prusecik[0]
elif status == 2: #podminka resici situaci, kdy je druha primka rovnobezna s osou y, pak je x-ova souradnice
    #pruseciku rovna x-ove souradnici jednoho z jejich bodu (v nasem pripade bodu C nebo D)
    prusecik[0] = C[0]
    a1 = A[1] - b1*A[0]
    prusecik[1] = a1 + b1 * prusecik[0]
elif status == 3:
    print ("primky jsou rovnobezne, prusecik neexistuje")
```

Průsečík dvou úseček - Python

```
else: #ani jedna z usecek neni rovnobezna s osou y, tedy pro obe z nich existuje parametr b
    #pocitam parametr a = svisly posun na ose y
    a1 = A[1] - b1*A[0]
    a2 = C[1] - b2*C[0]
    #pokud je smernice obou primek totozna, jsou rovnobezne
    if b1 == b2:
        print ("primky jsou rovnobezne, prusecik neexistuje")
    else:
        prusecik[0] = (a2-a1)/(b1-b2) #vypocet x-ove souradnice pruseciku
        prusecik[1] = a1 + b1*prusecik[0] #vyopocet y-ove souradnice pruseciku dosazenim do rovnice primky

print(prusecik)
```

Průsečík dvou úseček

- Ověření, že průsečík (ne)leží mezi dvěma body zadané úsečky:

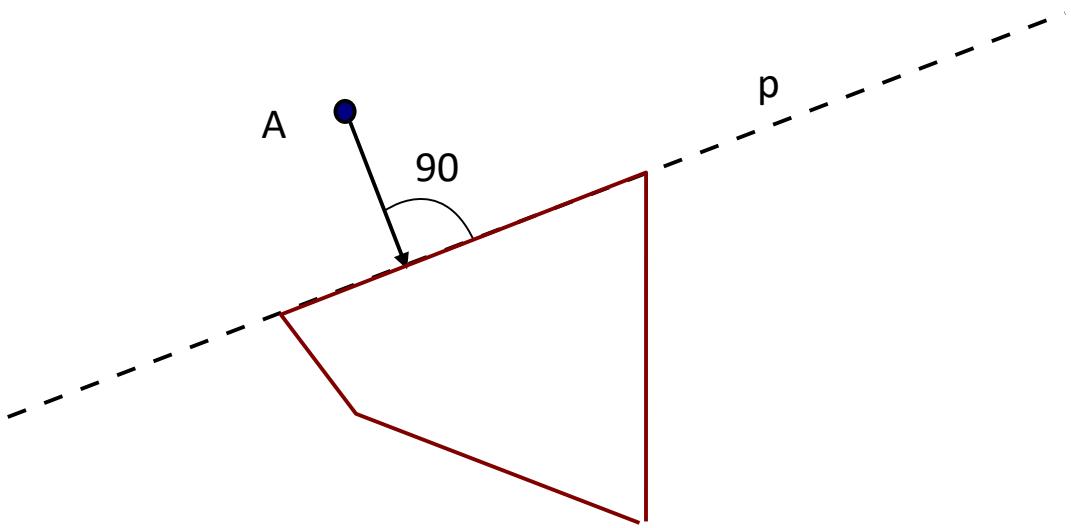
if (P1[X]-P[X])*(P[X]-P2[X])>=0 and (P1[Y]-P[Y])*(P[Y]-P2[Y])>=0

  - Porovnávají se rozsahy souřadnic

Vzdálenost bodu od přímky

Zadání:
Najděte nejmenší vzdálenost d bodu A od přímky p. Hledáme tedy $distance(A, p)$.
Řešení:
Připomeneme si analytické vzorce v eukleidovské rovině.
Označíme-li souřadnice bodu $A = [x_A, y_A]$,
a přímku p určíme body $P1$ a $P2$, kde $P1 = [x_1, y_1]$ a $P2 = [x_2, y_2]$,
pak vzorec pro vzdálenost bodu A od přímky p bude vypadat takto:

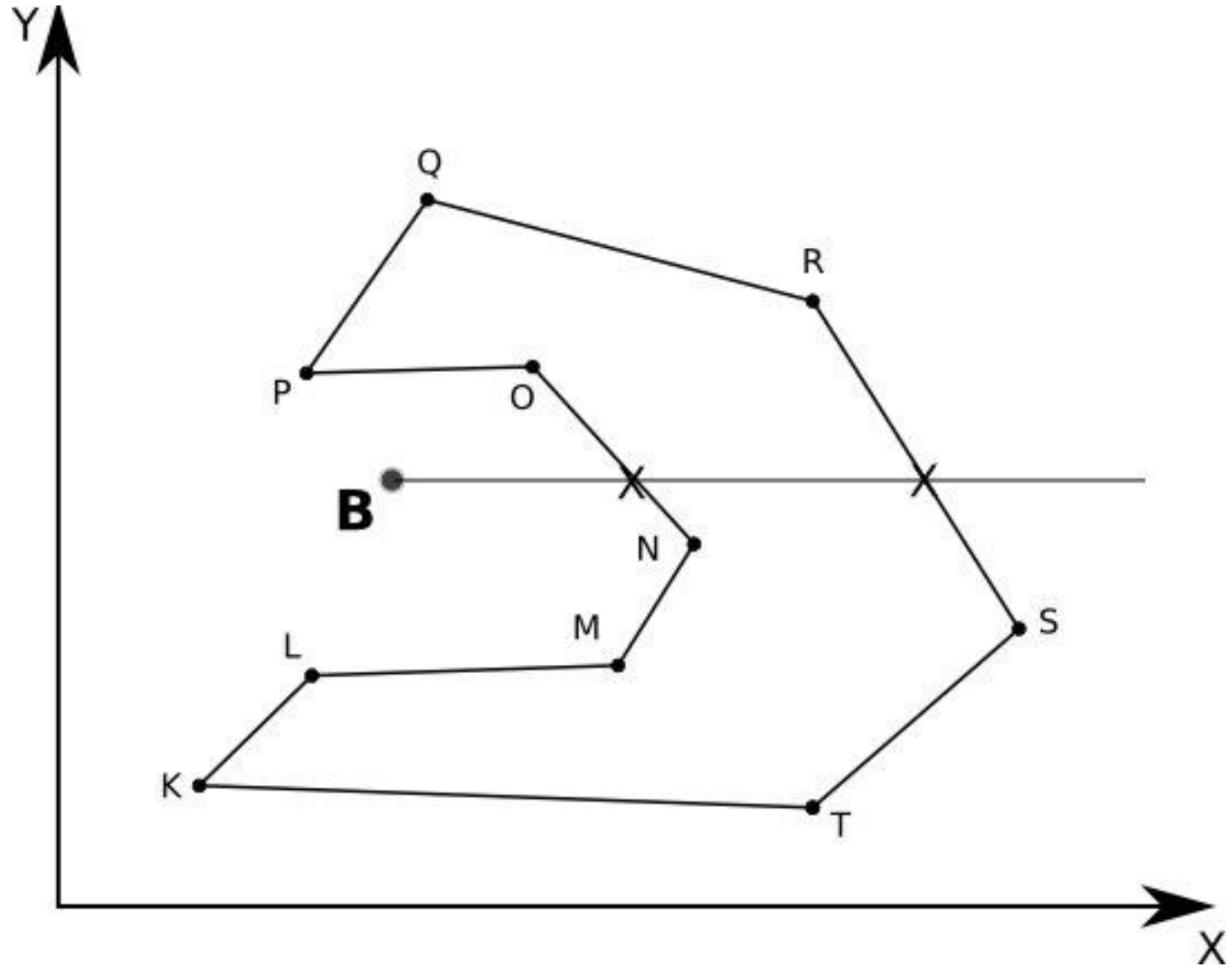
$$distance(A, p) = \frac{x_A(y_1 - y_2) + x_1(y_2 - y_A) + x_2(y_A - y_1)}{\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}}$$



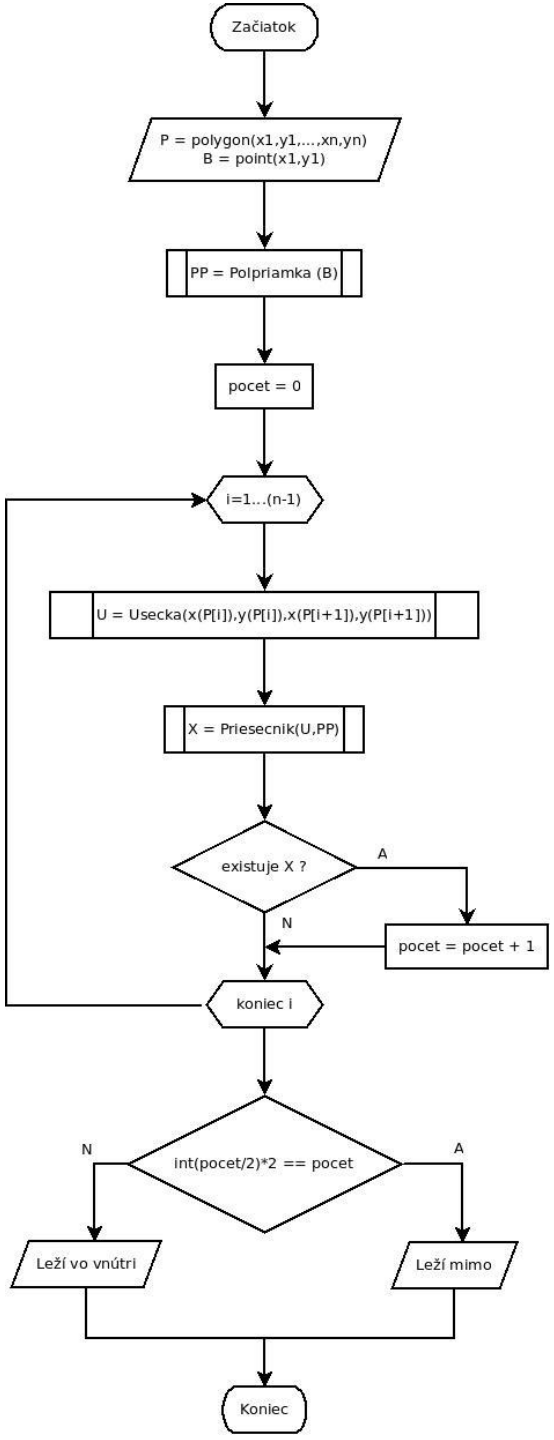
Bod v polygonu

- Testování zdali daný bod (B) leží uvnitř polygonu zadaného lomovými body
- Možno využít tzv. „Ray casting" algoritmus
- Princip:
  1. z testovaného bodu B vedeme polopřímku (úsečku), vhodný směr = tak, aby byla rovnoběžná s jednou z os
  2. v cyklu načítáme jednotlivé úsečky polygonu a počítáme počet průsečíků mezi těmito úsečkami a pomocnou úsečkou vedenou z bodu B
  3. pokud je počet průsečíků lichý -> bod v polygonu leží, pokud je sudý -> bod v polygonu neleží

Bod v polygonu



Bod v polygonu

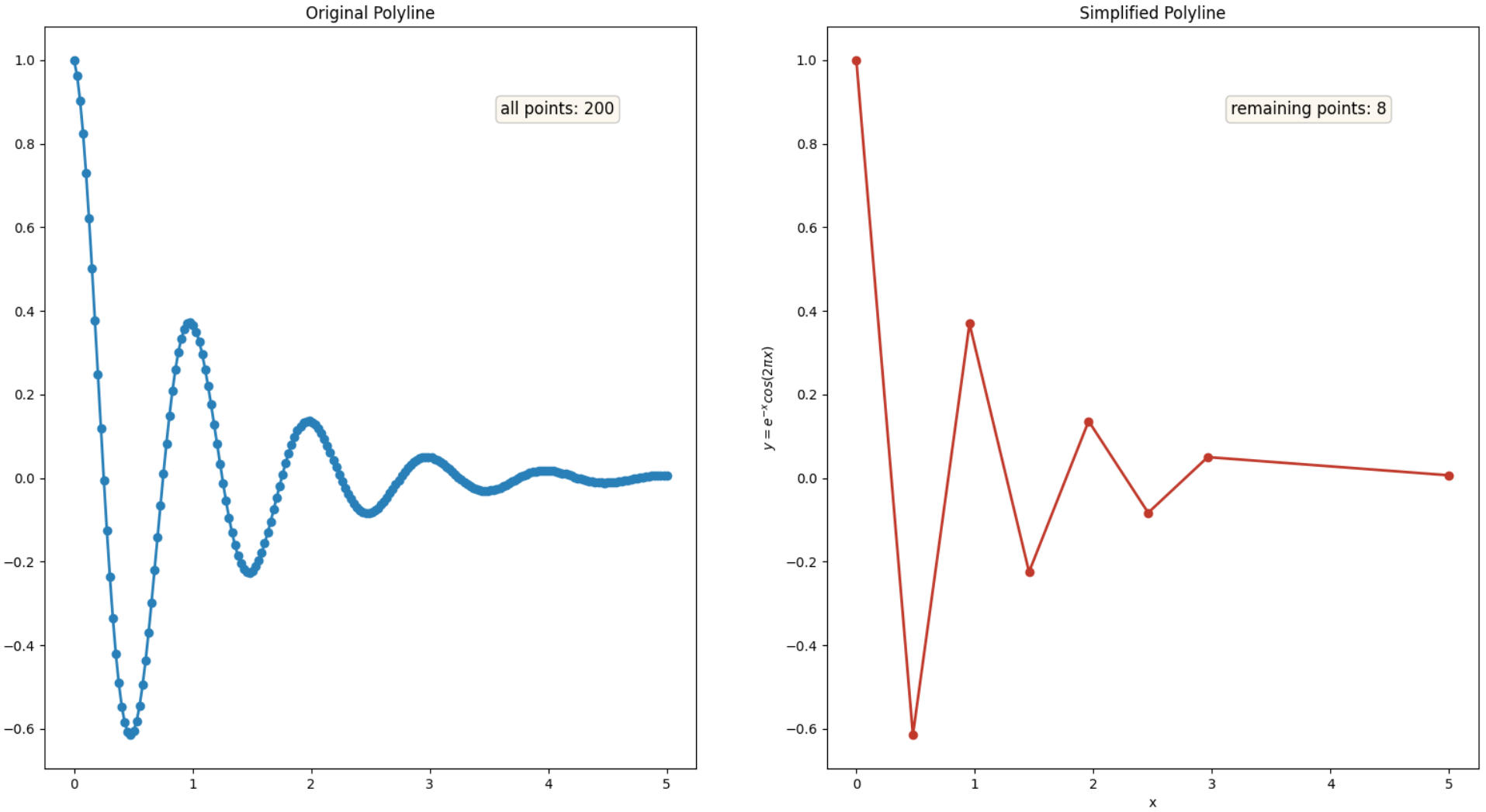


Bod v polygonu

Problémy popsaného algoritmu:

- Pokud leží bod přímo na hraně či velmi blízko k hraně polygonu, může dojít k chybnému napočítání průsečíků – kvůli použití datového typu float, případně špatnému zaokrouhlování

Generalizace geometrie linie



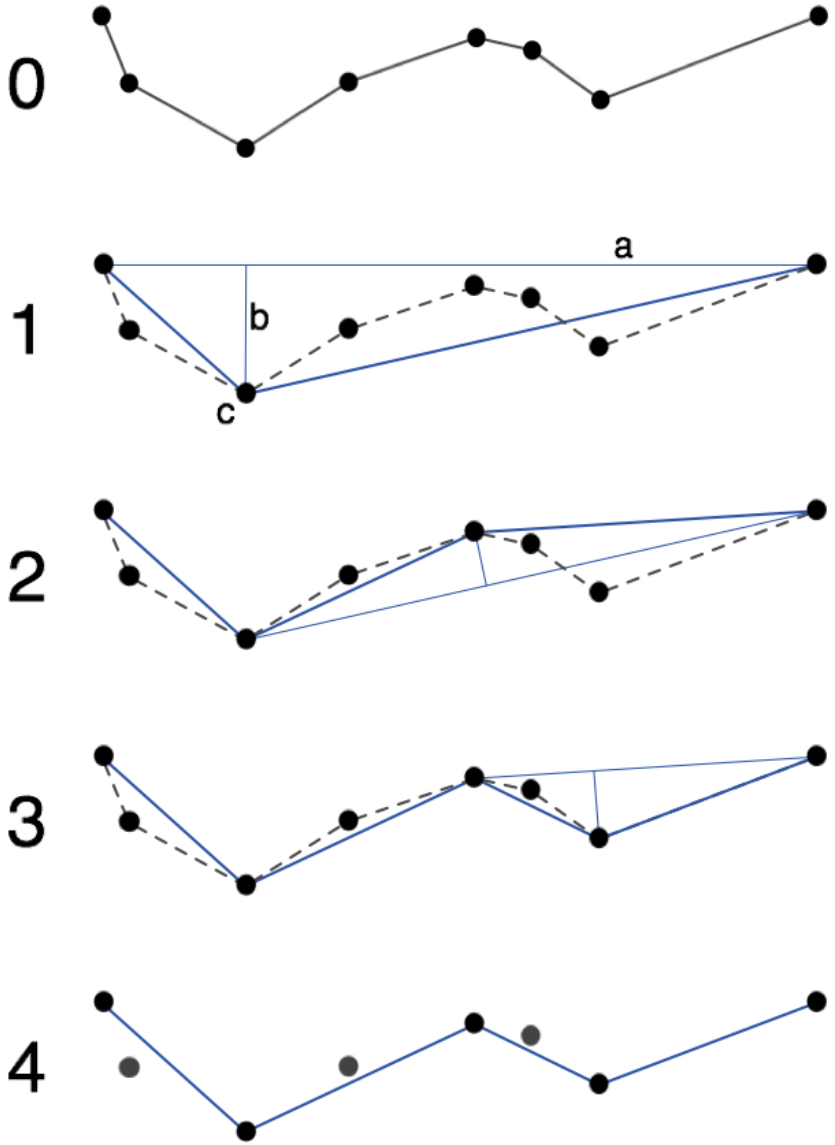
Obrázek převzat z: https://towardsdatascience.com/simplify-polylines-with-the-douglas-peucker-algorithm-ac8ed487a4a1

Generalizace geometrie linie

- Možné důvody:
  - Potřeba redukce množství dat (pro uložení, vizualizaci, atd.)
  - Generalizace kvůli měřítka mapy
- Velmi rozšířený je algoritmus Ramer–Douglas–Peucker (označovaný také jako Douglas–Peucker algoritmus)
  - jednoduchý
  - z linie na vstupu jsou postupně odebírány body (vrcholy) na základě stanovené maximální vzdálenosti definované uživatelem
  - výstupní linie se snaží co nejvíce zachovat tvar vstupní linie
  - algoritmus v základní podobě vyžaduje rozšíření, aby například nedocházelo ke tvorbě výstupní linie, která samu sebe protíná

Ramer–Douglas–Peucker algoritmus

- Vstup:
  - Linie daná sadou bodů (vrcholů)
  - Parametr epsilon specifikující maximální vzdálenost výstupní linie od vstupní



Obrázek převzat z: https://cs.m.wikipedia.org/wiki/Soubor:Douglas_Peucker.png

Ramer–Douglas–Peucker algoritmus

Princip:

- algoritmus vytvoří spojnici počátečního a koncového bodu linie a pro každý další bod vypočte kolmou vzdálenost k této spojnici
- maximální vypočtená vzdálenost je porovnána s parametrem epsilon:
  - pokud je vzdálenost větší, je vstupní linie v daném bodě rozdělena na dvě samostatné linie, které jsou dále zpracovávány samostatně
  - pokud je vzdálenost menší, bod je odstraněn

- algoritmus používá rekurzi – výše popsaný postup je opakován dokud vznikající dělené linie nemají již jen dva body, respektive dělené linie neobsahují žádné body, jejichž vzdálenost je větší než parametr epsilon

Výpočet plochy polygonu

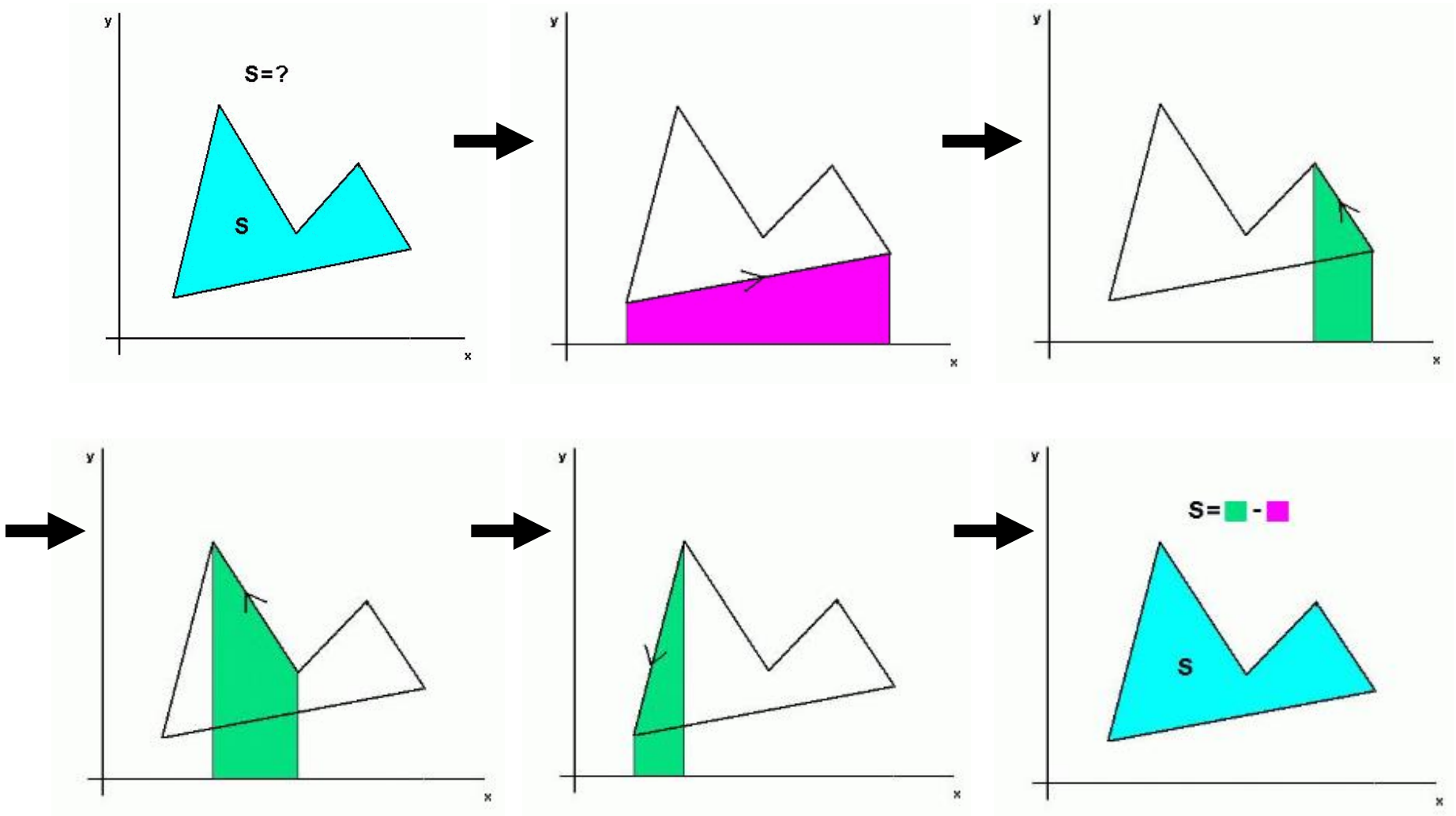
Jednoduchý způsob výpočtu plochy polygonu nepravidelného tvaru je:

1. rozdělit polygon na jednotlivé plochy reprezentované trapezoidy mezi dvojicemi bodů a osou x (či y)
2. jednoduše počítat plochu jednotlivých trapezoidů

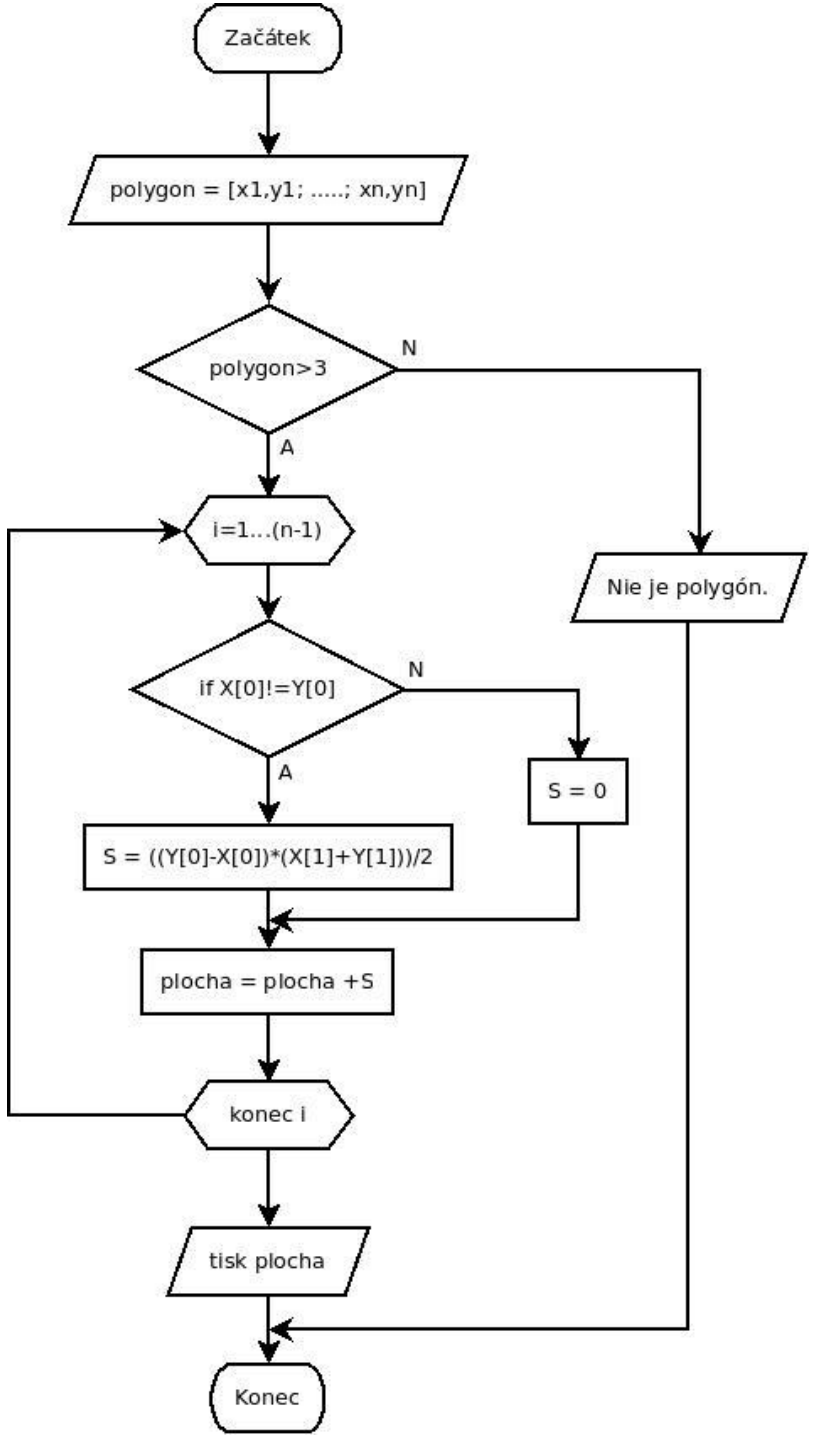
$$A = (x2-x1) * (y2+y1) / 2$$

3. součet ploch jednotlivých trapezoidů = plocha celého polygonu

Výpočet plochy polygonu



Výpočet plochy polygonu



Výpočet plochy polygonu

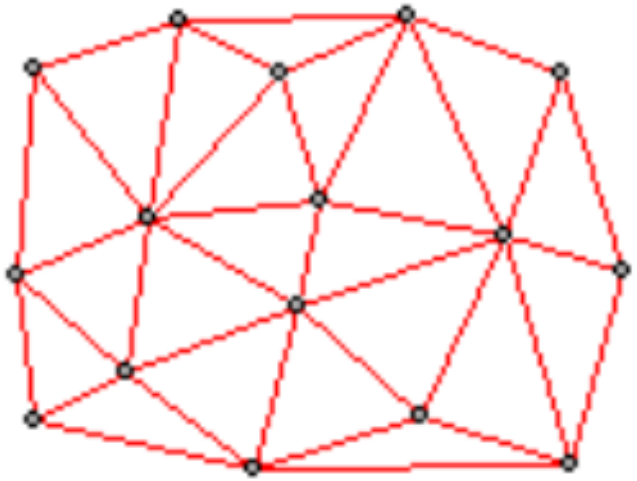
Speciální případy

- **Část polygonu nebo celý polygon leží v záporných souřadnicích** (řešení: převést celý polygon do kladných souřadnic)
- **Polygon je zapsaný v protisměru hodinových ručiček**: výsledek bude správný, ale v záporných souřadnicích (řešení: výsledek je nutné označit jako absolutní hodnotu)
- **Hranice polygonu se mezi sebou kříží**

Triangulace

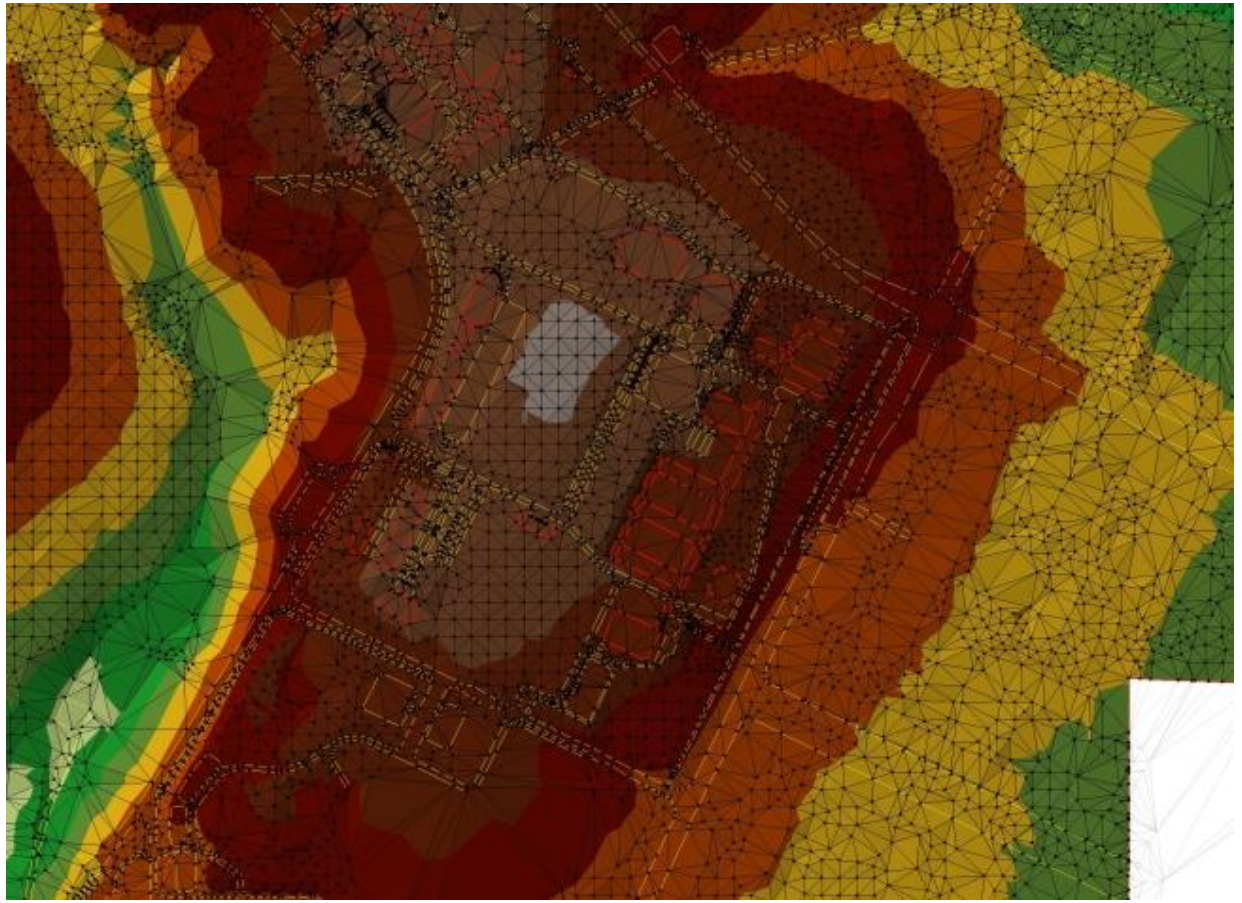
- Úloha triangulace spočívá v rozdělení roviny do sady trojúhelníků, jejichž vrcholy jsou určeny vstupní množinou bodů M = {P1, P2, …, PN}.
- Požadavky:
  - Sjednocení všech trojúhelníků tvoří konvexní obal množiny bodů M.
  - V trojúhelníku neleží žádný další bod z množiny M.
- V geoinformatice se používá zejména při tvorbě digitálních modelů terénu (DMT) či 3d modelů objektů.

Triangulace



Obrázky převzaty z
http://www.georeference.org/doc/transform_triangulation.htm

Triangulace v rovině



Obrázek převzat z http://gis.vsb.cz/vojtek/index.php?page=git-fast/cviceni02

Triangulace

Způsob geometrické konstrukce určuje metody řešení:

- Delaunayho triangulace,
- Greedy triangulace,
- triangulace s povinnými hranami (Constrained triangulace),
- ...

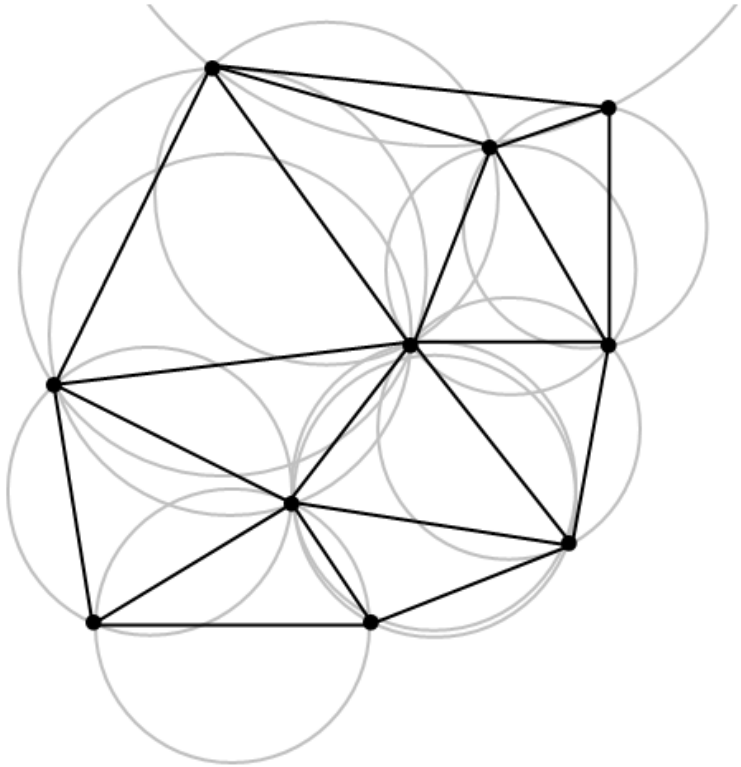
Delaunayho triangulace

Nejčastější triangulace, trojúhelníky se blíží rovnostranným (těm, co nejlépe reprezentují lokální povrch)

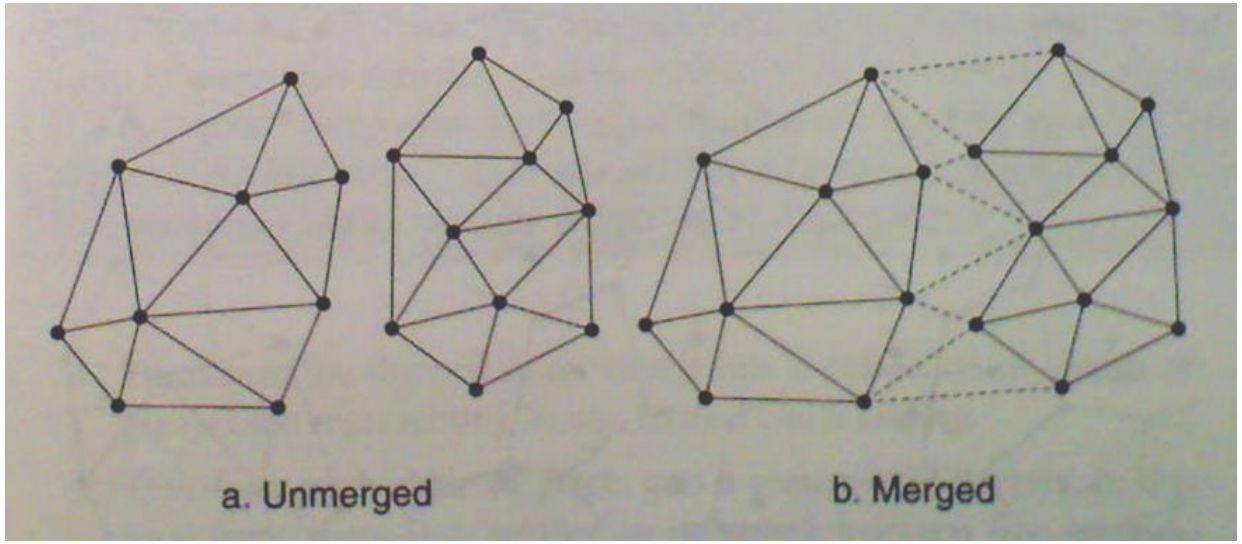
Princip:

- algoritmus vezme tři body, proloží jimi kružnici, když uvnitř kružnice neleží bod, vytvoří trojúhelník, pokud tam bod je, vybere jiné tři body
- tento proces je prováděn rekurzivně (možnost rozdělit celou oblast na menší části, ty řešit a pak poskládat dohromady)

Delaunayho triangulace



Obrázek převzat z
http://74fdc.wordpress.com/2012/03/01/delaunay-triangulation-creating-a-dynamic-design-expression/



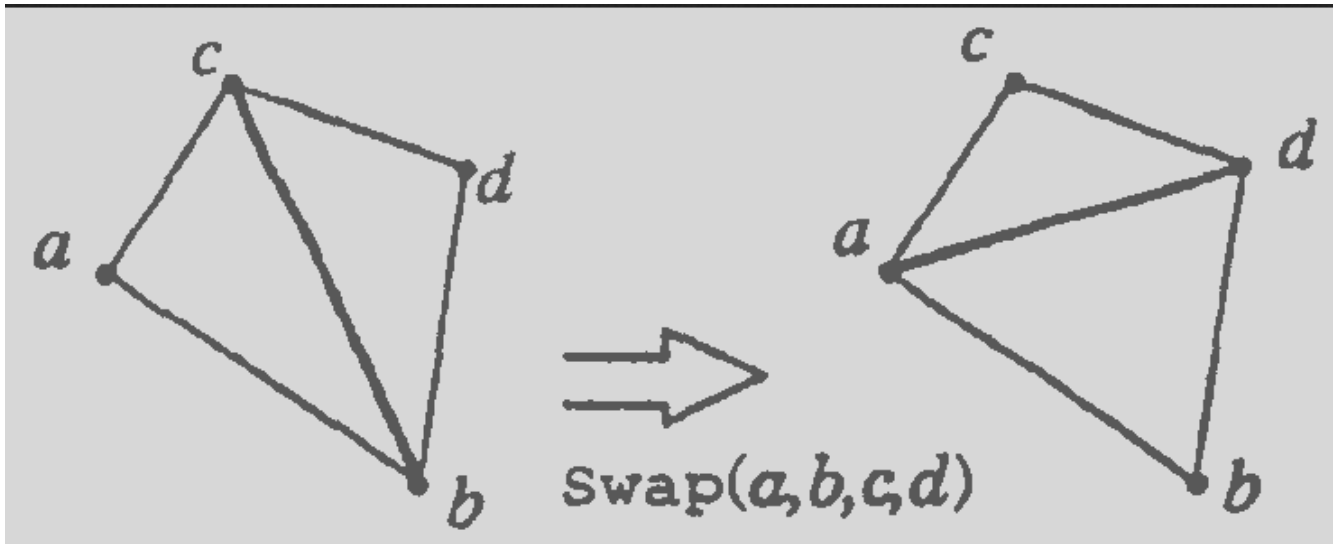
Delaunayho triangulace

Vlastnosti:

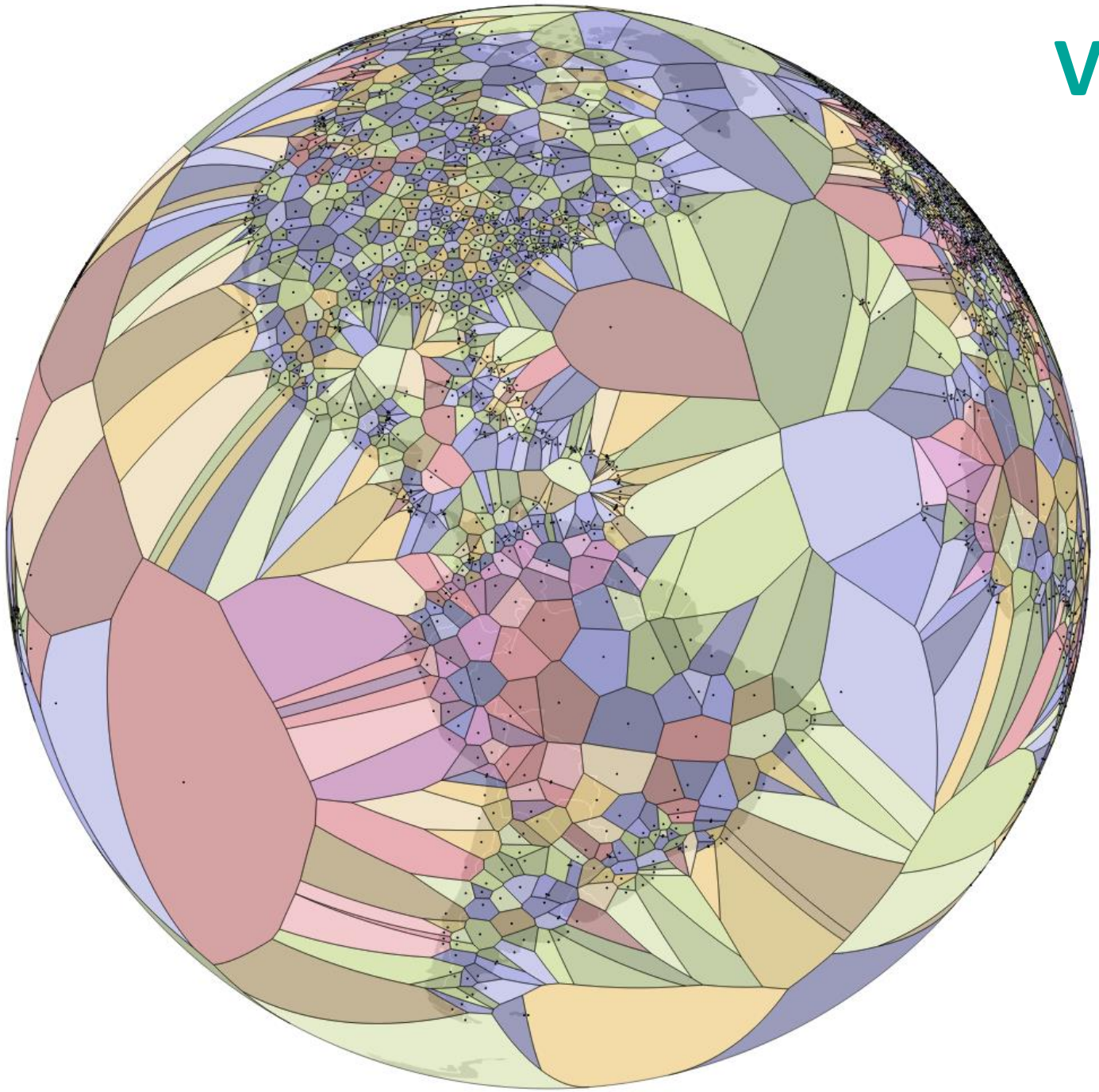
- Je maximalizován minimální úhel pro každý trojúhelník (není minimalizován maximální).
- Je dodržen princip optimálního i globálního kritéria minimálního úhlu.
- Výsledek je jednoznačný, pokud nejsou 4 body na kružnici.

Delaunayho triangulace

Prohazování hran řeší lokální optimalizaci. Označuje se pojmem legalizace a vychází z ověřování polohy protějších vrcholů vůči opsané kružnici.



Obrázek převzat z
http://patentimages.storage.googleapis.com/WO1999034336A1/imgf000143_0001.png



Voronoiovy diagramy

neboli Thiessenovy polygony

Vstup = sada bodů

Výstup = polygonová vrstva, polygony jsou vytvářeny okolo každého bodu tak, aby celá plocha polygonu byla nejblíže k bodu vstupní vrstvy, který v polygonu leží

https://www.jasondavies.com/maps/voronoi/airports/

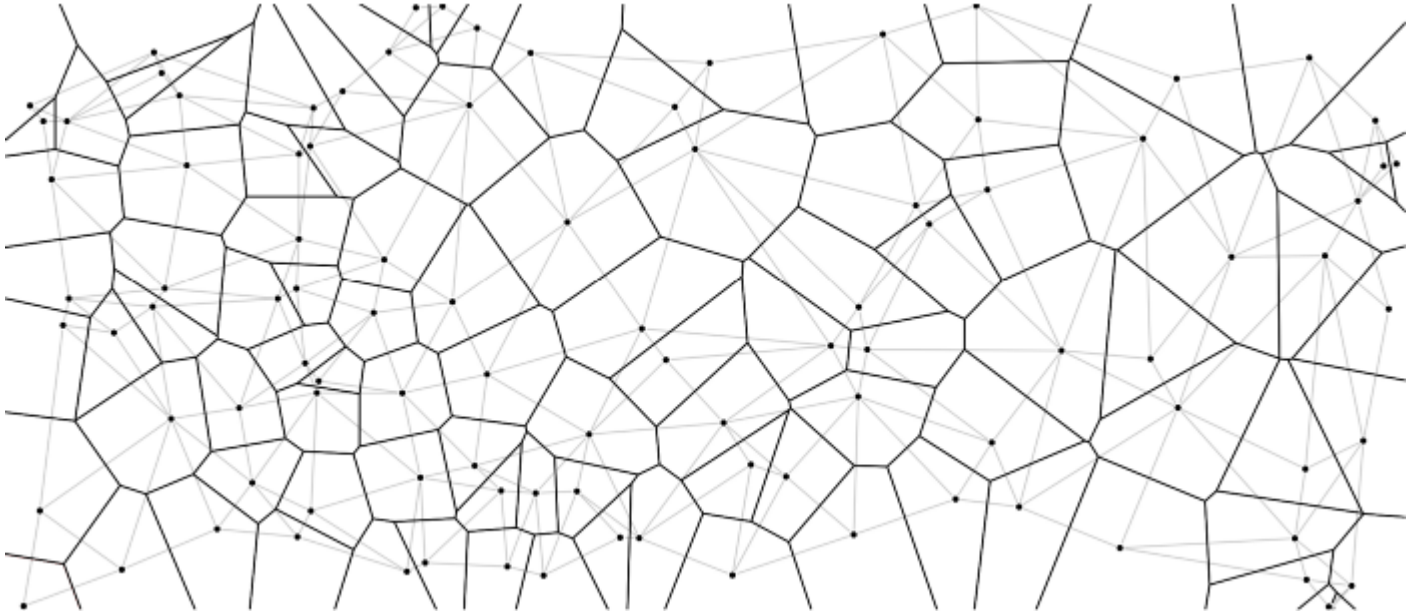
Voronoiovy diagramy

Pro tvorbu Voronoivých diagramů existuje několik algoritmů:

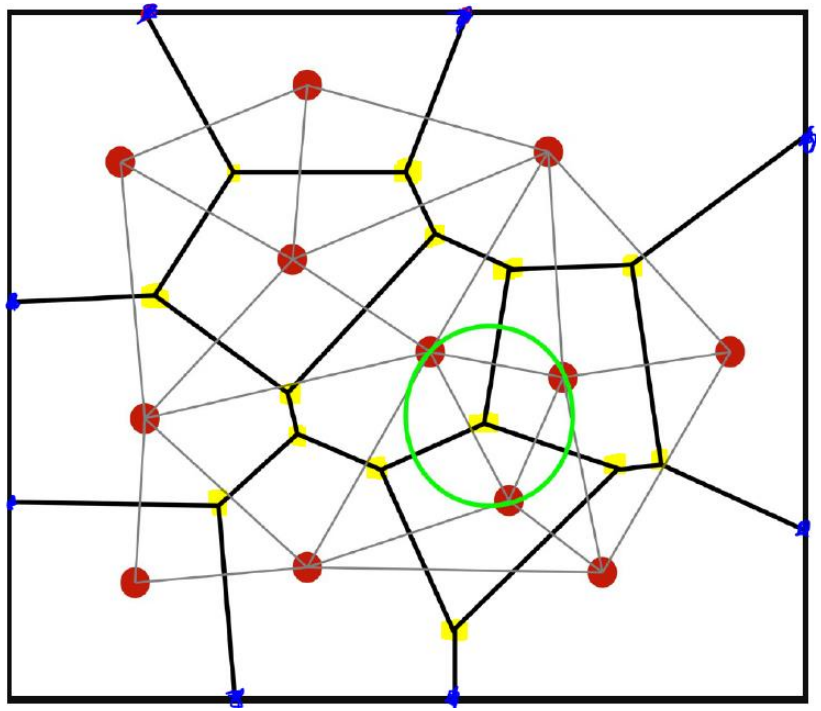
- Llyodův relaxační algoritmus (iterační)
- Fortuneův algoritmus
- Jump flooding algoritmus

- Většina algoritmů je přímo či nepřímo založena na vytvoření Delaunayho triangulace a následném odvození Voronoiova diagramu vytvořením duálního grafu z výsledku triangulace
  - středy kružnic Delaunayho triangulace tvoří vrcholy polygonů Voronoiva diagramu

Voronoiovy diagramy



Obrázek převzat z https://towardsdatascience.com/the-fascinating-world-of-voronoi-diagrams-da8fc700fa1b



Obrázek převzat z https://stackoverflow.com/questions/59869376/calculating-the-infinity-points-edges-of-the-delaunay-triangulation

Topologie

- topologie umožňuje modelovat prostorové vztahy mezi objekty jedné třídy (vrstvy) či objekty dvou samostatných tříd
- topologická pravidla lze použít pro sledování těchto vztahů a jejich dodržování
- nedodržení těchto pravidel lze pomocí editačních nástrojů najít a opravit.
- nelze ale předcházet vzniku chyb v datech samotných

Topologické vazby – bod

- bod – bod: pracuje se jedině se vzdáleností dvou bodů (nejbližší bod k danému bodu apod.)
- bod – linie:
  - Musí ležet na linii
  - Musí být koncovým bodem linie
  - Musí být dále/blíže než bod linie
- bod – polygon:
  - musí být uvnitř polygonů
  - musí ležet na hranici

Topologické vazby – linie

- Jedna linie:
  - nesmí mít volné konce
  - nesmí překrývat/protínat samy sebe

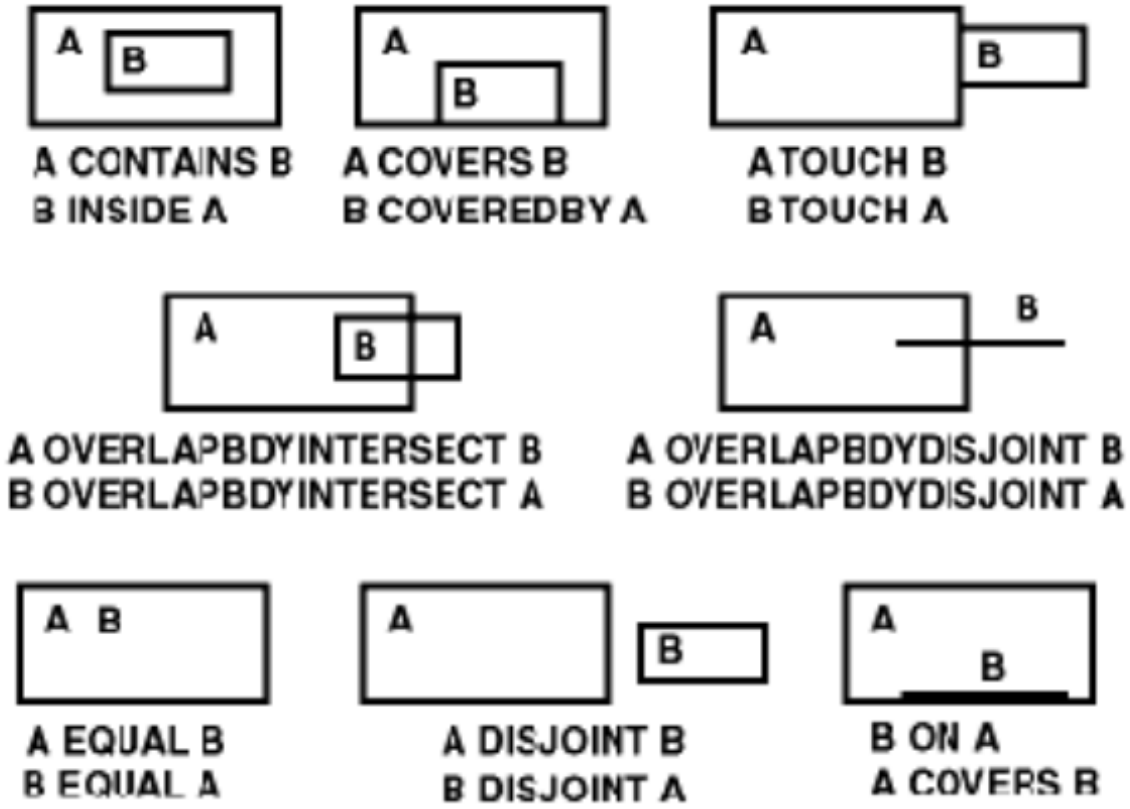
- Linie – linie:
  - nesmí se protínat/překrývat
  - musí/nesmí mít s jinou linií společný koncový bod (návaznost).

- Linie – polygon:
  - musí ležet na hranici polygonu

Topologické vazby – polygon

- Polygon – polygon:
  - polygony (jedné vrstvy) se nesmí překrývat,
  - obsažnost – jeden polygon obsahuje druhý nebo je obsažen (contains, overlaps),
  - musí mít totožné hranice (polygony dvou vrstev)

Implementace topologických vztahů



Obrázek převzat z http://docs.oracle.com/html/A88805_01/sdo_intr.htm

Zdroje

- Douglas, D. H., & Peucker, T. K. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: The International Journal for Geographic Information and Geovisualization, 1973, 10(2), 112–122. doi:10.3138/fm57–6770-u75u-7727
- Egenhofer, M., Sharma, J., David, M.: A critical comparison of the 4-intersection and 9-intersection models for spatial relations: formal analysis. Auto-Carto, 1993.
- Fuksa, M.: Delaunayova triangulace s omezením (CDT) v E2 a E3, DP. Plzeň 2006
- http://ibis.geog.ubc.ca/courses/klink/gis.notes/ncgia/u32.html#SEC32.3.4
- https://www.baeldung.com/cs/voronoi-diagram
- https://towardsdatascience.com/the-fascinating-world-of-voronoi-diagrams-da8fc700fa1b
- http://download.arcdata.cz/doc/TopologiePlakat.pdf
- https://towardsdatascience.com/simplify-polylines-with-the-douglas-peucker-algorithm-ac8ed487a4a1

Děkuji za pozornost

Michal Kačmařík

michal.kacmarik@vsb.cz

www.vsb.cz