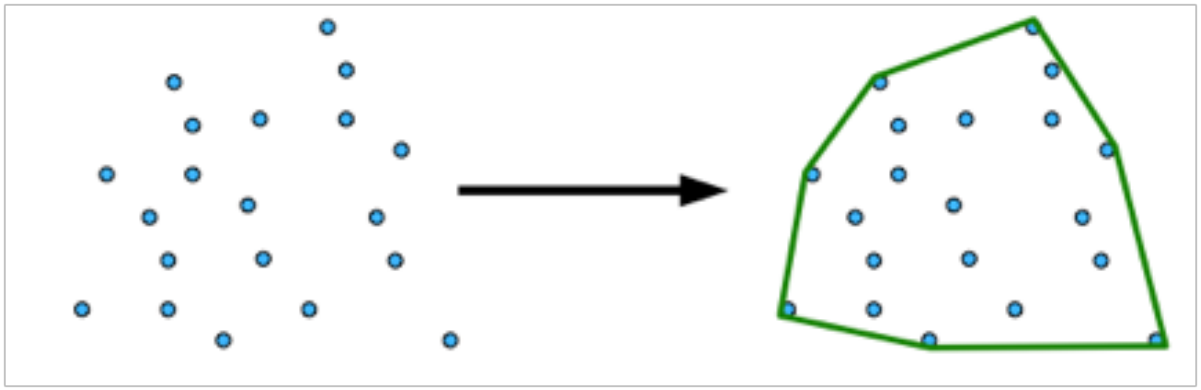
Ponthalmaz konvex burka

Szász Tünde

Konvex burok

Adott $Q \subseteq \mathbb{R}^d$ ($d = 1, 2, 3, \dots$), Q konvex burka: $CH(Q) =$

- Q-ban lévő pontok összes konvex kombinációinak halmaza
- a legkisebb konvex halmaz, ami tartalmazza Q-t
- az összes Q-t tartalmazó konvex halmaz metszete



Graham algoritmusa

- Ronald Grahamről nevezték el, aki az eredeti algoritmust 1972-ben publikálta.
- Graham-féle pásztázásként is ismert.
- Alapötlet: Minden $Q - beli$ pontot beírunk egyszer az S verembe. Azokat a pontokat, amelyek nem csúcsai a konvex buroknak, előbb-utóbb kivesszük a veremből. Amikor az algoritmus véget ér, S pontosan $CH(Q)$ csúcsait tartalmazza.
- Lényegében rendezés és pásztázás.
- Futási idő: $\mathcal{O}(n \log n)$

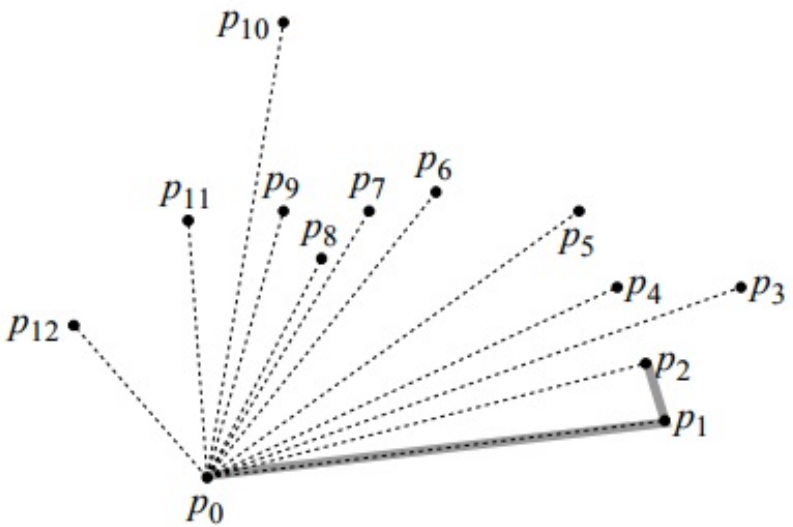
Az algoritmus

GRAHAM-PÁSZTÁZÁS(Q)

1 legyen p_0 a minimális y-koordinátájú Q-beli pont,
vagy egyezés esetén a bal szélső ilyen pont
2 legyen $\langle p_1, p_2, \ldots, p_m \rangle$ a többi Q-beli pont, p_0 körül poláris szög szerint az
óramutató járásával ellenkező sorrendben (ha több mint egy pontnak
ugyanaz a szöge, távolítsuk el mindet, a p_0-tól legtávolabbi kivételével)
3 VEREMBE(p_0, S)
4 VEREMBE(p_1, S)
5 VEREMBE(p_2, S)
6 **for** $i \leftarrow 3$ **to** m
7 **do while** a LEGFELSŐ-ALATTI(S), LEGFELSŐ(S) és p_i pontok
szöge nem fordul balra
8 **do** VEREMBŐL(S)
9 VEREMBE(p_i, S)
10 **return** S

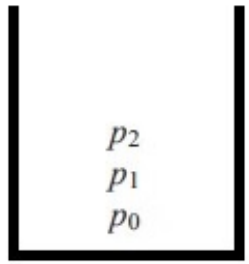
Az algoritmus

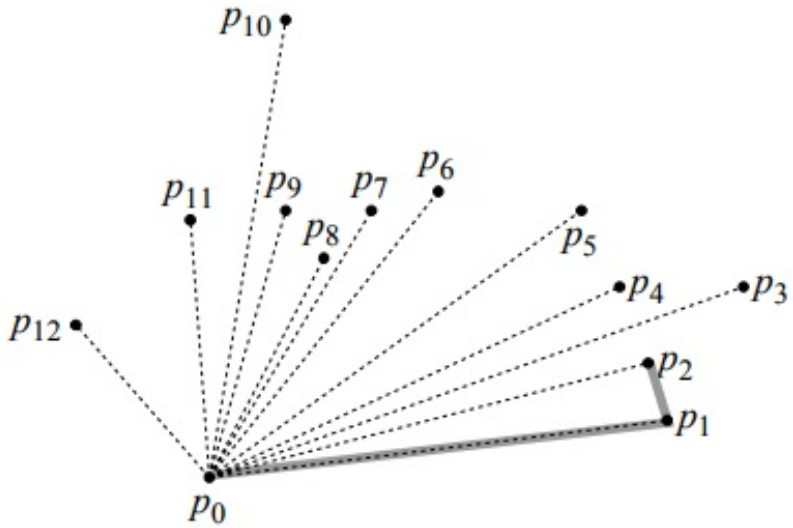
1 legyen p_0 a minimális y-koordinátájú Q-beli pont,
vagy egyezés esetén a bal szélső ilyen pont
2 legyen $\langle p_1, p_2, \ldots, p_m \rangle$ a többi Q-beli pont, p_0 körül poláris szög szerint az óramutató járásával ellenkező sorrendben (ha több mint egy pontnak ugyanaz a szöge, távolítsuk el mindet, a p_0-tól legtávolabbi kivételével)



Az algoritmus

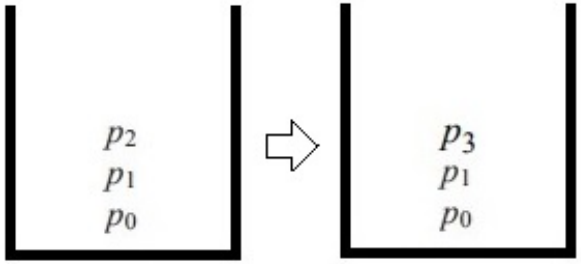
3 VEREMBE(p_0, S)
4 VEREMBE(p_1, S)
5 VEREMBE(p_2, S)

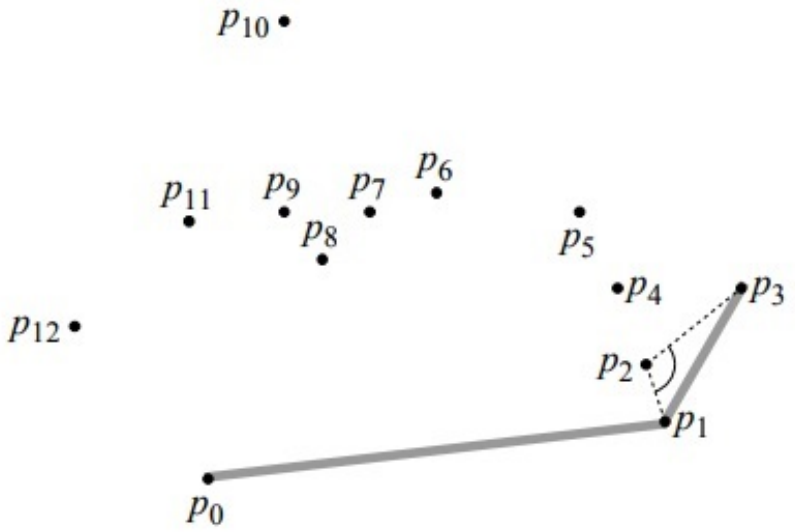




Az algoritmus

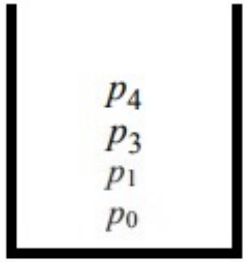
```
6  for i ← 3 to m
7      do while a LEGFELSŐ-ALATTI(S), LEGFELSŐ(S) és p_i pontok
                szöge nem fordul balra
8              do VEREMBŐL(S)
9          VEREMBE(p_i, S)
```

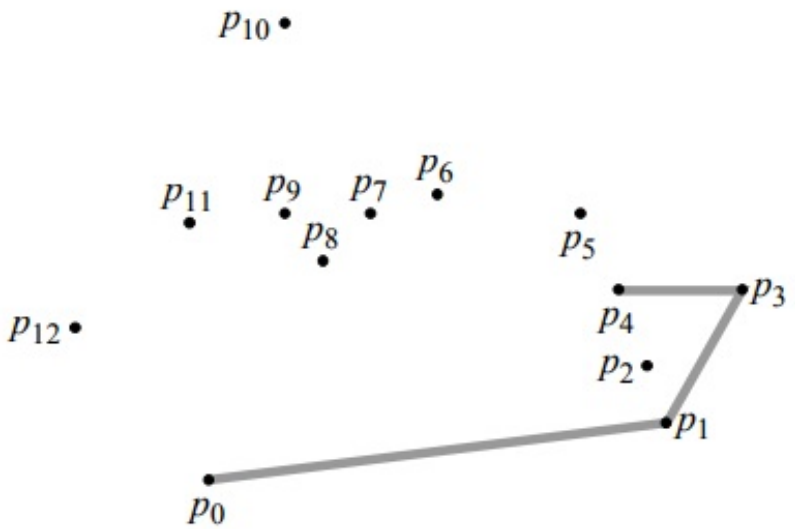




Az algoritmus

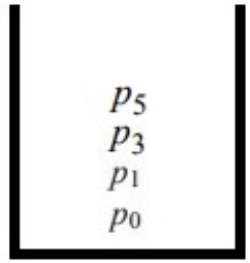
6 **for** $i \leftarrow 3$ **to** m
7 **do while** a LEGFELSŐ-ALATTI(S), LEGFELSŐ(S) és p_i pontok szöge nem fordul balra
8 **do** VEREMBŐL(S)
9 VEREMBE(p_i, S)

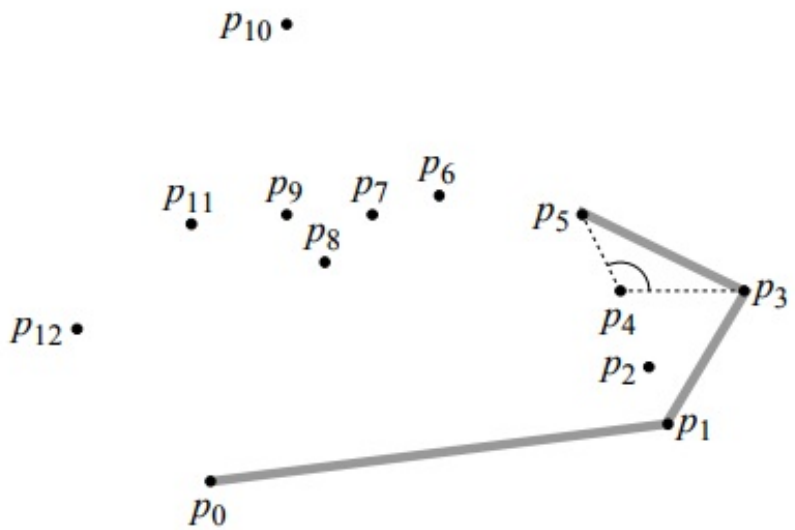




Az algoritmus

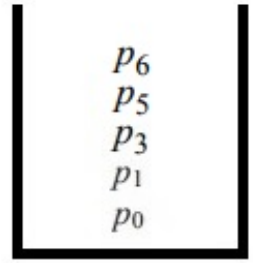
```
6  for i ← 3 to m
7      do while a LEGFELSŐ-ALATTI(S), LEGFELSŐ(S) és p_i pontok
                 szöge nem fordul balra
8              do VEREMBŐL(S)
9          VEREMBE(p_i, S)
```

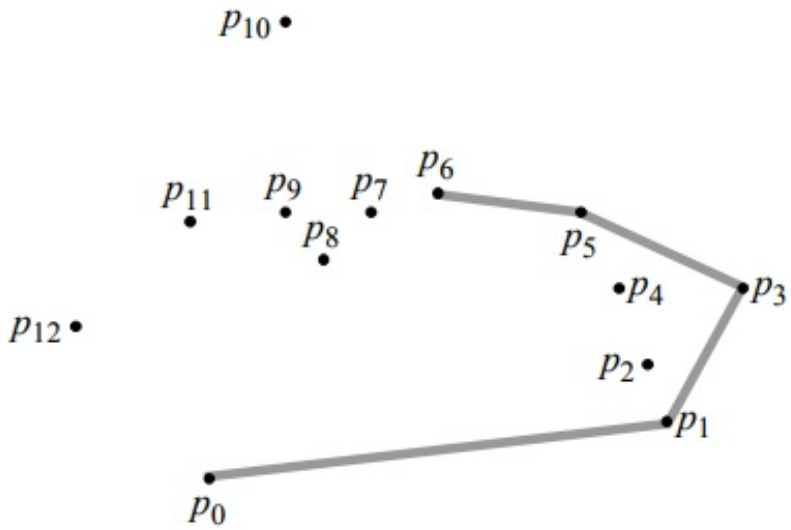




Az algoritmus

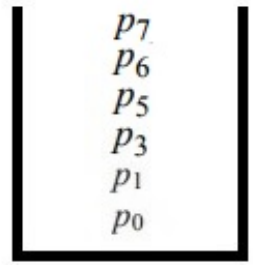
```
6  for i ← 3 to m
7      do while a LEGFELSŐ-ALATTI(S), LEGFELSŐ(S) és p_i pontok
               szöge nem fordul balra
8              do VEREMBŐL(S)
9          VEREMBE(p_i, S)
```

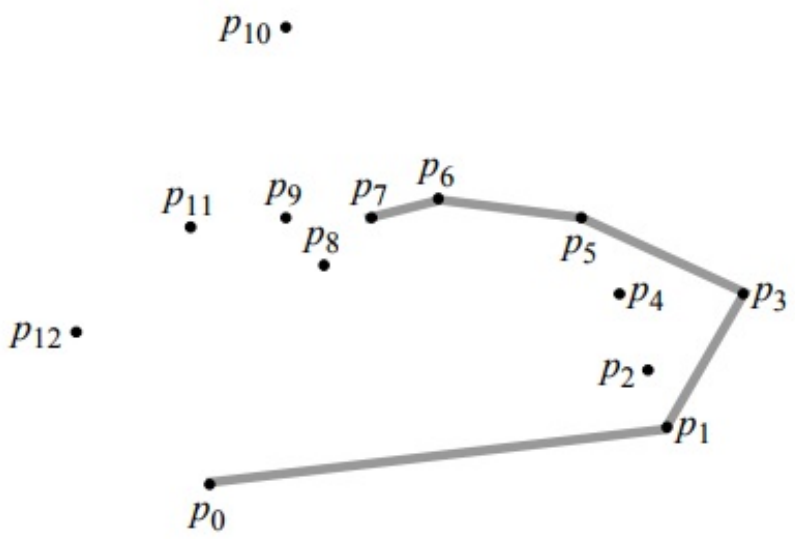




Az algoritmus

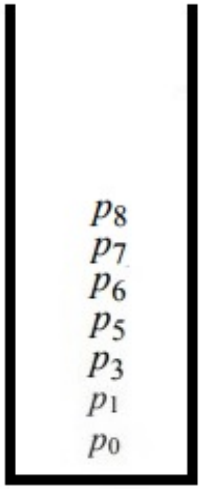
```
6  for i ← 3 to m
7      do while a LEGFELSŐ-ALATTI(S), LEGFELSŐ(S) és p_i pontok
               szöge nem fordul balra
8              do VEREMBŐL(S)
9          VEREMBE(p_i, S)
```

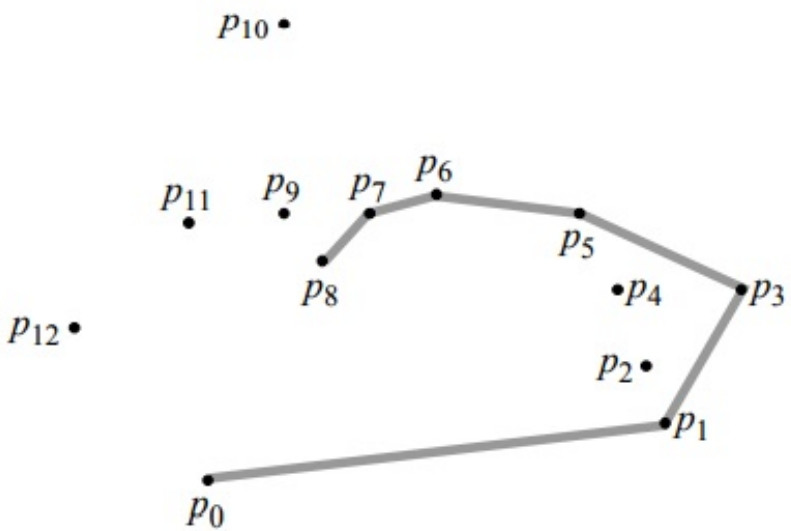




Az algoritmus

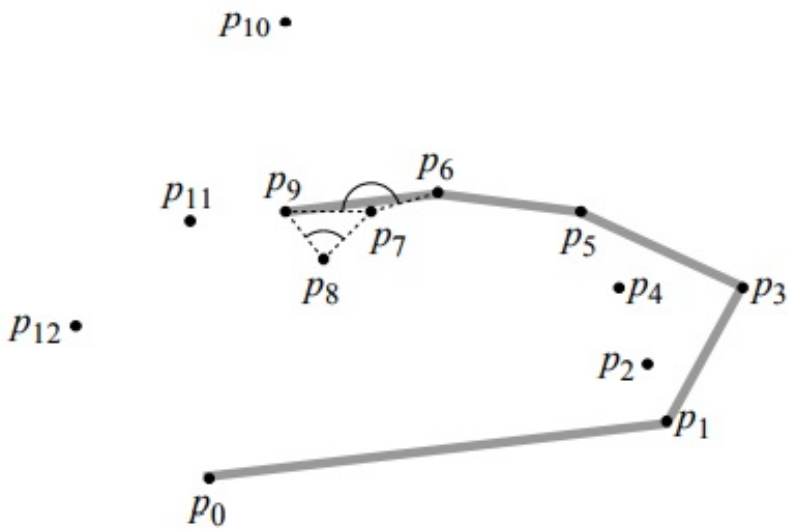
```
6  for i ← 3 to m
7      do while a LEGFELSŐ-ALATTI(S), LEGFELSŐ(S) és p_i pontok
                szöge nem fordul balra
8              do VEREMBŐL(S)
9          VEREMBE(p_i, S)
```

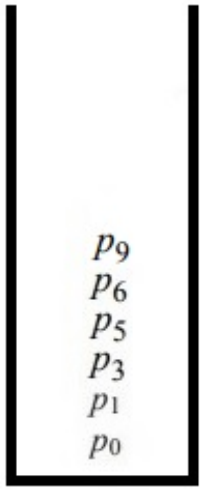




Az algoritmus

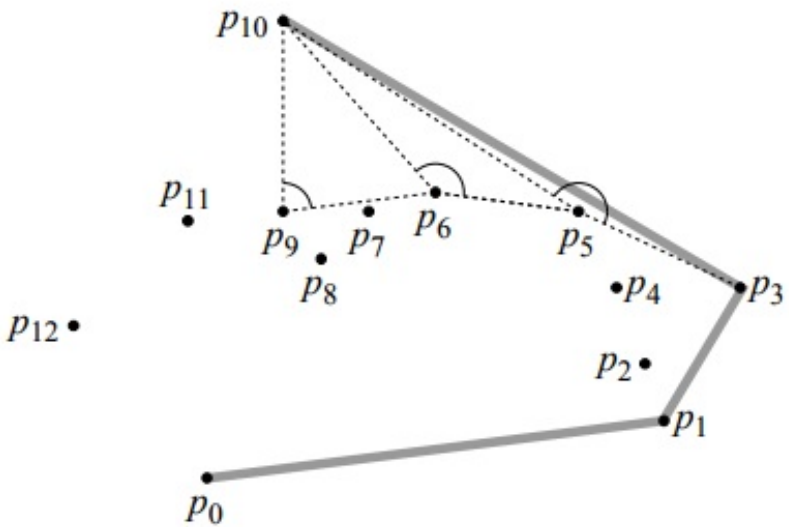
6 **for** $i \leftarrow 3$ **to** m
7 **do while** a LEGFELSŐ-ALATTI(S), LEGFELSŐ(S) és p_i pontok szöge nem fordul balra
8 **do** VEREMBŐL(S)
9 VEREMBE(p_i, S)

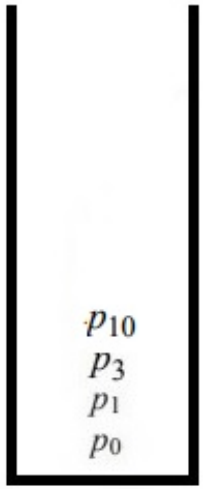




Az algoritmus

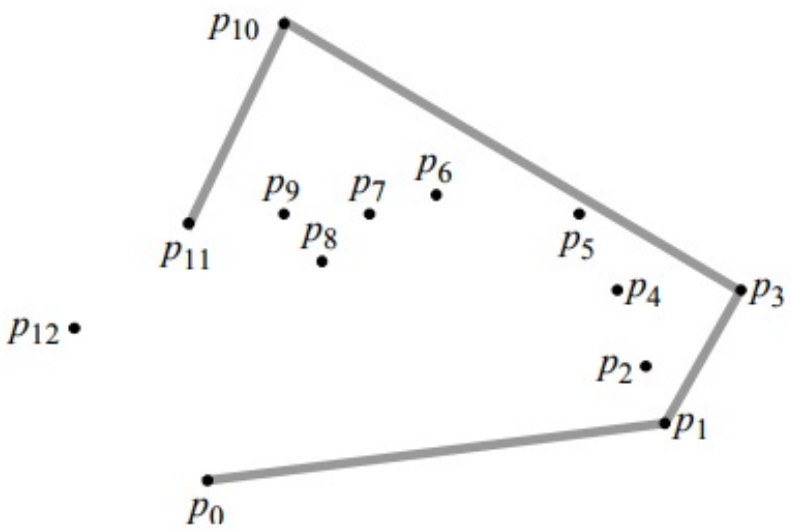
6 **for** $i \leftarrow 3$ **to** m
7 **do while** a LEGFELSŐ-ALATTI(S), LEGFELSŐ(S) és p_i pontok szöge nem fordul balra
8 **do** VEREMBŐL(S)
9 VEREMBE(p_i, S)

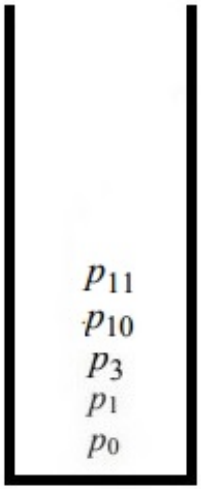




Az algoritmus

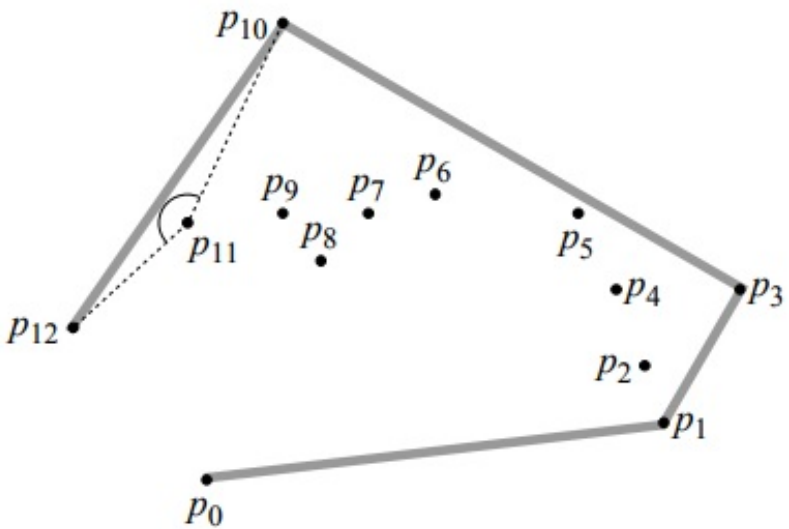
```
6  for i ← 3 to m
7      do while a LEGFELSŐ-ALATTI(S), LEGFELSŐ(S) és p_i pontok
                szöge nem fordul balra
8              do VEREMBŐL(S)
9          VEREMBE(p_i, S)
```

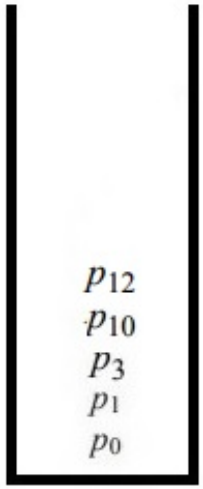




Az algoritmus

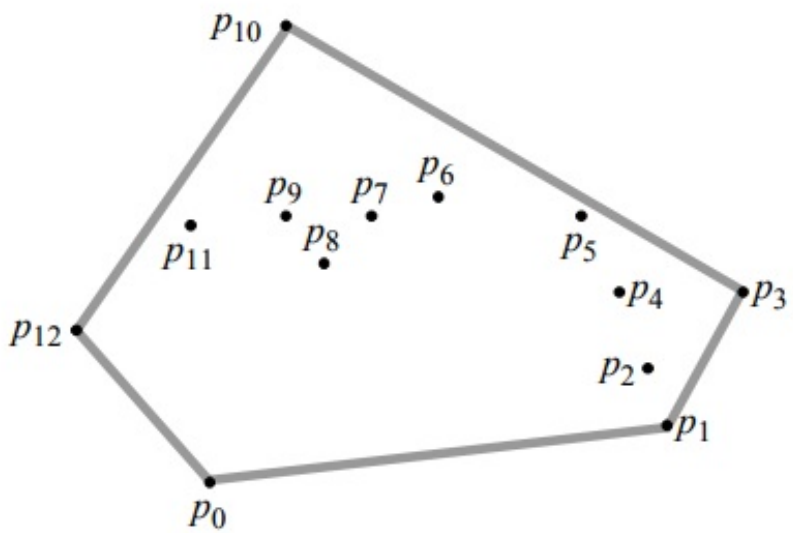
6 **for** $i \leftarrow 3$ **to** m
7 **do while** a LEGFELSŐ-ALATTI(S), LEGFELSŐ(S) és p_i pontok szöge nem fordul balra
8 **do** VEREMBŐL(S)
9 VEREMBE(p_i, S)

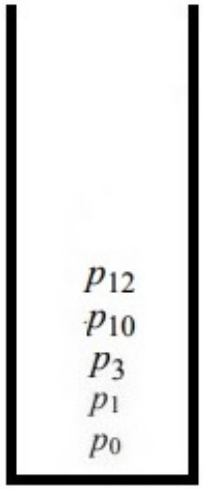




Az algoritmus

6 **for** $i \leftarrow 3$ **to** m
7 **do while** a LEGFELSŐ-ALATTI(S), LEGFELSŐ(S) és p_i pontok szöge nem fordul balra
8 **do** VEREMBŐL(S)
9 VEREMBE(p_i, S)





Helyesség

Ha a Graham-pásztázás eljárást egy olyan Q ponthalmazra futtatjuk, melynek elemszáma legalább 3, akkor befejezésekor az S verem pontosan CH(Q) csúcsaiból áll, alulról felfelé az óramutató járásával ellentétes sorrendben:

- Legyen $Q_i = p_0, p_1, \ldots, p_i$.
- $CH(Q_m) = CH(Q)$, mert az eltávolított pontok nincsenek benne $CH(Q)$-ban.

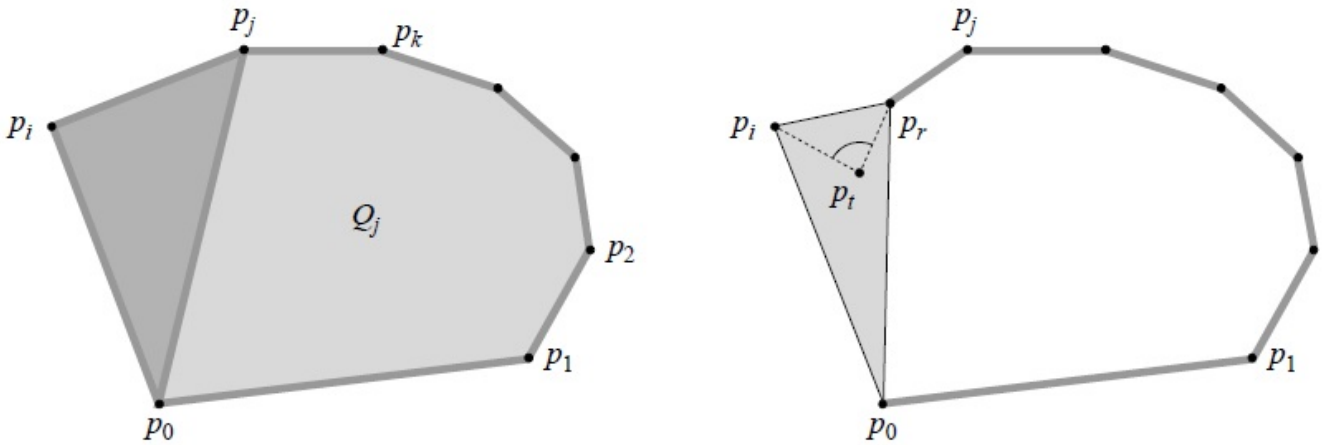
Helyesség

- Ciklusinvariáns: a for ciklus magjának minden végrehajtása elején az S verem pontosan $CH(Q_{i-1})$ csúcsaiból áll.
- Az invariáns fennáll az első végrehajtásnál, mert akkor az S verem pontosan a $Q_2 = Q_{i-1}$ csúcsokból áll, és a három csúcs saját konvex burkát alkotja.

Helyesség

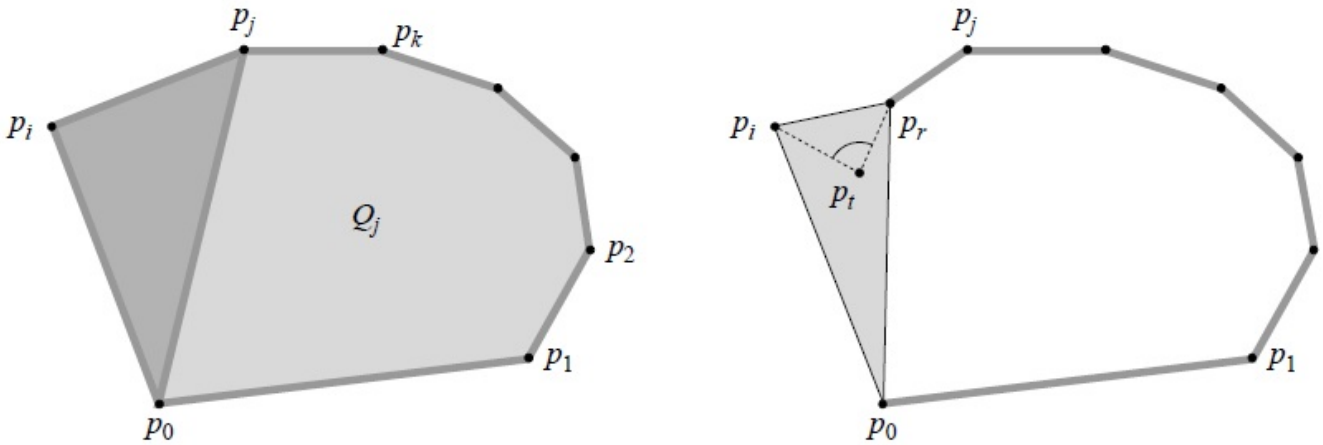
- A for ciklus magjába való belépéskor az S verem legfelső pontja p_{i-1}, legyen p_j a while ciklus végrehajtása után a legfelső, a p_k az S-ben közvetlenül alatta lévő pont.
- Amikor p_j a legfelső S-ben és még nem tettük be p_i-t, az S ugyanazokat a pontokat tartalmazza, amelyek a for ciklus j-re való végrehajtása után benne voltak $\rightarrow$ S pontosan $CH(Q_j)$ csúcsait tartalmazza.
- Mielőtt p_i a verembe kerül, p_i poláris szöge p_0 körül nagyobb, mint p_j-é és a $p_k p_j p_i$ szög balra fordul (különben kivettük volna p_j-t) $\rightarrow$ ha beszúrjuk p_i-t, S pontosan $CH(Q_j \cup p_i)$-t tartalmazza.

Helyesség



- $CH(Q_j \cup p_i)$ ugyanaz a ponthalmaz, mint $CH(Q_i)$.
  - Legyen p_t olyan, amit a for i-re való végrehajtása során vettünk ki, és p_r a közvetlenül p_t alatti pont a veremben.
  - A $p_r p_t p_i$ szög nem fordul balra és p_t poláris szöge p_0 körül nagyobb, mint p_r poláris szöge.

Helyesség



- $CH(Q_j \cup p_i)$ ugyanaz a ponthalmaz, mint $CH(Q_i)$.
  - p_t a p_0, p_r és p_i által alkotott háromszög belsejében vagy oldalán van $\rightarrow$ nem csúcsa $CH(Q_i)$-nek.
  - P_i az összes ilyen kivett pont halmaza: $CH(Q_i - P_i) = CH(Q_i)$.
  - $Q_i - P_i = Q_j \cup p_i \rightarrow CH(Q_j \cup p_i) = CH(Q_i - P_i) = CH(Q_i)$

Helyesség

- Ciklusinvariáns i növelésével a ciklusmag következő végrehajtására is fennáll.
- Amikor a ciklus befejeződik, $i = m + 1$, az S verem pontosan $CH(Q_m)$, alulról fölfelé az óramutató járásával ellentétes irányban.

Futási idő

- Első sor (extremális pont) $\theta(n)$ ideig tart.
- Második sor (rendezés) $\mathcal{O}(nlogn)$ futási idejű, ha a poláris szögek rendezésére az összefésülő rendezést vagy a kupacrendezést használjuk, a szögek összehasonlítására pedig a keresztszorzatos módszert.
- p_0-t, p_1-t és p_2 verembe tétele $\mathcal{O}(1)$

Futási idő

- Mivel $m \leq n - 1$, a for ciklusok magját legfeljebb $n - 3$ alkalommal hajtjuk végre.
- A ciklusmag mindegyik végrehajtása $\mathcal{O}(1)$ ideig tart, kivéve a while ciklusban töltött időt.
- Egész for ciklus $\mathcal{O}(n)$ ideig tart, kivéve a beágyazott while ciklust.
- While ciklus összesen $\mathcal{O}(n)$ ideig tart.

$\rightarrow$ Graham-pásztázás $\mathcal{O}(nlogn)$ ideig tart.

Jarvis algoritmusa

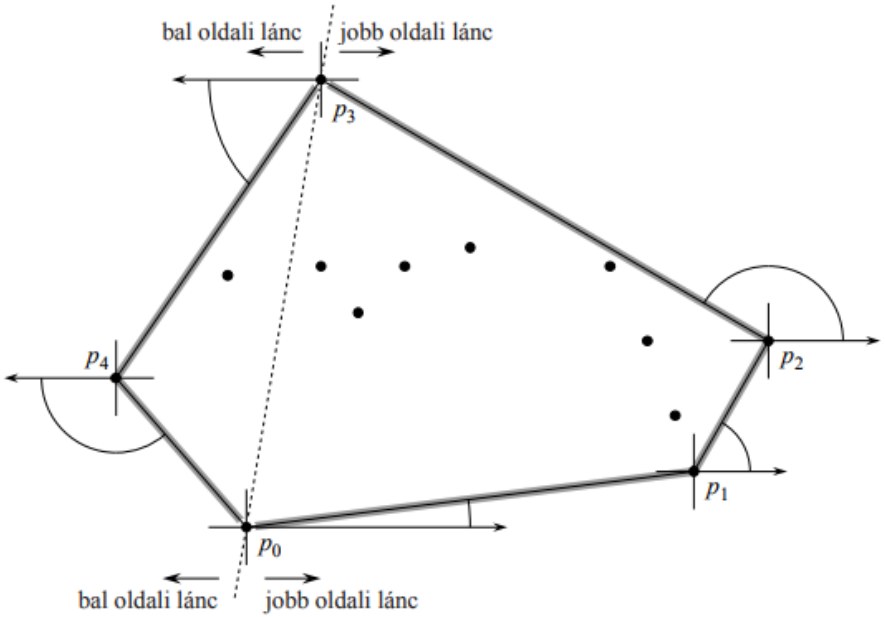
- Az egyik legegyszerűbb algoritmus a konvex burok kiszámítására.
- Kétdimenziós esetben az algoritmust Jarvis menetelésének is nevezik, R. A. Jarvis után, aki 1973-ban publikálta az algoritmust.
- Vizuálisabb elnevezése: ajándékcsomagolás.
- Futási idő: $\mathcal{O}(nh)$, ahol h a $CH(Q)$ csúcsainak száma.

Jarvis algoritmusa

- Az egyik legegyszerűbb algoritmus a konvex burok kiszámítására.
- Kétdimenziós esetben az algoritmust Jarvis menetelésének is nevezik, R. A. Jarvis után, aki 1973-ban publikálta az algoritmust.
- Vizuálisabb elnevezése: ajándékcsomagolás.
- Futási idő: $\mathcal{O}(nh)$, ahol h a $CH(Q)$ csúcsainak száma.

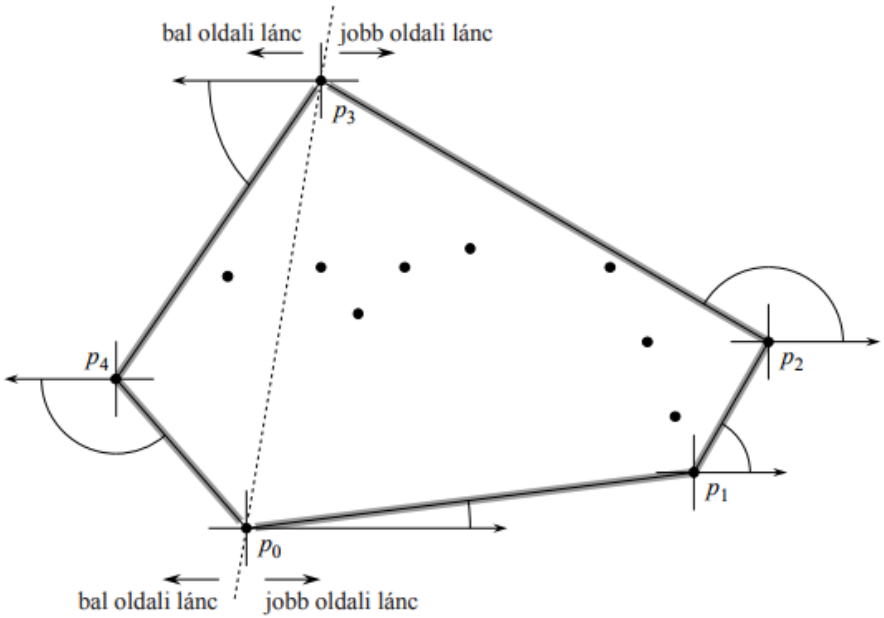
Működés

- p_0-ból indulunk a Graham-féle pásztázáshoz hasonlóan.



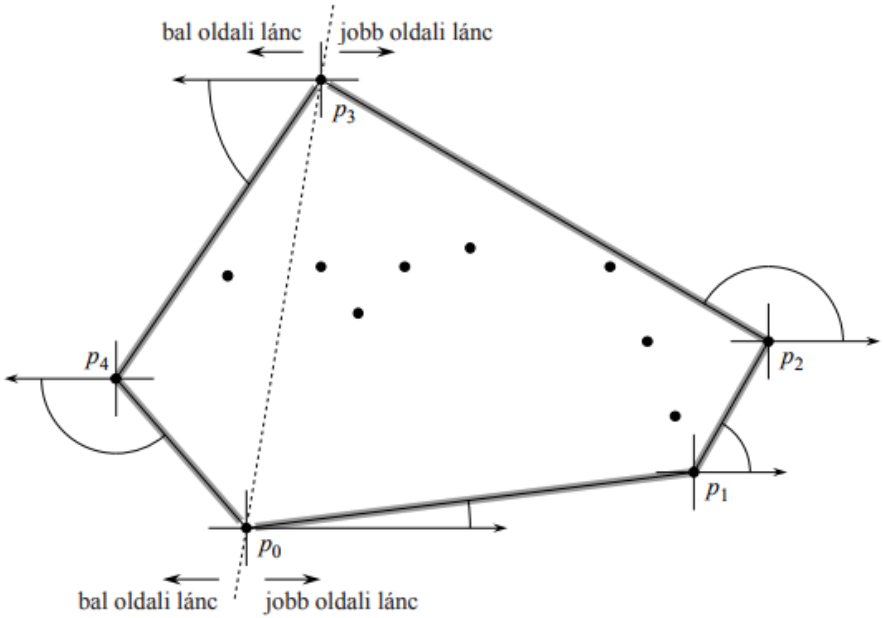
Működés

- A konvex burok következő csúcsa (p_1) az a pont, amelynek a legkisebb a p_0 körüli poláris szöge (vagy ezek közül a legtávolabbi).



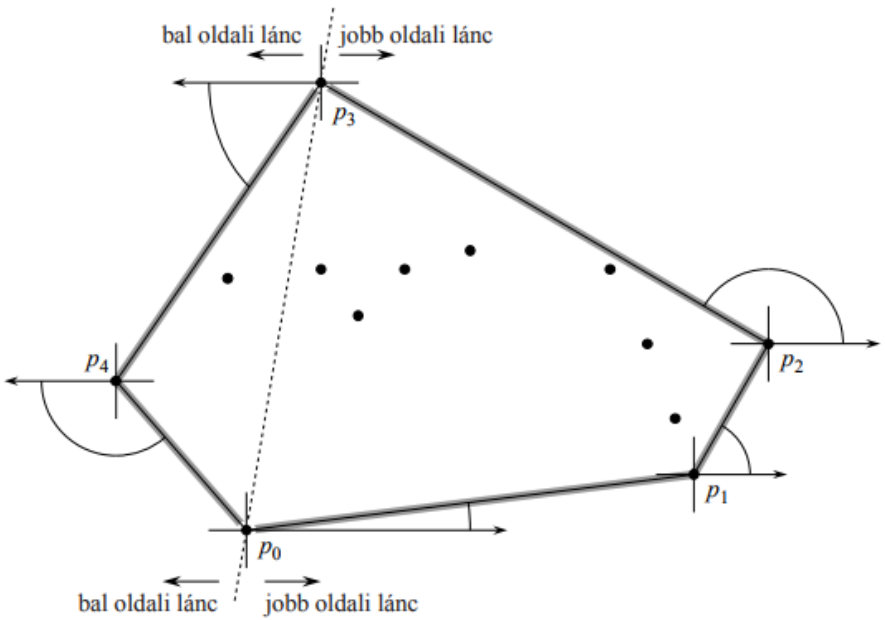
Működés

- Hasonlóan a p_2 a p_1 körül legkisebb poláris szögű pont.



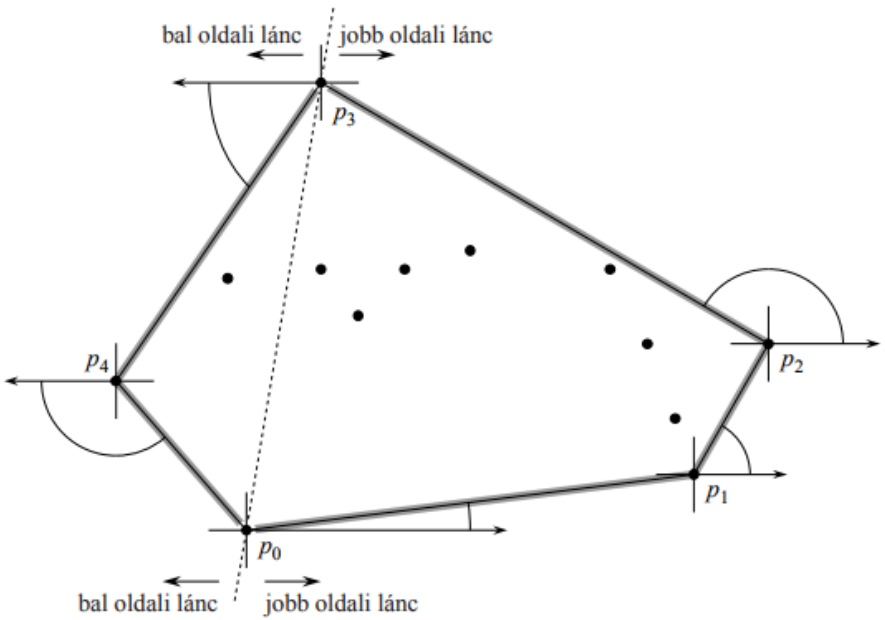
Működés

- Így tovább... mire elérjük a legfölső p_k csúcsot, megszerkesztjük a konvex burok jobb oldali láncát.



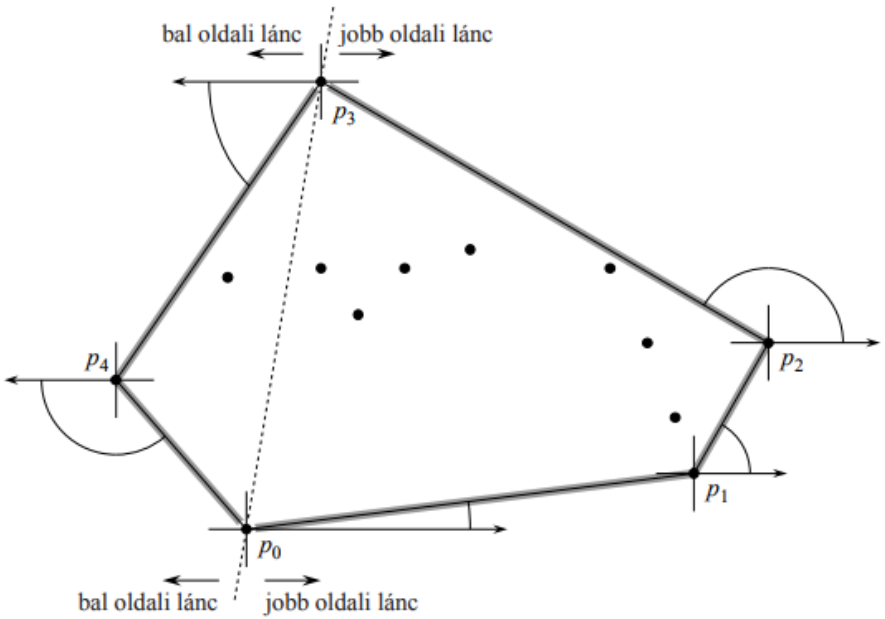
Működés

- Bal oldali lánchoz induljunk p_k-ból és legyen p_{k+1} az a pont, amelynek a negatív x-tengelytől mérve legkisebb a poláris szöge p_k körül.



Működés

- Ezt folytatjuk, míg vissza nem érünk a kiindulási p_0 csúcsba.



Futási idő

- $CH(Q)$ minden h csúcsára megkeressük a legkisebb poláris szögű csúcsot.
- A poláris szögek közötti minden összehasonlítás $\mathcal{O}(1)$ idő.
- n érték minimumát $\mathcal{O}(n)$ időben kereshetjük, ha minden összehasonlítás $\mathcal{O}(1)$ idejű.

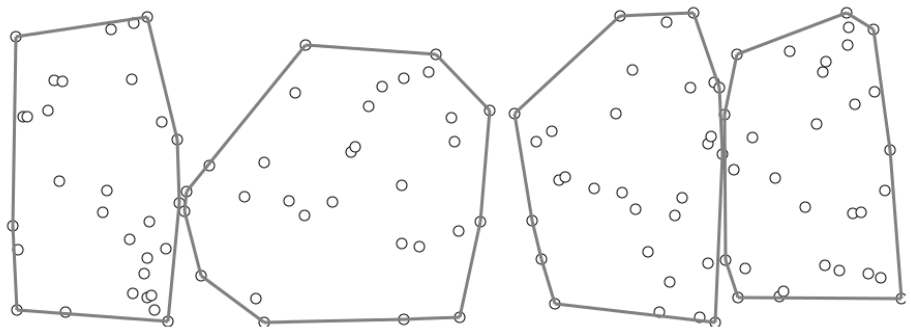
$\rightarrow$ Jarvis menetelése $\mathcal{O}(nh)$ ideig tart.

Chan algoritmusa

- Timothy Chan találta ki 1993-ban.
- Oszd-meg-és uralkodj, Jarvis menetelése és Graham-pásztázás kombinációja.
- A futási idő az output méretétől függ.

Hogyan működik?

1. Keressünk egy "mágikus" m értéket, amely minden pontot n/m részhalmazba oszt, ahol minden részhalmaz m pontot tartalmaz (rendezés nélkül).
2. Minden részhalmazra használjuk Graham algoritmusát a részburok kiszámítására.
3. Minden részburokra használjuk Jarvis algoritmusát a végső konvex burok meghatározására.



Mágikus m

- Az algoritmus lényege, hogyan határozzuk meg m értékét.
- Chan a következő trükköt javasolta:
  - Beállítunk egy $t = 1$ paramétert és legyen $m = 2^{2^t}$.
  - Használjuk ezt az m-et Chan algoritmusában, számlálót indítunk a burok outputpontjaira. Ha a számláló eléri az m értéket, leállítja az algoritmust és 1-gyel növeli t-t, újraszámítja m értékét, és újra elvégzi ezt az eljárást.
- Így az algoritmus futási ideje mindig kisebb lesz, mint $\mathcal{O}(n \log m)$.

Futási idő

- n/m részre osztottuk a pontokat, minden résznek m pontja van.
- Mivel az algoritmus leáll, ha az output száma eléri m-et, ez a lépés legfeljebb $\mathcal{O}(n \log m)$ költségű, ha m kicsi.
- Részhalmazokra Graham az alburkok kiszámítására: $\mathcal{O}((n/m) * m \log m) = \mathcal{O}(n \log m)$.

Futási idő

- Jarvis a részburkokra:
  - Válasszunk ki egy extremális pontot: $\mathcal{O}(n)$
  - A legjobboldalibb pontot válasszuk következőnek: $\mathcal{O}((n/m) * m \log m)$
  - Következő pontra ugyanez: h pont van a burokban $\rightarrow$ $\mathcal{O}(h * (n/m) * m \log m) = \mathcal{O}(n \log h)$, ha $h = m$.

$\rightarrow$ Összesen $\mathcal{O}(n \log h)$ futási idő.

Köszönöm a figyelmet!