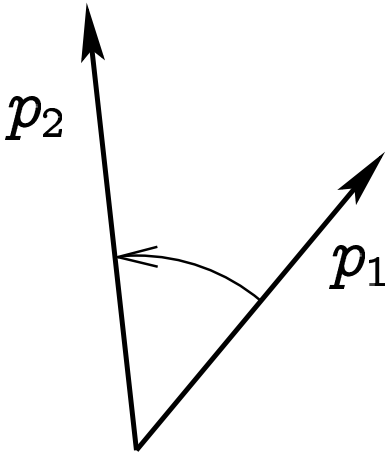
Tasandi punktide hulk $\cong \mathbb{R} \times \mathbb{R}$.

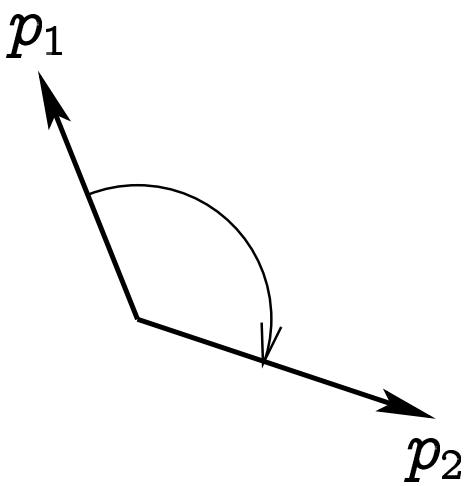
(Sirg)lõiku kujutame tema otspunktide paarina.

Vektorit kujutame ühe punktina — sellena, milleks ta teisendab koordinaatide alguspunkti.

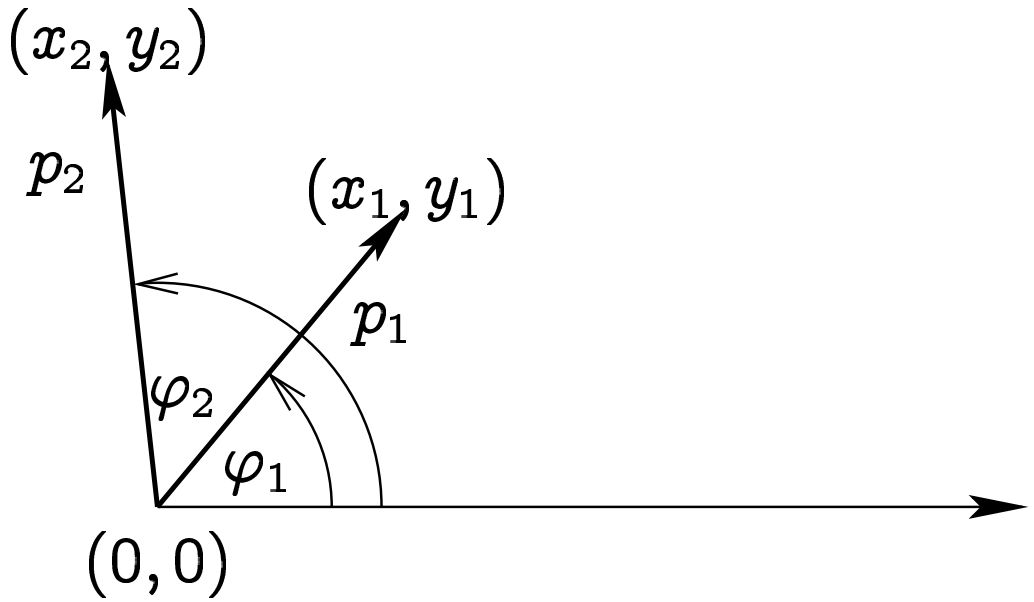
Antud n sirglõiku. Kas nende seas leidub kaks tükki, mis teineteisega lõikuvad?

Antud kaks vektorit p_1 ja p_2. Vektorit p_1 tuleb pöörata ülimalt 180° ühes või teises suunas, selleks, et ta langeks kokku vektoriga p_2.





Kummas suunas?



Meid huvitab $\varphi_2 - \varphi_1$. Meid huvitab, kas see vahe on $180°$-st suurem või väiksem.

Meid huvitab, kas $\sin(\varphi_2 - \varphi_1)$ on positiivne või negatiivne.

$$\sin(\varphi_2 - \varphi_1) = \sin\varphi_2 \cos\varphi_1 - \cos\varphi_2 \sin\varphi_1 = \frac{y_2}{|p_2|} \cdot \frac{x_1}{|p_1|} - \frac{x_2}{|p_2|} \cdot \frac{y_1}{|p_1|} \ .$$

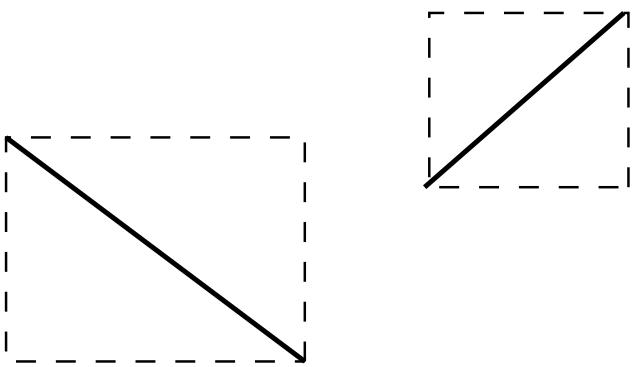
Kuna $|p_1|$ ja $|p_2|$ on positiivsed, siis huvitab meid avaldise $x_1 y_2 - x_2 y_1$ märk.

Kui see on positiivne, siis on p_2 p_1-st „vasakul“, kui negatiivne, siis „paremal“. Kui ta on 0, siis on p_1 ja p_2 samasihilised.

Kuidas kontrollida, kas lõigud $\overline{p_1p_2}$ ja $\overline{p_3p_4}$ lõikuvad?

Ebatäpsusi sissetoovaid tehteid (näiteks jagamine) ei tahaks seejuures kasutada.

Kontrollime kõigepealt, kas neid piiravad horisontaalsete ja vertikaalsete külgedega ristkülikud lõikuvad.



Kui ei, siis ei lõiku ka sirglõigud.

Olgu $p_i = (x_i, y_i)$ (kus $1 \leqslant i \leqslant 4$). Olgu

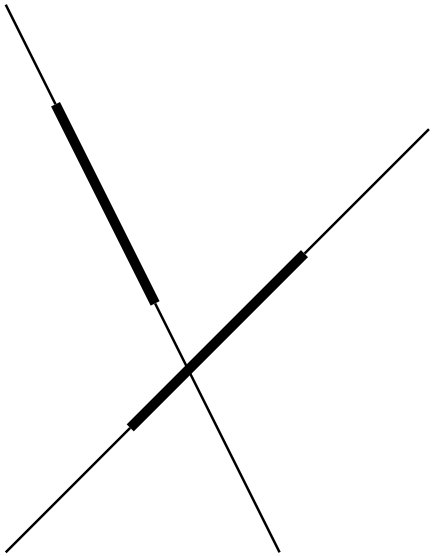
$$\begin{aligned}
\hat{x}_1 &= \min(x_1, x_2) & \hat{y}_1 &= \min(y_1, y_2) \\
\hat{x}_2 &= \max(x_1, x_2) & \hat{y}_2 &= \max(y_1, y_2) \\
\hat{x}_3 &= \min(x_3, x_4) & \hat{y}_3 &= \min(y_3, y_4) \\
\hat{x}_4 &= \max(x_3, x_4) & \hat{y}_4 &= \max(y_3, y_4),
\end{aligned}$$

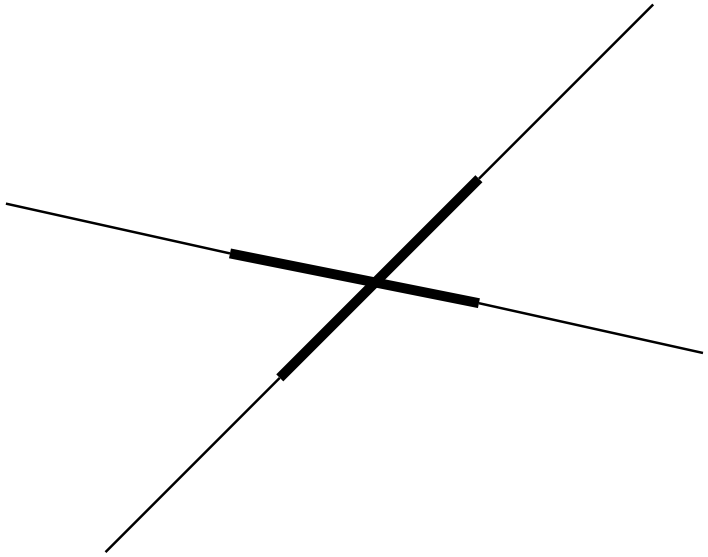
Siis need ristkülikudd on vastavalt $\hat{x}_1 \leqslant x \leqslant \hat{x}_2$, $\hat{y}_1 \leqslant y \leqslant \hat{y}_2$ ja $\hat{x}_3 \leqslant x \leqslant \hat{x}_4$, $\hat{y}_3 \leqslant y \leqslant \hat{y}_4$. Nad lõikuvad parajasti siis, kui nad lõikuvad mõlemas dimensioonis.

Lõikuvad, kui $z_3 \leqslant z_2$ ja $z_1 \leqslant z_4$.

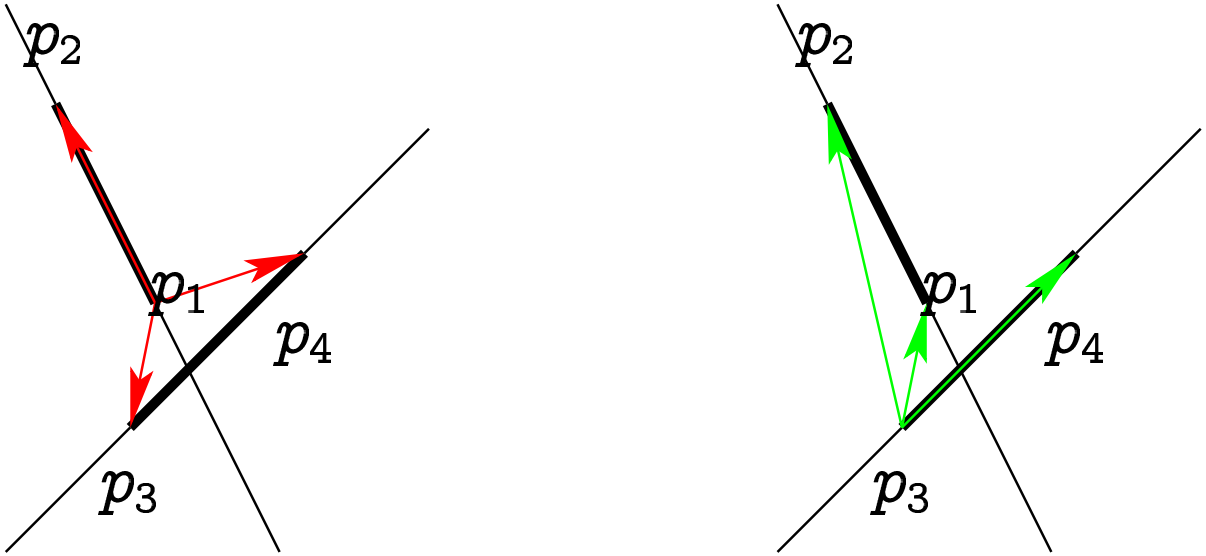
Oletame, et piiravad ristkülikud lõikuvad. Vaatame sirgeid, mille määravad need sirglõigud.

Kui üks lõik asub teisega määratud sirgest ühel pool, siis nad ei lõiku. Kui kumbki lõikab teisega määratud sirget, siis nad lõikuvad.





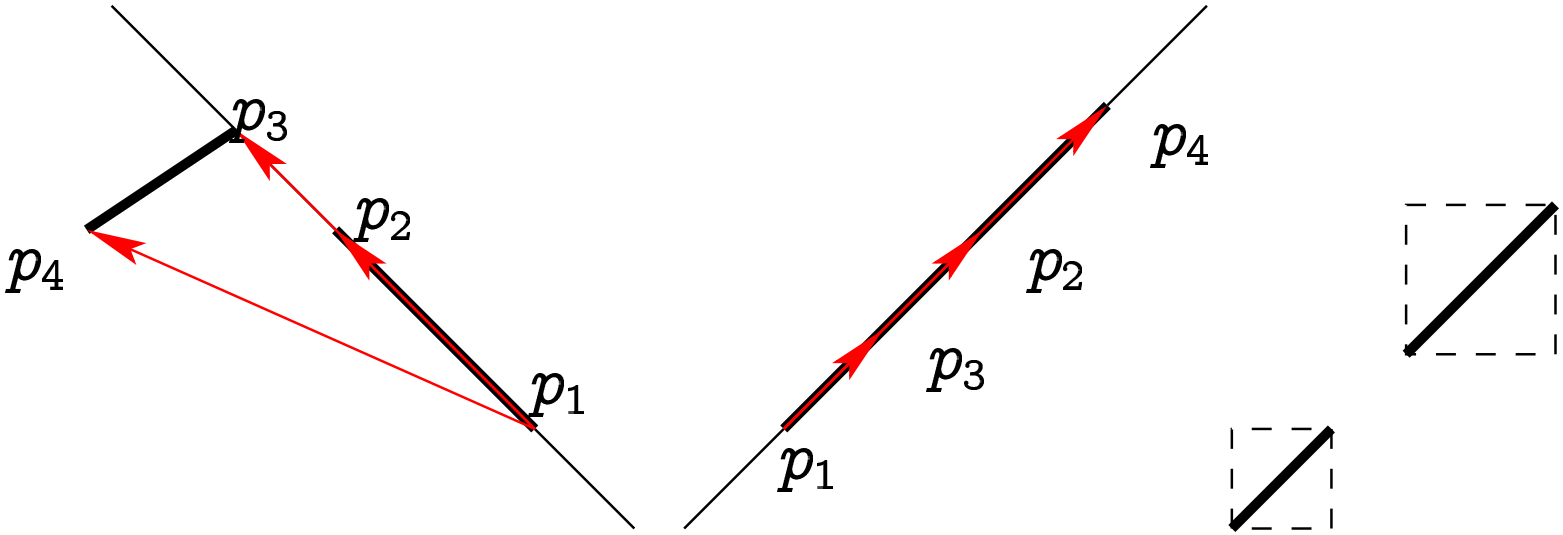
$\overline{p_3p_4}$ asub $\overline{p_1p_2}$-ga määratud sirgest ühel pool parajasti siis, kui tema mõlemad otspunktid asuvad sellest ühel pool.



Meil tuleb leida, kummale poole jäävad vektorist $\overrightarrow{p_1p_2}$ vektorid $\overrightarrow{p_1p_3}$ ja $\overrightarrow{p_1p_4}$.

Samuti, kummale poole jäävad vektorist $\overrightarrow{p_3p_4}$ vektorid $\overrightarrow{p_3p_1}$ ja $\overrightarrow{p_3p_2}$.

Mis siis, kui mõned neist vektoritest on samasihilised?



Siis võime lugeda, et üks lõik lõikub teise poolt määratud sirgega.

Parempoolne variant pole siinkohal võimalik, sest me oleme juba kontrollinud, kas piiravad ristkülikud lõikuvad.

vekt_asend$((x_1, y_1), (x_2, y_2))$ on

1 $d := x_1 y_2 - x_2 y_1$
2 **if** $d > 0$ **then return** „+“
3 **if** $d < 0$ **then return** „−“ **else return** „0“

p3_asend$((x_1, y_1), (x_2, y_2), (x_3, y_3))$ on

1 **return** vekt_asend$((x_2 - x_1, y_2 - y_1), (x_3 - x_1, y_3 - y_1))$

lõik_sirge?$((x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4))$ on

1 $a :=$ p3_asend$((x_1, y_1), (x_2, y_2), (x_3, y_3))$
2 $b :=$ p3_asend$((x_1, y_1), (x_2, y_2), (x_4, y_4))$
3 **if** $a =$ „0“ **or** $b =$ „0“ **then return** true
4 **return** $(a \neq b)$

lõik_bbox?$((x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4))$ on

1 $\hat{x}_1 := \min(x_1, x_2)$; $\hat{y}_1 := \min(y_1, y_2)$
2 $\hat{x}_2 := \max(x_1, x_2)$; $\hat{y}_2 := \max(y_1, y_2)$
3 $\hat{x}_3 := \min(x_3, x_4)$; $\hat{y}_3 := \min(y_3, y_4)$
4 $\hat{x}_4 := \max(x_3, x_4)$; $\hat{y}_4 := \max(y_3, y_4)$
5 **return** $(\hat{x}_3 \leqslant \hat{x}_2) \wedge (\hat{x}_1 \leqslant \hat{x}_4) \wedge (\hat{y}_3 \leqslant \hat{y}_2) \wedge (\hat{y}_1 \leqslant \hat{y}_4)$

lõikuvad_lõigud?(p_1, p_2, p_3, p_4) on

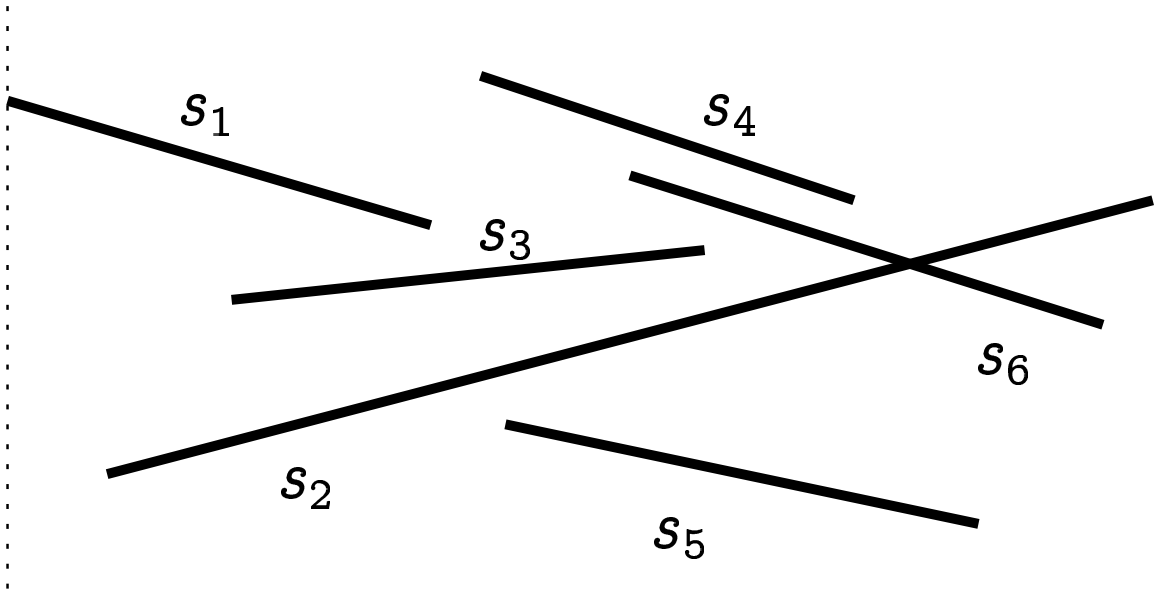
1 **return** lõik_bbox?(p_1, p_2, p_3, p_4)
and lõik_sirge?(p_1, p_2, p_3, p_4)
and lõik_sirge?(p_3, p_4, p_1, p_2)

Olgu meil nüüd antud sirglõigud $s_1, \ldots, s_n$. Tahame leida, kas nende seas mõned omavahel lõikuvad. Kõiki lõikuvaid paare ei taha leida.

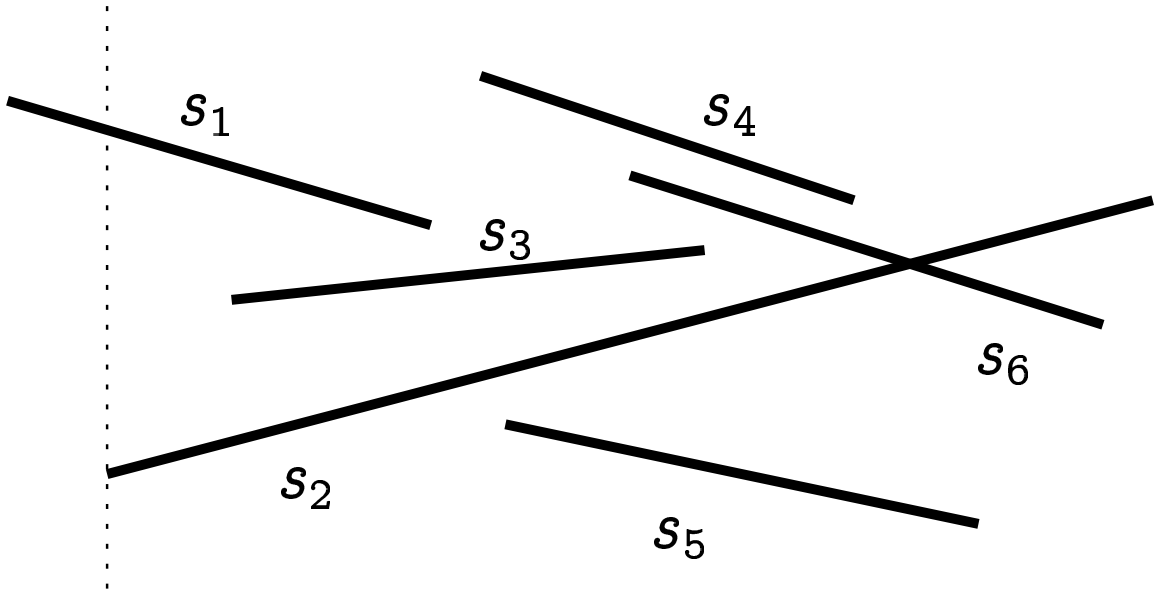
Loeme, et ükski lõik ei ole vertikaalne ning ükski kolmik ei lõiku ühes punktis.

Lahendusidee: libistame vertikaalset joont üle nende lõikude vasakult paremale, kuni leiame lõikepunkti (või jõuame kõigist lõikudest üle).

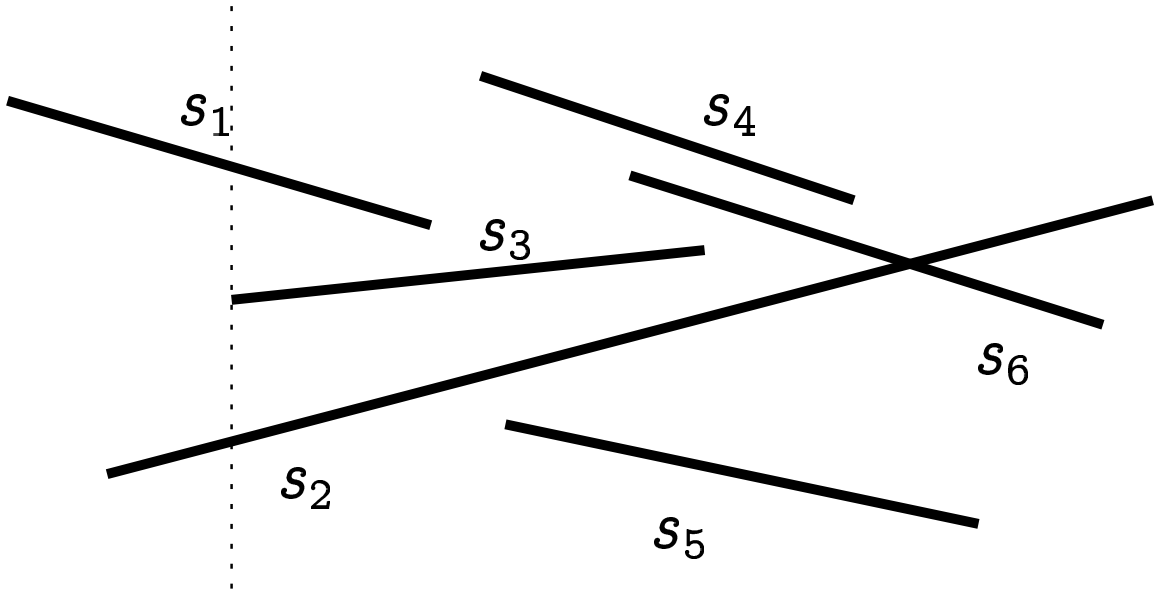
Peame arvet, mis järjekorras antud sirglõigud selle vertikaalse joonega lõikuvad.



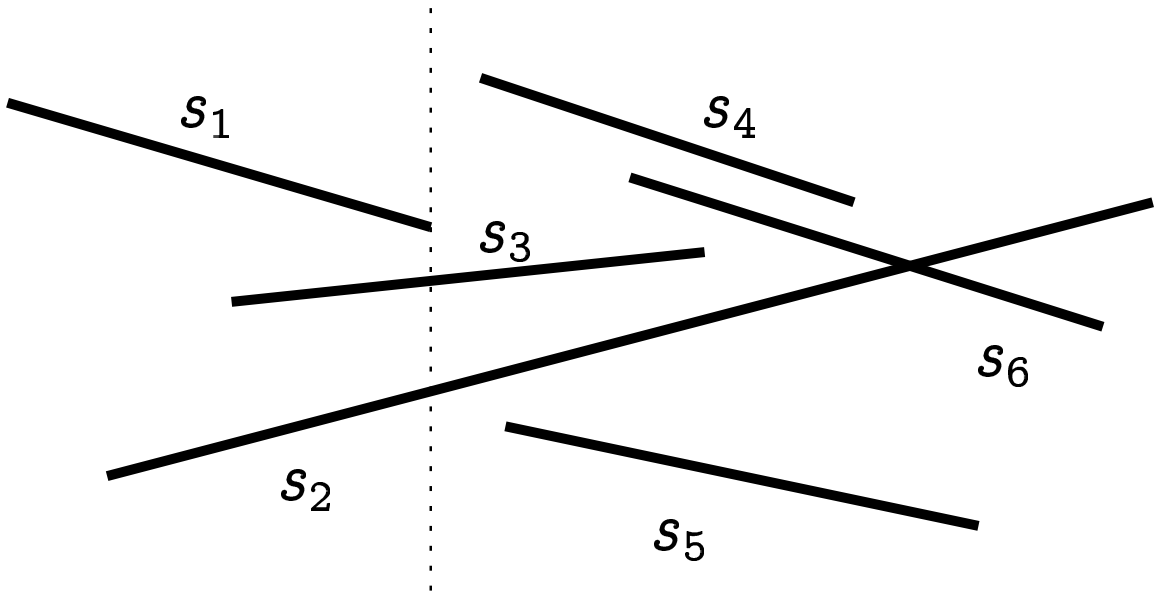
Lõigatavad lõigud: s_1



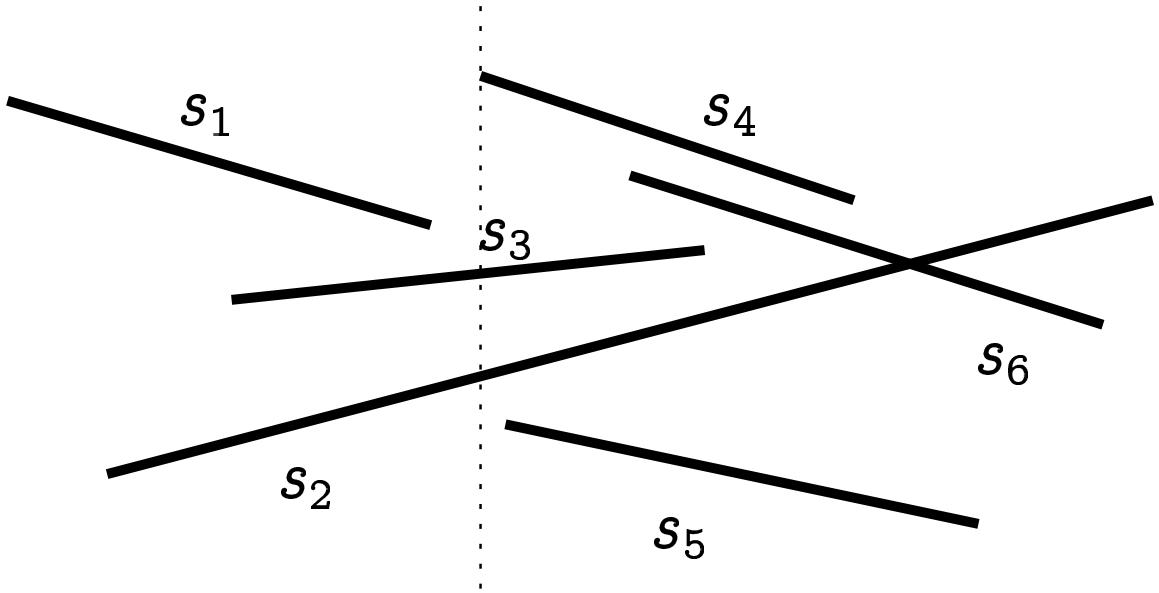
Lõigatavad lõigud: s_1, s_2



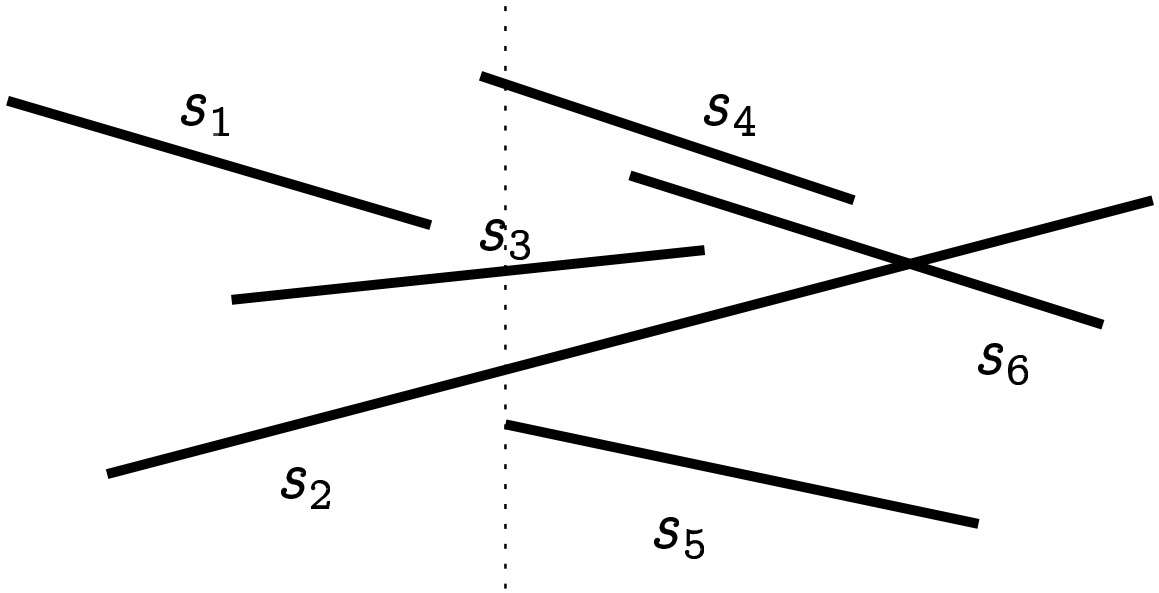
Lõigatavad lõigud: s_1, s_3, s_2



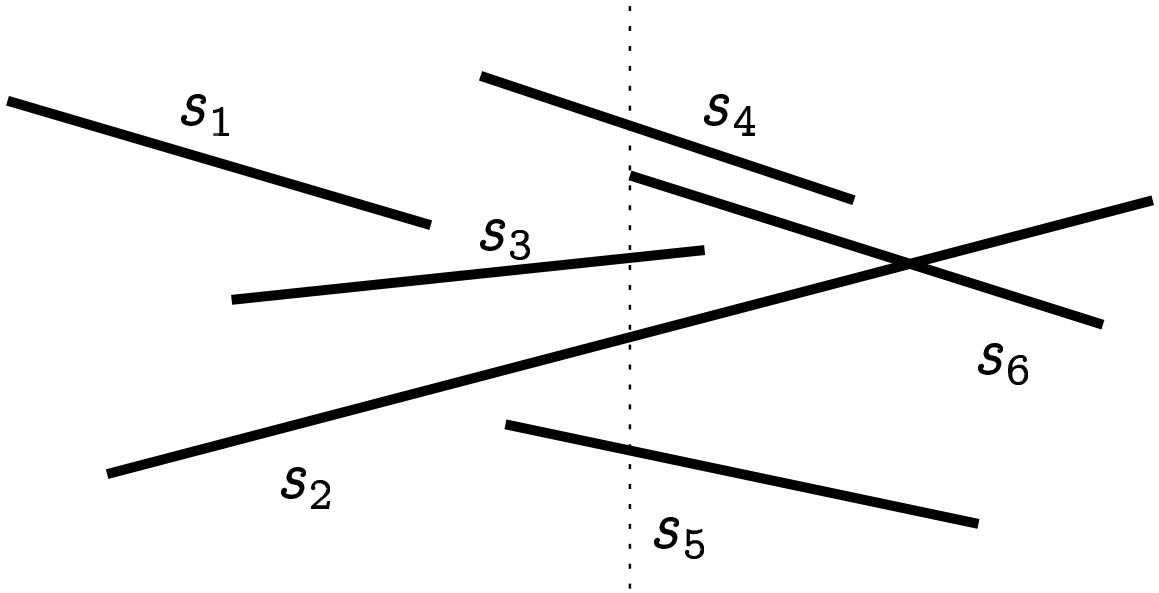
Lõigatavad lõigud: s_3, s_2



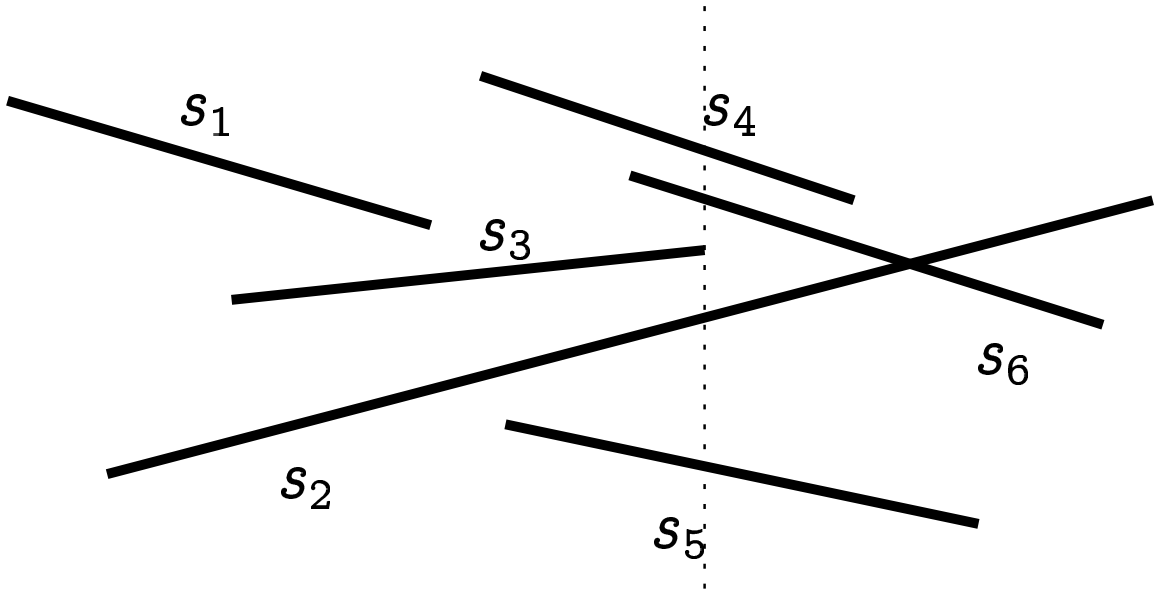
Lõigatavad lõigud: s_4, s_3, s_2



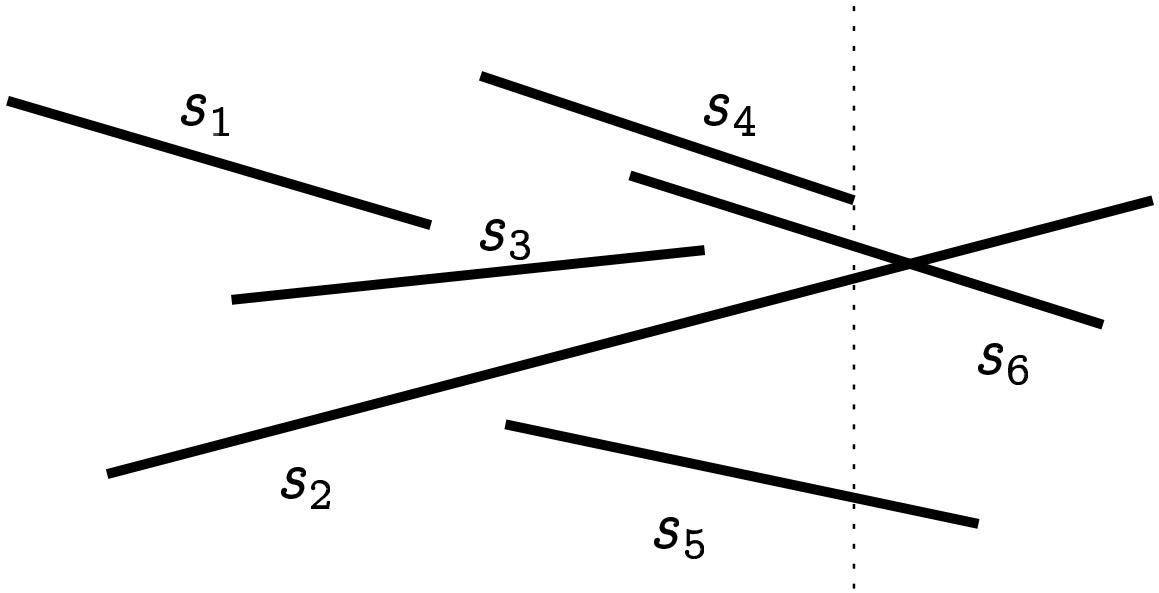
Lõigatavad lõigud: s_4, s_3, s_2, s_5



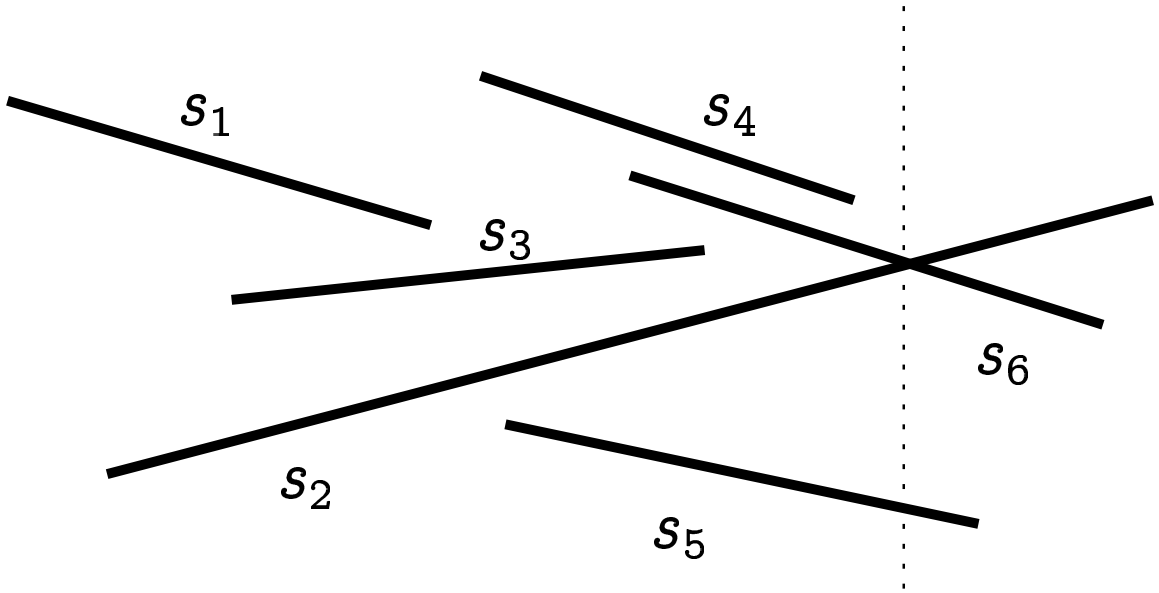
Lõigatavad lõigud: s_4, s_6, s_3, s_2, s_5



Lõigatavad lõigud: s_4, s_6, s_2, s_5



Lõigatavad lõigud: s_6, s_2, s_5



Lõigatavad lõigud: s_2, s_6, s_5

Paneme tähele, et senikaua, kuni pole olnud ühtegi lõikepunkti, lõikude suhteline asend vertikaalsel joonel ei muutu.

Enne, kui mingid kaks lõiku lõikuvad, peavad nad vertikaalsel joonel järjest esinema.

Algoritmis me võtame mingite kohtade peal vertikaalsed jooned, nendel järjest mingid kaks lõiku ning kontrollime, kas need lõigud lõikuvad.

Kohtade valik on selline, et me ühtegi lõikumist maha ei maga.

Olgu $p_1, \ldots, p_{2n}$ lõikude $s_1, \ldots, s_n$ otspunktid, mis on sorteeritud x-koordinaadi järgi (kui kahe punkti x-koordinaadid on võrdsed, siis y-koordinaadi järgi).

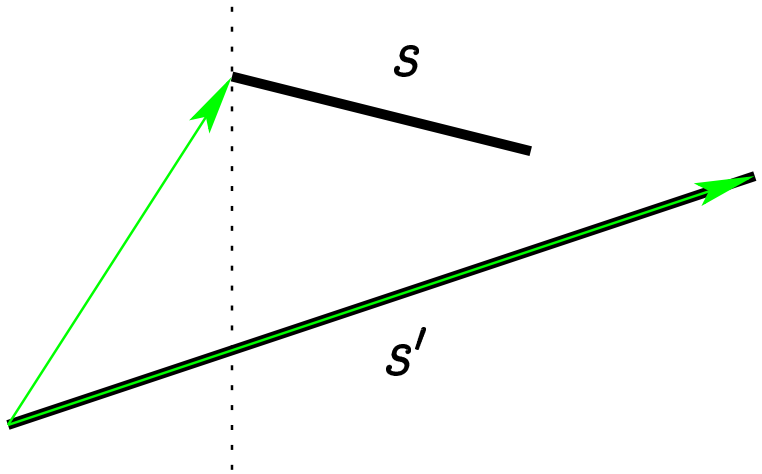
Olgu T dünaamiline järjend, mis toetab järgmisi operatsioone:

- lisa(T, s),
- kustuta(T, s),
- eelmine(T, s),
- järgmine(T, s).

T-s hoiame lõike. T-ks sobib kahendpuu. Kui ta on AVL-puu, siis on kõik operatsioonid keerukusega $O(\log n)$.

lisa(T, s) peab panema s-i kohta, mis vastab tema positsioonile vertikaalsel joonel.

Võrdlemaks, kas s asub juba T-sse kuuluvast s'-st üleval või allpool, uurime, kumba pidi asuvad allpool kujutatud vektorid.



Käi punktid $p_1, \ldots, p_{2n}$ vasakult paremale läbi ja

- Kui p_i on mingi s-i vasakpoolne otspunkt, siis
  - lisa(T, s)
  - kui eelmine(T, s) / järgmine(T, s) leidub, siis kontrolli, kas ta lõikub s-ga. Kui jah, siis tagasta „jah“ ja lõpeta.
- Kui p_i on mingi s-i parempoolne otspunkt, siis
  - kui eelmine(T, s) ja järgmine(T, s) mõlemad leiduvad, siis kontrolli, kas nad lõikuvad. Kui jah, siis tagasta „jah“ ja lõpeta.
  - kustuta(T, s).

Kui lõikumisi ei leitud, tagasta „ei“.

Ilmne on, et kui algoritm tagastab „jah“, siis leiduvad lõigud, mis lõikuvad. Algoritm on siis just ühe paari leidnud.

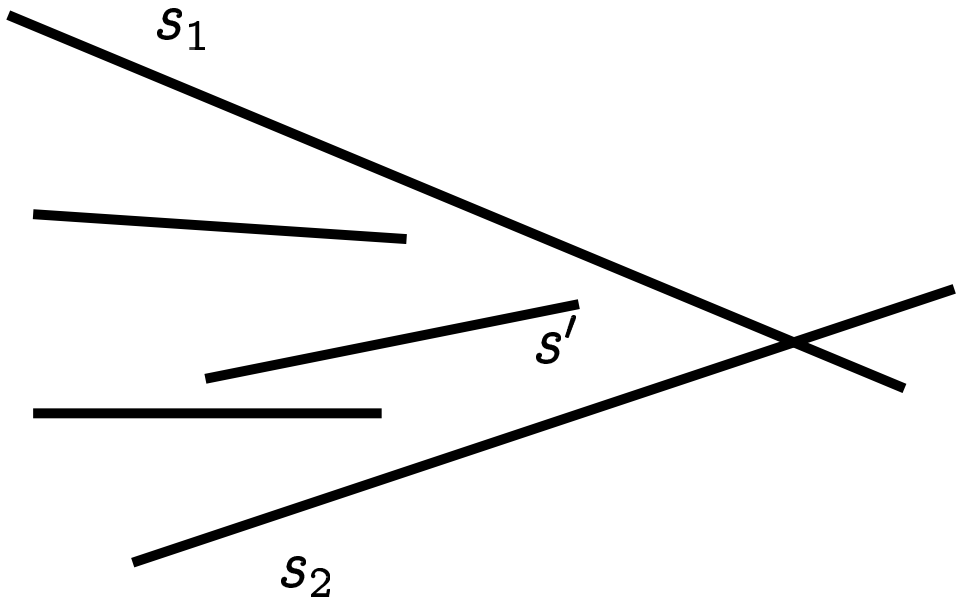
Näitame, et kui ta tagastab „ei“, siis selliseid lõike ei leidu.

Oletame vastuväiteliselt, et algoritm tagastab „ei“, aga mingid lõigud s_1 ja s_2 lõikuvad.

Lõikugu s_1 ja s_2 nii vasakul kui võimalik. Kui vähima x-koordinaadiga lõikepunkte on mitu, siis valime neist alumise.

S.t. s_1 ja s_2 lõikepunktist vasakul ei toimu ühtegi lõikumist.

Alaku s_1 enne s_2-e. Kui s_2 alguspunkti töötlemisel ei leita üles tema lõikumist s_1-ga, siis peavad mingid lõigud veel s_1 ja s_2 vahel asuma.



Vaatame neid lõike. Nad kõik lõppevad enne s_1 ja s_2 lõikumist. Olgu s' lõik, mis neist kõige hiljem lõppeb. Tema lõppemisel avastatakse s_1 ja s_2 lõikumine.

Algoritmi tööaeg:

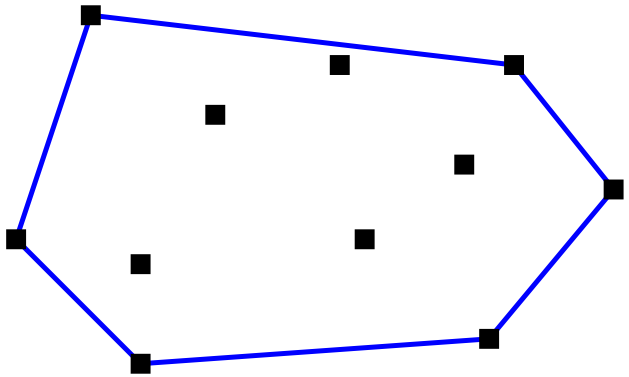
- $2n$ punkti sorteerimine — $O(n \log n)$.
- Tsükkel üle $2n$ punkti, igal iteratsioonil
  - kolm operatsiooni järjendiga T, keerukus $O(\log n)$.
  - kuni kaks lõikude lõikumise kontrolli, keerukus $O(1)$.

  Seega kokku $O(n \log n)$.

Kokku on siis tööaeg $O(n \log n)$.

Kõikvõimalikke lõikumisi pole selle ajaga võimalik üles lugeda, neid võib olla $\Omega(n^2)$.

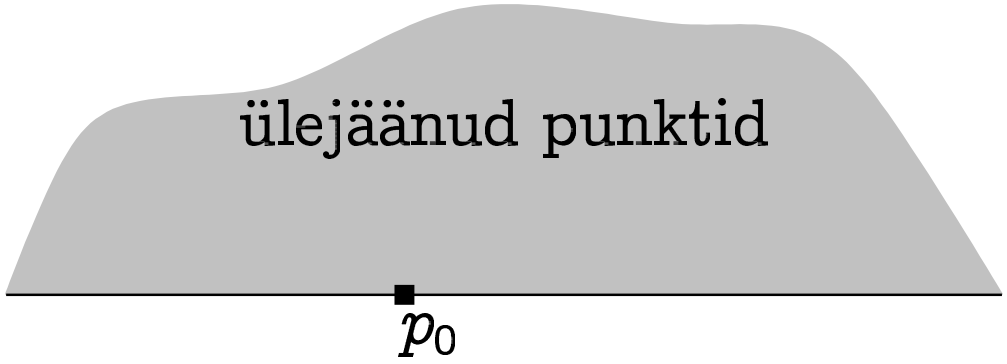
Olgu antud mingid punktid $p_0, p_1, \ldots, p_n$. Nende *kumeraks katteks* nimetatakse vähimat kumerat hulknurka, mis kõiki neid punkte sisaldab.



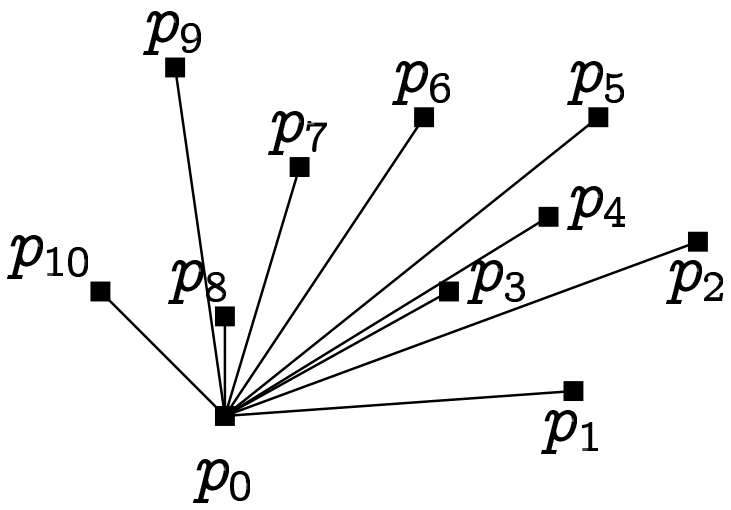
Kumera katte tipud on mingites neist punktidest.

Kuidas leida punktihulga kumerat katet?

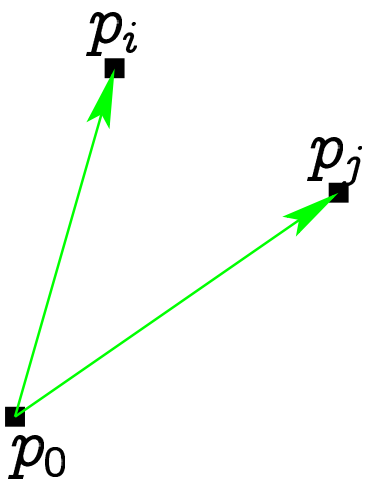
Grahami seiremeetod. Olgu p_0 kõige väiksema y-koordinaadiga punkt. Kui neid on mitu, siis kõige väiksema x-koordinaadiga.



Sorteerime ülejäänud punktid p_i vektorite $\overrightarrow{p_0p_i}$ tõusu järgi.



Punktide võrdlemiseks vaatame, kumba pidi on järgmised vektorid.



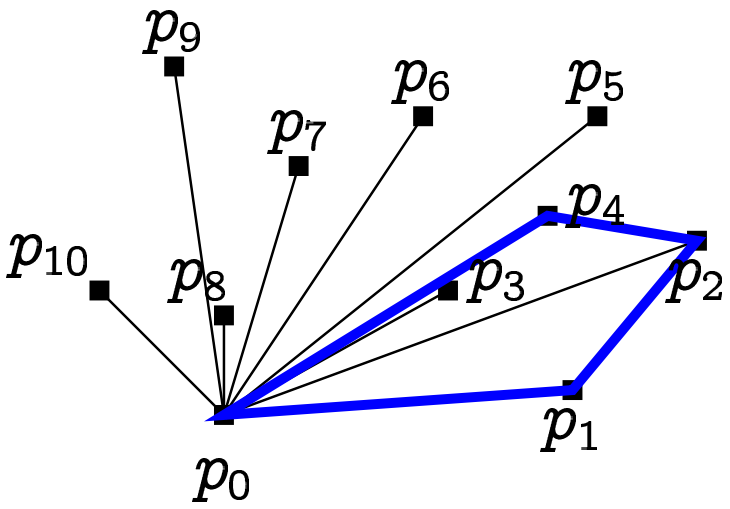
S.t. tõusunurki ei tule välja arvutada.

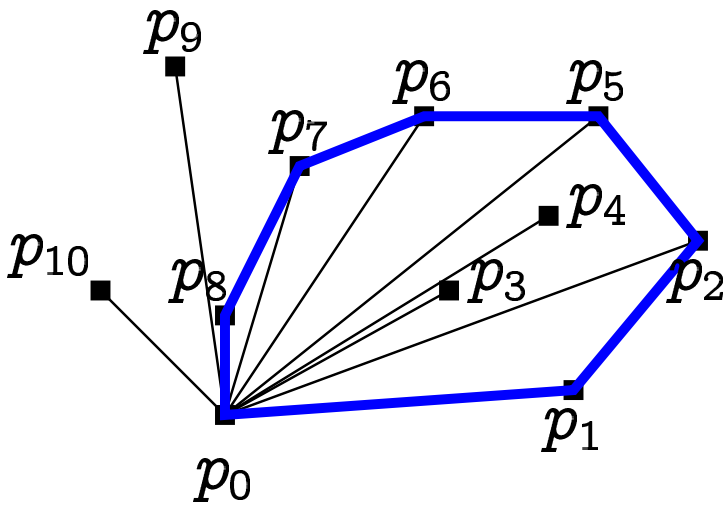
Kui mitmel punktil on sama tõusunurk, siis võime arvesse võtta ainult kaugeima punkti. Lähemad punktid jäävad kindlasti kumeraks katteks oleva hulknurga sisemusse.

Kaugeim punkt — suurima y-koordinaadiga.

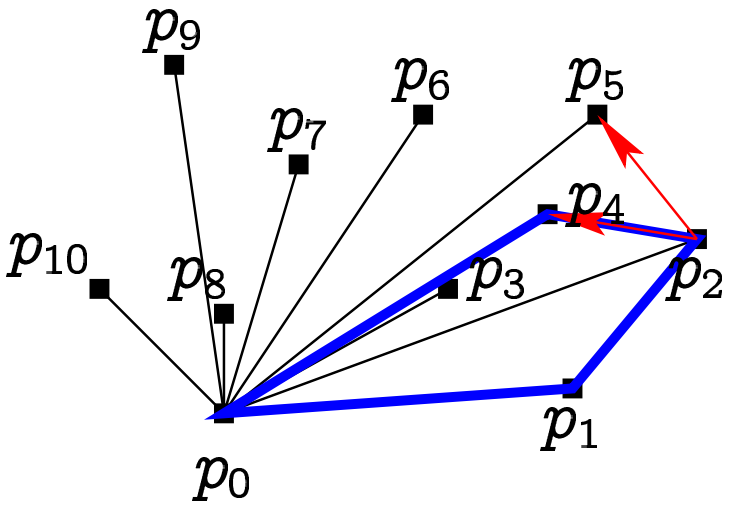
Punktide p_0, p_1, p_2 kumer kate on (p_0, p_1, p_2). (kui p_0, p_1 ja p_2 ei asu ühel sirgel)

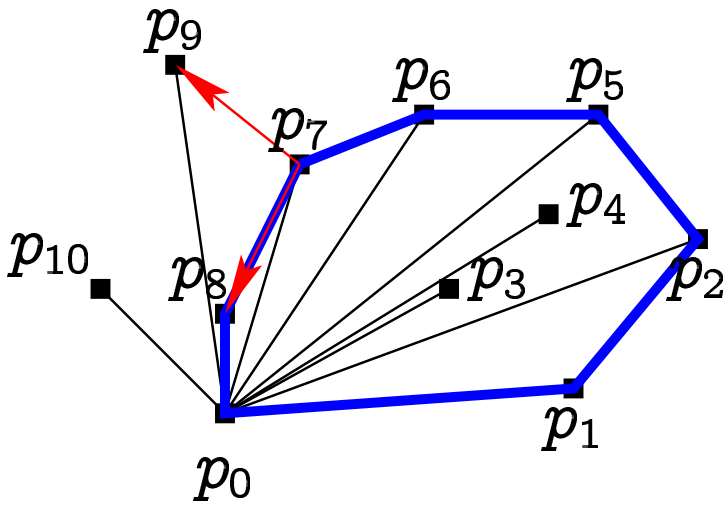
Olgu me juba leidnud punktide $p_0, \ldots, p_i$ kumera katte $(q_0, \ldots, q_j)$. Vaatame punkti p_{i+1}. Ta kuulub kindlasti punktide $p_0, \ldots, p_{i+1}$ kumerasse kattesse.





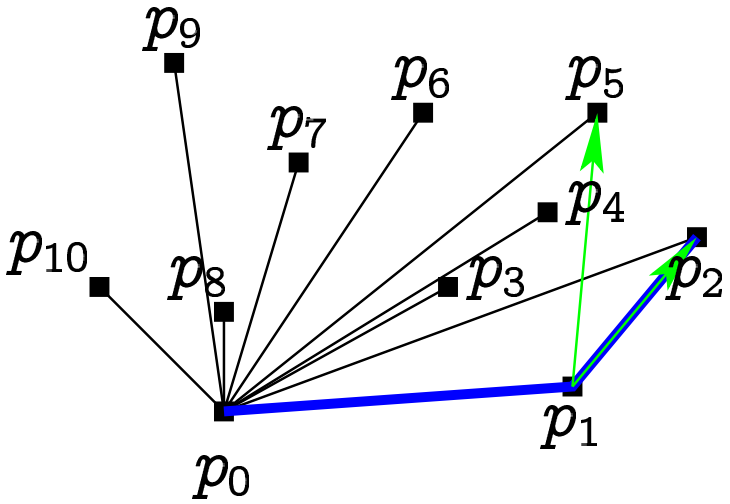
Kontrollimaks, kas q_j kuulub $p_0, \ldots, p_{i+1}$ kumerasse kattesse, vaatame vektoreid $\overrightarrow{q_{j-1}q_j}$ ja $\overrightarrow{q_{j-1}p_{i+1}}$.

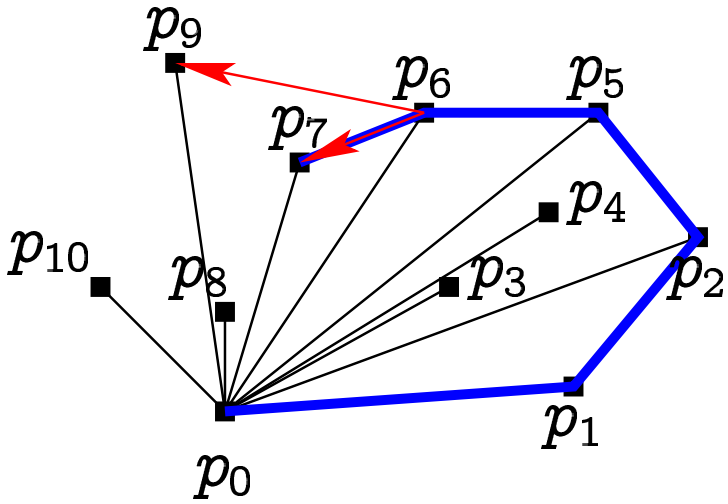




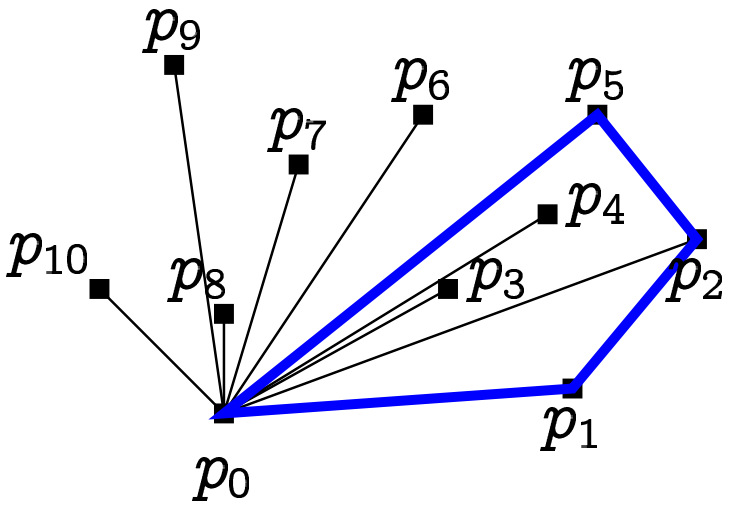
Kui $\overrightarrow{q_{j-1}p_{i+1}}$ asub $\overrightarrow{q_{j-1}q_j}$-st paremal, siis ei kuulu q_j punktide $p_0, \ldots, p_{i+1}$ kumerasse kattesse.

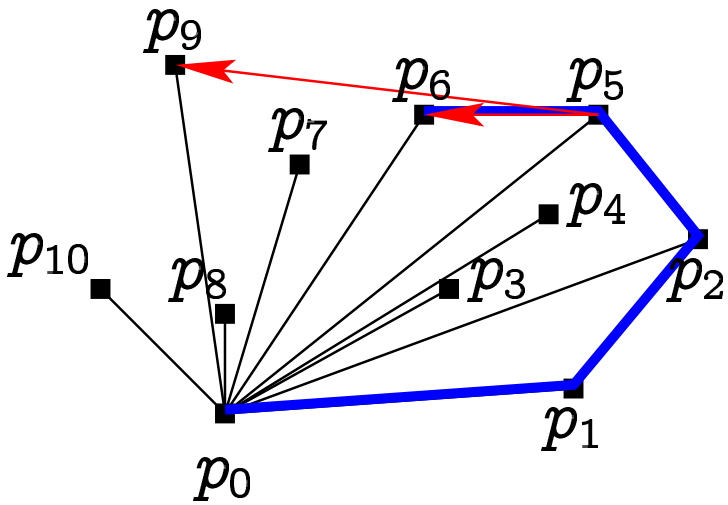
Sel juhul vähendame j-i ühe võrra ja proovime uuesti.

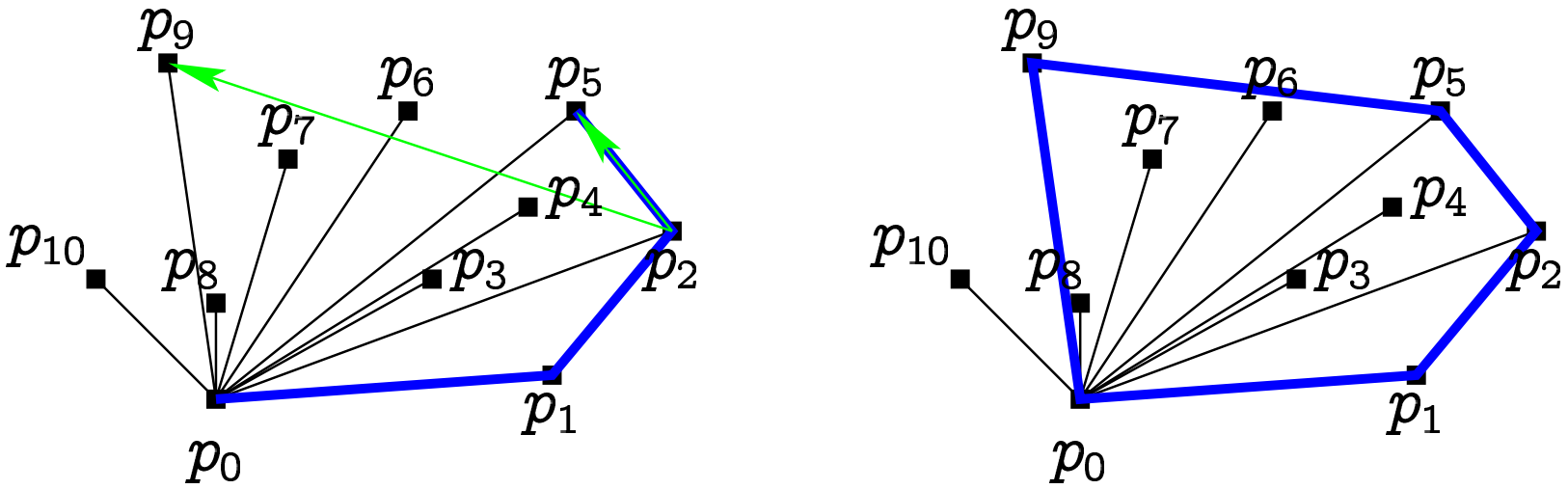




Kui $\overrightarrow{q_{j-1}p_{i+1}}$ asub $\overrightarrow{q_{j-1}q_j}$-st vasakul, siis oleme $p_0, \ldots, p_{i+1}$ kumera katte leidnud. See on $(q_0, \ldots, q_j, p_{i+1})$.







Algoritmi realiseerides on mõttekas punkte $q_0, \ldots, q_j$ pinus hoida, nii et q_j on pealmine.

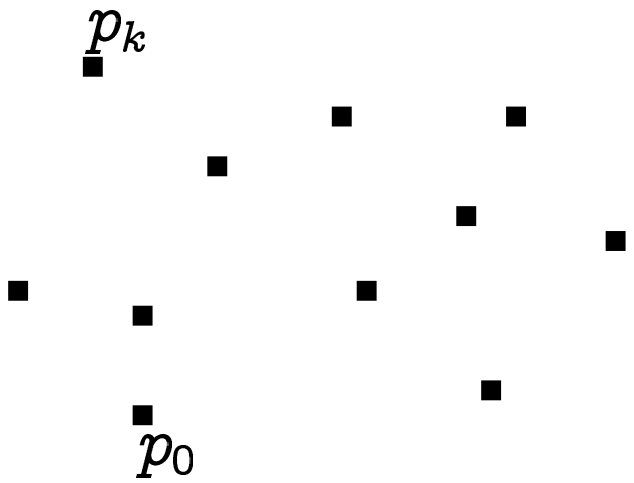
Keerukus:

- Vähima y-koordinaadiga punkti otsimine — $O(n)$.
- Sorteerimine — $O(n \log n)$.
- Kumera katte leidmisel võib iga punkt
  - (täpselt) ühel korral kattesse sattuda;
  - ülimalt ühel korral kattest eemaldatud saada.

  Seega võtab punktide lisamine/eemaldamine aega $O(n)$.
- Igale vektorite omavahelise asendi võrdlemisele järgneb kas mingi punkti lisamine kattesse või eemaldamine sealt. Seega on võrdlusi $O(n)$.

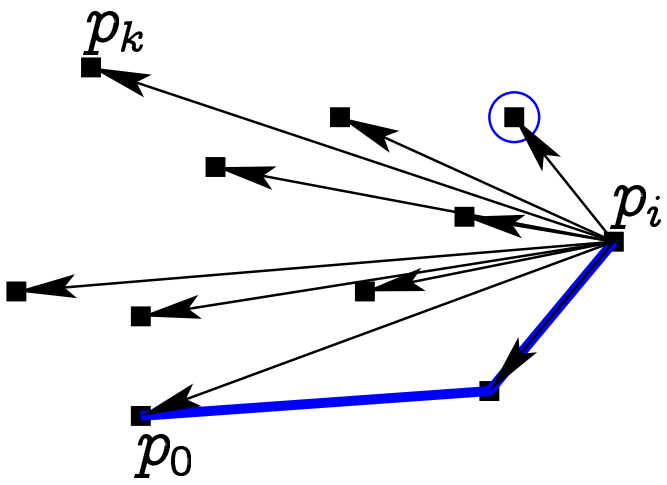
Kokku seega $O(n \log n)$.

Jarvise mähkimismeetod. Olgu p_0 kõige väiksema ja p_k kõige suurema y-koordinaadiga punkt. (kui neid on mitu, siis kõige vasakpoolsem(ad))



Nii p_0 kui ka p_k kuuluvad kumerasse kattesse. Neist kahest punktist „paremale“ jääva kumera katte osa leidmiseks:

Esimeseks punktiks kumera katte parempoolsel osal võtame p_0-i. Kui p_i on viimane leitud punkt kumera katte parempoolsel osal, siis vaatame vektoreid p_i-st kõigisse teistesse punktidesse.



Järgmiseks punktiks võtame parempoolseima vektori teise otstipu. Jätkame, kuni jõuame p_k-ni.

Samamoodi leiame kumera katte vasakpoolse osa.

Keerukus: Punkti lisamisi kumerasse kattesse on samapalju kui kumeras kattes punkte on.

Igal lisamisel tuleb leida parem-/vasakpoolseim punkt. Leidmine on analoogiline massiivi minimaalse elemendi leidmisega. Ühe punkti lisamise keerukus on seega $O(n)$.

Kogukeerukus on $O(nh)$, kus h on tippude arv kumeras kattes.

Valides Grahami ja Jarvise meetodite vahel tuleks võrrelda suurusi h ja $\log n$ (s.t. nende eeldatavate väärtuste proportsioone konkreetses rakenduses).

Antud punktid $p_1, \ldots, p_n$ (kõik erinevad). Leida nende seast kaks teineteisele lähimat punkti.

Lihtne lahendus — vaatame kõik punktipaarid läbi, leiame nendevaheliste kauguste seast minimaalse. Keerukus: $\Theta(n^2)$.

Punktide $p_1 = (x_1, y_1)$ ja $p_2 = (x_2, y_2)$ vaheline kaugus $d(p_1, p_2)$ on

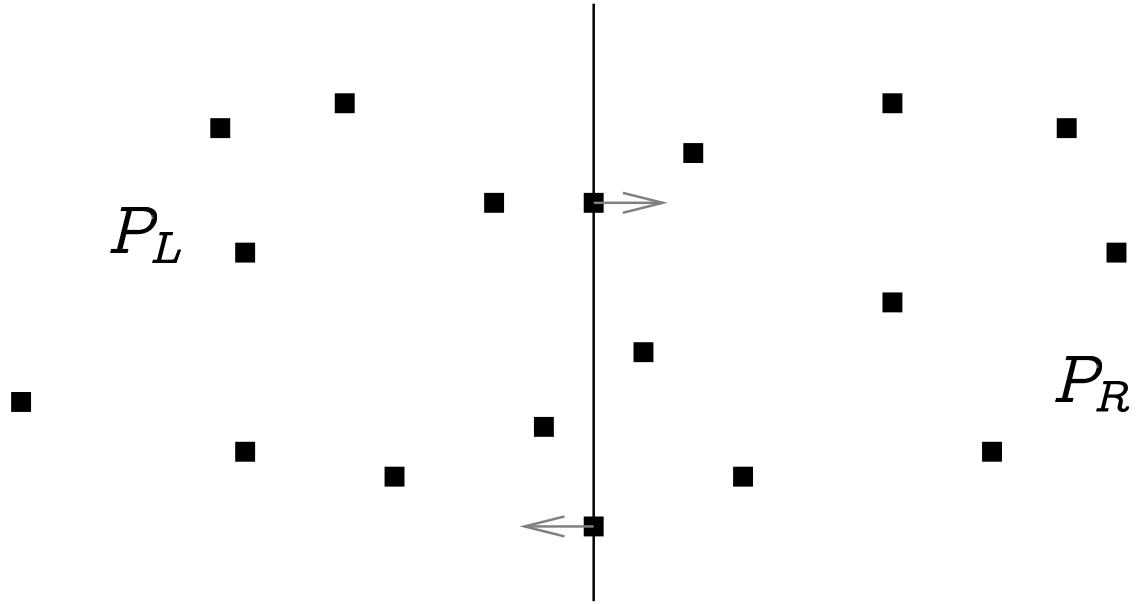
$$\sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \ .$$

Kui me vaid kaugusi võrrelda tahame, siis piisab kauguste ruutude võrdlemisest. S.t. ruutjuurt pole tarvis võtta.

Rekursiivne lahendus:

1. Jaga punktihulk P kaheks võrdse suurusega osaks P_L ja P_R, leia kummastki osast vähima kaugusega punktipaar.
   - Rekursioon lõppeb, kui $|P| \leqslant 3$.
2. Vali neist kahest paarist lähem, olgu nendevaheline kaugus δ.
3. Kontrolli, kas leiduvad punktid $p_L \in P_L$ ja $p_R \in P_R$, nii et $d(p_L, p_R) \leqslant \delta$.
   - Piisab selliste punktide $p_L \in P_L$ vaatamisest, mis on hulgale P_R lähemal kui δ. Sama $p_R \in P_R$ jaoks...

Jaotamise teeme vertikaalse sirgega.

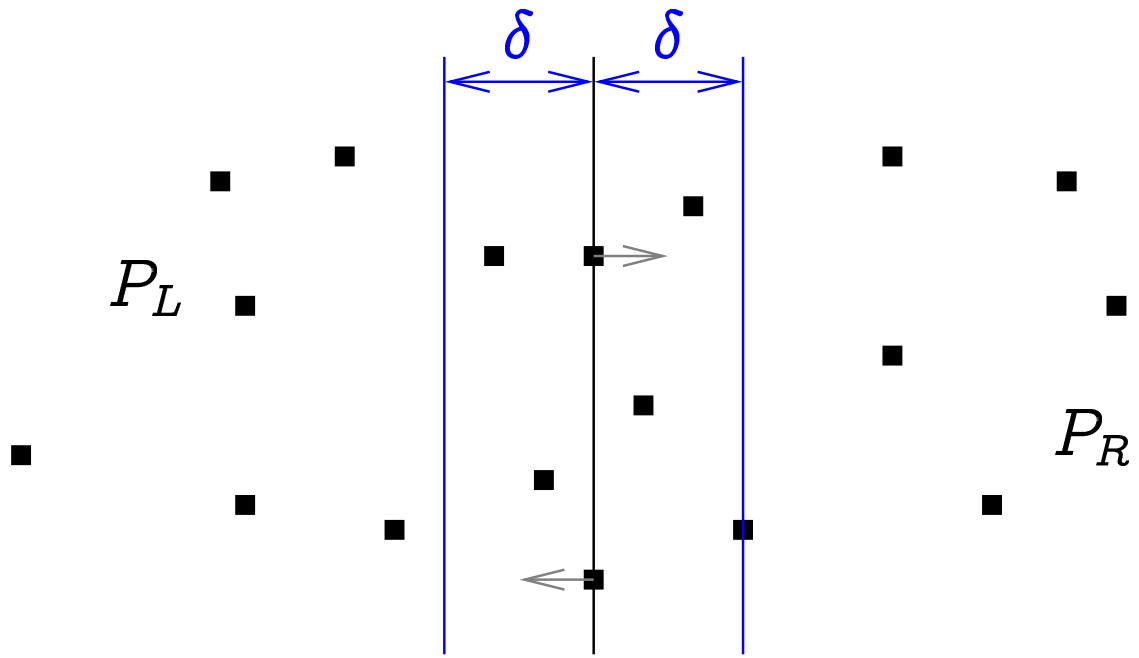


Sirgele jäävad punktid võivad kuuluda nii ühte kui ka teise poolde.

Olgu $X_1, \ldots, X_n$ punktid $p_1, \ldots, p_n$, sorteeritud x-koordinaadi järgi. Massiivi X kasutame vertikaalse sirge valikul.

Olgu δ_L vähima kaugusega punktipaari kaugus osas P_L ja δ_R sama asi osas P_R. Olgu $\delta = \min(\delta_L, \delta_R)$.

Kui $p_L \in P_L$ ja $p_R \in P_R$ on sellised, et $d(p_L, p_R) < \delta$, siis on p_L ja p_R eraldavast vertikaalsest joonest kaugusel $\leqslant \delta$.

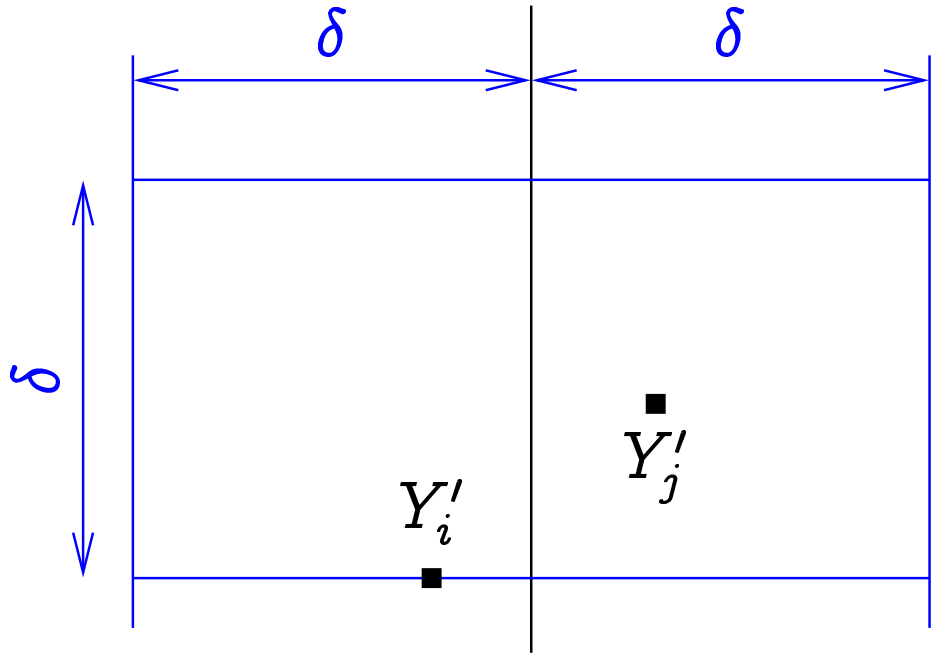


Olgu $P' \subseteq P$ kõigi selliste punktide hulk, mis on valitud vertikaalsest sirgest kaugusel $\leqslant \delta$. Olgu Y' nendesamade punktide järjend, y-koordinaadi järgi sorteeritud.

Olgu Y'_i ja Y'_j $(j > i)$ teineteisele lähimad punktid P'-st. Kui $d(Y'_i, Y'_j) \leqslant \delta$, siis $j - i \leqslant 6$.

S.t. (kui eelmine lõik on õige, siis) selleks, et leida teineteisele lähim punktipaar P'-st tuleb vaadata läbi massiiv Y' ja leida mingi punkti Y'_i kaugused ainult punktidest $Y'_{i+1}, \ldots, Y'_{i+6}$. S.t. teineteisele lähima punktipaari leidmine on tehtav lineaarses ajas.

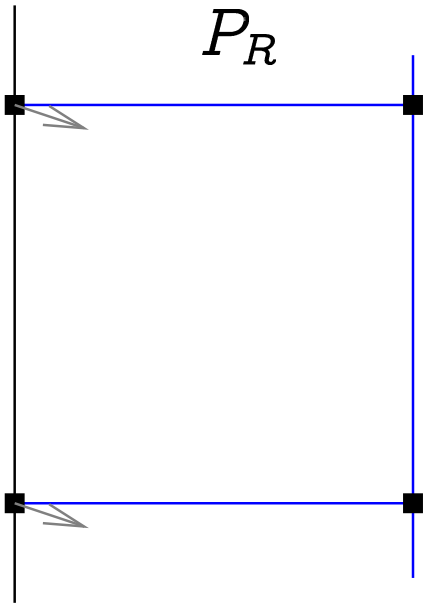
Kui $d(Y_i', Y_j') \leqslant \delta$, siis asuvad punktid Y_i' ja Y_j' ristkülikus mõõtmetega $2\delta \times \delta$.



Üldisust kitsendamata loeme, et $Y_i' \in P_L$. Vaatame juhtu, kus Y_i' ei asu vertikaalsel sirgel, mis eraldab P_L-i ja P_R-i.

Vaatame kõiki P_R-i kuuluvaid punkte, mis sinna ristkülikusse kuuluvad. Iga kahe P_R-i kuuluva punkti vaheline kaugus on vähemalt δ.

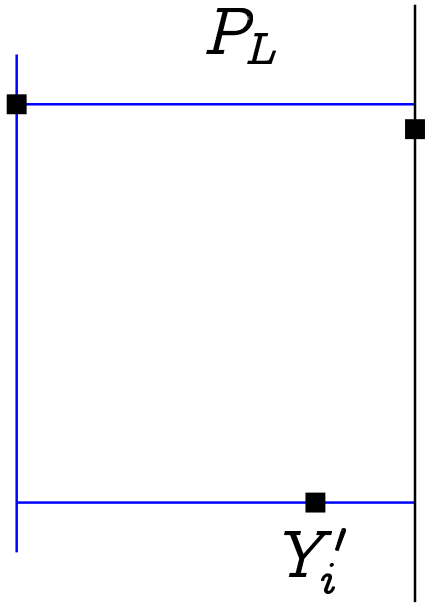
Nad kõik kuuluvad ruutu mõõtmetega $\delta \times \delta$.



Mahub ülimalt neli tükki.

Vaatame kõiki P_L-i kuuluvaid punkte, mis sinna ristkülikusse kuuluvad. Iga kahe P_L-i kuuluva punkti vaheline kaugus on vähemalt δ.

Nad kõik kuuluvad ruutu mõõtmetega $\delta \times \delta$.



Mahub ülimalt kolm tükki (kaasa arvatud Y_i').

Kui Y_i' asuks P_L-i ja P_R-i eraldaval vertikaalsel sirgel, siis mahuks vasakule poole neli punkti (k.a. Y_i') ja paremale poole kolm.

Seega kuulub vaadeldavasse ristkülikusse mõõtmetega $2\delta \times \delta$ ülimalt kuus punkti peale Y_i'.

Algoritm:

Eeltöö: sorteeri punktid $p_1, \ldots, p_n$ x-koordinaadi järgi massiivi X ja y-koordinaadi järgi massiivi Y. Kui kahel punktil on koordinaat võrdne, siis võta ettepoole see punkt, mille teine koordinaat on väiksem.

Põhitöö teeb ära funktsioon lähimad_punktid(X, Y, n).

lähimad_punktid(X, Y, n) on

```
 1  if n = 2 then return (X_1, X_2)
 2  if n = 3 then
 3      return lähim_kolmest(X_1, X_2, X_3)
 4  k := ⌊n/2⌋
 5  X^L := X_{1...k}; X^R := X_{k+1...n}
 6  l^L := 0; l^R := 0
 7  for i := 1 to n do
 8      if Y_i.x < X_k.x or Y_i.x = X_k.x ∧ Y_i.y ⩽ X_k.y then
 9          l^L := l^L + 1; Y^L_{l^L} := Y_i
10      else
11          l^R := l^R + 1; Y^R_{l^R} := Y_i
12  (q^L_1, q^L_2) := lähimad_punktid(X^L, Y^L, k)
13  (q^R_1, q^R_2) := lähimad_punktid(X^R, Y^R, n − k)
    ...
```

```
14  δ^L2 := (q_1^L.x − q_2^L.x)^2 + (q_1^L.y − q_2^L.y)^2
15  δ^R2 := (q_1^R.x − q_2^R.x)^2 + (q_1^R.y − q_2^R.y)^2
16  if δ^L2 < δ^R2 then
17      δruut := δ^L2; q_1 := q_1^L; q_2 := q_2^L
18  else
19      δruut := δ^R2; q_1 := q_1^R; q_2 := q_2^R
20  l' := 0
21  for i := 1 to n do
22      if (Y_i.x − X_k.x)^2 ⩽ δruut then
23          l' := l' + 1; Y'_l' := Y_i
    ...
```

```
24  for i := 1 to l' − 1 do
25      for j := 1 to min(6, l' − i) do
26          druut := (Y'_i.x − Y'_j.x)^2 + (Y'_i.y − Y'_j.y)^2
27          if druut < δruut then
28              δruut := druut; q_1 := Y'_i; q_2 := Y'_j
29  return (q_1, q_2)
```

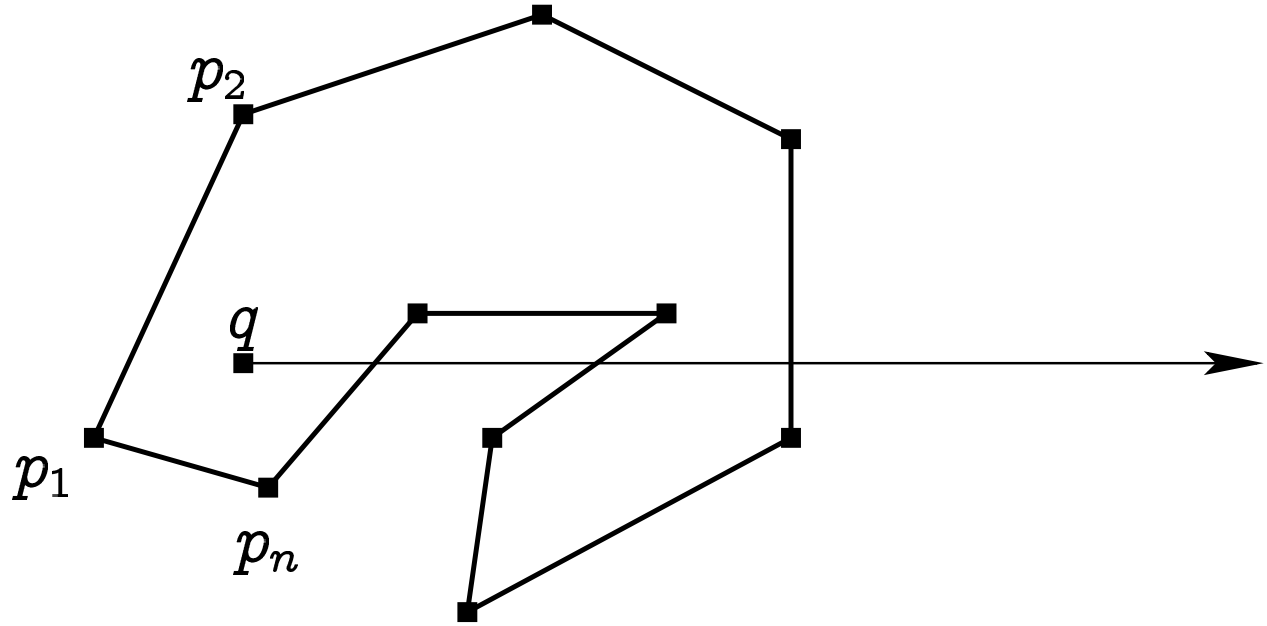
Keerukus:

- Eeltöö (sorteerimine) — $O(n \log n)$.
- Kui protseduuri lähimad_punktid tööaeg sisendil suurusega n on ülimalt $T(n)$, siis $T(n) = 2T(n/2) + O(n)$.
  - See $O(n)$ tuleb sellest, et meil on tsükleid üle kõigi punktide, pole aga kahekordseid tsükleid.
    - $*$ **for** $j := 1$ **to** $\min(6, l' - i)$ **do** ei loe.

  See rekurrents on lahendatav esimeses loengus tõestatud teoreemiga. Saame $T(n) = O(n \log n)$.

Kokku seega $O(n \log n)$.

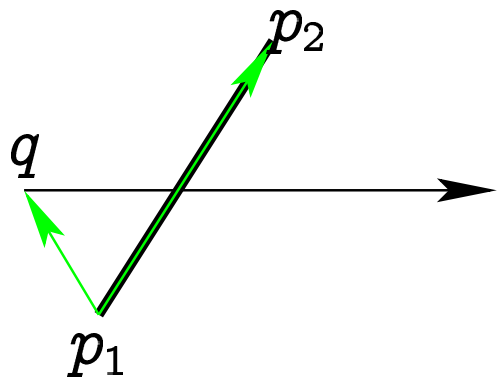
Olgu antud hulknurk tippudega $p_1, \ldots, p_n$ ning punkt q. Kas q asub hulknurga $p_1 p_2 \cdots p_n$ sees?

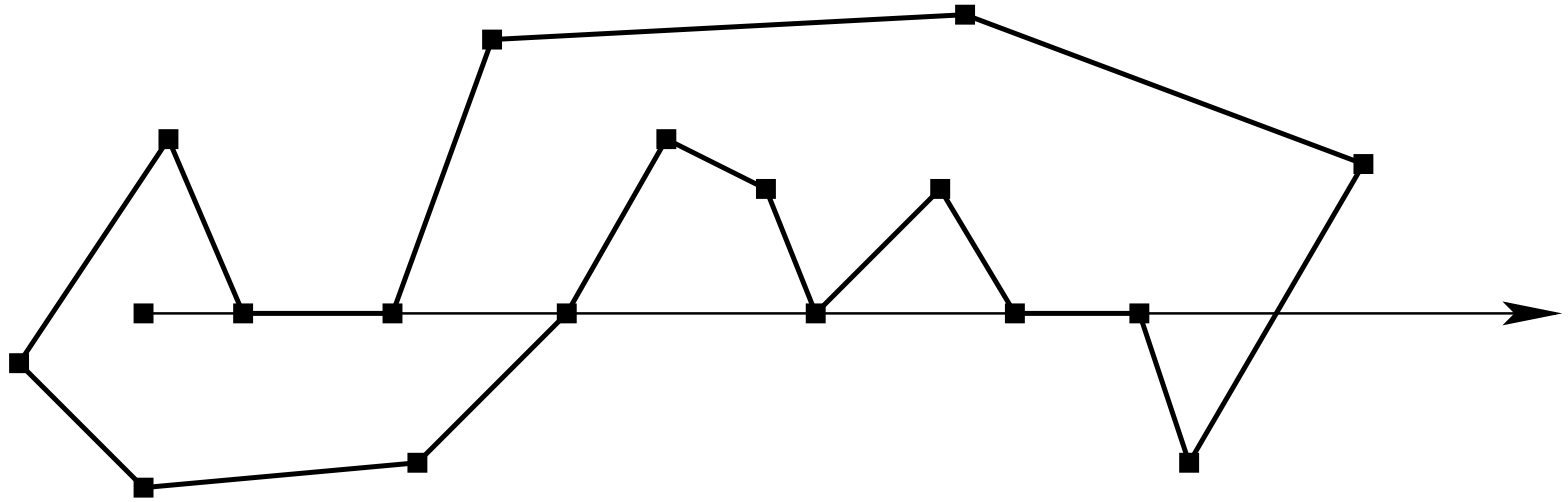


Vaatame mingit kiirt, mis lähtub punktist q. Kui q asub hulknurga sees, siis lõikab see kiir seda hulknurka paaritu arv kordi, muidu paarisarv kordi.

Millal lõikab lõik $\overline{p_1p_2}$ (loeme, et $p_1.y \leqslant p_2.y$) horisontaalset kiirt, mis läheb punktist q paremale?

- $p_1.y \leqslant q.y \leqslant p_2.y$;
- $\overrightarrow{p_1q}$ jääb $\overrightarrow{p_1p_2}$-st vasakule.





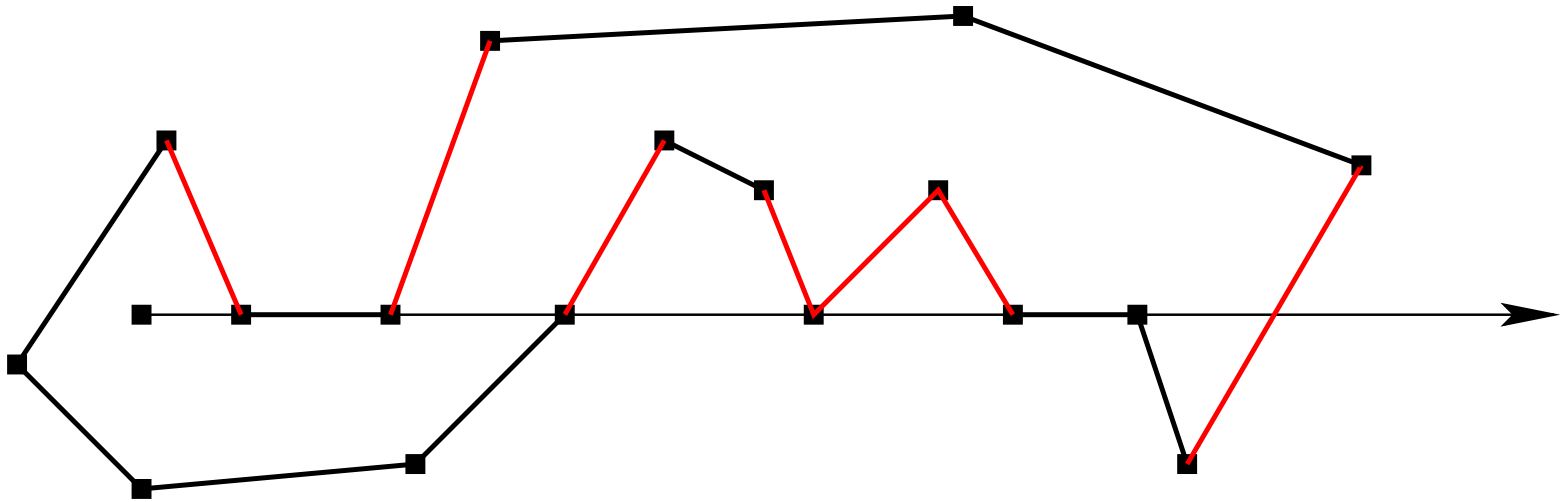
Mida teha siis, kui külje $\overline{p_1 p_2}$ (kus $p_1.y \leqslant p_2.y$) mõni tipp on q-st algaval kiirel?

Loeme nii:

- Kui $p_2.y = q.y$, siis ei lõiku.
- Kui $p_1.y = q.y < p_2.y$, siis lõikub.

S.t. kiir on „kaduvväikesel määral ülevalpool“ oma tegel i-kust asukohast.

Näide:

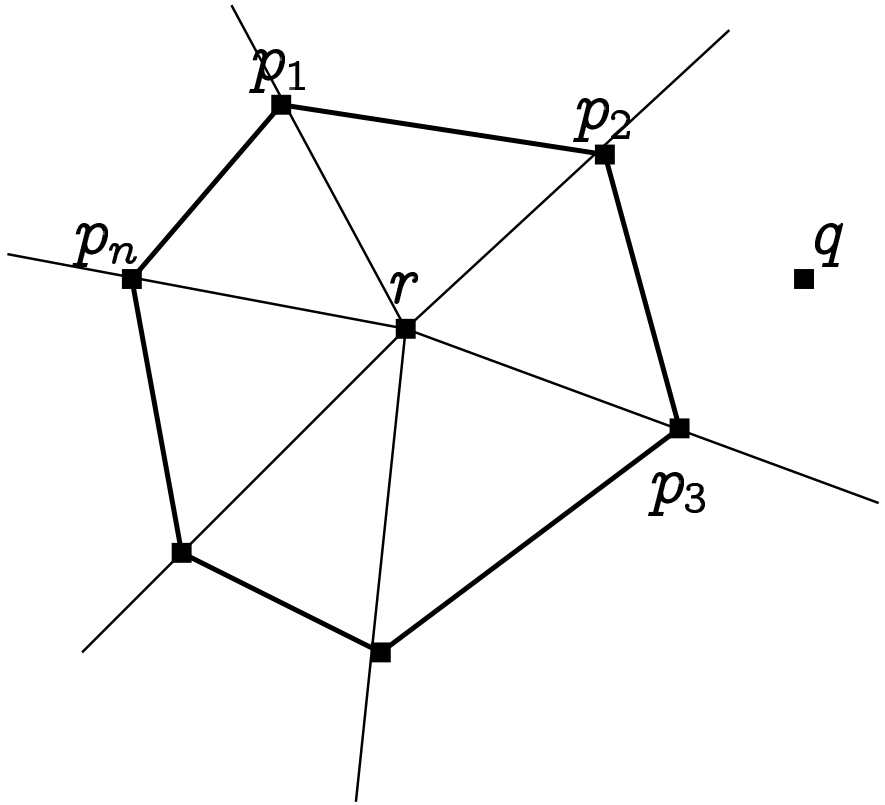


Seitse lõikumist $\Rightarrow$ sees.

Kõdunud juhu muutsime mittekõdunuks väikese häirituse lisamisega.

Keerukus — $O(n)$.

Kumera hulknurga puhul on punkti kuulumist hulknurka võimalik välja selgitada kiiremini, ajaga $O(\log n)$. Loeme, et $p_0, \ldots, p_n$ on antud kellaosuti liikumise suunas.



1. Olgu r mingi punkt hulknurga sisemuses. Sobib suvalise kolme tipu keskmine.
2. Leia, millisesse sektorisse $p_i r p_{i+1}$ kuulub punkt q. See on tehtav kahendotsimisega.
3. Leia, kas $\overline{rq}$ ja $\overline{p_i p_{i+1}}$ lõikuvad.

Olgu antud hulknurk $p_1 p_2 \cdots p_n$, tipud olgu kellaosuti liikumise suunas.

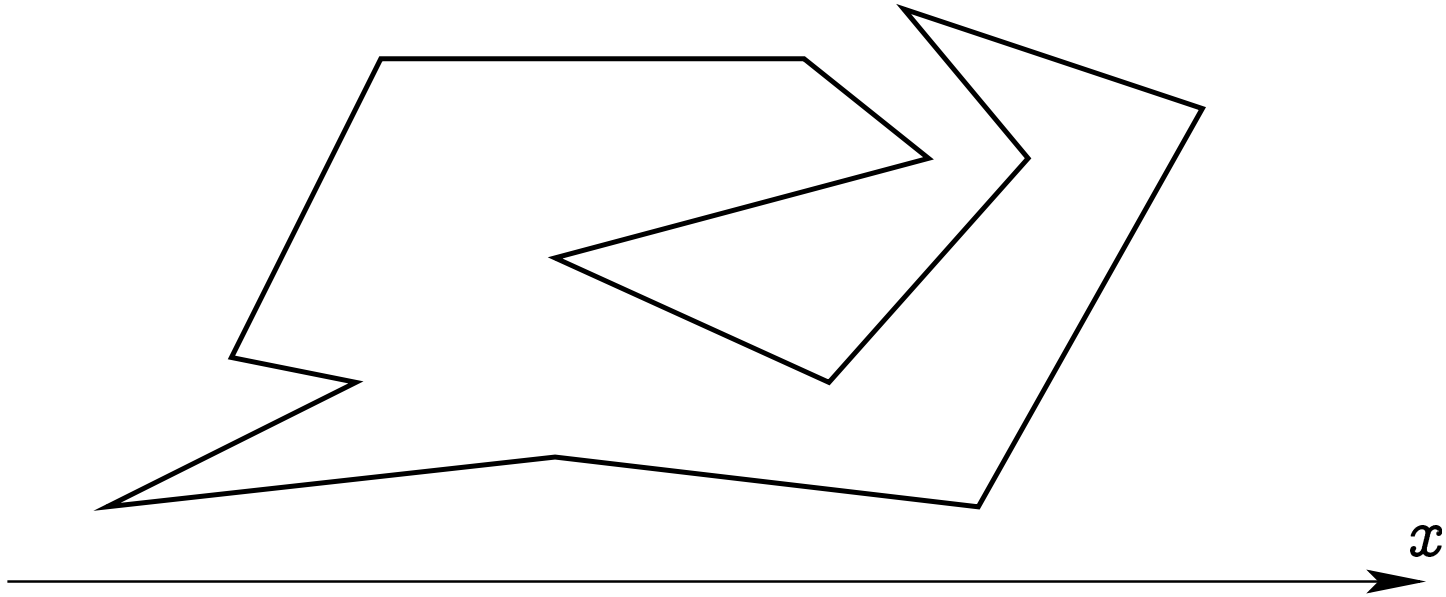
Kuidas leida selle hulknurga pindala?

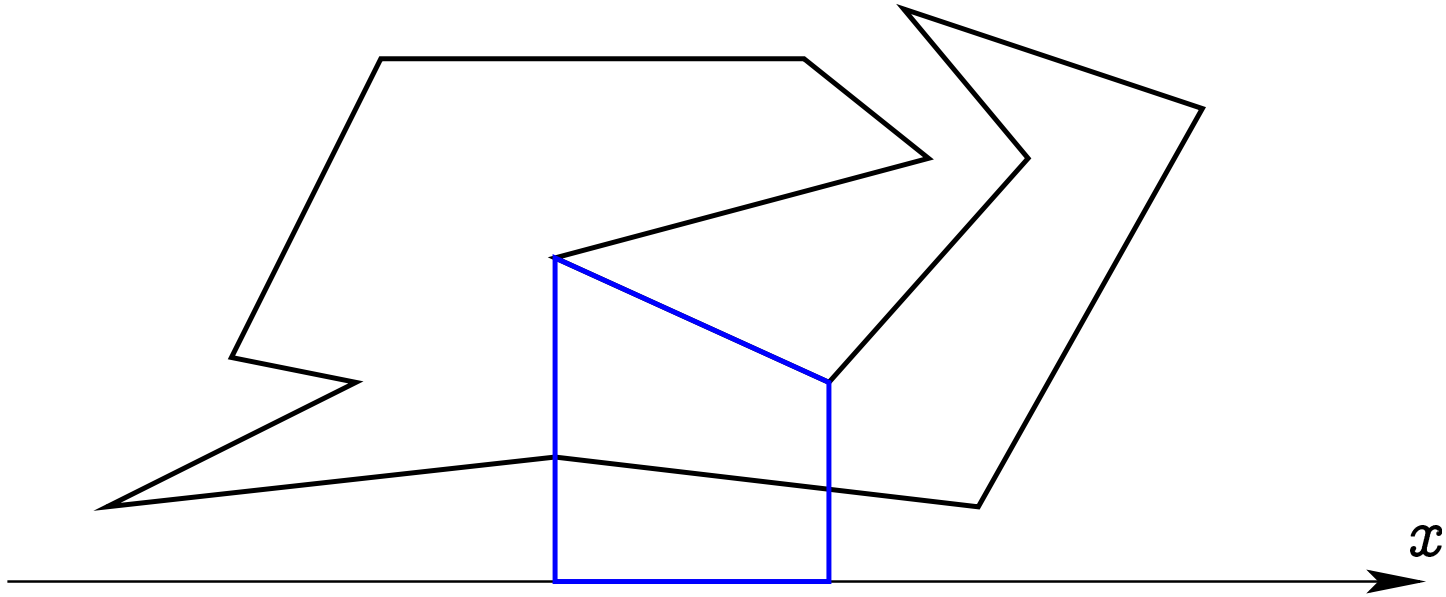
Olgu $p_i = (x_i, y_i)$. Loeme, et $y_i \geqslant 0$. Leiame trapetsite

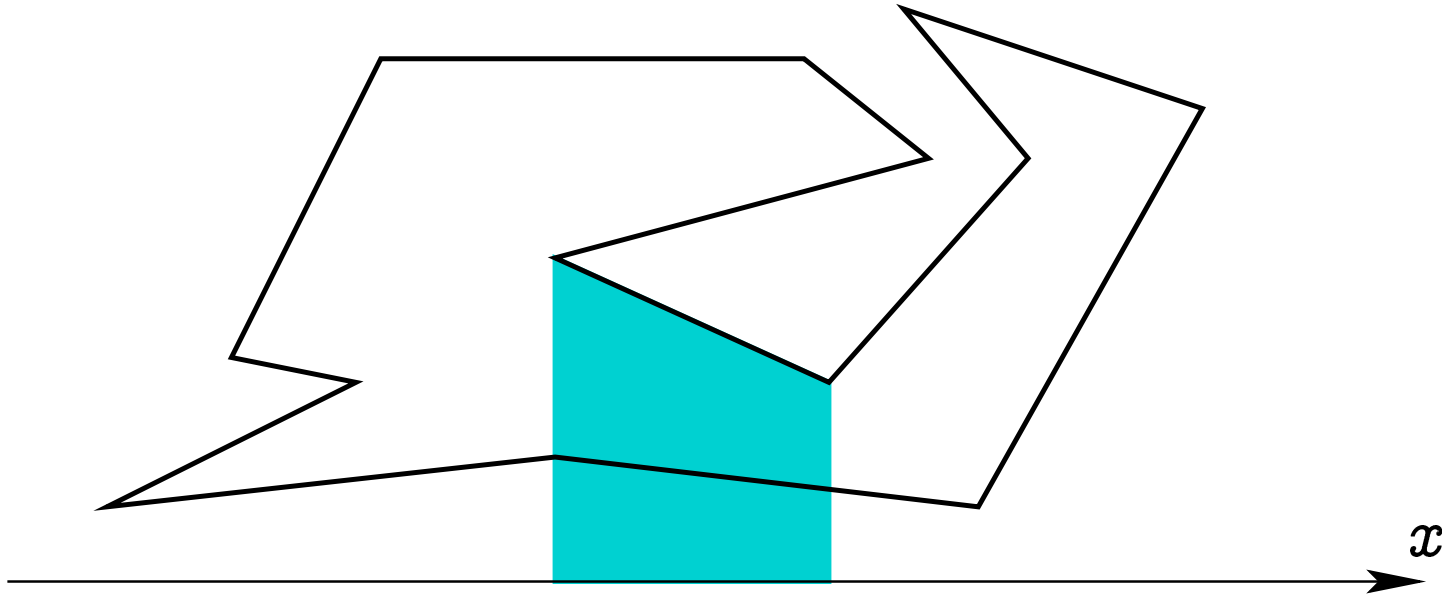
$$(x_i, 0)—(x_i, y_i)—(x_{i+1}, y_{i+1})—(x_{i+1}, 0)—(x_i, 0)$$

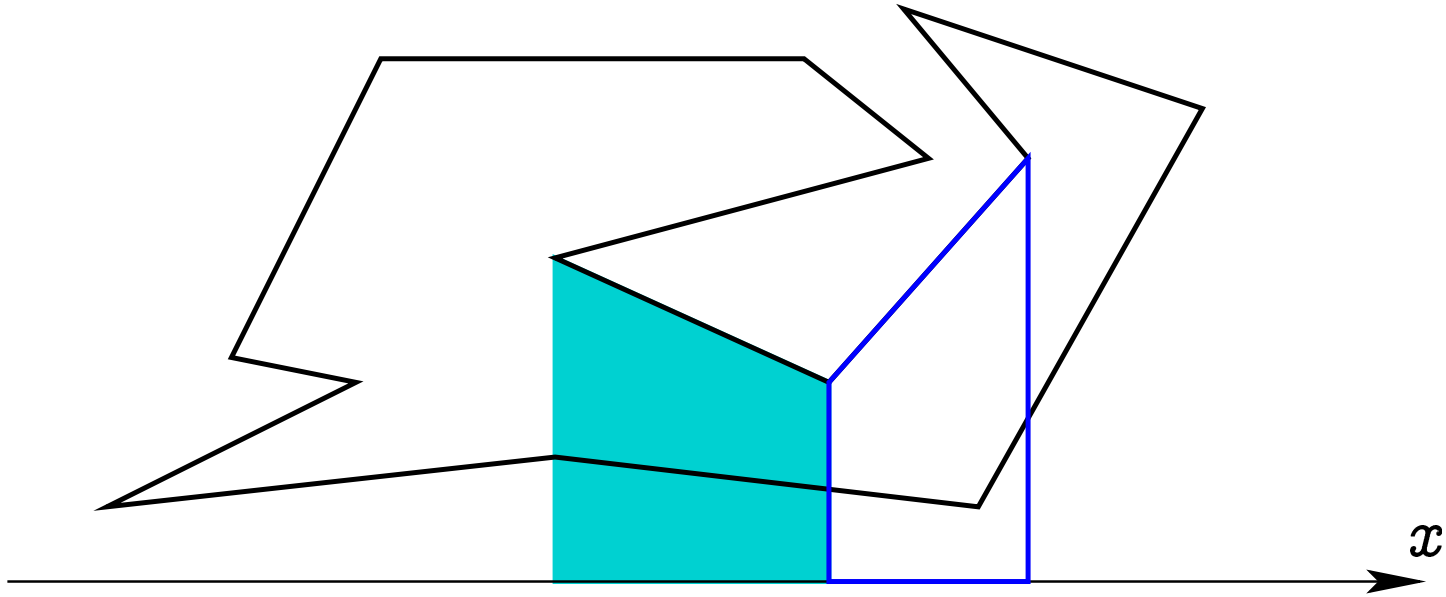
pindalad. Siin $1 \leqslant i \leqslant n$ ja $i+1$ on leitud mod n. Pindalad on *märgiga* — kui $x_{i+1} > x_i$, siis on pindala positiivne, muidu negatiivne.

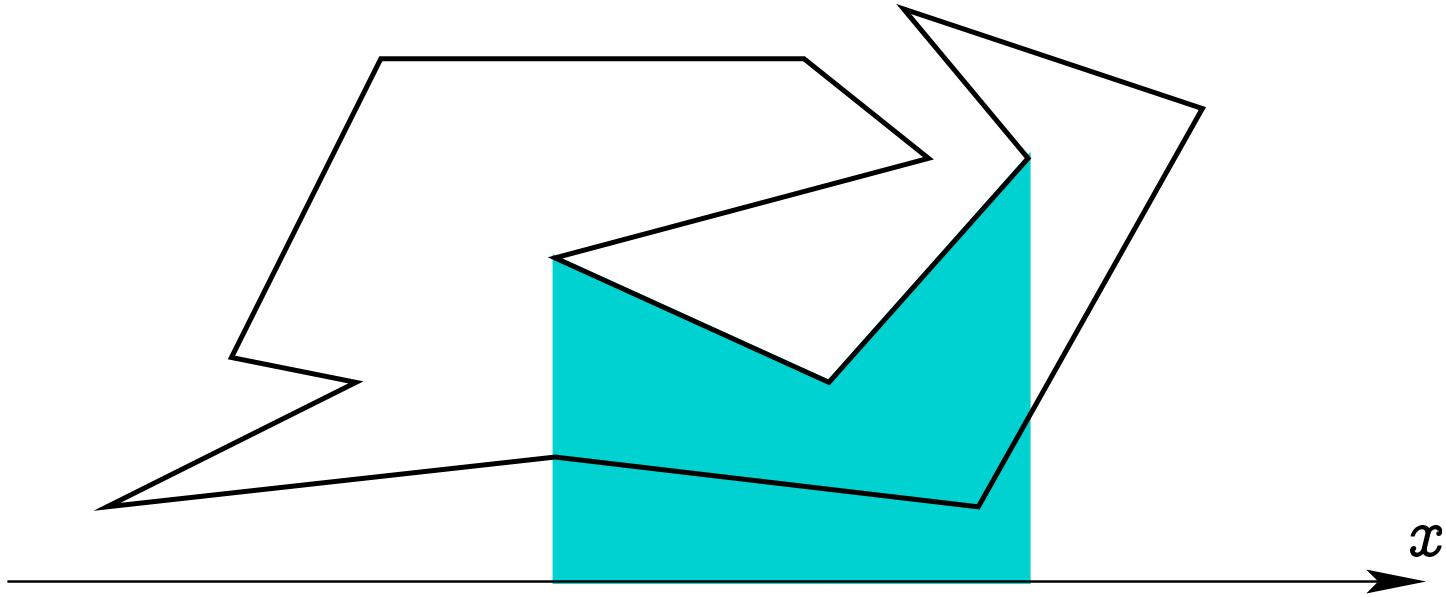
Trapetsite pindalate summa on hulknurga pindala.

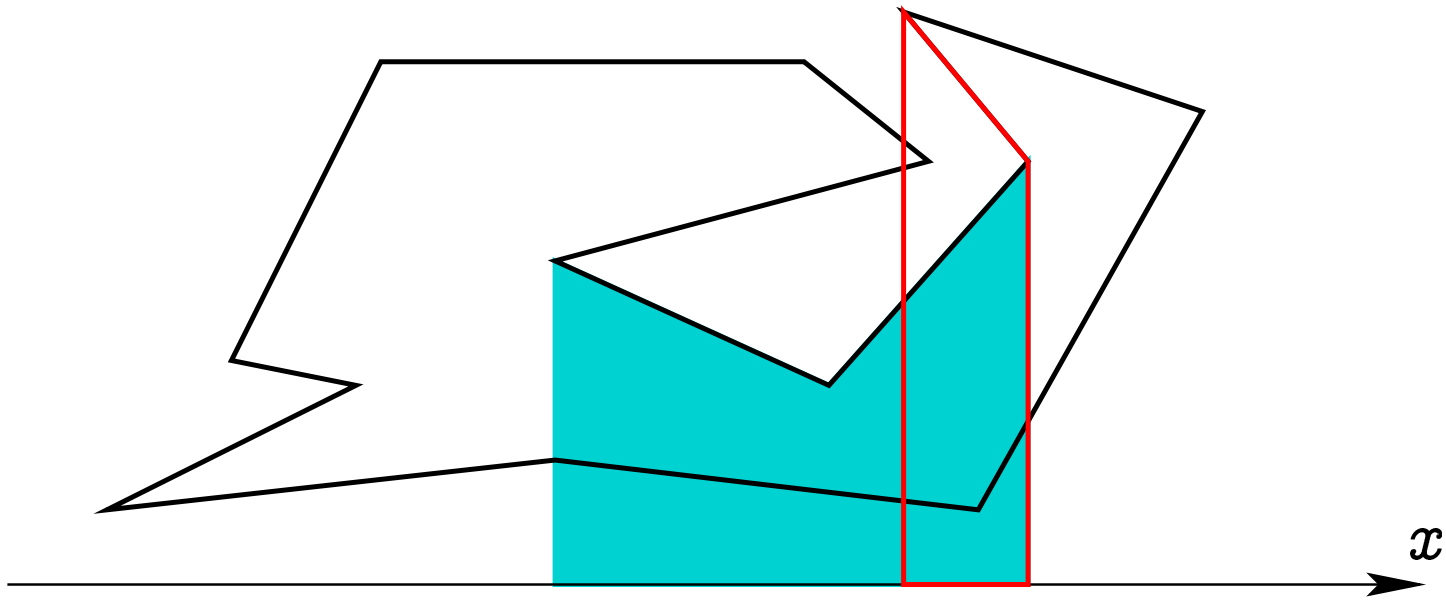

x

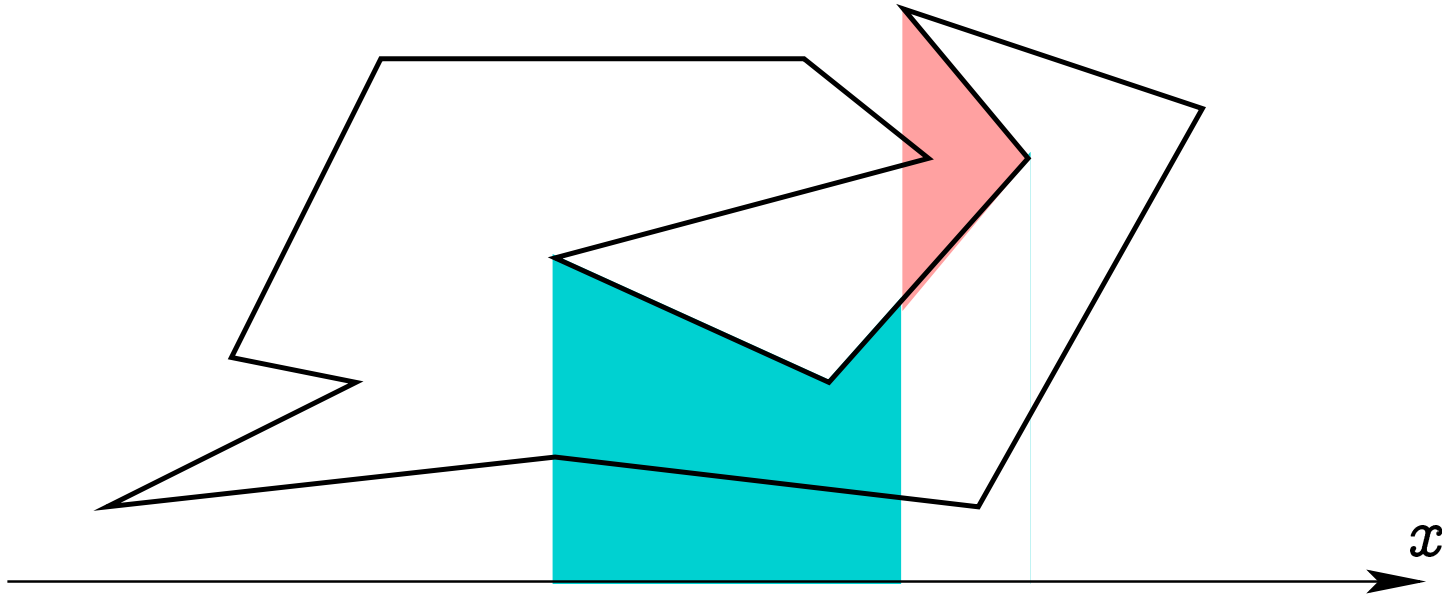

x

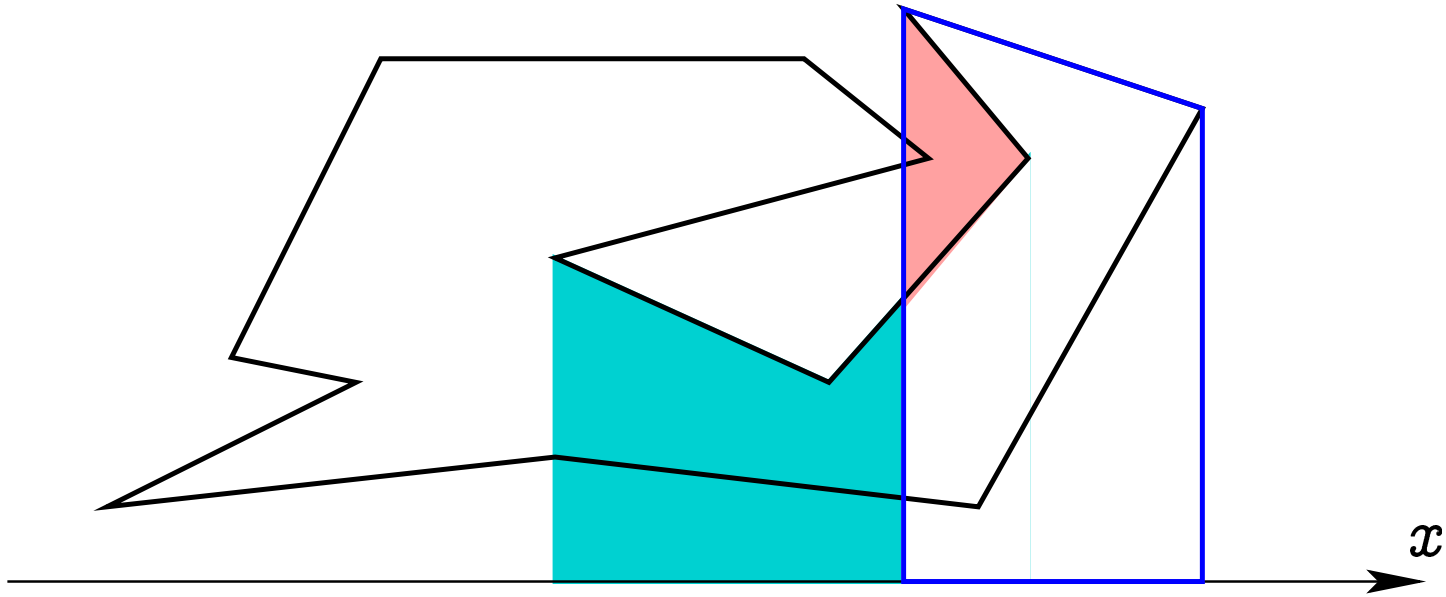

x

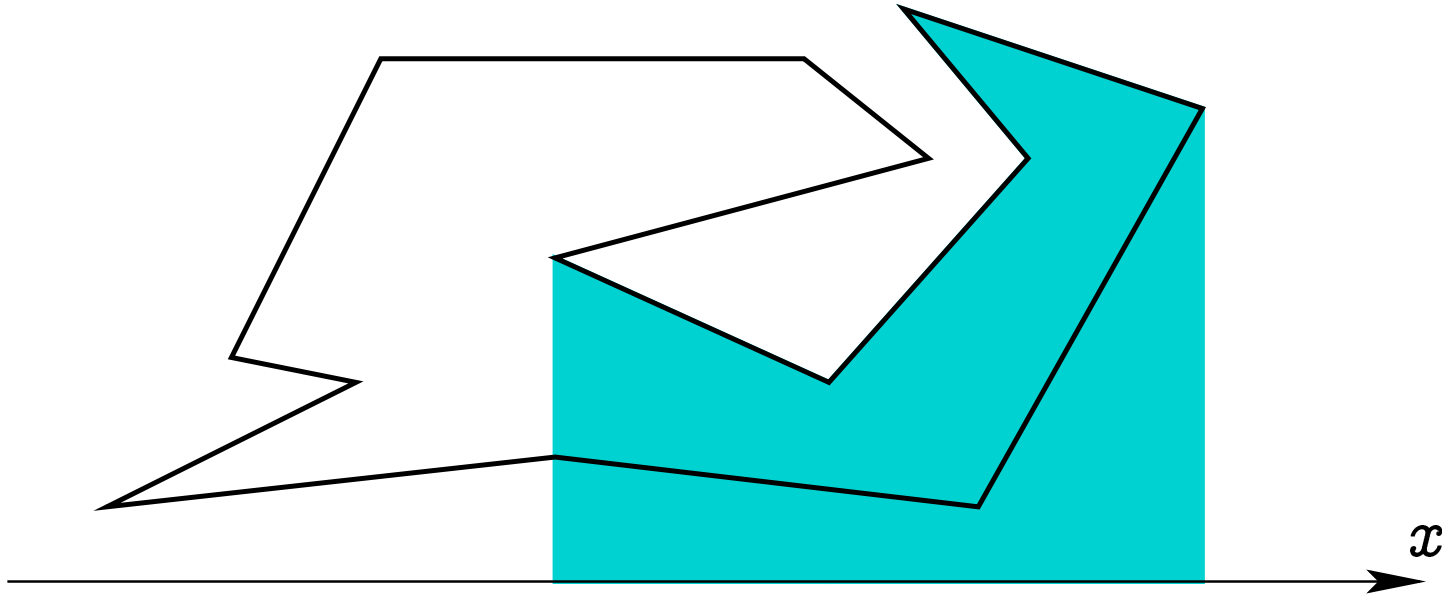

x

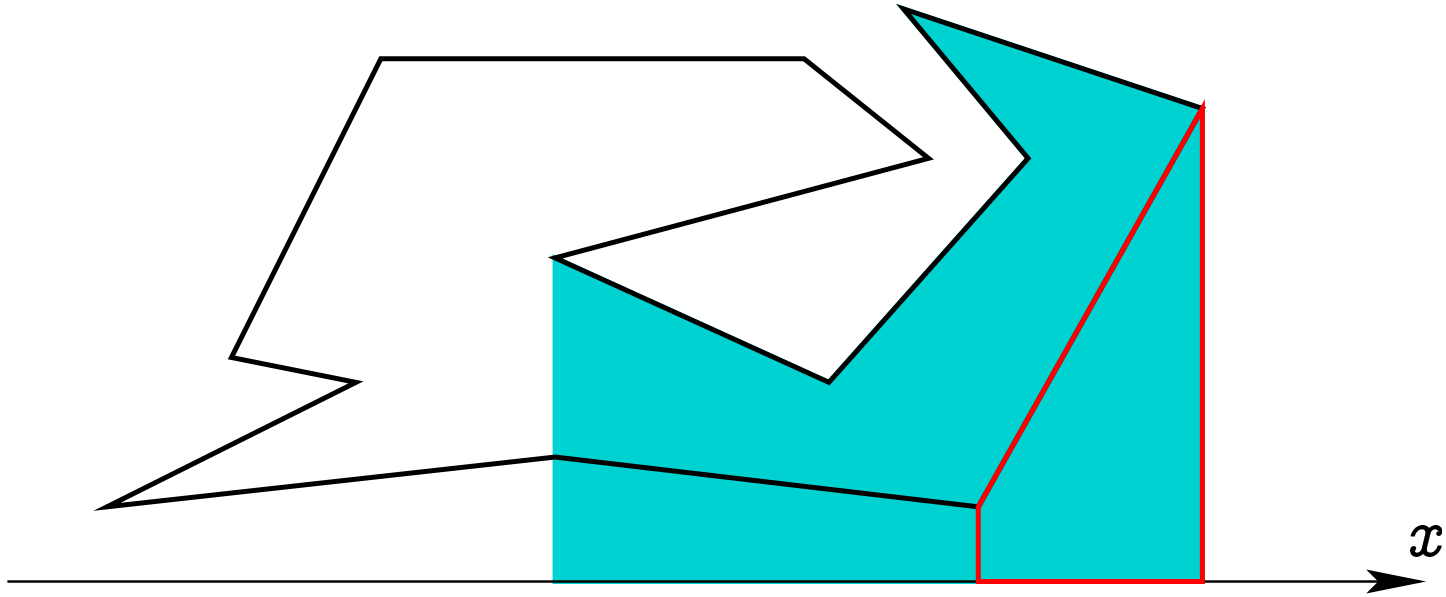

x

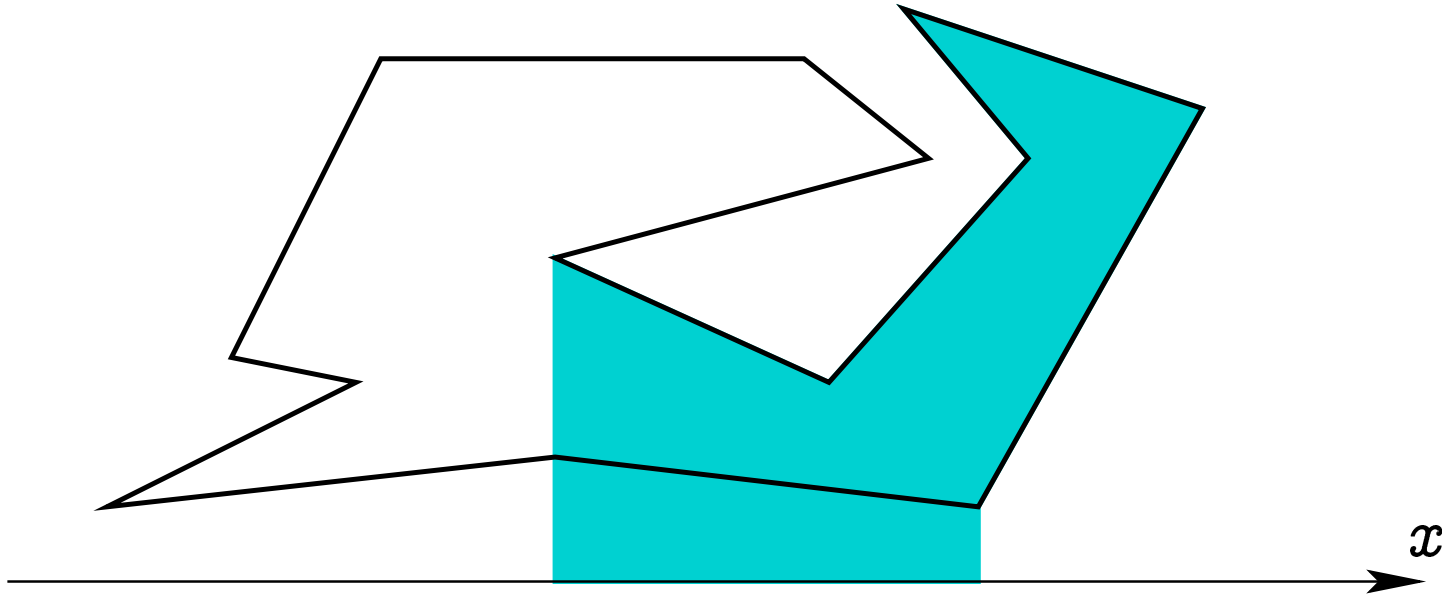

x

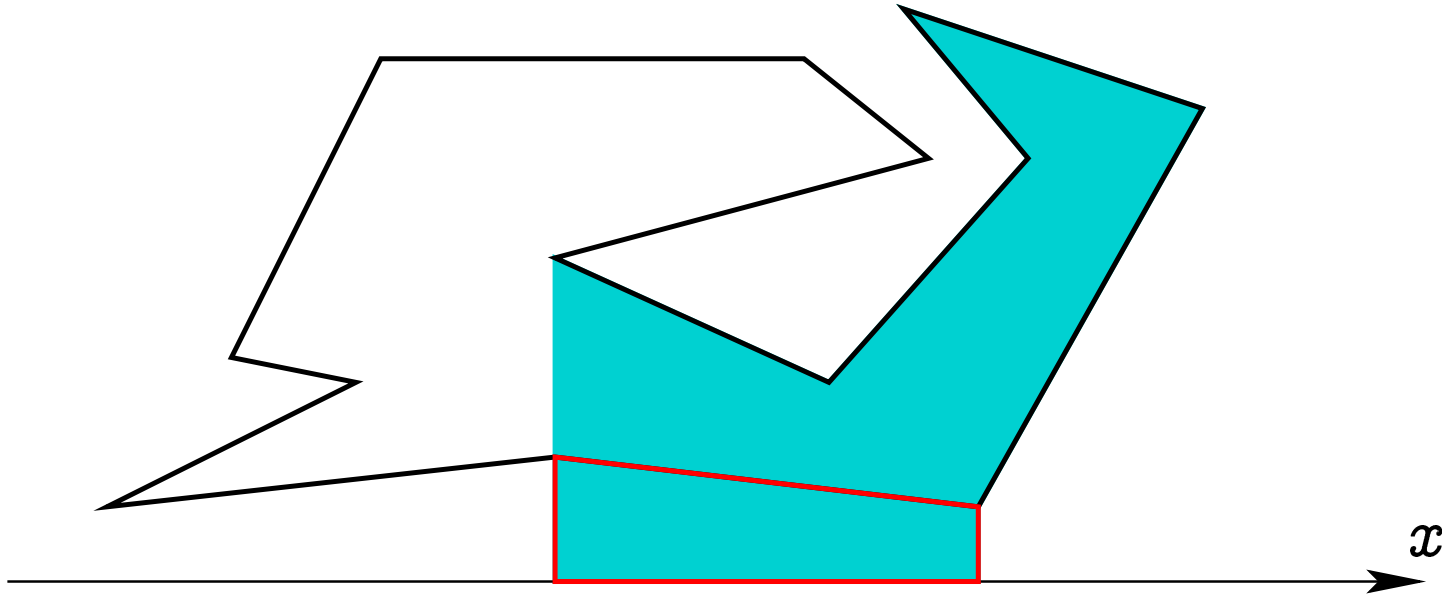

x

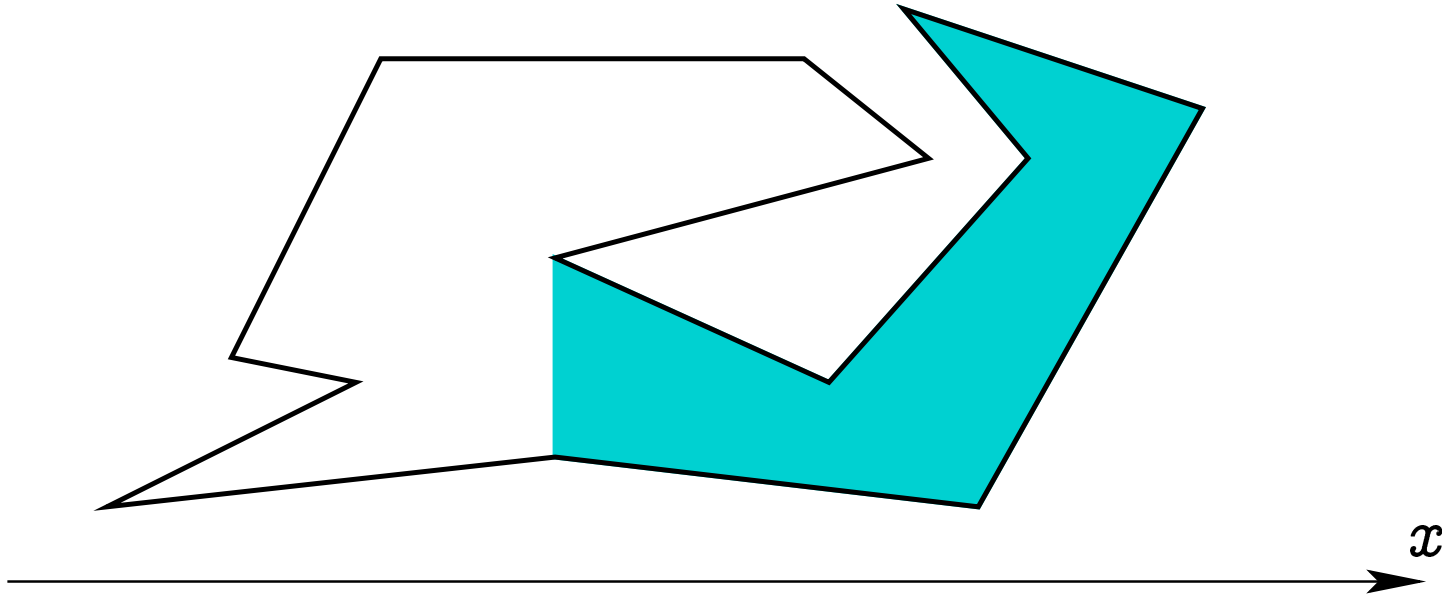

x

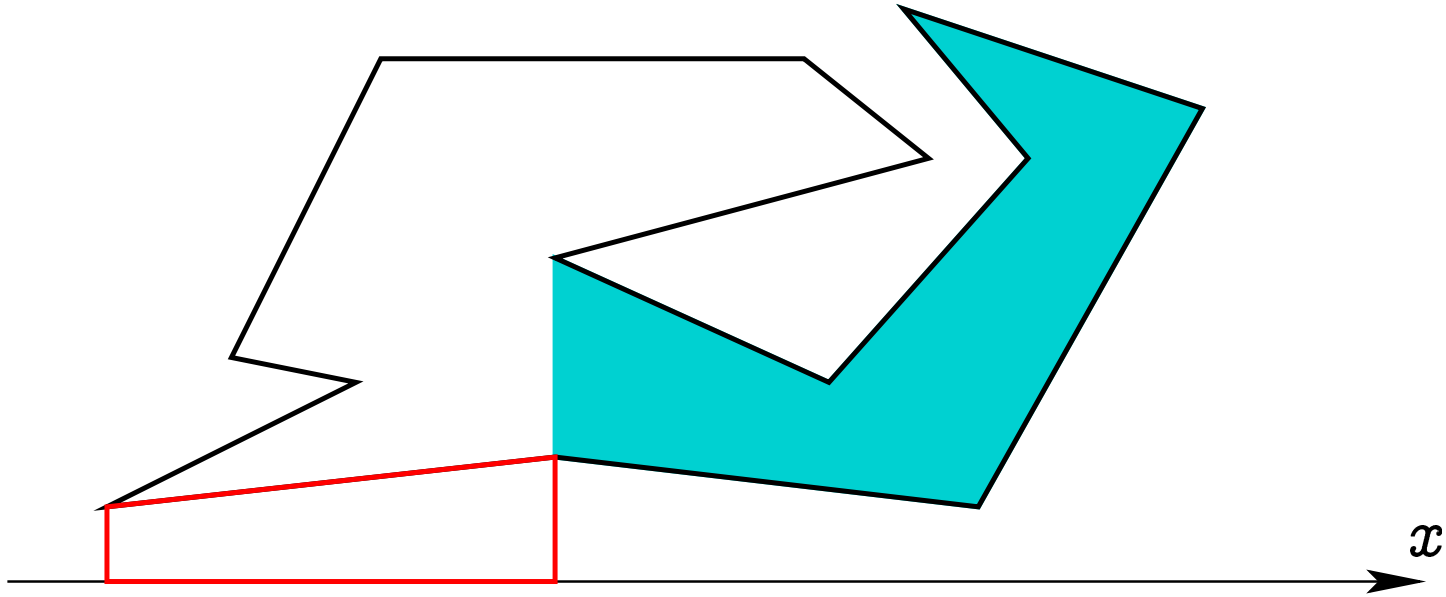

x

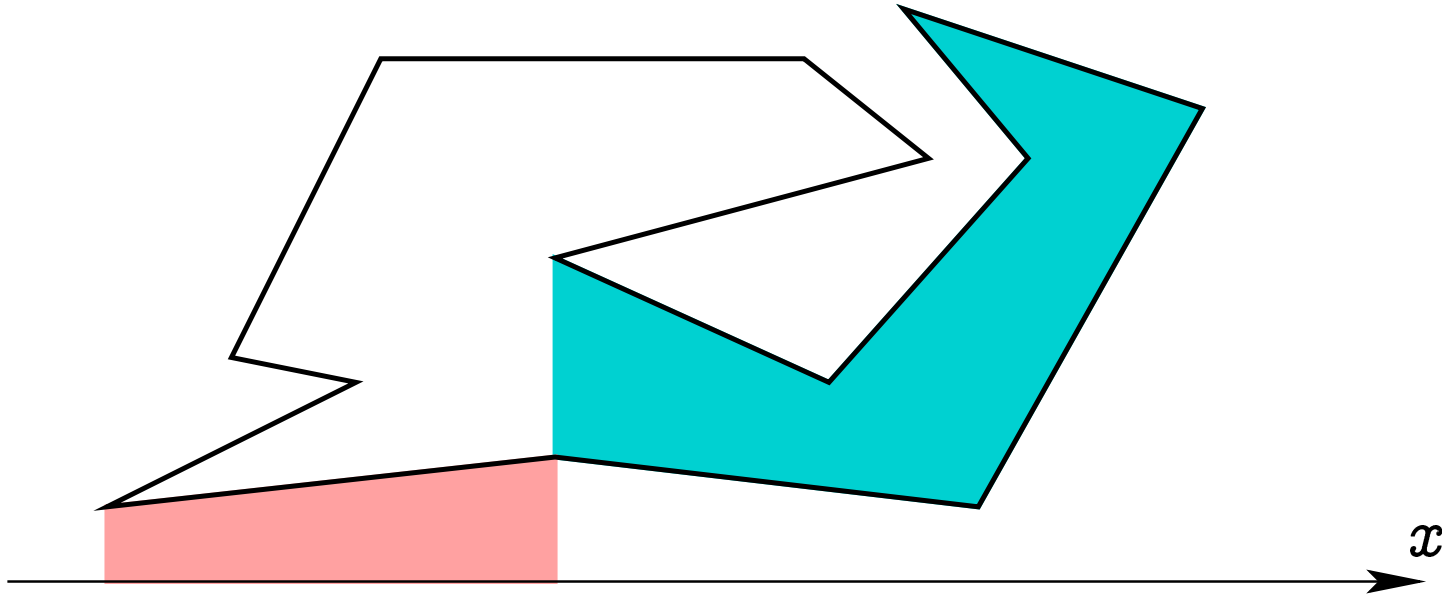

x

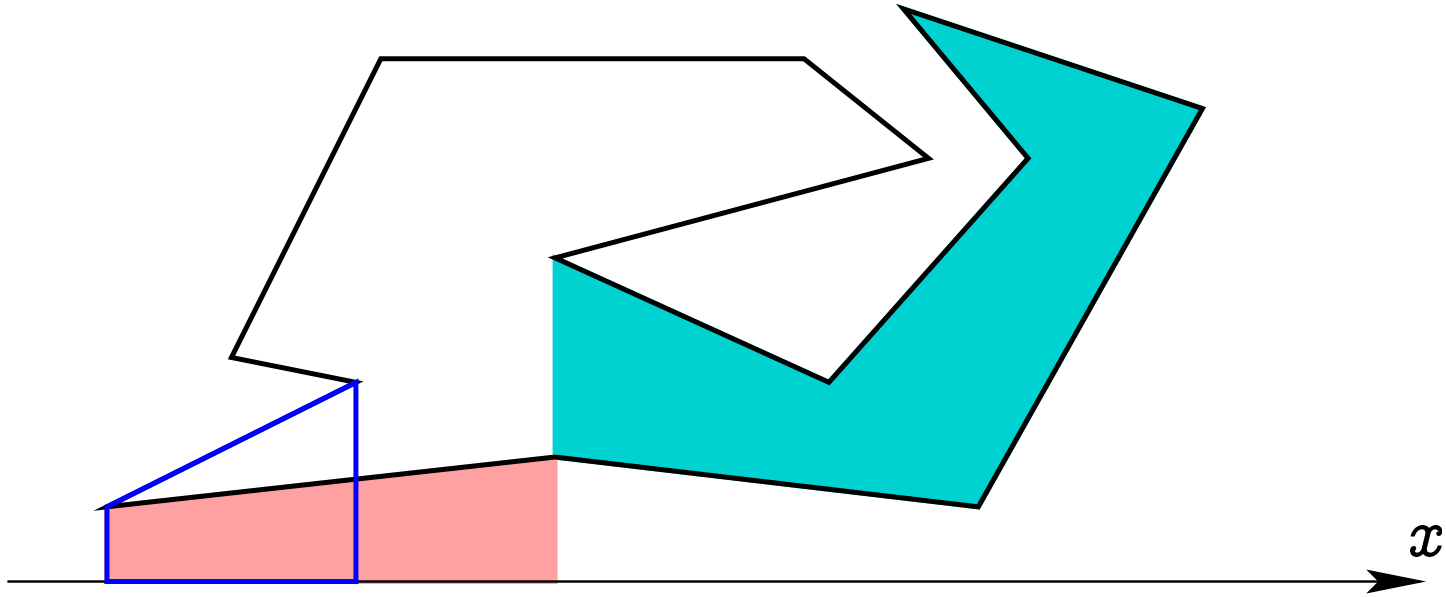

x

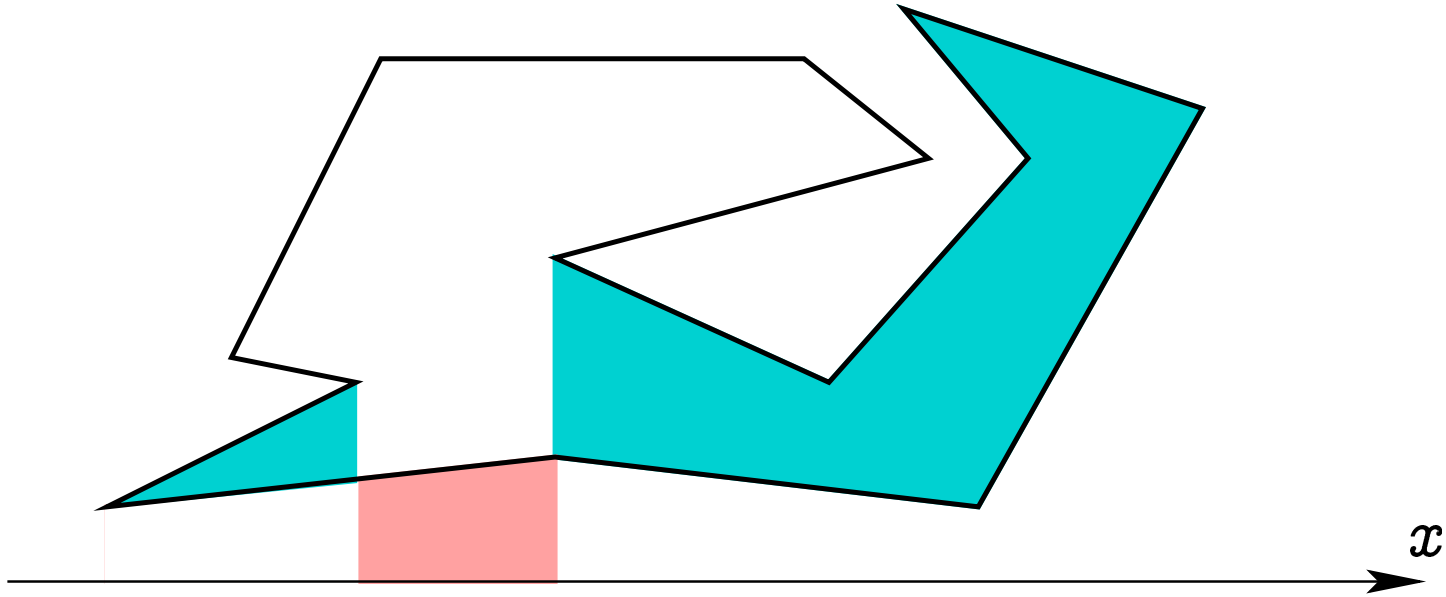

x

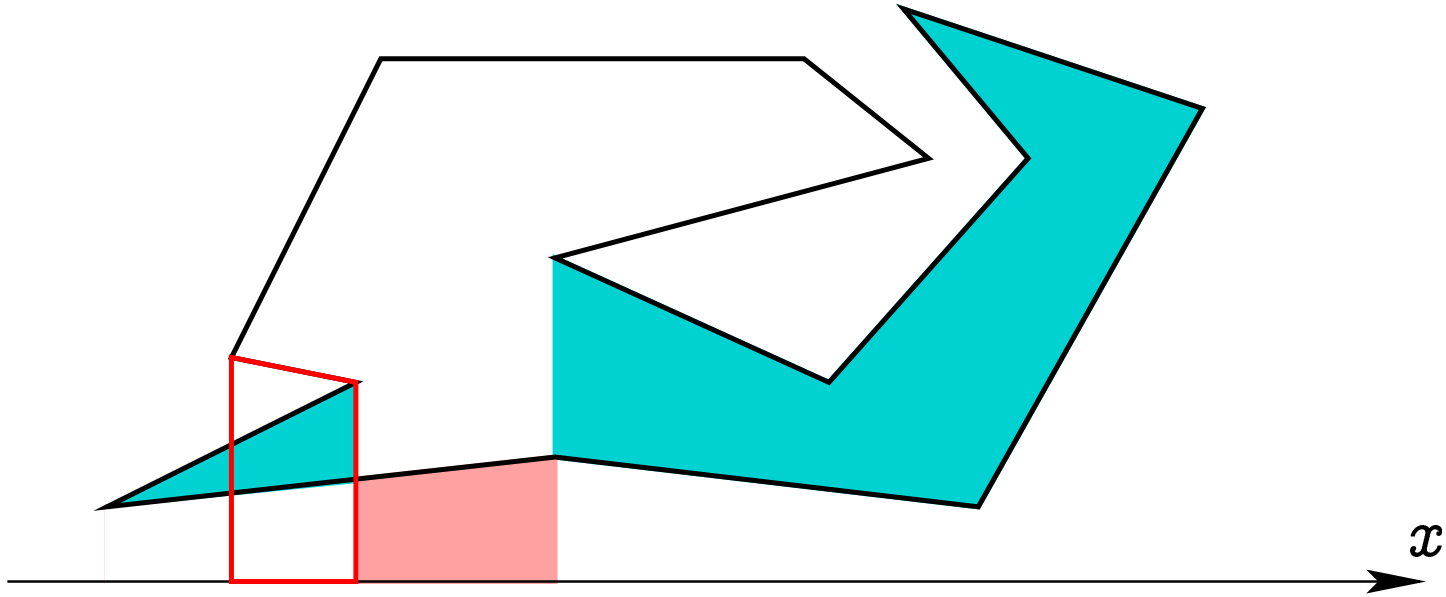

x

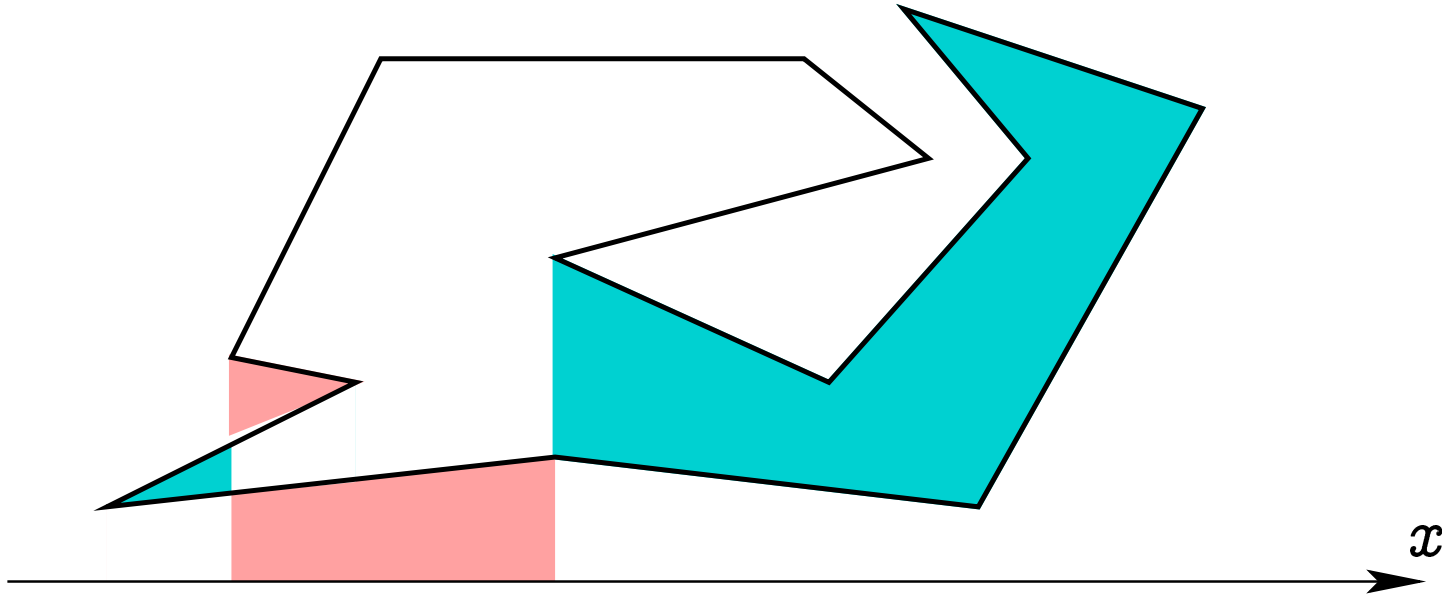

x

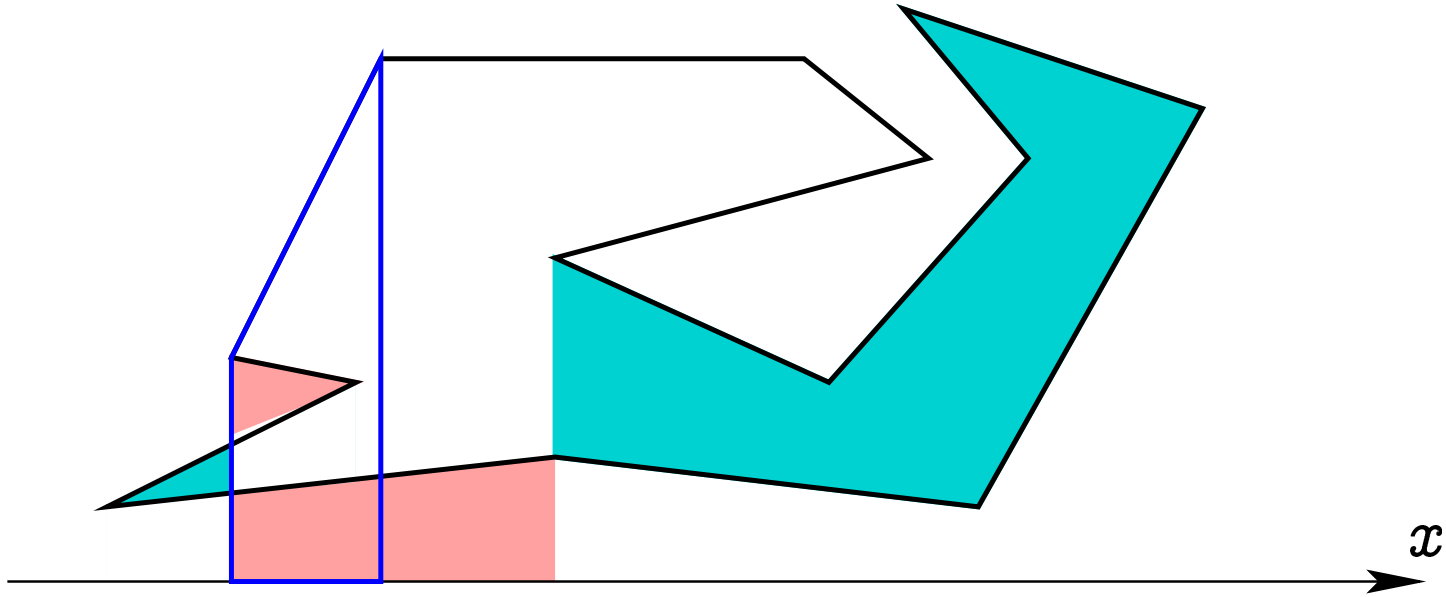

x

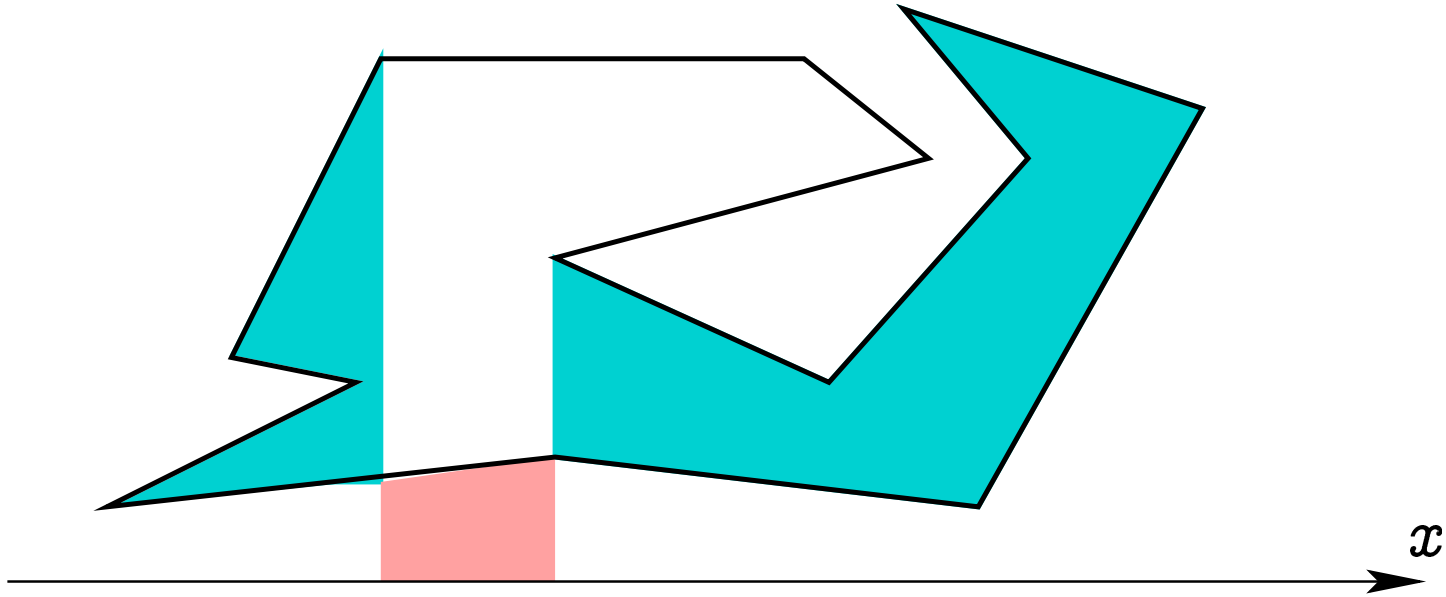

x

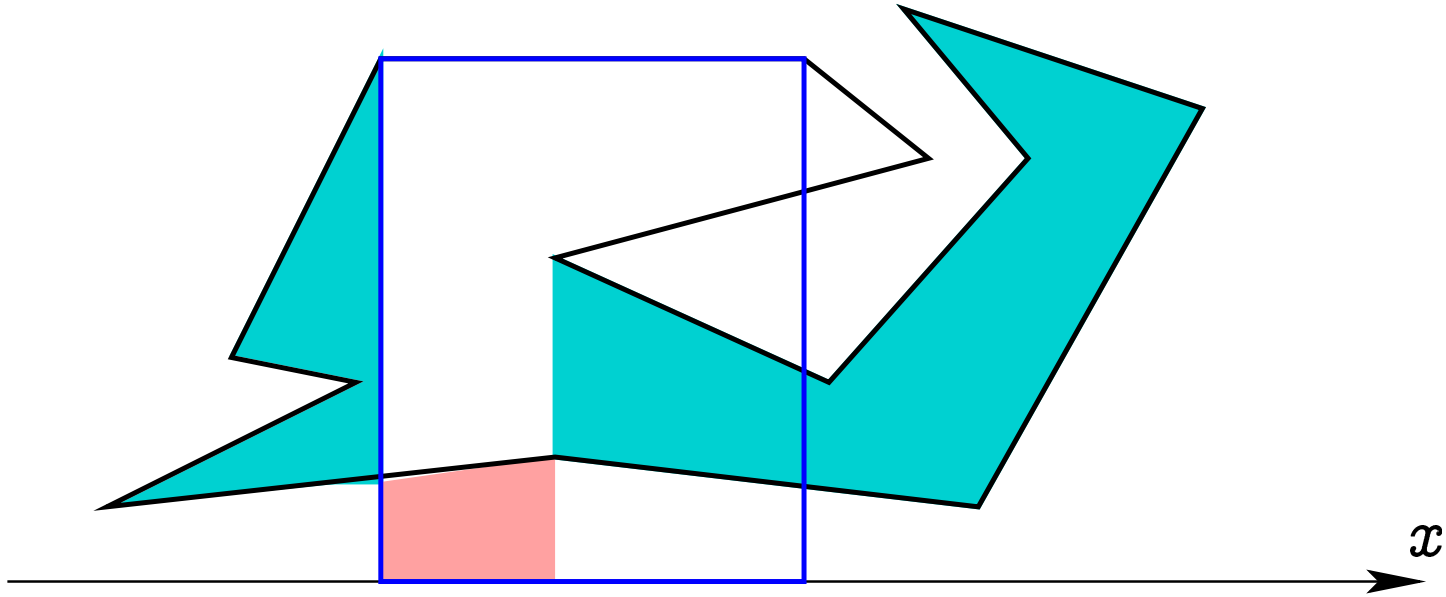

x

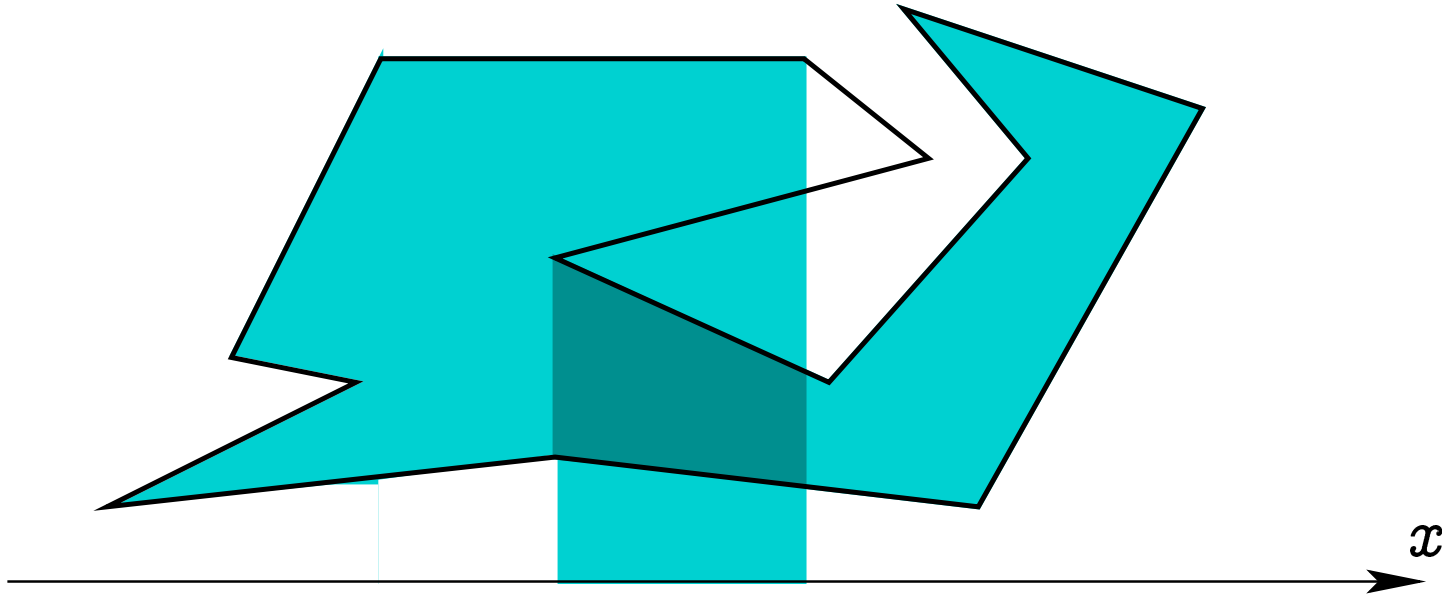

x

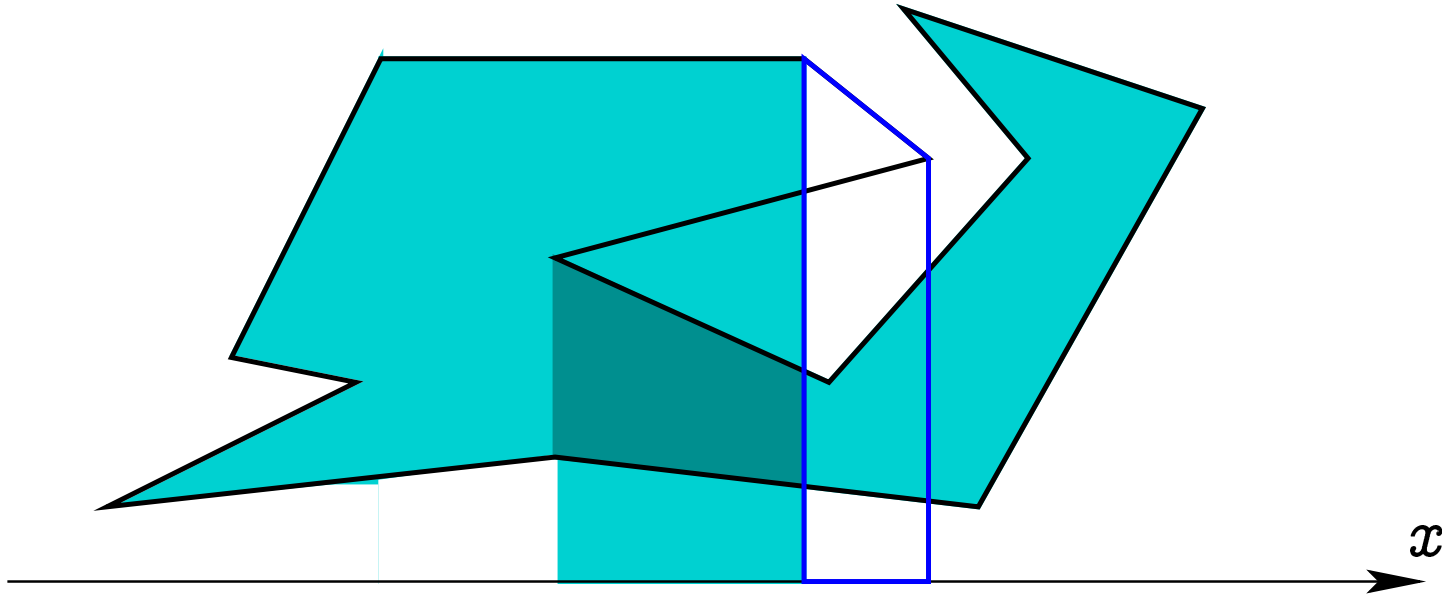

x

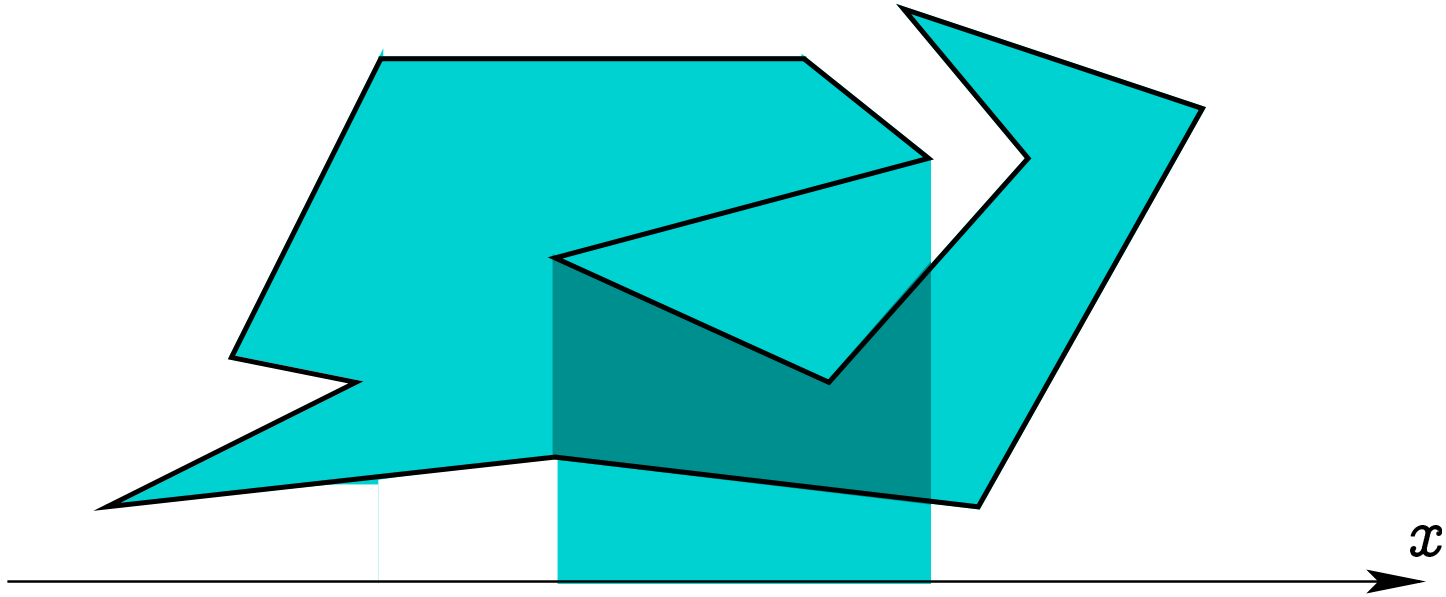

x

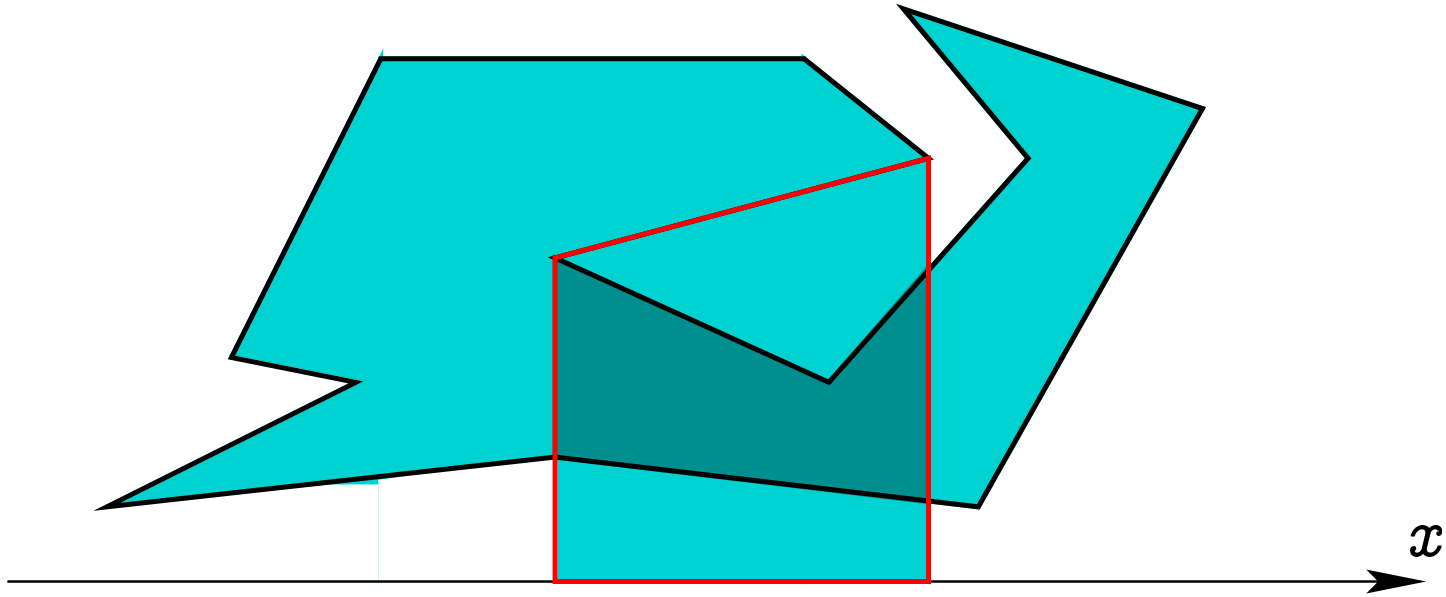

x

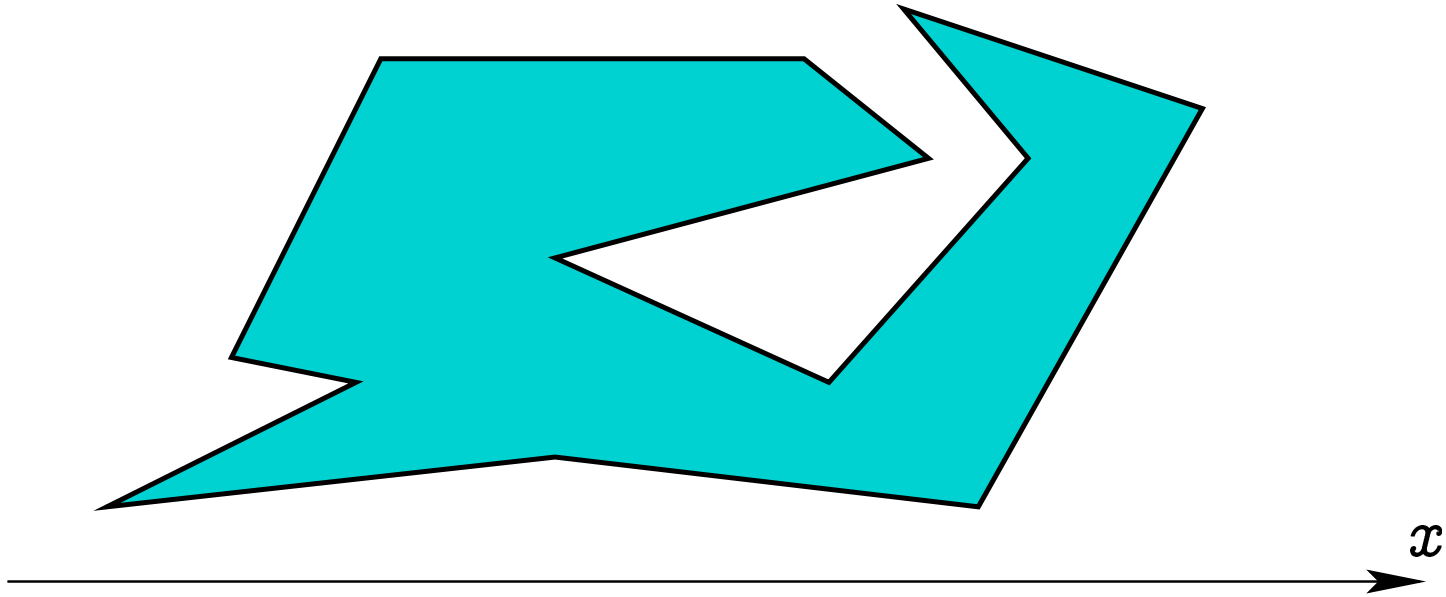

x


x


x


x

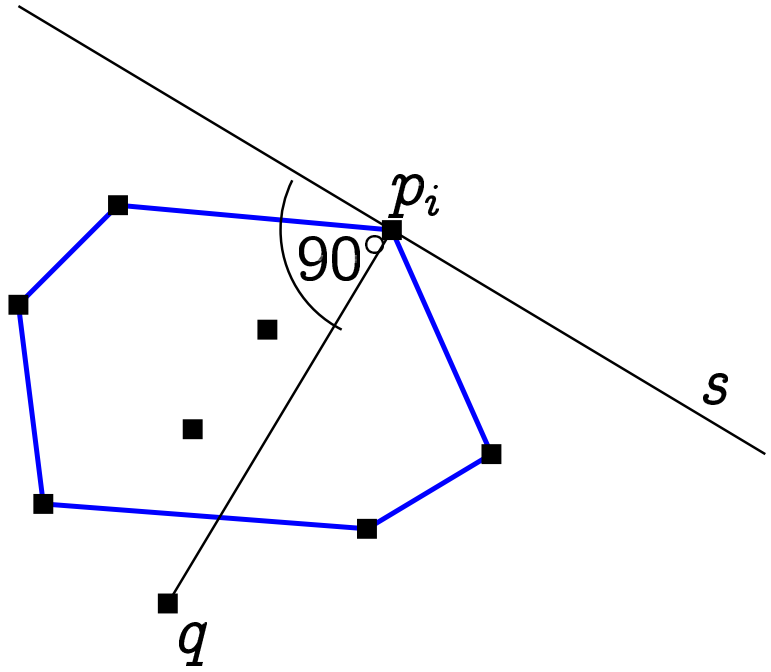

x


x

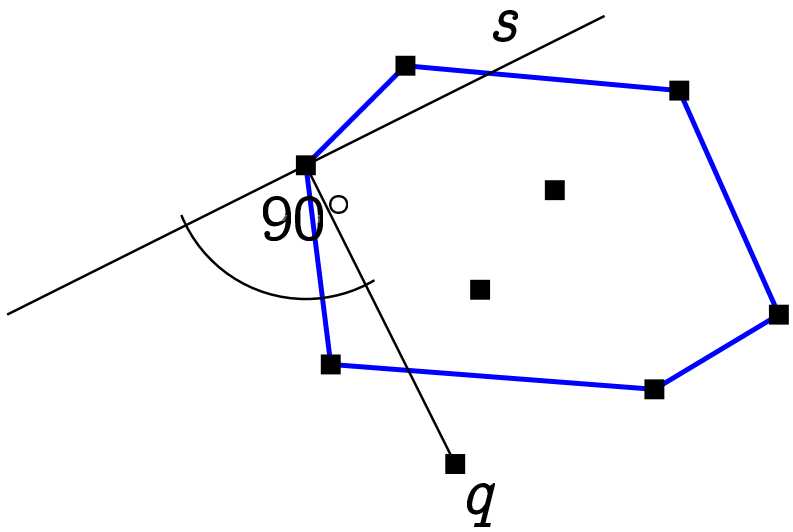
Antud punktid $p_1, \ldots, p_n$. Leida nende seast kaks teineteisest kõige kaugemat punkti.

On üsna ilmne, et need kaks punkti peavad asuma nende punktide kumeral kattel, aga see järeldub ka järgmisest lausest.

Lause. Olgu q mingi punkt ning olgu p_i punktist q kaugeim punkt punktide $p_1, \ldots, p_n$ seas. Olgu s sirge, mis on risti lõiguga $\overline{qp_i}$ ja läbib punkti p_i. Siis s-i ja punktide $p_1, \ldots, p_n$ kumera katte ainus ühine punkt on p_i.

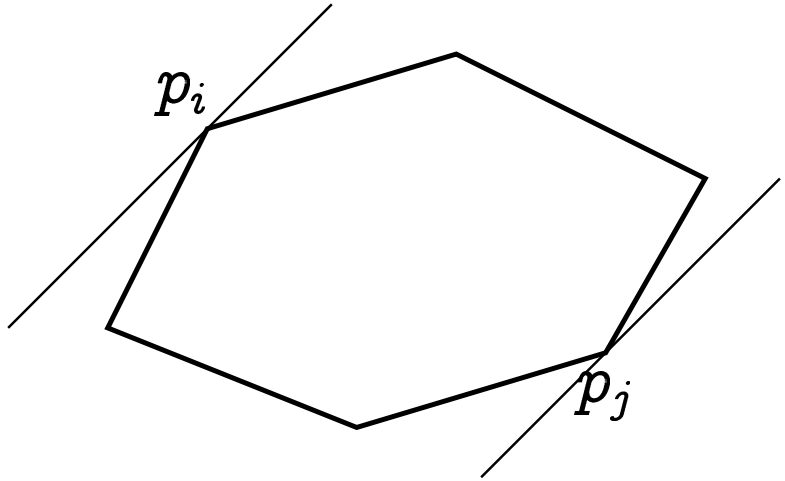


Vastasel juhul võiks mööda katte sisse jäävat s-i osa liikudes q-st kaugemate punktideni jõuda.

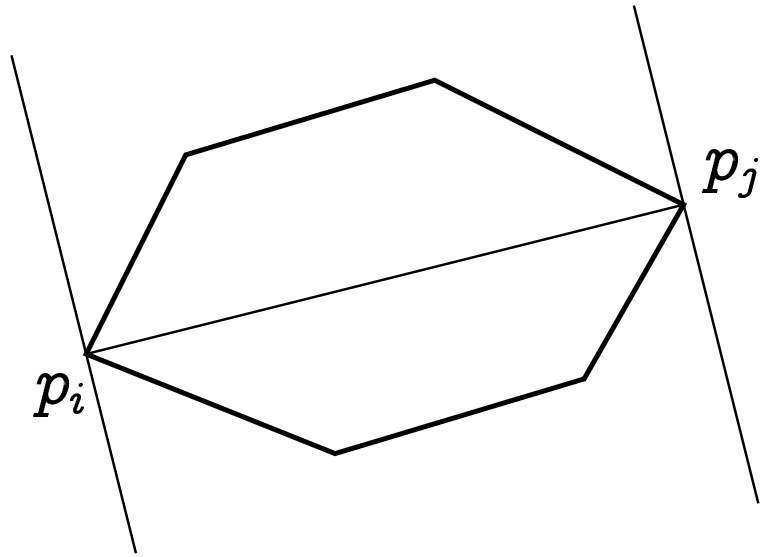


Olgu meil antud kumer hulknurk $p_1 p_2 \cdots p_n$, olgu punktid $p_1, \ldots, p_n$ kellaosuti liikumise vastassuunas. (Grahami seiremeetod annab kumera katte just sellises järjekorras) Loeme, et tal ei ole paralleelseid külgi.

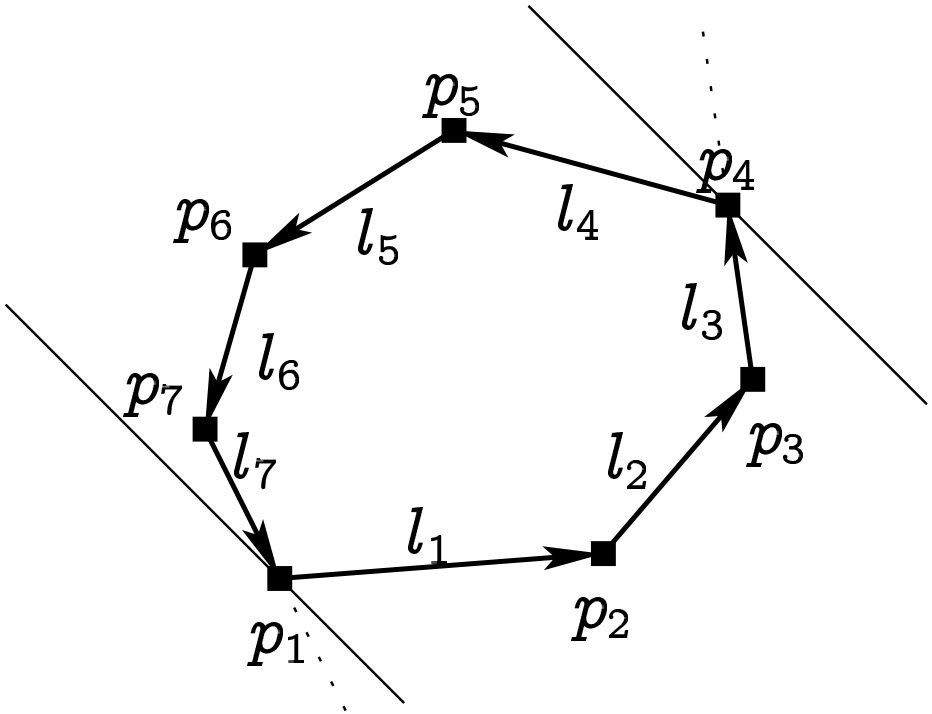
Punktid p_i ja p_j on *antipoodsed*, kui leiduvad kaks paralleelset sirget nii, et ühe ühisosaks selle hulknurgaga on p_i ja teise ühisosaks selle hulknurgaga on p_j.

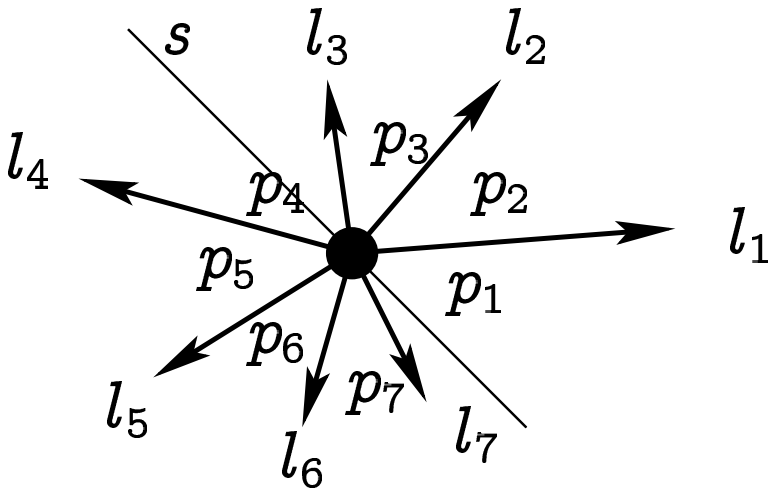


Lause. Kui p_i ja p_j on teineteisest kõige kaugemal asetsevad punktid, siis nad on antipoodsed.



Sobivad näiteks punktides p_i ja p_j võetud lõiguga $\overline{p_i p_j}$ ristuvad sirged.

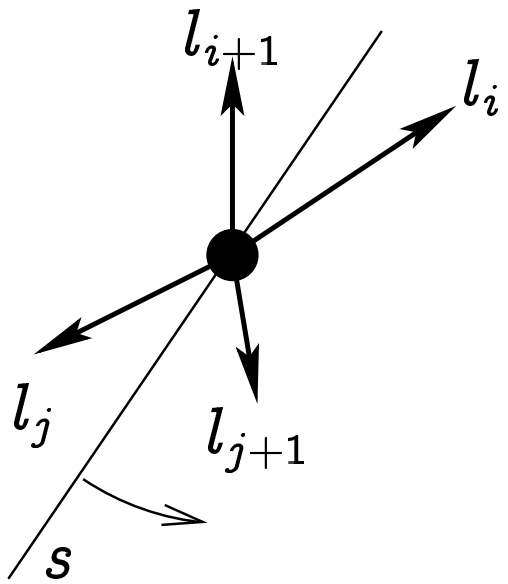




Antipoodsete paaride leidmine.

Keerutame sirget s.

Kui s on ühelt poolt l_i ja l_{i+1} vahel ja teiselt poolt l_j ja l_{j+1} vahel, siis



huvitab meid, kumba pidi tuleb keerata l_{i+1}-e, et viia ta samasuunaliseks l_{j+1}-ga.

Selle järgi otsustame, kas järgmine piirkond on l_{i+1} ja l_{i+2} vahel või l_{j+1} ja l_{j+2} vahel.