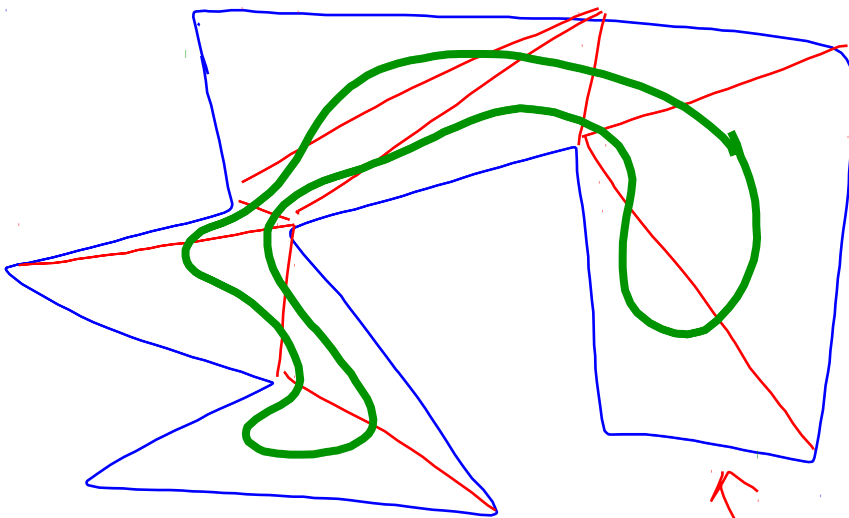
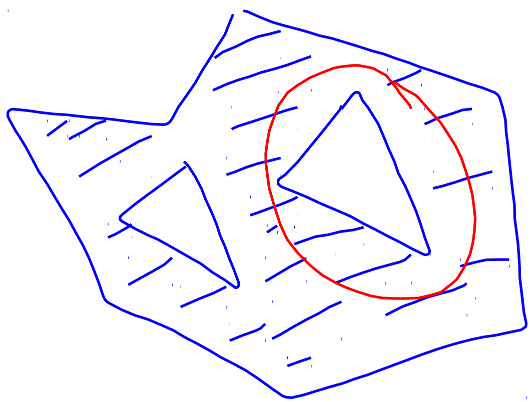
TRIANGULACE MNOHOÚHELNÍKŮ

Triangulace jednoduchých mnohoúhelníků

– každou uzavřenou křivku v mnohoúhelníku lze stáhnout do bodu

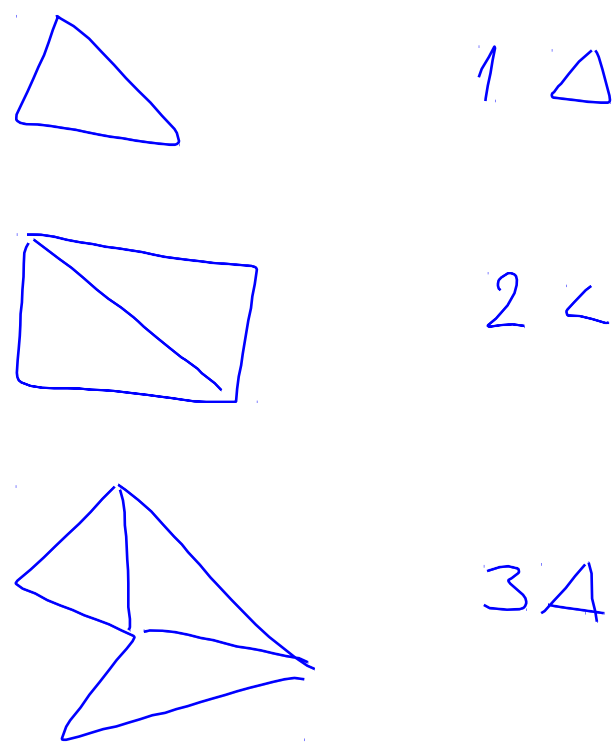


jednoduchý



není jednoduchý

2

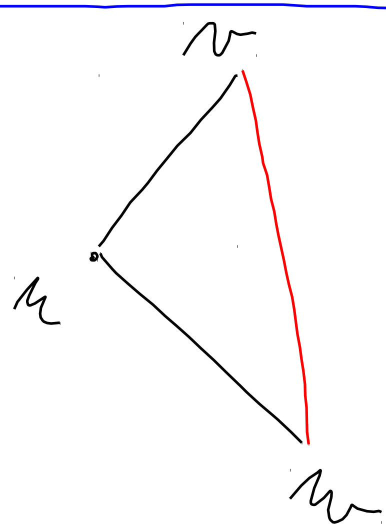


Lemma: Každý jednoduchý n-úhelník lze triangulovat.
Každá triangulace má právě $\underline{n-2}$ trojúhelníků.

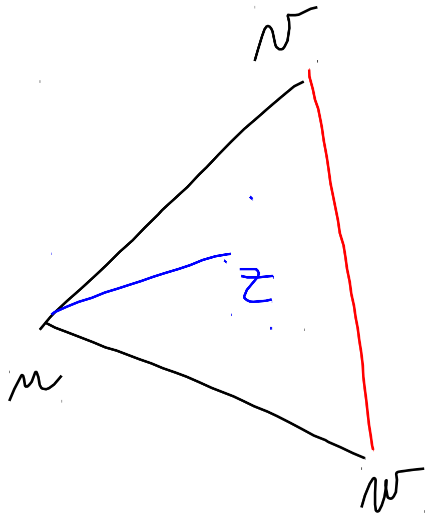
Indukcí

Vezmeme vrchol $\underline{u}$ mnohoúhelníka, který leží nejvíce vlevo. Ten je stranami spojen s vrcholy v w

(1) vw leží v mnohoúhelníku. Dělí mnohoúhelník na $\triangle uvw$ a $(n-1)$-úhelník. Podle ind. před. lze tento triangulovat.



3



(1) vw neleží celá v n-úhelníku

V Δuvw leží nějaké vrcholy mnohoúhelníku. Z nich vybereme nejvzdálenější od přímky vw. Označme ho z.

Úsečka uz leží celá v mnohoúhelníku.

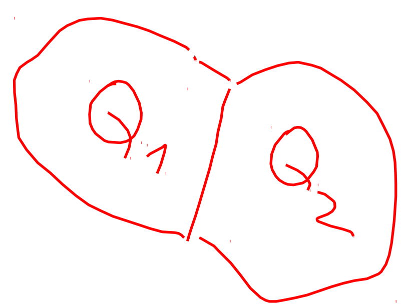
Potom uz dělí n-úhelník na dva mnohoúhelníky a ty lze triangulovat podle ind. předpokladu

Všechny triangulace mají $(n-2)$ trojúhelníků.

Indukcí. Pro $n=3$ zřejmé. Nechť platí pro $i<n$, $n \geq 4$.

V dané triangulaci vezmeme jednu stranu trojúhelníka, která je úhlopříčka.

Ta rozdělí n-úhelník na n_1 a n_2-úhelník.



$$n_1 + n_2 = n + 2$$

(4)

Podle ind. předpokladu triangulace Q_1 má $(n_1-2)\triangle$

— " — Q_2 má $(n_2-2)\triangle$

Triangulace našeho n-úhelníka má

$$(n_1-2)+(n_2-2) = n_1+n_2-4 = n-2 \text{ trojúhelníků.}$$

ALGORITMUS

Konvexní mnohoúhelník rozdělíme na $\triangle$ jednoduše.

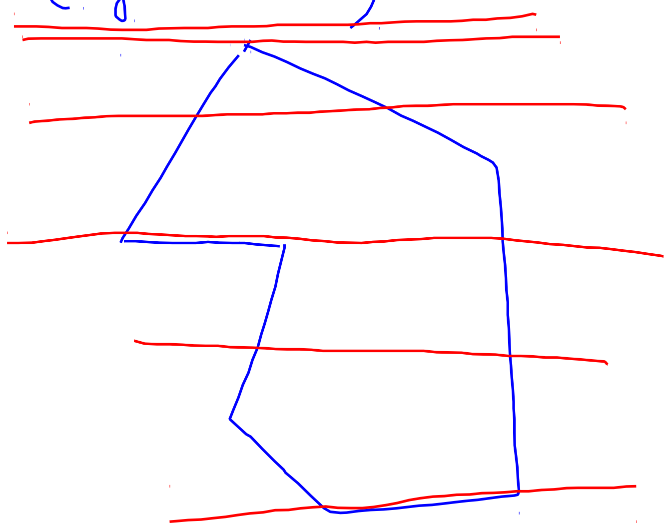
2 kroky

(1) rozdělení n-úhelníka na jednodušší mnohoúhelníky tzv. monotónní

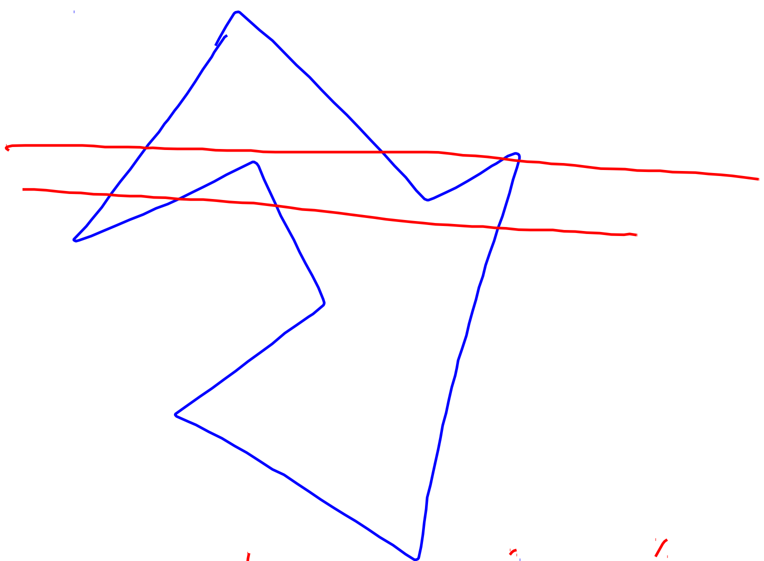
(2) triangulace mon. mnohoúhelníků

Monotónní mnohoúhelník (vzhledem k ose y)

- geometricky
  průnik mon. mnohoúhelníka s každou vodorovnou přímkou ($y = const$) je konv. množina ($\emptyset$, bod, úsečka)



monotónní podle naší definice



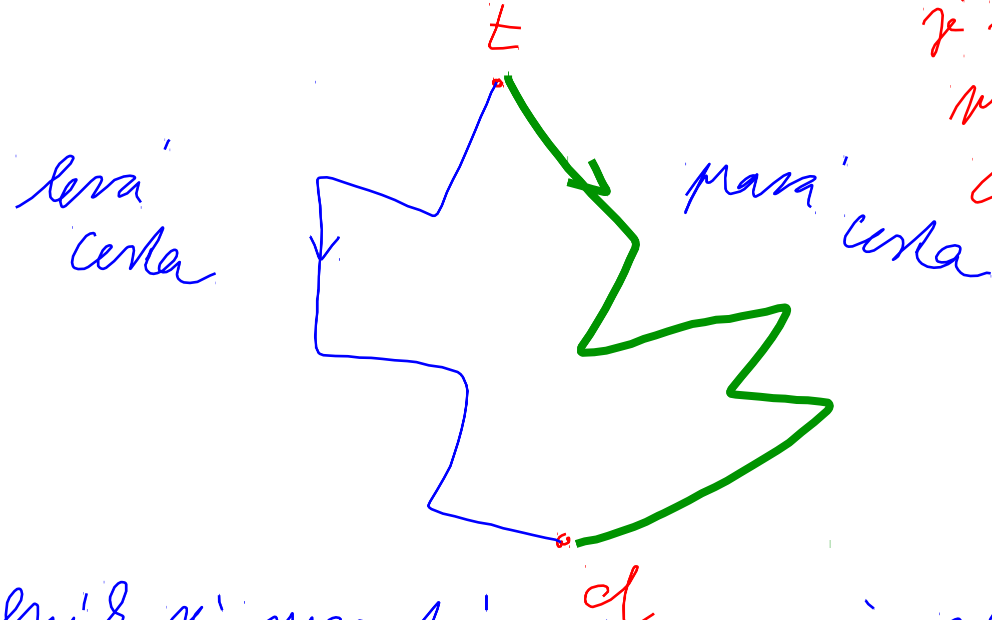
není monotónní

6

- zjednodušení: Vezmeme lexikografické uspořádání (po zavedení přímky)

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

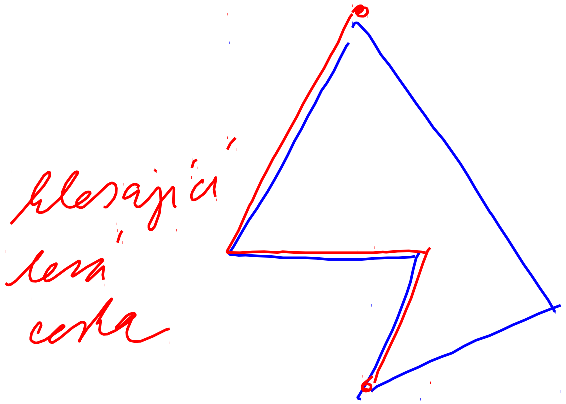
Máme-li mnohoúhelník vezmeme v tomto uspořádání jeho nejvyšší vrchol a nejnižší vrchol. Hranice mnohoúhelníka je rozdělena na levou a pravou část (levá a pravá cesta)



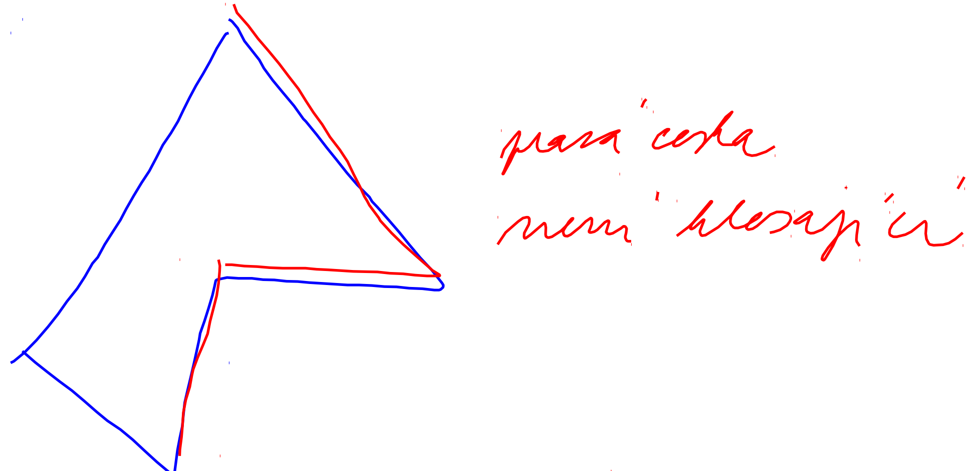
Mnohoúhelník je monotónní, jestliže obě cesty od t k d jsou v daném uspořádání klesající

7

To úplně nekoresponduje s geom. definicí.

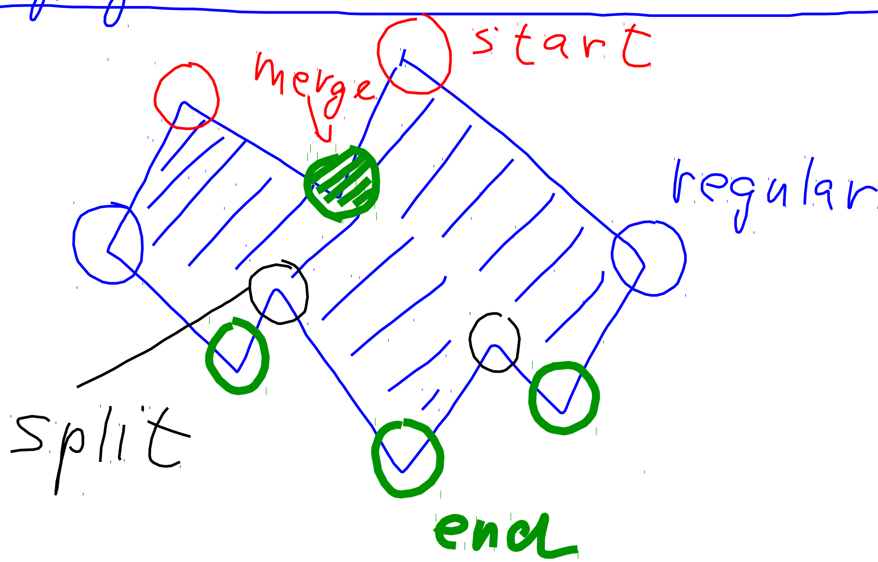


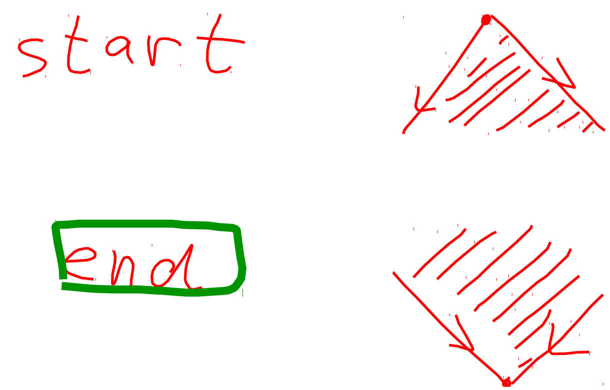
je monotónní



není monotónní
podle zpřesněné definice

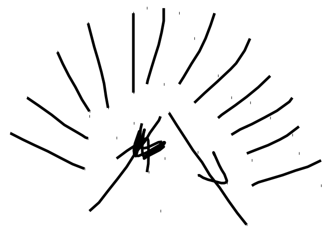
Typy vrcholů mnohoúhelníka





regulární

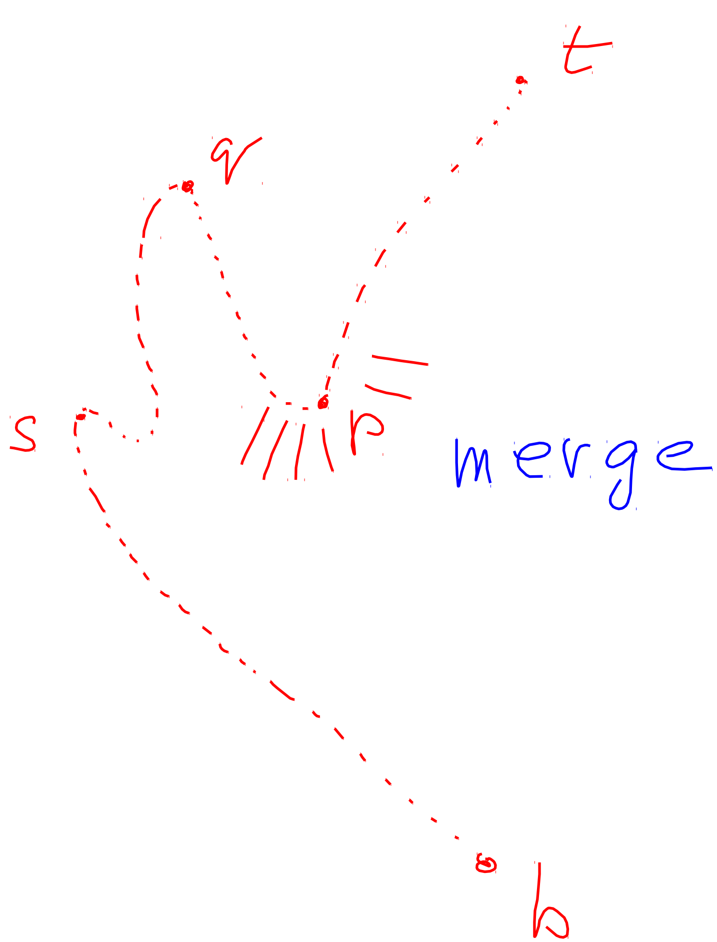
split



merge

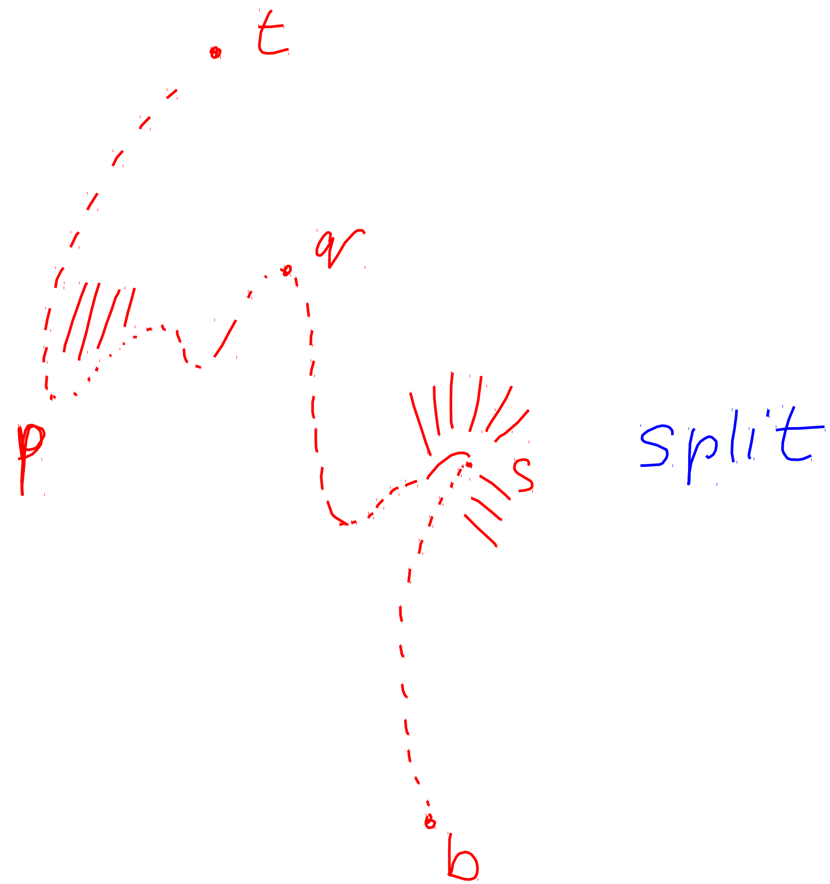
Lemma Mnohoúhelník je monotónní, právě když neobsahuje vrcholy typu split a merge.

Dk: monotónní $\Rightarrow$ pouze vrcholy typu start, end, regulární.



Mnohoúhelník nad p

p je merge uzel

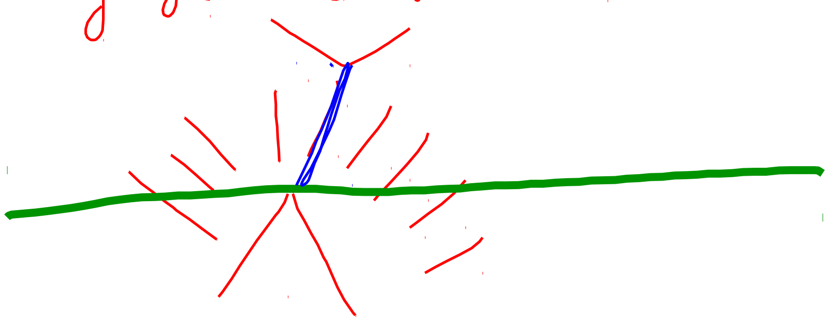


Mnohoúhelník nad p => nad s

s je split uzel

Algoritmus odstraňuje split a merge vrcholy.
Provádí to metodou zametací přímky.
Události jsou pouze vrcholy mnohoúhelníka.
Ty jsou lexikograficky uspořádány do fronty.

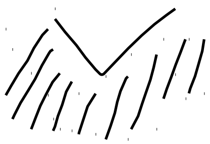
(1) odstraňování split vrcholů provádíme v okamžiku kdy jimi prochází zametací přímka



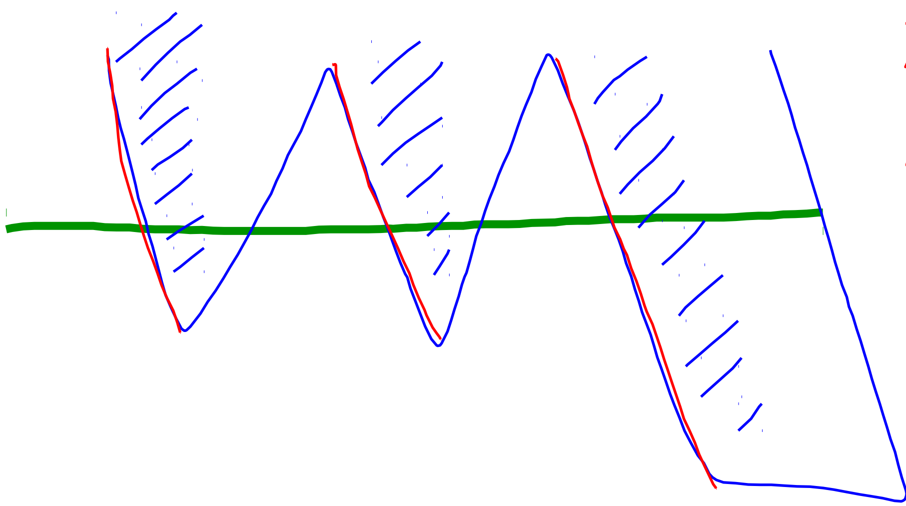
Spojíme s nějakým vrcholem nad ním.

12

2) merge vrcholy musíme až když padáme nějakým vrcholem pod nimi. Zpětně.



Budou nás zajímat strany mnohoúhelníka, které jsou nejblíže vlevo od události a mají mnohoúhelník zprava.



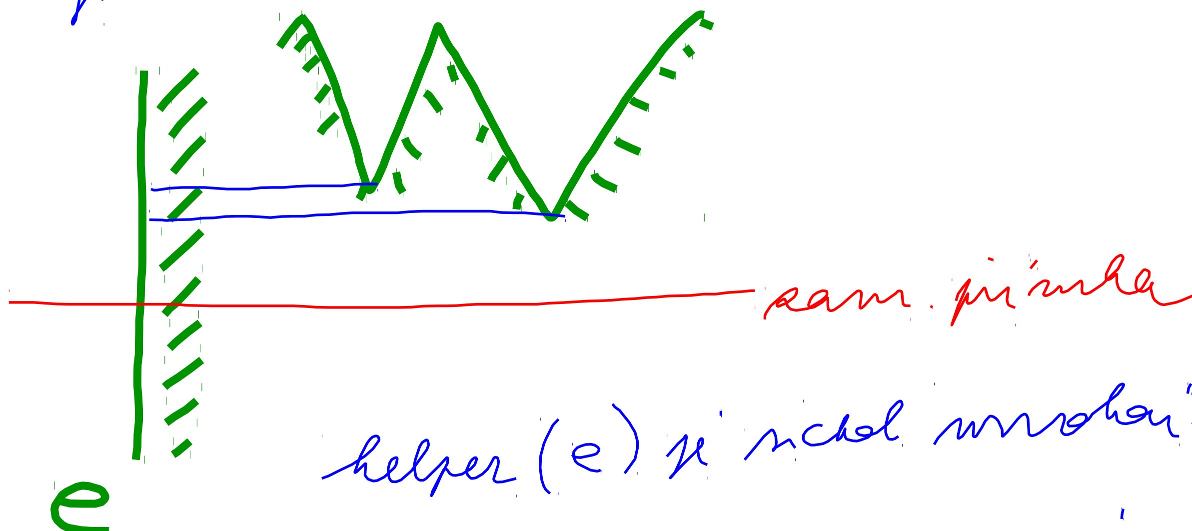
Vytvoříme binární vyvážený strom, který určuje pořadí stran mnohoúhelníka, které

1) protínají zam. přímku (pořadí zleva doprava)
2) mají mnohoúhelník zprava

13

helper – pomocník hrany e (která má mnohoúhelník zprava) (= strana mnohoúhelníka)

je dán touto stranou a polohou zametací přímky takto:



helper(e) je vrchol mnohoúhelníka, který

(1) lze spojit vodorovnou úsečkou se stranou e uvnitř mnohoúhelníka nad zametací přímkou

(2) z vrcholů splňujících (1) vybereme nejbližší k zametací přímce

Ve stromu T držíme nejen strany mnohoúhelníka, ale také jejich helpery. Při průchodu zametací přímky vrcholem provádíme tyto typy akcí:

(1) vyřadíme vrchol z fronty

(2) za pomocí určených podmínek spojíme vrchol s helperem stran, které jsou blízko vrcholu

(3) měníme strom

(4) měníme pomocníky stran ve stromě

15

start vrchol

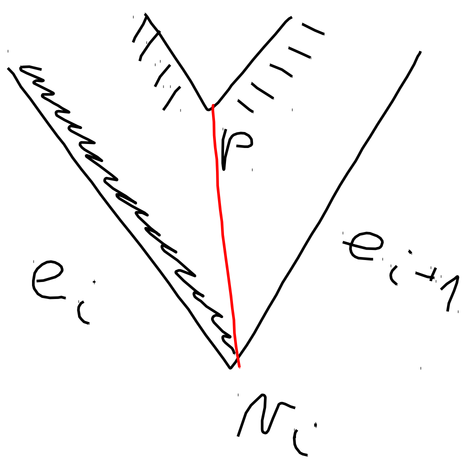
nechť $v_1, v_2, \dots, v_n, v_{n+1} = v_1$

hrany $e_i = v_i v_{i+1}$

(1) zařadím e_i do stromu

(2) $helper(e_i) := v_i$

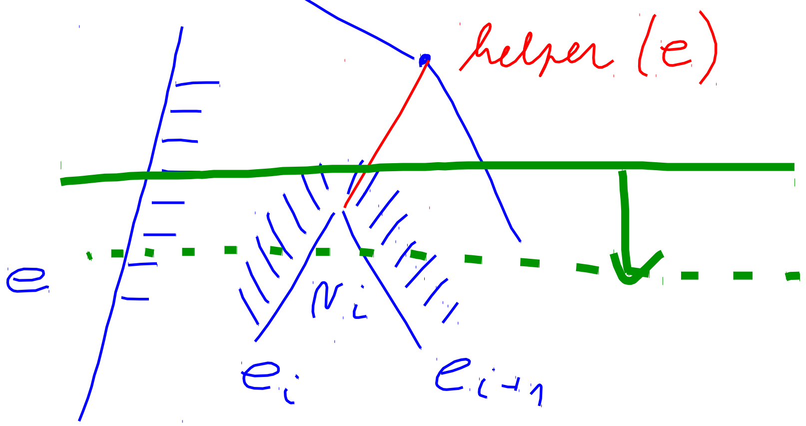
end vrchol



(1) je-li pomocník hrany e_{i-1} typu merge, spojíme jej s v_i

(2) vyjmeme e_{i-1} ze stromu T.

split vrchol



e je nejbližší levá hrana k v_i

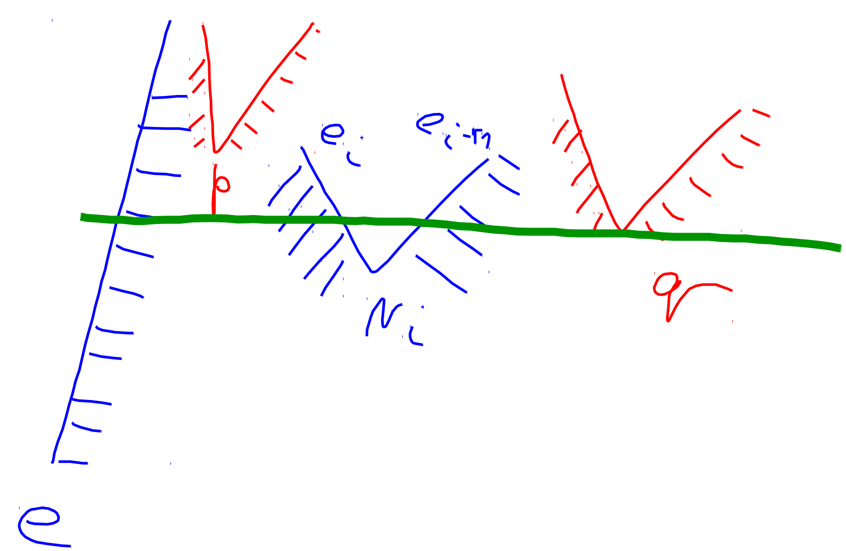
(1) spojíme v_i s helper(e)

(2) položíme helper(e) = v_i

(3) zařadíme e_{i+1} do stromu T

(4) položíme helper e_{i+1} = v_i

merge vrchol



e nejbližší levá hrana ve stromě

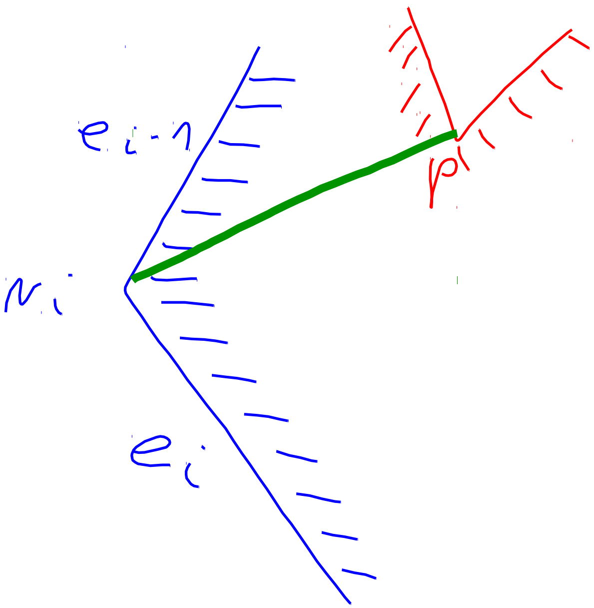
(1) je-li helper(e) typu merge, spojíme ho s v_i.

(2) položíme helper(e) = v_i

(3) je-li helper e_{i+1} typu merge, spojíme ho s v_i.

(4) vyjmeme e_{i+1} ze stromu T.

Regulární uzel

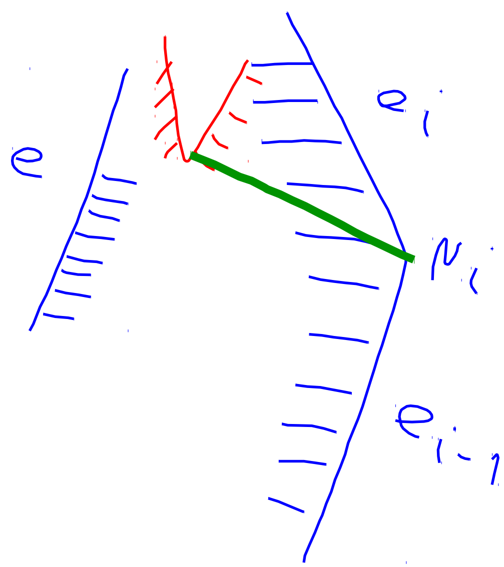


(1) je-li helper(e_{i-1}) typu merge, spojíme ho s v_i

(2) vyjmeme e_{i-1} ze stromu T.

(3) zařadíme e_i do stromu T

(4) položíme helper e_i = v_i



e nejbližší zleva k v_i ve stromě T

(1) je-li helper(e) typu merge, spojíme ho s v_i

(2) Položíme helper(e) = v_i.