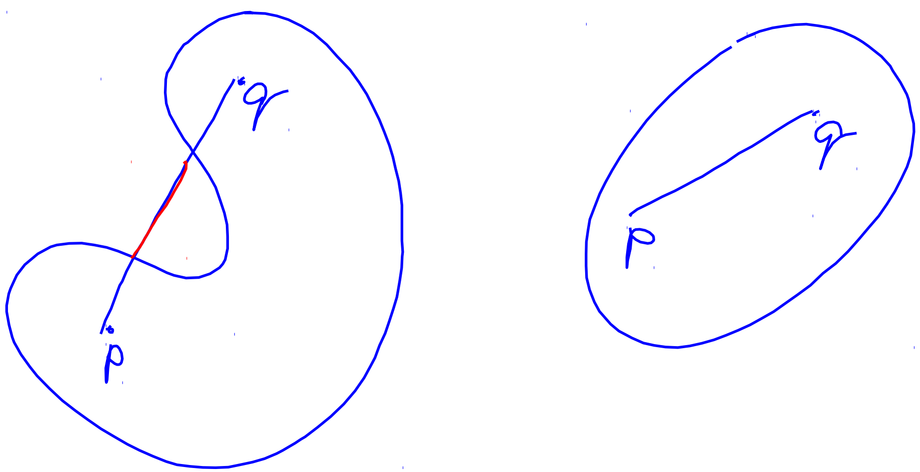
Konvexní obal v rovině

K množina v R^2 je konvexní, jestliže s každými dvěma body obsahuje i úsečku, kterou určují:



Bod na úsečce pq lze zapsat takto

$$s = \lambda p + (1-\lambda) q \qquad \lambda \in [0,1]$$

$$p = [p_x, p_y]$$

$$s_x = \lambda p_x + (1-\lambda) q_x$$

$$s_y = \lambda p_y + (1-\lambda) q_y$$

Konvexní obal množiny M – nejmenší konvexní množina obsahující množinu M

$$CH(M) = \bigcap_{K \supseteq M \text{ konvexní}} K$$

Pro nás většinou bude množina M konečná

$$CH(M) = \bigcap P$$

P je polorovina obsahující M

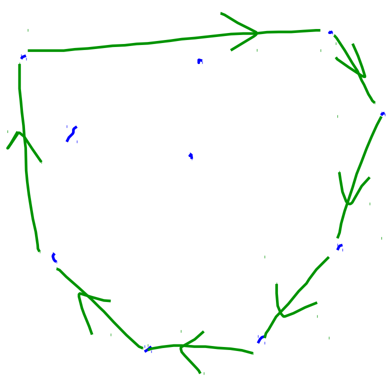
Obalem konečné množiny je konvexní mnohoúhelník.

Typická úloha:

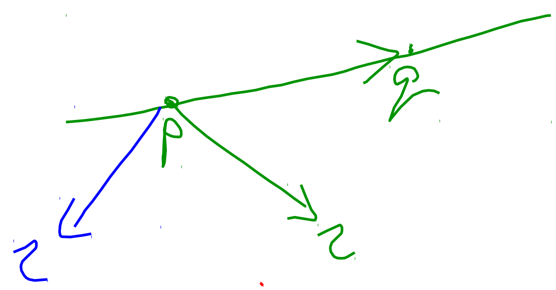
Vstup: body množiny M

Výstup: vrcholy konvexního obalu množiny M zapsané ve směru hod. ručiček

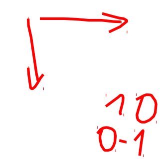
SLOW CONVEX HULL ALGORITHM



$\vec{pq}$ je hranicí konv. obalu, jestliže pro žádné $r \in M$ leží r vpravo od $\vec{pq}$



$$\det\begin{pmatrix} q_x - p_x & q_y - p_y \\ r_x - p_x & r_y - p_y \end{pmatrix} < 0$$



$$\det\begin{pmatrix} q_x - p_x & q_y - p_y \\ r_x - p_x & r_y - p_y \end{pmatrix} = (q_x - p_x)(r_y - p_y) - (r_x - p_x)(q_y - p_y) = \ldots\ldots$$

$$= \det\begin{pmatrix} 1 & p_x & p_y \\ 1 & q_x & q_y \\ 1 & r_x & r_y \end{pmatrix}$$

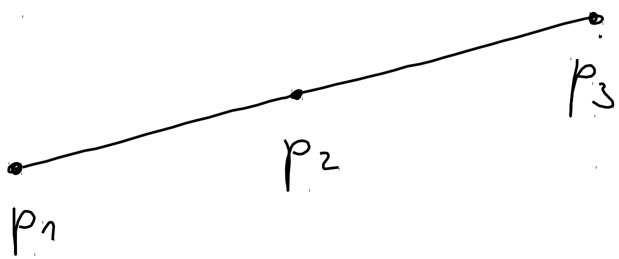
Časová náročnosť SLOW CONVEX HULL algoritmu je $\underline{O(n^3)}$ pri n bodoch na vstupe.

Význam symboliky s veľkým O

Časová náročnosť algoritmu je $O(n^3)$ znamená že existuje konštanta C, že pri n vstupoch (vstupných bodoch) je čas potrebný k prevedeniu algoritmu

$$T(n) \leq C \cdot n^3$$

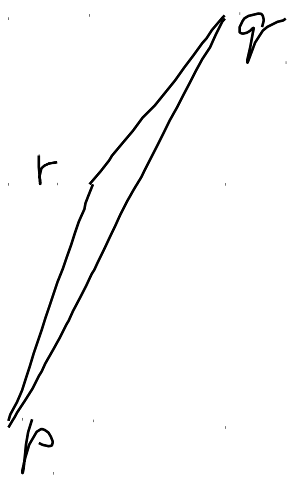
Další nedostatky – je nestabilní



$\overrightarrow{p_1 p_2} \in E$
$\overrightarrow{p_2 p_3} \in E$
$\overrightarrow{p_2 p_3} \in E$

$\overrightarrow{p_1 p_3} \in E$

Chyby v zaokrouhlování



Při zaokrouhlování se může stát, že
$\overrightarrow{pq}, \overrightarrow{rq}, \overrightarrow{pr} \in E$.

Lepší algoritmus

- hledá konv. horní a dolní konvexní obal

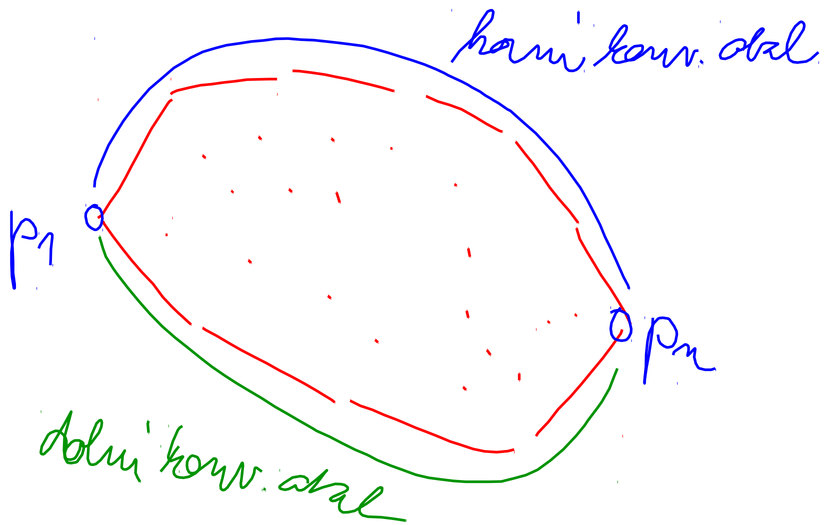
$p_1, p_2, \dots p_n \in M$ najdeme bod nejvíce vlevo (nejvíce dole) ..... p_1

dále najdeme bod nejvíce vpravo (nejvíce nahoře) ... p_n

p_1 a p_n jsou určitě vždy konv. obalu

Hranice konvexního obalu je jimi dělena na horní a dolní část

- horní a dolní konvexní obal

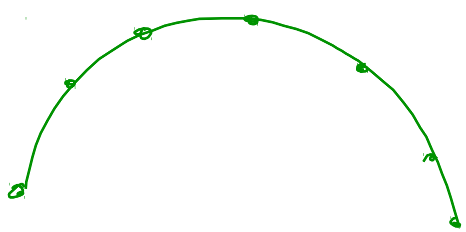


Algoritmus využívá lexikografického uspořádání bodů množiny M

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

V lexikografickém uspořádání zapíšeme body množiny M

$$p_1 < p_2 < \dots\dots < p_{n-1} < p_n$$

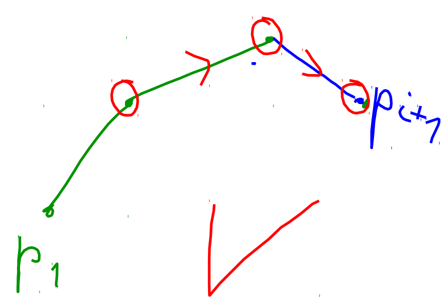


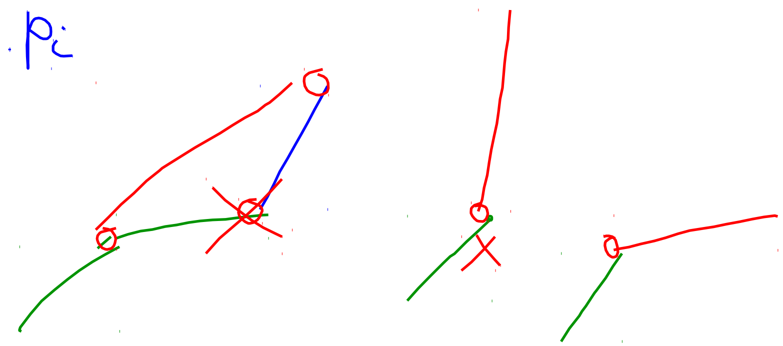
body na horním konv. obalu jsou rovněž uspořádány lexikograficky

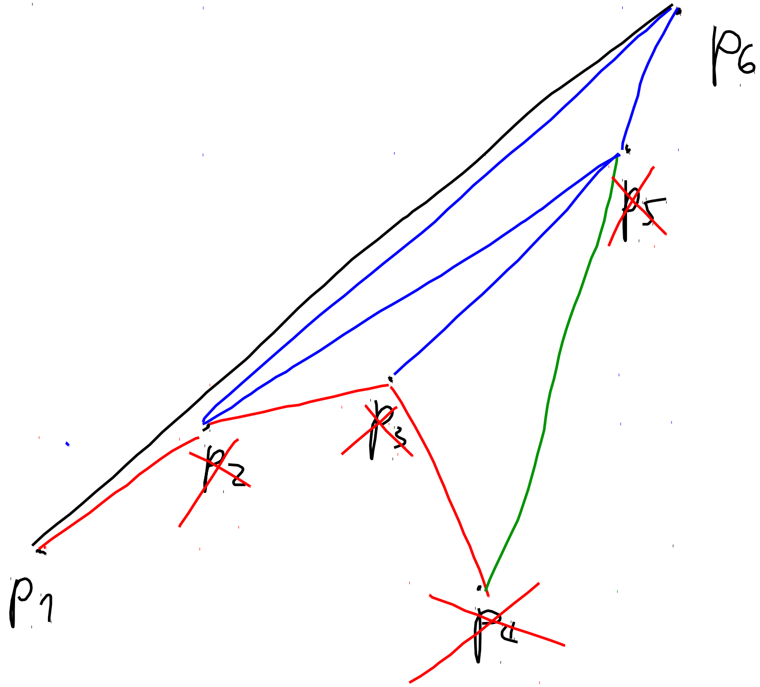
Algoritmus pracuje takto:

Mějme horní konv. obal pro množinu $p_1, p_2, \dots p_i$

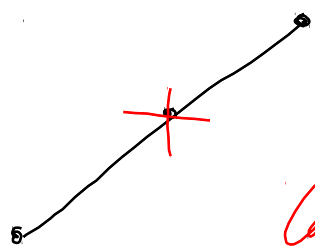
Přidáme bod p_{i+1}





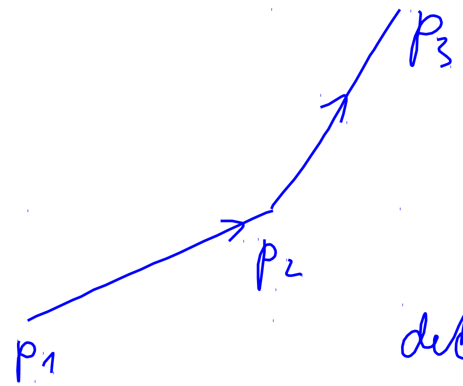


Po nalezení horního konvexního obalu, vytvoříme analogicky dolní konvexní obal začínající p_n a končící p_1, tedy zprava doleva.



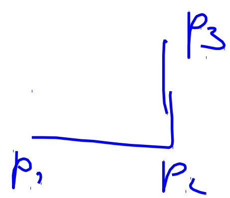
Časová náročnost je $O(n \log n)$

- seřazení bodů $O(n \log n)$
- konv. obal $O(n)$

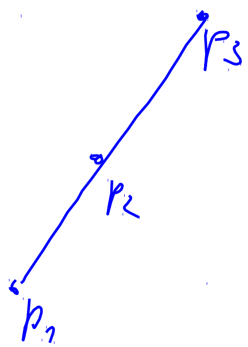


$p_1 p_2 p_3$ nedělají zatáčku doprava

$$\det\begin{pmatrix} p_{2x}-p_{1x} & p_{2y}-p_{1y} \\ p_{3x}-p_{2x} & p_{3y}-p_{2y} \end{pmatrix} \geq 0$$



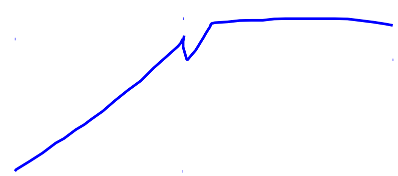
$$\det\begin{pmatrix} 1 & 0 \\ 1-1 & 1-0 \end{pmatrix} = 1$$



V konvexním obalu podle algoritmu bude $p_1 p_3$

Zaokrouhlování → někdy dá nějaký výsledek

3 body velice blízko – řešení může vypadat



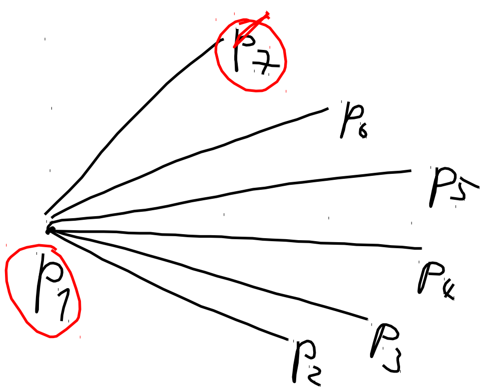
Jiný algoritmus

Gift wrapping — vhodný v případě, že konvexním obalem je k-úhelník s $k \ll n$.

Časová náročnost je $O(n k)$

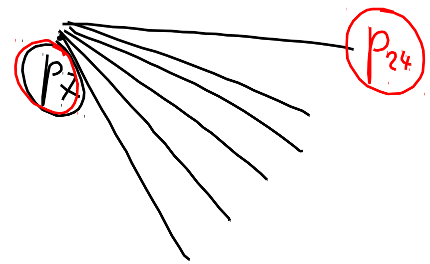
Najdeme bod nejníže vlevo p_1 $\quad O(n)$

p_1 je určitě na hranici konvexního obalu



$O(n)$

$$\operatorname{tg} \alpha_i = \frac{p_{1y} - p_{iy}}{p_{1x} - p_{ix}}$$



$O(n)$

najdeme i s největší směrnicí

Děláme tak dlouho až se dostaneme do p_1.

dohromady $O(kn)$

Metoda rozděl a panuj

n bodů   dva konv. mnohoúhelníky – obaly pro $\frac{n}{2}$ bodů

z nich vyrobíme konv. obal pro všechny body

$$T(n) = 2\,T\left(\frac{n}{2}\right) + O(n)$$

Tato rekurentní formule pro časovou náročnost dává

$$T(n) = O(n \log n)$$

$$T(2^k) = 2 \cdot T(2^{k-1}) + 2^k$$

$$\Rightarrow T(2^k) = 2^k k$$

$$T(2) = 2$$

$$T(2^{k-1}) = (k-1)2^{k-1}$$