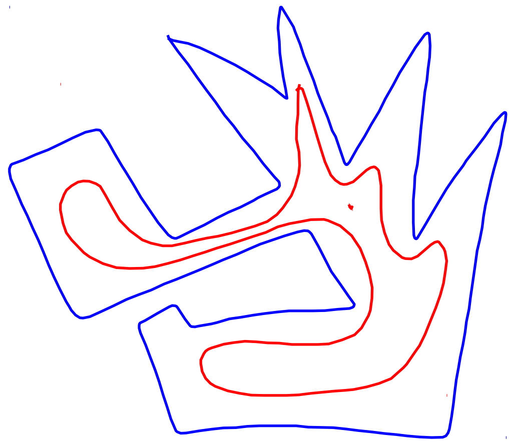
TRIANGULACE MNOHOÚHELNÍKŮ

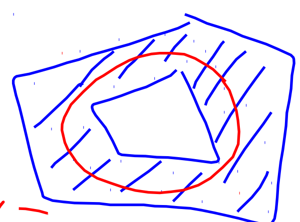
Mnohoúhelník P – jednoduchý (jednoduše souvislý)

– každá uzavřená křivka v P lze stáhnout do bodu

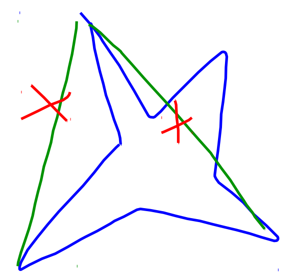
je

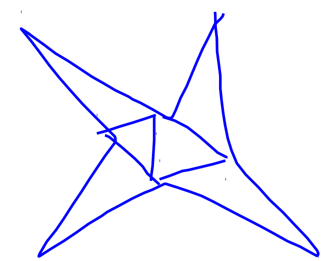


není jednoduchý

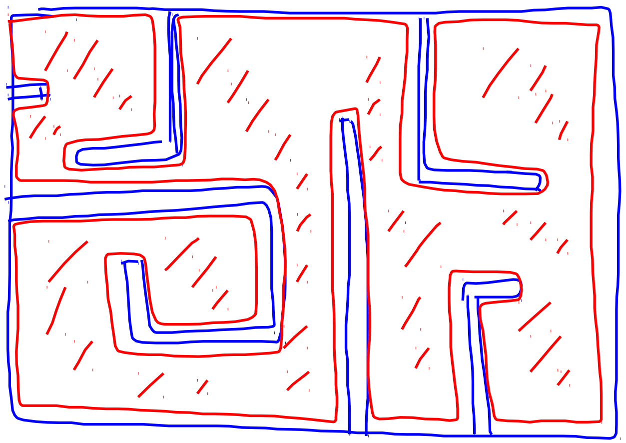


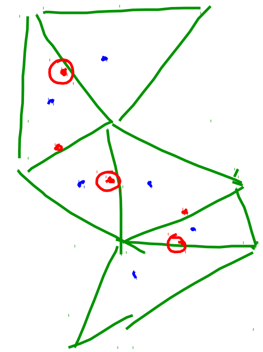
Triangulace mnohoúhelníka je jeho rozdělení na trojúhelníky, jejichž strany jsou spojnice vrcholů a leží v P.





Motivace pro triangulaci – hlídání galerie pomocí kamer.



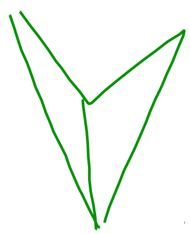


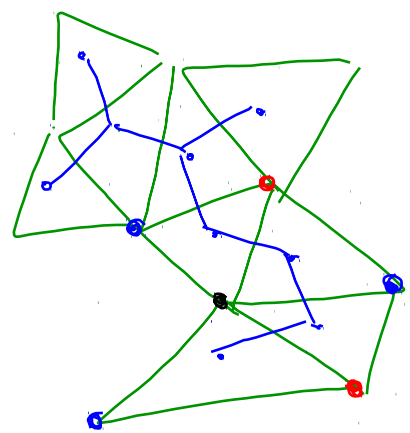
počet kamer

$=$ počet trojúhelníků

$=\dfrac{\text{počet trojúhelníků}}{2}$

dá se ukázat, že je $n-2$





obarvíme vrcholy pomocí 3 barev tak, aby každý △ měl od každé 1.

duální graf je strom

při procházení stromu barvíme vrcholy jedné barvy.

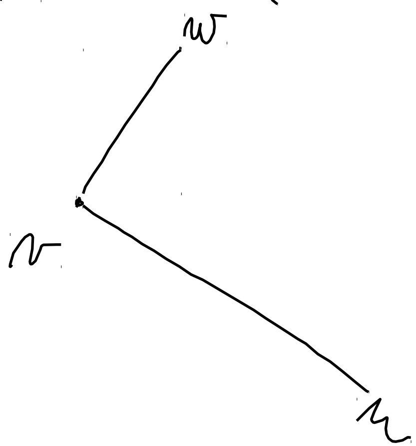
Kamery dáme do vrcholů jedné barvy.

n-úhelník kamer je $\lfloor \frac{n}{3} \rfloor$.

Věta: Každý jednoduchý n-úhelník lze triangulovat pomocí $(n-2)$ trojúhelníků.

Dk: Nechť to platí pro nějaká $k < n$.

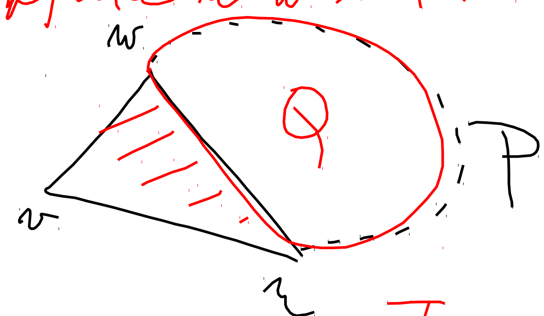
V n-úhelníku nalezneme vrchol nejvíce vlevo



Spoj vu, vw

2 možnosti

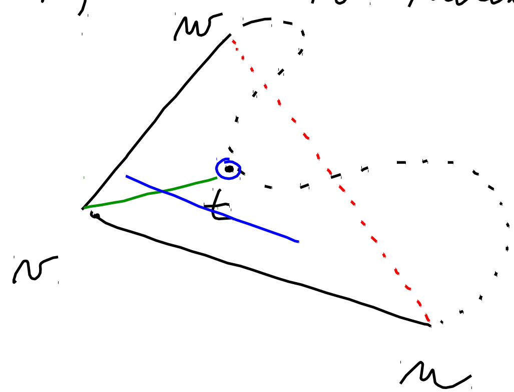
1. spojnice u w leží v P



Q je $(n-1)$-úhelník a ten jde rozdělit na $(n-3)$ $\triangle$

Tedy P lze rozdělit na $(n-2)$ $\triangle$.

(2) Spojnice $u\,w$ neleží celá v P



Vezmeme vrchol mnohoúhelníka $t \in P$ ležící v $\Delta\, v\,u\,w$ nejdále od strany $u\,w$. Pak spojnice $t\,v$ leží celá v P.

Kdyby neležela, protínala by nějakou stranu mnohoúhelníka P a t by nebyl vrchol P ležící nejdál od $u\,w$.

vt rozděluje P na 2 mnohoúhelníky P_1 a P_2 o počtu vrcholů n_1, n_2, $n_1 + n_2 = n+2$. Použijeme ind. předpoklad

P_1 rozdělíme na $n_1 - 2$ Δ

P_2 rozdělíme na $n_2 - 2$ Δ

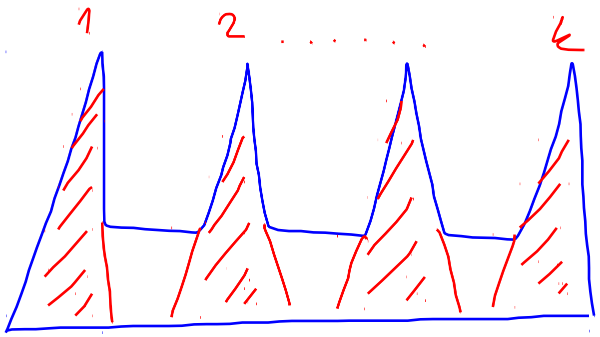
Tedy P rozdělíme na $n_1 - 2 + n_2 - 2 =$
$= n + 2 - 2 - 2 = n - 2$ Δ.

Malá poznámka ke kamerám:

Nemůže jich obecně být méně než $\left[\frac{n}{3}\right]$.

$n = 3k$ n-úhelník

$k = 4$



Potřebujeme aspoň $k = \frac{n}{3}$ kamer.

ALGORITMUS PRO TRIANGULACI

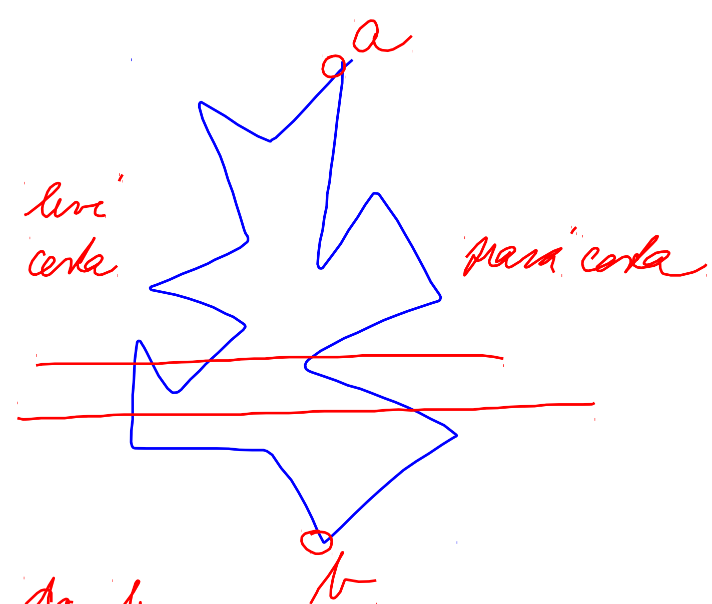
Triangulace konvexních mnohoúhelníků je jednoduchá.

Obecnější mnohoúhelníky, které lze jistě dobře triangulovat jsou tzv. monotónní mnohoúhelníky vzhledem k ose y.

<u>Mnohoúhelník</u>

nejvyšší vrchol

nejnižší vrchol



Monotónní mnohoúhelník vzhledem k ose y

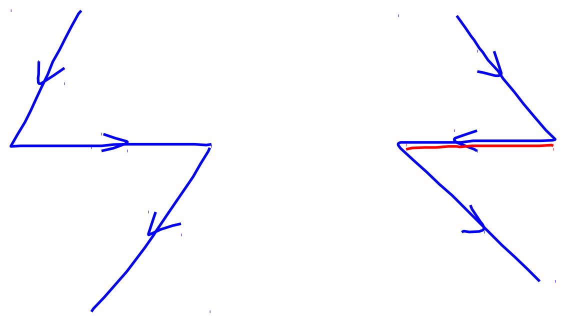
- po cestě levé i pravé od a do b klesá souřadnice y

<u>Fakt</u>: Každá přímka s rovnicí $y=k$ protíná monotónní mnohoúhelník v konvexní množině.

Přesná definice vyžaduje lexikografické uspořádání

$$p > q \iff p_y > q_y \text{ nebo } p_y = q_y \text{ a } p_x < q_x$$

Mnohoúhelník je monotónní, jestliže levá i pravá cesta jsou klesající ve smyslu lexikograf. uspořádání.

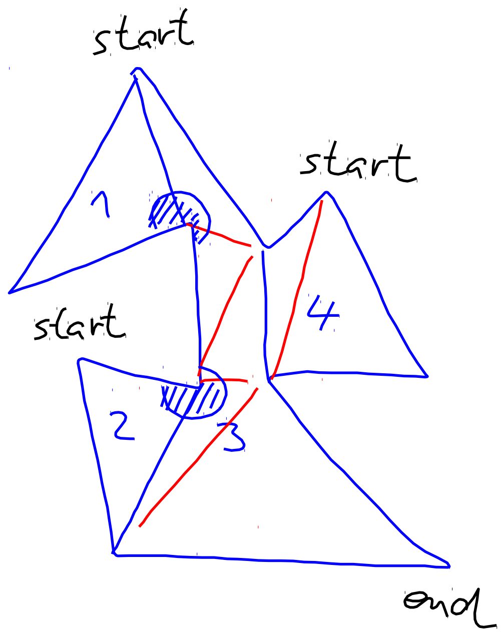


klesající

není klesající

ALGORITMUS PRO TRIANGULACI MÁ 2 ČÁSTI:

1. Rozdělení mnohoúhelníku P na monotónní části
2. Rozdělení monotónních mnohoúhelníků na trojúhelníky.



Nyní k části 1.

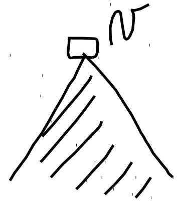
Popis mnohoúhelníka je dán dvojitě zřetězeným seznamem.

Rozdělení mnohoúhelníka na monotónní části je opět rovinné podrozdělení, které lze popsat dvojitě zřetězeným seznamem.

Algoritmus je založen na metodě zametací přímky.

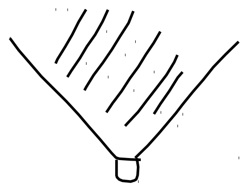
Události budou vrcholy mnohoúhelníka – rozdělíme je do několika typů.

start



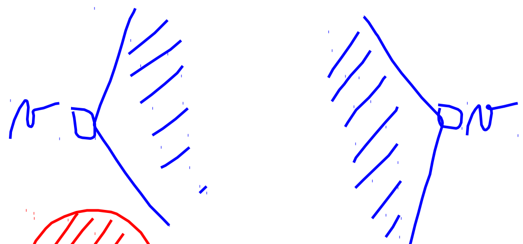
cesta přes v je zleva: nahoru dolů

end



cesta přes v je zleva: dolů - nahoru

regulární



dolů - dolů

split

mnohoúhelník nad v úhlu > 180°

merge

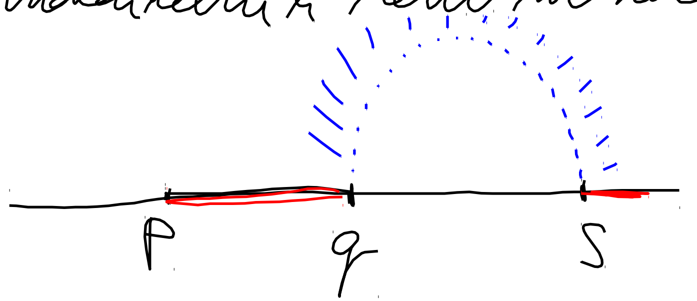
mnohoúhelník pod v úhlu > 180°

Lemma: Mnohoúhelník je monotónní, právě když nemá split a merge vrcholy.

Důkaz: Má-li split nebo merge => není monotónní

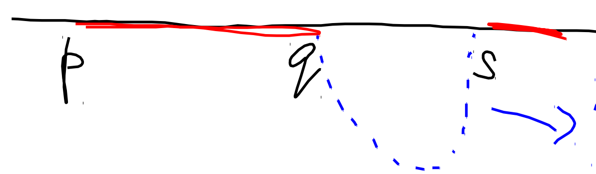
průnik není konvexní

Mnohoúhelník není monotónní => má split nebo merge vrchol



nejvyšší vrchol této cesty je split

q a s spojíme obvodem mnohoúhelníka



nejnižší vrchol je merge

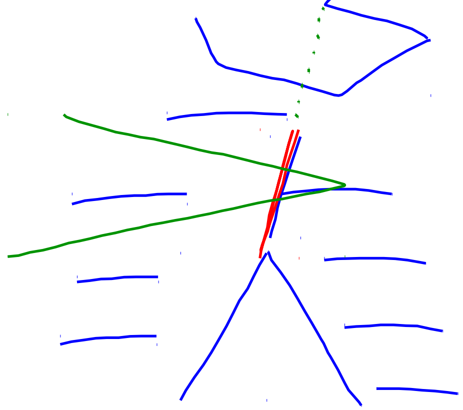
Cílem algoritmu je dělit mnohoúhelník tak, abychom odstranili split a merge vrcholy.

Fronta událostí Q … uspořádané (v daném lexikografickém uspořádání) řadí vrcholů.

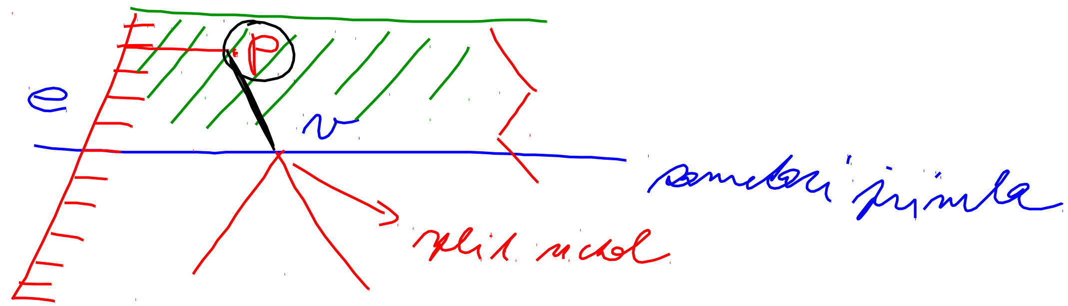
Binární vyvážený strom T bude udržovat pořadí hran mnohoúhelníka, které protínají zametací přímku (časem upřesníme)

V událostech
- přidáváme hrany
- měníme T
+ další věci – bude upřesněno

Idea odstranění split vrcholů



chceme ho spojit s nejbližším vrcholem nad ním.



p se nazývá pomocník (helper) hrany e (ta má vnitřek zprava.

p = vrchol splňující

(1) vlevo od něj je hrana e, nejbližší levý soused po v

(2) z těchto vrcholů je p nejblíže k zametací přímce (má nejmenší souřadnici y

Upřesnění stromu T .... drží pořadí hran protínajících zametací přímku. Tj. ta však pouze hrany, které mají mnohoúhelník zprava.

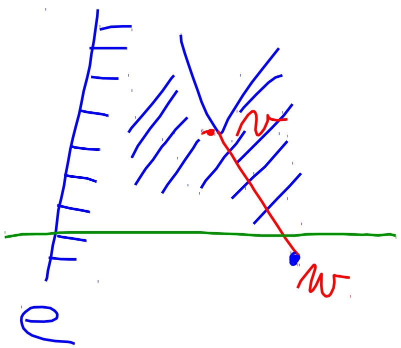
Ke každé hraně určíme její helper
helper(e) = vrchol mnohoúhelníku, který lze s e spojit vodorovnou úsečkou ležící v P a je nejblíže nad zametací přímkou.

Ve stromu T držíme s každou hranou její helper.

Při průchodu událostí z T některé hrany vypustíme
některé hrany přidáme
u některých změníme helper

Odstranění split vrcholu se provádí v okamžiku, kdy jím zametací přímka prochází.

Odstranění merge vrcholu nemůžeme provést, když v něm jsme, neboť nemůžeme dělat spojnice směrem dolů

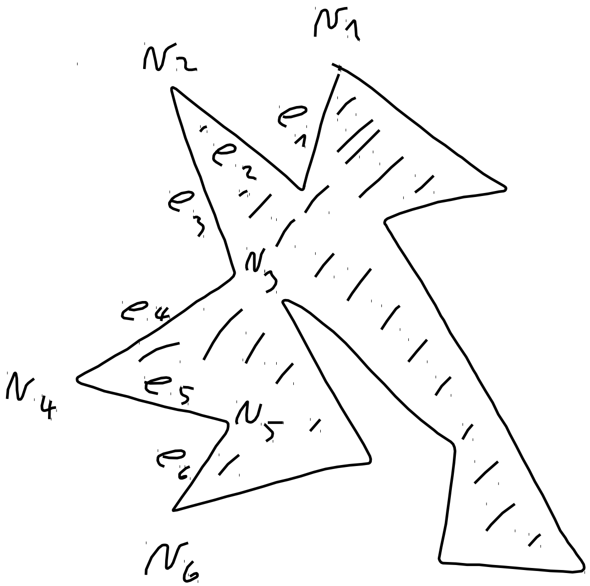


helper(e) = v

Merge vrchol v odstraníme až ve vrcholu w

Strategie … při průchodu vrcholem zkoumáme, zda helpery „blízkých" stran jsou merge. Pokud ano, spojíme je s daným vrcholem

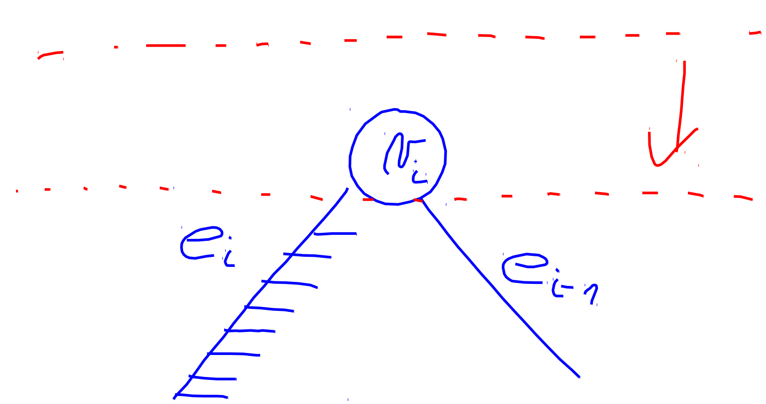
Vrcholy označíme $v_1, v_2, \dots v_n$ (proti směru hod. ručiček)
hrany $e_i = v_i v_{i+1}$ $e_n = v_n v_1$



ALGORITMUS 7 v pseudo.pdf

Procedura pro start vrchol

1. e_i vložíme do T
2. $v_i \rightarrow helper(e_i)$



Procedura pro end vrchol

1. Je-li $helper(e_{i-1})$ merge vrchol, spojíme

   v_i s $helper(e_{i-1})$
2. e_{i-1} odstraníme z T