Základní geometrické algoritmy

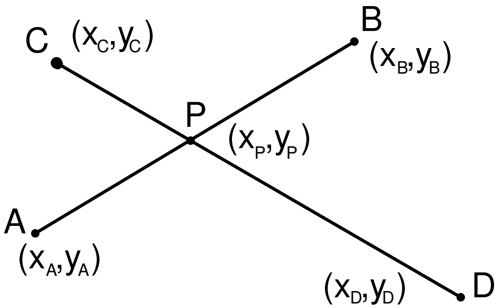
Radek Pelánek

IV122

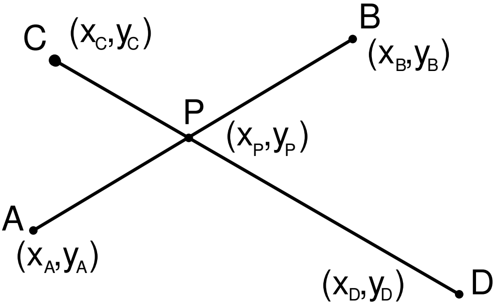
Geometrické algoritmy a speciální případy

- geometrické algoritmy často vyžadují ošetření speciálních případů (vedou na dělení nulou apd), např.:
  - rovnoběžné přímky
  - kolmé přímky
  - průsečík v koncovém bodě
- pro zjednodušení budeme zde ignorovat (při náhodně generovaných vstupech nastává s velmi malou pravděpodobností)

Hledání průsečíku



Hledání průsečíku



$$x_P = \frac{(x_A y_B - y_A x_B)(x_C - x_D) - (x_A - x_B)(x_C y_D - y_C x_D)}{(x_A - x_B)(y_C - y_D) - (y_A - y_B)(x_C - x_D)}$$

Hledání průsečíku

The intersection P of line L_1 and L_2 can be defined using determinants.

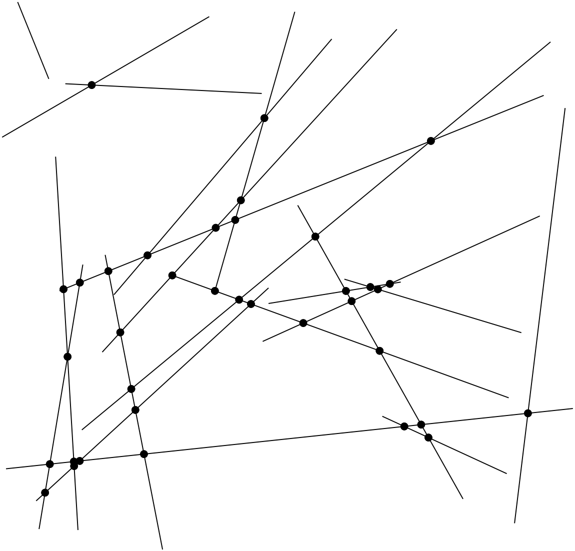
$$P_x = \frac{\begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} \\ \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} \end{vmatrix}}{\begin{vmatrix} \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \\ \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}} \qquad P_y = \frac{\begin{vmatrix} \begin{vmatrix} x_1 & y_1 \\ x_2 & y_2 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \\ \begin{vmatrix} x_3 & y_3 \\ x_4 & y_4 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}}{\begin{vmatrix} \begin{vmatrix} x_1 & 1 \\ x_2 & 1 \end{vmatrix} & \begin{vmatrix} y_1 & 1 \\ y_2 & 1 \end{vmatrix} \\ \\ \begin{vmatrix} x_3 & 1 \\ x_4 & 1 \end{vmatrix} & \begin{vmatrix} y_3 & 1 \\ y_4 & 1 \end{vmatrix} \end{vmatrix}}$$

The determinants can be written out as:

$$(P_x, P_y) = \left(\frac{(x_1y_2 - y_1x_2)(x_3 - x_4) - (x_1 - x_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)}, \frac{(x_1y_2 - y_1x_2)(y_3 - y_4) - (y_1 - y_2)(x_3y_4 - y_3x_4)}{(x_1 - x_2)(y_3 - y_4) - (y_1 - y_2)(x_3 - x_4)} \right)$$

http://en.wikipedia.org/wiki/Line-line_intersection

Hledání všech průsečíků



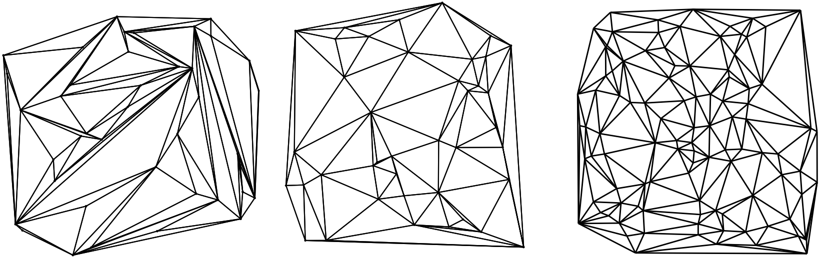
Hledání všech průsečíků

- vstup: N úseček
- výstup: seznam všech průsečíků
- algoritmus:
  - přímočarý: $O(N^2)$
  - „sweep line“: $O((N + k) \log N)$, kde k je počet průsečíků)

Triangulace

- rozdělení dvojrozměrného objektu/prostoru na trojúhelníky
- motivace: s trojúhelníky se snadno pracuje
- častý první krok u složitějších geometrických operací

Triangulace



Jak sestrojit triangulaci?

Co je „pěkná“ triangulace?

Triangulace: základní algoritmus

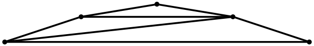
```
def triangulace(body):
  inicializuj prázdný výběr
  pro všechny úsečky U mezi body:
    pokud se U neprotíná s žádnou úsečkou ve výběru:
      přidej U do výběru
  return výběr
```

Minimální triangulace konvexního mnohoúhelníku

- vstup: konvexní mnohoúhelník
- výstup: minimální triangulace
- kritérium: minimální součet délek hran
- hladový algoritmus:
  - v každém kroku bereme nejkratší hranu, která neprotíná žádnou z již přidaných hran
  - není optimální – najděte protipříklad
- optimální řešení – dynamické programování

Minimální triangulace konvexního mnohoúhelníku

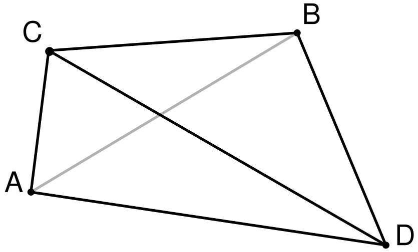
hladová triangulace



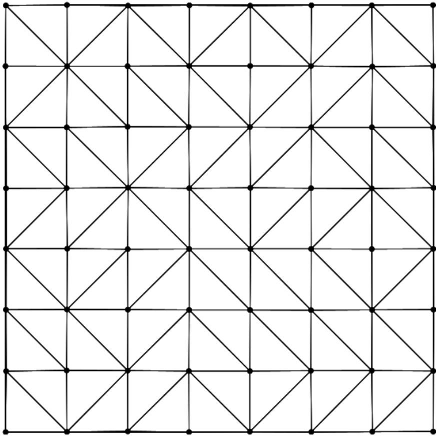
minimální triangulace

Delaunayova triangulace

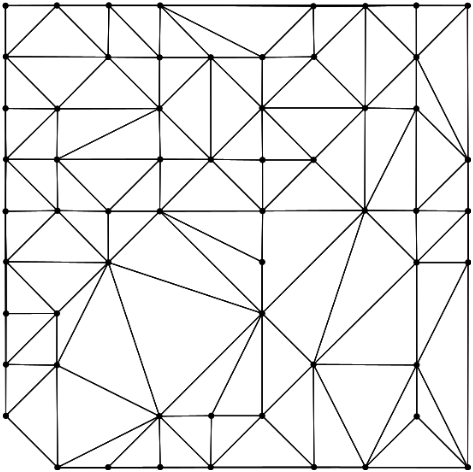
- kritérium: maximalizace minimálního úhlu
- alternativně: žádný bod uvnitř kružnice opsané trojúhelníku v triangulaci
- spojitost Voronei diagram
- algoritmus: prohazování hran



Hrátky s triangulací



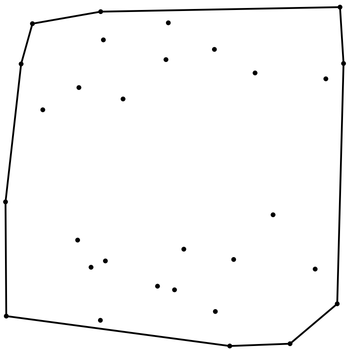
Hrátky s triangulací

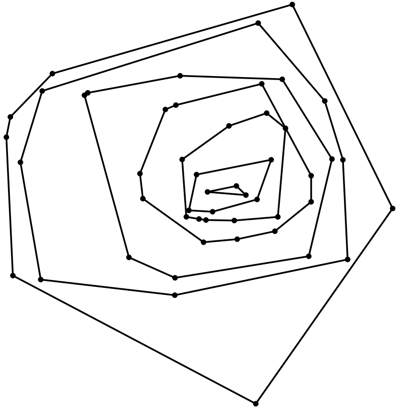


Konvexní obal

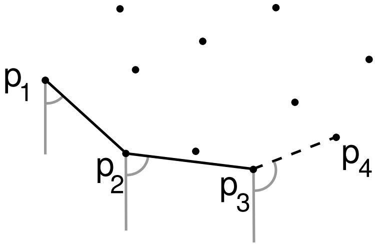
- Množina M je konvexní, pokud pro každé dva body z této množiny platí, že všechny body na jejich spojnici leží v M.
- Konvexní obal množiny bodů je nejmenší konvexní množina, která obsahuje všechny dané body.

Konvexní obal



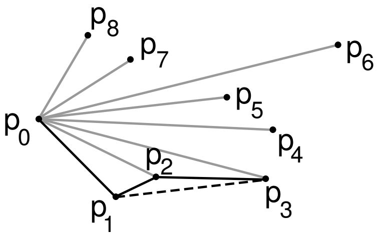


Konvexní obal: Jarvisův algoritmus



časová složitost: $O(nh)$, kde n je celkový počet bodů a h je počet bodů tvořících konvexní obal

Konvexní obal: Grahamův algoritmus



časová složitost: $O(n \log n)$