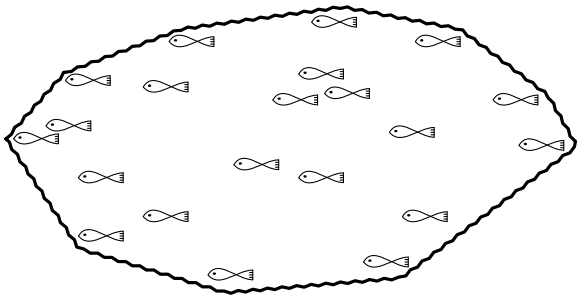
7. Geometrické algoritmy *(sepsal Pavel Klavík)*

Ukážeme si několik základních algoritmů na řešení geometrických problémů v rovině. Proč zrovna v rovině? Inu, jednorozměrné problémy bývají triviální a naopak pro vyšší dimenze jsou velice komplikované. Rovina je proto rozumným kompromisem mezi obtížností a zajímavostí.

Celou kapitolou nás bude provázet pohádka ze života ledních medvědů. Pokusíme se vyřešit jejich „každodenní“ problémy ...

Hledání konvexního obalu

Daleko na severu žili lední medvědi. Ve vodách tamního moře byla hojnost ryb a jak je známo, ryby jsou oblíbenou pochoutkou ledních medvědů. Protože medvědi z naší pohádky rozhodně nejsou ledajací a ani chytrost jim neschází, rozhodli se všechny ryby pochytat. Znají přesná místa výskytu ryb a rádi by vyrobili obrovskou síť, do které by je všechny chytili. Pomozte medvědům zjistit, jaký nejmenší obvod taková síť může mít.

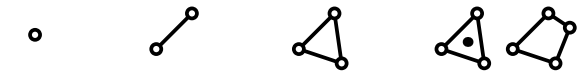


Problém ledních mědvědů: Jaký je nejmenší obvod sítě?

Neboli v řeči matematické, chceme pro zadanou množinu bodů v rovině nalézt její konvexní obal. Co je to konvexní obal? Množina bodů je *konvexní*, pokud pro každé dva body obsahuje i celou úsečku mezi nimi. *Konvexní obal* je nejmenší konvexní podmnožina roviny, která obsahuje všechny zadané body.[1] Z algoritmického hlediska nás však bude zajímat jenom jeho hranice, kterou budeme dále označovat jako konvexní obal.

[1] Pamatujete si na lineární obaly ve vektorových prostorech? Lineární obal množiny vektorů je nejmenší vektorový podprostor, který tyto vektory obsahuje. Není náhoda, že tato definice připomíná definici konvexního obalu. Na druhou stranu každý vektor z lineárního obalu lze vyjádřit jako lineární kombinaci daných vektorů. Podobně platí i pro konvexní obaly, že každý bod z obalu je konvexní kombinací daných bodů. Ta se liší od lineární v tom, že všechny koeficienty jsou v intervalu $[0, 1]$ a navíc součet všech koeficientů je 1. Tento algebraický pohled může mnohé věci zjednodušit. Zkuste si dokázat, že obě definice konvexního obalu jsou ekvivalentní.

Naším úkolem je nalézt konvexní obal konečné množiny bodů. To je vždy konvexní mnohoúhelník, navíc s vrcholy v zadaných bodech. Řešením problému tedy bude posloupnost bodů, které tvoří konvexní obal. Pro malé množiny je konvexní obal nakreslen na obrázku, pro více bodů je však situace mnohem složitější.



$n = 1$ $n = 2$ $n = 3$ $n = 4$

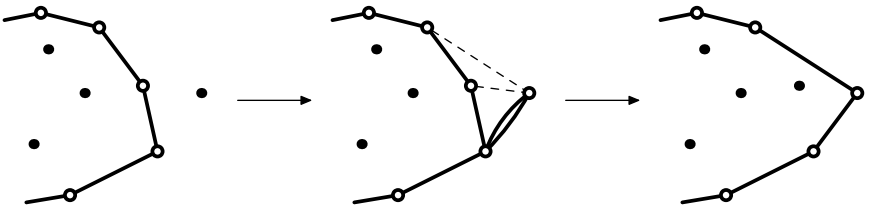
Konvexní obaly malých množin.

Pro jednoduchost budeme předpokládat, že všechny body mají různé x-ové souřadnice. Tedy utřídění bodů zleva doprava je určené jednoznačně.⟨2⟩ Tím máme zajištěné, že existují dva body, nejlevější a nejpravější, pro které platí následující invariant:

Invariant: Nejlevější a nejpravější body jsou vždy v konvexním obalu.

Algoritmus na nalezení konvexního obalu funguje na následujícím jednoduchém principu, kterému se někdy říká *zametání roviny*. Procházíme body zleva doprava a postupně rozšiřujeme doposud nalezený konvexní obal o další body. Na začátku bude konvexní obal jediného bodu samotný bod. Na konci k-tého kroku algoritmu známe konvexní obal prvních k bodů. Když algoritmus skončí, známe hledaný konvexní obal. Podle invariantu musíme v k-tém kroku přidat do obalu k-tý nejlevější bod. Zbývá si jen rozmyslet, jak přesně tento bod přidat.

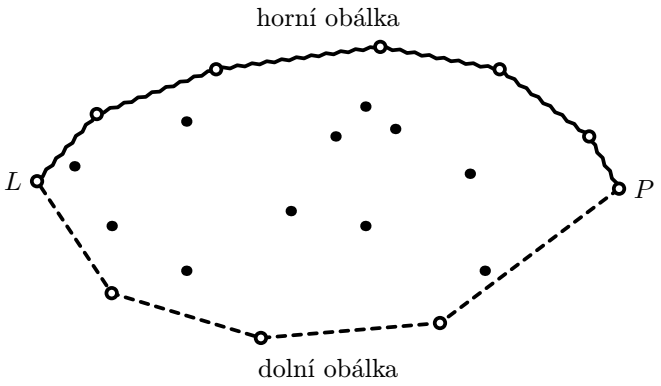
Přidání dalšího bodu do konvexního obalu funguje, jak je naznačeno na obrázku. Podle invariantu víme, že bod nejvíc vpravo je součástí konvexního obalu. Za něj napojíme nově přidávaný bod. Tím jsme získali nějaký obal, ale zpravidla nebude konvexní. To lze však snadno napravit, stačí odebírat body, v obou směrech podél konvexního obalu, tak dlouho, dokud nezískáme konvexní obal. Na příkladu z obrázku nemusíme po směru hodinových ručiček odebrat ani jeden bod, obal je v pořádku. Naopak proti směru hodinových ručiček musíme odebrat dokonce dva body.



Přidání bodu do konvexního obalu.

⟨2⟩ To si můžeme dovolit předpokládat, neboť se všemi body stačí nepatrně pootočit. Tím konvexní obal určitě nezměníme. Avšak jednodušší řešení je naprogramovat třídění lexikograficky (druhotně podle souřadnice y) a vyřadit identické body.

Pro případnou implementaci a rozbor složitosti si nyní popíšeme algoritmus detailněji. Aby se lépe popisoval, rozdělíme si konvexní obal na dvě části spojující nejlevější a nejpravější bod obalu. Budeme jim říkat *horní obálka* a *dolní obálka*.



Horní a dolní obálka konvexního obalu.

Obě obálky jsou lomené čáry, navíc horní obálka pořád zatáčí doprava a dolní naopak doleva. Pro udržování bodů v obálkách stačí dva zásobníky. V k-tém kroku algoritmu přidáme zvlášť k-tý bod do horní i dolní obálky. Přidáním k-tého bodu se však může porušit směr, ve kterém obálka zatáčí. Proto budeme nejprve body z obálky odebírat a k-tý bod přidáme až ve chvíli, kdy jeho přidání směr zatáčení neporuší.

Algoritmus:

1. Setřídíme body podle x-ové souřadnice, označme body $b_1, \ldots, b_n$.
2. Vložíme do horní a dolní obálky bod b_1: $H = D = (b_1)$.
3. Pro každý další bod $b = b_2, \ldots, b_n$:
4. Přepočítáme horní obálku:
5. Dokud $|H| \geq 2$, $H = (\ldots, h_{k-1}, h_k)$ a úhel $h_{k-1}h_kb$ je orientovaný doleva:
6. Odebereme poslední bod h_k z obálky H.
7. Přidáme bod b do obálky H.
8. Symetricky přepočteme dolní obálku (s orientací doprava).
9. Výsledný obal je tvořen body v obálkách H a D.

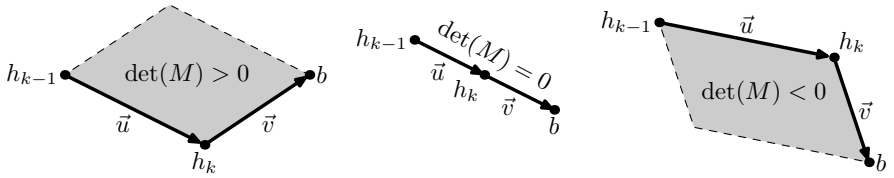
Rozebereme si časovou složitost algoritmu. Setřídit body podle x-ové souřadnice dokážeme v čase $\mathcal{O}(n \log n)$. Přidání dalšího bodu do obálek trvá lineárně vzhledem k počtu odebraných bodů. Zde využijeme obvyklý postup: Každý bod je odebrán nejvýše jednou, a tedy všechna odebrání trvají dohromady $\mathcal{O}(n)$. Konvexní obal dokážeme sestrojit v čase $\mathcal{O}(n \log n)$, a pokud bychom měli seznam bodů již utříděný, dokážeme to dokonce v $\mathcal{O}(n)$.

Algebraický dodatek: Existuje jednoduchý postup, jak zjistit orientaci úhlu? Ukážeme si jeden založený na lineární algebře. Budou se hodit vlastnosti determinantu. Absolutní hodnota determinantu je objem rovnoběžnostěnu určeného řádkovými vektory matice. Důležitější však je, že znaménko determinantu určuje „orientaci" vektorů, zda je levotočivá či pravotočivá. Protože náš problém je rovinný, budeme uvažovat determinanty matic 2×2.

Uvažme souřadnicový systém v rovině, kde x-ová souřadnice roste směrem doprava a y-ová směrem nahoru. Chceme zjistit orientaci úhlu $h_{k-1}h_kb$. Položme $\vec{u} = (x_1, y_1)$ jako rozdíl souřadnic h_k a h_{k-1} a podobně $\vec{v} = (x_2, y_2)$ je rozdíl souřadnic b a h_k. Matice M je definována následovně:

$$M = \begin{pmatrix} \vec{u} \\ \vec{v} \end{pmatrix} = \begin{pmatrix} x_1 & y_1 \\ x_2 & y_2 \end{pmatrix}.$$

Úhel $h_{k-1}h_kb$ je orientován doleva, právě když $\det M = x_1y_2 - x_2y_1$ je nezáporný,[3] a spočítat hodnotu determinantu je jednoduché. Možné situace jsou nakresleny na obrázku. Poznamenejme, že k podobnému vzorci se lze také dostat přes vektorový součin vektorů $\vec{u}$ a $\vec{v}$.



Jak vypadají determinanty různých znamének v rovině.

Šlo by to vyřešit rychleji? Také vám vrtá hlavou, zda existují rychlejší algoritmy? Na závěr si ukážeme něco, co na přednášce nebylo.[4] Nejrychlejší známý algoritmus, jehož autorem je T. Chan, funguje v čase $\mathcal{O}(n \log h)$, kde h je počet bodů ležících na konvexním obalu, a přitom je překvapivě jednoduchý. Zde si naznačíme, jak tento algoritmus funguje.

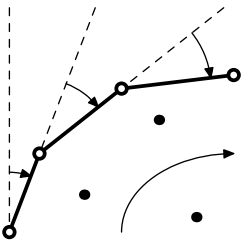
Algoritmus přichází s následující myšlenkou. Předpokládejme, že bychom znali velikost konvexního obalu h. Rozdělíme body libovolně do $\lceil \frac{n}{h} \rceil$ množin $Q_1, \ldots, Q_k$ tak, že $|Q_i| \leq h$. Pro každou z těchto množin nalezneme konvexní obal pomocí výše popsaného algoritmu. To dokážeme pro jednu v čase $\mathcal{O}(h \log h)$ a pro všechny v čase $\mathcal{O}(n \log h)$. V druhé fázi spustíme hledání konvexního obalu pomocí provázkového algoritmu a pro zrychlení použijeme předpočítané obaly menších množin. Nejprve popíšeme jeho myšlenku. Použijeme následující pozorování:

[3] Neboli vektory $\vec{u}$ a $\vec{v}$ odpovídají roztažení a zkosení vektorů báze $\vec{x} = (1, 0)$ a $\vec{y} = (0, 1)$, pro něž je determinant nezáporný.

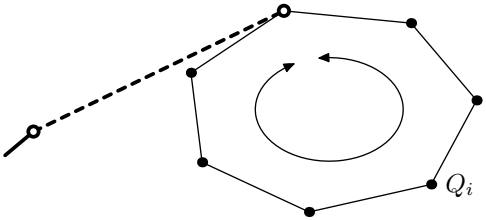
[4] A také se nebude zkoušet.

Pozorování: Úsečka spojující dva body a a b leží na konvexním obalu, právě když všechny ostatní body leží pouze na jedné její straně.⟨5⟩

Algoritmu se říká *provázkový*, protože svojí činností připomíná namotávání provázku podél konvexního obalu. Začneme s bodem, který na konvexním obalu určitě leží, to je třeba ten nejlevější. V každém kroku nalezneme následující bod po obvodu konvexního obalu. To uděláme například tak, že projdeme všechny body a vybereme ten, který svírá nejmenší úhel s poslední stranou konvexního obalu. Nově přidaná úsečka vyhovuje pozorování a proto do konvexního obalu patří. Po h krocích se dostaneme zpět k nejlevějšímu bodu a výpočet ukončíme. V každém kroku potřebujeme projít všechny body a vybrat následníka, což dokážeme v čase $\mathcal{O}(n)$. Celková složitost algoritmu je tedy $\mathcal{O}(n \cdot h)$.



Provázkový algoritmus.



Hledání kandidáta v předpočítaném obalu.

Provázkový algoritmus funguje, ale má jednu obrovskou nevýhodu – je totiž ukrutně pomalý. Kýženého zrychlení dosáhneme, pokud použijeme předpočítané konvexní obaly. Ty umožní rychleji hledat následníka. Pro každou z množin Q_i najdeme zvlášť kandidáta a poté z nich vybereme toho nejlepšího. Možný kandidát vždy leží na konvexním obalu množiny Q_i. Využijeme toho, že body obalu jsou „uspořádané“, i když trochu netypicky do kruhu. Kandidáta můžeme hledat metodou půlení intervalu, i když detaily jsou maličko složitější než je obvyklé. Jak půlit zjistíme podle směru zatáčení konvexního obalu. Detaily si rozmyslí čtenář sám.

Časová složitost půlení je $\mathcal{O}(\log h)$ pro jednu množinu. Množin je nejvýše $\mathcal{O}(\frac{n}{h})$, tedy následující bod konvexního obalu nalezneme v čase $\mathcal{O}(\frac{n}{h} \log h)$. Celý obal nalezneme ve slibovaném čase $\mathcal{O}(n \log h)$.

Popsanému algoritmu schází jedna důležitá věc: Ve skutečnosti většinou neznáme velikost h. Budeme proto algoritmus iterovat s rostoucí hodnotou h, dokud konvexní obal nesestrojíme. Pokud při slepování konvexních obalů zjistíme, že konvexní obal je větší než h, výpočet ukončíme. Zbývá ještě zvolit, jak rychle má h růst. Pokud by rostlo moc pomalu, budeme počítat zbytečně mnoho fází, naopak při rychlém růstu by nás poslední fáze mohla stát příliš mnoho.

⟨5⟩ Formálně je podmínka následující: Přímka ab určuje dvě poloroviny. Úsečka leží na konvexním obalu, právě když všechny body leží v jedné z polorovin.

V k-té iteraci položíme $h = 2^{2^k}$. Dostáváme celkovou složitost algoritmu:

$$\sum_{m=0}^{\mathcal{O}(\log\log h)} \mathcal{O}(n \log 2^{2^m}) = \sum_{m=0}^{\mathcal{O}(\log\log h)} \mathcal{O}(n \cdot 2^m) = \mathcal{O}(n \log h),$$

kde poslední rovnost dostaneme jako součet prvních $\mathcal{O}(\log\log h)$ členů geometrické řady $\sum 2^m$.