# 9. Geometrické algoritmy

*(B. Maslowski, J. Návrat, M. Staša)*

Budeme se teď bavit o geometrických problémech v rovině. Většina algoritmů, které zde uvedeme, má sice své obdoby i pro prostory vyšší nebo nižší dimenze, ale jednorozměrné případy bývají triviální a vícerozměrné jsou zase většinou moc složité.

Budeme se tedy zabývat tím, jak tyto problémy řešit v dimenzi dva (v Euklidovské rovině).

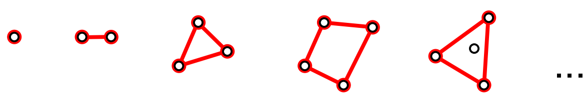
## Hledání konvexního obalu

Ptáte se o co půjde? Zkusme si to přiblížit na problému ledních medvědů :) *Lední medvědi si po dlouhé době zmapovali vody severního moře a zjistili přesně místa, kde se nacházejí jejich oblíbené ryby. No a protože to jsou medvědi chytří, rozhodli se všechny tyto rybky pochytat najednou do jedné velké sítě. A problém, který tady mají, je následující: jaký nejmenší obvod může mít taková síť, aby se dovnitř vešly všechny rybky?!*

Neboli budeme řešit, jak nějakou zadanou množinu bodů v rovině obalit co nejkratší uzavřenou křivkou, do které se ještě všechny body vejdou.

Intuice nám napovídá že výsledek bude nějaký konvexní[1] mnohoúhelník, který bude mít vrcholy v některých uvedených bodech. Ostatní vrcholy pak budou buď někde na hranách mnohoúhelníku, nebo uvnitř. Tomuto mnohoulehníku se říká *konvexní obal* dané množiny.

Možná by se teď hodilo předvést názorně, jak vypadají nejmenší konvexní obaly:



Základní obaly.

- Konvexní obal prázdné množiny je prázdná množina.
- Konvexní obal 1 bodu je bod samotný.
- Konvexní obal 2 bodů je úsečka spojující tyto body.
- Konvexní obal 3 bodů je trojúhleník s vrcholy v těchto bodech.
- Konvexní obal 4 bodů . . . to už je složitější. . .

Konvexní obaly 4 a více bodů, jak si můžeme všimnout, už nejsou jednoznačné. Pro $N$-prvkovou množinu bude konvexní obal mnohoúhelník se třemi až $N$ vrcholy.

Jeden dobrý způsob, jak tento konvexní obal sestrojit se nazývá *Zametání roviny.*

Algoritmus funguje tak, že si v rovině zvolíme nějaký směr, a v tomto směru začneme posouvat přímku. Budeme takto potkávat body ležící v naší množině.

[1] Množina bodů v rovině je konvexní, pokud platí, že pro každé dva body této množiny leží úsečka spojující tyto dva body také celá v této množině.

V každém okamžiku budeme chtít, aby body v části, kterou jsme již zametli, už měli spočítaný konvexní obal. Vždy když pak zametací přímkou narazíme na nový bod, už si jen rozmyslíme, jak ho do konvexního obalu přidat.

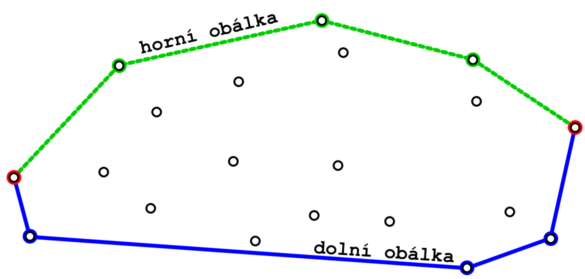
BÚNO předpokládáme body v obecné poloze, tedy takové, že žádné tři neleží na jedné přímce. Dále také budeme předpokládat, že budeme zametat ve směru $x$-ové osy a že všechny body mají různou $x$-ovou souřadnici.

Je také vidět, že bod s nejmenší a největší $x$-ovou souřadnicí bude ležet na konvexním obalu.

**Myšlenka algoritmu:**

1. Setřídíme body podle jejich $x$-ové souřadnice.
2. Vezmeme první tři body a sestrojíme jejich konvexní obal.
3. Opakuj: Najdeme další bod a podíváme se, jestli ho můžeme do konvexního obalu rovnou přidat:
4. Pokud jej můžeme rovnou přidat, tak jej přidáme.
5. Pokud jej přidat rovnou nemůžeme, pak je potřeba nejdříve nějaké body odzadu odebrat a pak teprve připojit náš nový bod.

Každá iterace tedy bude probíhat tak, že nějaké body z původního konvexního obalu pozapomínáme a přidáme nový bod. Aby se to lépe popisovalo, tak celý konvexní obal rozdělíme na *horní obálku* a *dolní obálku.*




Obrázek obálek.

Vidíme teď, že dolní obálka je nějaká lomená čára, která zatáčí doleva. Horní obálka zatáčí doprava. V našem algoritmu si budeme obálky pamatovat jako dva seznamy vrcholů. Když pak v algoritmu narazíme na nový bod, budeme zvlášť řešit jak to ovlivní horní obálku a jak ovlivní dolní. Je vidět že bod nejvíce vlevo a bod nejvíce vpravo leží v obou obálkách. Ostatní body buď leží jen v jedné z obálek, nebo neleží v žádné z nich (tedy nejsou součástí konvexního obalu).

**Algoritmus:**

1. Setřídíme body podle souřadnice $x$, dostaneme množinu bodů $b_1 - b_n$.

2. Spočítáme konvexní obal $b_1, b_2, b_3$, z toho získáme horní a dolní obálku[2].
3. Pro $b$ postupně zpracováváme $b_3 - b_n$:
4. Přepočítáme Horní obálku:
5. Dokud $(|H| \geq 2)$ a úhel $(H[-2], H[-1], b)$ je orientovaný doleva:
6. Odebereme poslední prvek z obálky.
7. Přidáme do obálky nový vrchol.
8. Přepočítáme dolní obálku:
9. Dokud $(|D| \geq 2)$ a úhel $(D[-2], D[-1], b)$ je orientovaný doprava:
10. Odebereme poslední prvek z obálky.
11. Přidáme do obálky nový vrchol.

Setřídit body podle $x$-ové souřadnice a sestrojit konvexní obal prvních třech bodů stihneme v čase $\mathcal{O}(n \log n)$. Zbytek pak už uděláme dokonce v čase lineárním $\mathcal{O}(n)$[3]. Platí tedy:

**Věta:** Konvexní obal dokážeme sestrojit v čase $\mathcal{O}(n \log n)$.

Naši lední medvědi se tedy již naučili, jak si efektivně obstarat potravu a mohly se pustit do řešení dalšího velmi důležitého problému. Pojďme se na něj podívat s nimi. A o co že to půjde?

*Lední medvědi nejsou na antarktidě sami, kromě nich tam taky bydlí kamarádi eskymáci ve svých iglů. Medvědi by si teď rádi udělali mapu, podle které by hned poznali, ke kterému ekymákovi to mají nejblíže na návšťevu.*

My tuhle medvědí mapu od teď budeme nazývat *Voroneho diagramem.*

## Voroného diagramy

Před tím, než vás vystraším nějakou definicí, si řekneme, co jsi pod tímto, na první pohled ne zřejmým pojmem, představit. Mějme množinu teček $T$ rozmístěných náhodně po papíru. Ke každému bodu nakreslíme okraje tak, aby vniklá ploška obsahovala body, které jsou nejblíže právě té naší vybrané tečce. Samozřejmě „sousední“ tečky budou mít tyto hranice společné. Výsledkem našeho dlouhého snažení pak bude právě Voroného diagram. V dalších odstavcích se budeme zajímat o to, jak takový útvar správně popsat, jak ho sestrojit a jaké datové struktury k tomu použít.

**Definice:** *Voroného diagram* pro konečnou množinu $M = \{m_1, \ldots, m_n\} \in \mathbb{R}^2$ míst je systém množin $O_1, \ldots, O_n$ takových, že pro všechna $i$ a $j$ a pro všechna $x \in M_i$ je

---

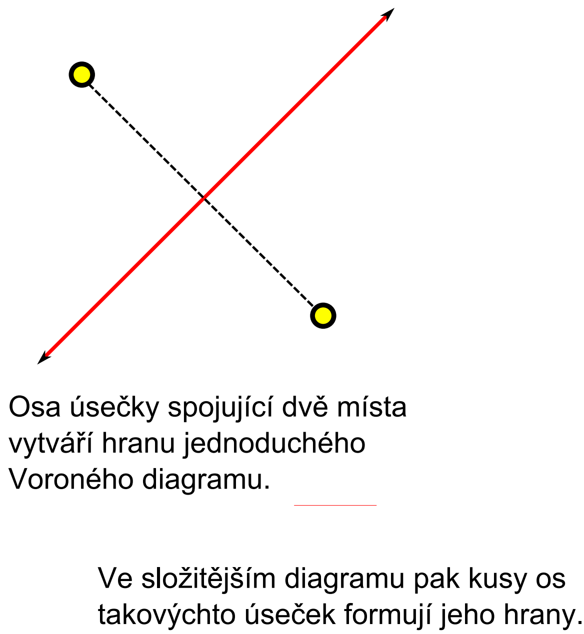
[2] Body $b_1$ a $b_3$ budou v obou obálkách. Bod $b_2$ bude v horní obálce pokud leží nad přímkou spojující $b_1$ a $b_3$, v dolní obálce bude pokud leží pod přímkou.

[3] V této části už jen do obálek přidáváme a odebíráme body.Přidáváme jich $N$. A odebrat jich můžeme maximálně tolik kolik jsme jich přidali. Tedy zase maximálně $N$.

vzdálenost $x$ od $m_i$ menší nebo rovna vzdálenosti $x$ od $m_j$ a zároveň sjednocení $O_i$ přes všechna $i$ je celý prostor $\mathbb{R}^2$, neboli:

$$d(x, m_i) \leq d(x, m_j) \wedge \bigcup_i O_i = \mathbb{R}^2.$$

Jednoduchý Voroného diagram:




Voroneho diagramu pro dvě místa.

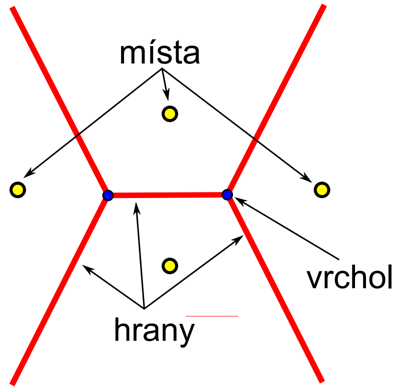
Voroného diagram se tedy skládá z nějakých míst, oblastí a hran, které ty oblasti oddělují.

**Definice:** Řekneme, že *hrana* $H$ je taková množina bodů, že pro každý bod $x \in H$ platí, že existují dvě místa $m_i$ a $m_j$, od kterých má bod $x$ stejnou vzdálenost. Tyto dvě místa jsou pro všechny body $x$ stejná a platí, že všechny ostatní místa mají od každého bodu $x$ delší vzdálenost.

**Definice:** Řekneme, že *vrchol* je takový bod, kde se potkávají alespoň dvě hrany.

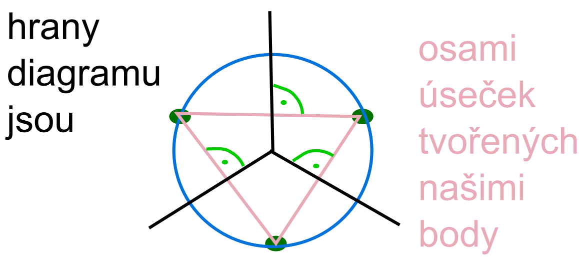
**Pozorování:**

- Voroneho diagramem pro dvě místa jsou dvě poloroviny odělené takovou přímkou, že každý bod přímky je stejně vzdálený od obou míst.
- Každá množina $M_i$ je ohraničena konvexní lomenou čarou, takže oblasti mají tvar konvexních mnohoúhelníků, ale je možné, že jsou oteveřené do neknečna.




Části Voroneho diagramu.

- Pro každou hranu $h$ ve Voroného diagramu existuje $i$ a $j$ takové, že když $x \in H$, pak vzdálenost $d(x, m_i) = d(x, m_j)$.
- Pro každý vrchol $v$ Voroného diagramu existují alespoň tři místa ležící na kružnici se středem $v$.




Body na kružnici.

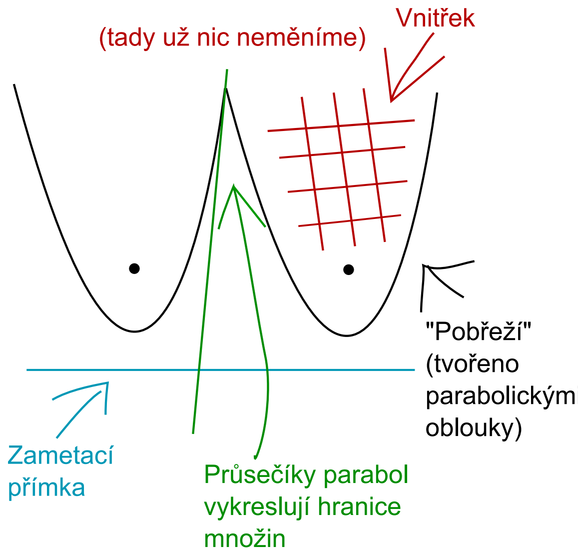
- Počet vrcholů a hran je lineární k počtu míst[4].
- Počet krajních oblastí je tak velký, jak velký je konvexní obal té zadané množiny. (Je to dobré vědět, ale asi to nebudeme potřebovat.)

Pojďme teď vymyslet, jak takový diagram vyrobit. Mohli bychom zkonstruovat všechny dělící přímky a poslepovat je, ale vznikl by nám kvadratický algoritmus a to nám nemůže stačit.

Mluvili jsme o zametání roviny, a tak bychom tento trik mohli využít právě při řešení našeho problému. Ovšem tentokrát má zametání jeden podstatný háček. Když

[4] Voroneho diagram si lze představit jako graf, kde místa Voroneho diagramu odpovídají stěnám, vrcholy diagramu vrcholům a hrany odpovídají hranám grafu. Pokud si teď někam mimo graf přidáme ještě jeden vrchol a všechny přímky vedoucí do nekonečna navedeme do toho bodu, vidíme, že náš graf je roviný. Pro roviný graf platí Eulerova formule a z ní už plyne že naše linearita.

si vezmeme nějakou zametací přímku a pojedeme s ní shora dolů, tak nad ní máme nějakou už zkonstruovanou část diagramu a když narazíme na další bod, tak se nám může právě tato část diagramu poměrně složitě změnit. Pomůžeme si malým trikem. Nebudeme považovat za hotovou celou oblast nad zametací přímkou, ale jen takové body, které mají blíže k místům ($m_i$) než k zametací přímce. Tak dostaneme nějakou posloupnost (množinu) parabol. Všechno, co jsme spočítali uvnitř této oblasti nám už nikdo nepokazí (ani nevylepší), je tam bezpečno. Vezmeme si tedy dolní obálku těchto parabol, budeme jí říkat *pobřeží*.




Pobřežní linie.

Pobřeží je tedy nějaká posloupnost parabolických oblouků s tím, že nejlevější a nejpravější jdou do nekonečna. Průsečíky těchto oblouků vykreslují hrany diagramu. Proč? Odpověď na tuto otázku není těžká, stačí vyjít z definice paraboly tak, jak jí zde používáme. Nebo-li je to množina bodů, která je od ohniska (pro nás místa) stejně vzdálená jako od přímky (řídící). A tedy průnik dvou parabol je místo, které je stejně vzdálené od obou ohnisek, což je vlastně definice bodu ležícího na nějaké hraně.

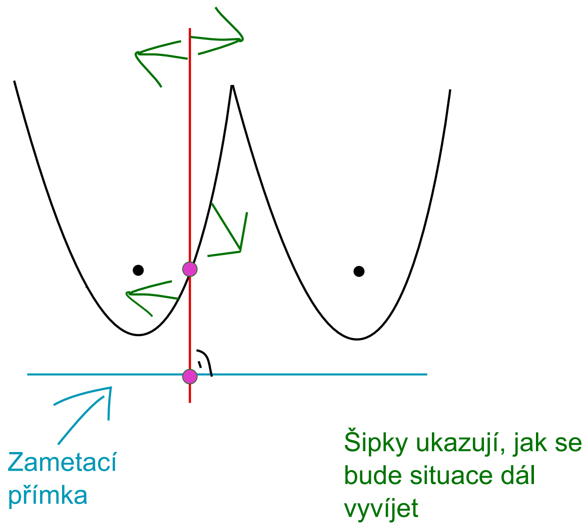
Kdykoliv v průběhu zametání narazíme na nějaký bod, může nám ovlivnit už jen část diagramu pod pobřežím. Dostáváme se tedy k tomu, co se děje, když hýbeme zametací přímkou. Pakliže nenarážíme na žádné body, tak se v podstatě neděje nic zajímavého. Zajímavá situace nastává až tehdy, narazíme-li na další bod. V tom okamžiku vzniká nová parabola. V tuto chvíli je značně degenerovaná. Je to totiž zatím jen polopřímka kolmá na zametací přímku. S dalším pohybem se začne parabola rozevírat. Všimněme si, že průnik oné přímky a pobřeží je vlastně vrchol Voroného diagramu.

Může nastat ještě jeden problém. Nějaká parabola se může natolik rozevřít, že pohltí jiné a ty zmizí z pobřežní linie. V takovém případě, se nám ale netriviálně změní vzhled pobřeží, a proto se této situaci budeme muset více věnovat.

Shrneme-li naše úvahy, mohou se dít celkem tři věci. Jedna z nich je posun přímky. To se vlastně děje pořád. Pobřeží se téměř nemění a průsečíky parabol nám kreslí hrany. To můžeme počítat najednou. Navíc nejen že bychom mohli s přímkou skočit o nějaké epsilon, ale my dokonce můžeme skočit o pořádný kus a prostě pouze dopočítat, jak se pobřeží změnilo a co se vykreslilo. Důležitým místům, kde se budeme zastavovat, budeme říkat *události*.

*Místní událost*

Pokud narazíme na bod, musíme najít místo, kde pobřeží rozetnou a kam vklínit další výběžek (parabolu). Takovéto události budeme říkat místní událost.

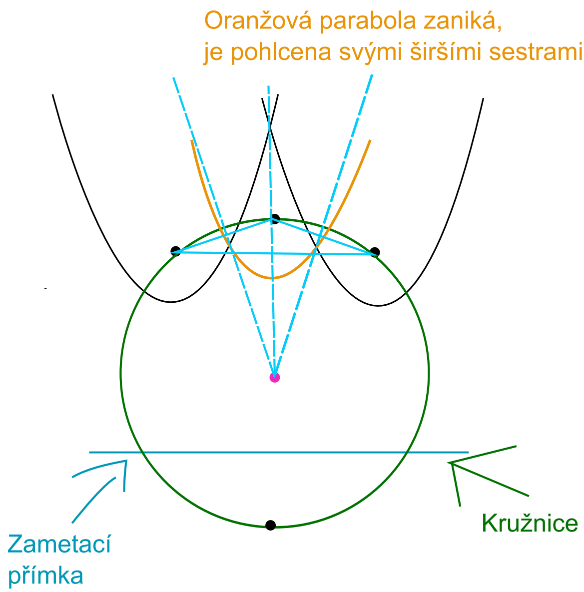



Místní událost – červená kolmice je nově vznikající parabola, při postupu zametací přímky dále se bude rozevírat a vytvoří další parabolu.

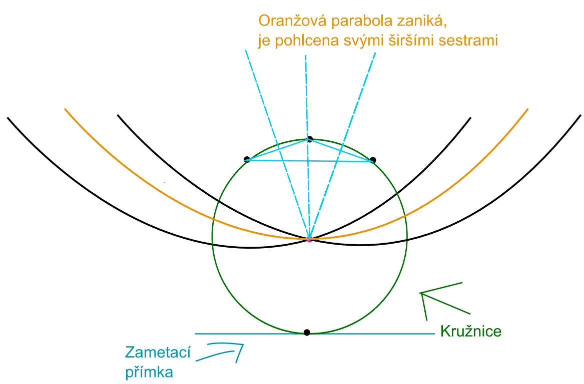
*Kružnicová událost*

Poslední situace, která může nastat, je, že se nějaká parabola schová za jiné. Koukněme se na první obrázek níže, fialový bod je střed kružnice opsané trojúhelníku tvořenému třemi místy. Jak víme, ten leží na osách stran takového trojúhelníku. Po těchto osách se však budou i pohybovat průsečíky parabol a s posunováním řídící přímky se pak dostanou všechny tři do středu této kružnice. Stane se to právě tehdy, když se zametací přímka dotkne kružnice zespodu. Je možné nahlédnout, že při postupu dále se pak dvě krajní paraboly rozšíří natolik, že prostřední pohltí a ta zanikne. Takto vzniklé události budeme říkat kružnicová.

Pojďme si to ukázat lépe na následujících dvou obrázcích. První představuje situaci před kružnicovou událostí a druhý právě kružnicovou událost. Mimo jiné by tedy z obrázků mělo být patrné, že kružnicové události jsou určeny trojicemi sousedních oblouků v pobřeží.




Před kružnicovou událostí.




Kružnicová událost.

**Datové struktury**

Budeme potřebovat haldu událostí (místních i kružnicových dohromady).

Dále bude zapotřebí udržovat si pobřežní linii, neboli posloupnost míst v ohniscích parabolických oblouků. Zde je potřeba si definovat operace *Insert, Delete* a

*FindX*, jinak řečeno přidat a odebrat oblouk a najít oblouk podle x-sové souřadnice nebo-li oblouk do kterého jsme se trefili při místní události. Navíc budeme potřebovat vyhledávací strom nad průsečíky s implicitní reprezentací, což znamená, že si ve vrcholech nebudeme pamatovat přímo souřadnice průsečíků, ale jen instrukci, jak je spočítat. Takže jakkoli se mění poloha průsečíků, tak struktura stromu zůstává stejná.

S haldou událostí lze pracovat s logaritmickou časovou složitostí na operaci a ve stejném čase dokážeme pracovat s pobřežní linií. Když si pobřeží ještě reprezentujeme zvlášť jako seznam tak Insert a Delete budou v konstatním čase a operace se stromem pak $\mathcal{O}(\log n)$.

Poslední datovou strukturou bude samotný diagram, reprezentovaný grafem se souřadnicemi a vazbami hran na průsečík.

**Fortunův algoritmus**

1. Připravíme si haldu $H$ a vložíme do ní všechny místní události.
2. Připravíme pobřežní linii $P \leftarrow 0$.
3. Dokud existuje $h \leftarrow DeleteMin(H)$:
4. Je-li na řadě místní událost:
5. Najdeme průsečík s $P(FindX(P))$.
6. Vložíme do $P$ novou parabolu.
7. Poznamenáme do $D$ vznik hran.
8. Přepočítáme kružnicové události.
9. Je-li na řadě kružnicová událost:
10. Smažeme oblouk z $P$.
11. Poznamenáme vznik a zánik do $D$.
12. Přepočítáme kružnicové události.

**Složitost:**

Počet místních událostí je roven $n$ (na každé místo narazíme právě jednou). Počet kružnicových událostí není větší než $n$, protože kružnicová událost smaže parabolu a ty vznikají jen při místních událostech, takže kružnicových událostí není více než místních. Speciálně z toho plyne, že velikost pobřežní linie je vždy lineární, protože s každou místní událostí přibudou dva úseky do pobřežní linie, takže velikost pobřežní linie je maximálně $2n$. Velikost haldy je pak také $2n$, takže pak operace určitě zvládneme v čase $\mathcal{O}(\log n)$. Jelikož diagram je lineárně velký tak i jeho struktura je lineárně velká. Operace se strukturami nás stojí nejvíce $\mathcal{O}(\log n)$. Takže místní i kružnicové události zvládneme v čase $\mathcal{O}(\log n)$ na jednu na konstantním počtu struktur. Halda má velikost $2n$, takže maximálně provedeme $\mathcal{O}(2n \log n)$ operací. Celý algoritmus potřebuje na inicializaci maximálně $\mathcal{O}(n \log n)$ (i kdybychom ji dělali neefektivně) a $\mathcal{O}(2n \log n)$ výpočet.

Pokud tedy shrneme všechny naše odhady, pak časová složitost algoritmu je $\mathcal{O}(n \log n)$ a prostorová $\mathcal{O}(n)$.