Algoritmi avansați

C8 - Acoperiri convexe

Mihai-Sorin Stupariu

Sem. al II-lea, 2023 - 2024

Introducere

Criterii numerice. Raport și test de orientare

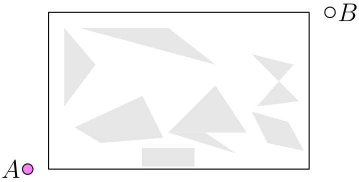
Acoperiri convexe

Introducere

- Algoritmii geometrici sunt legați de Geometria Computațională: *complexitatea computațională a problemelor geometrice în contextul analizei algoritmilor* [Lee & Preparata, 1984]

Introducere

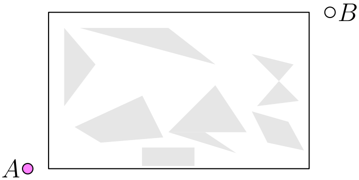
- Algoritmii geometrici sunt legați de Geometria Computațională: *complexitatea computațională a problemelor geometrice în contextul analizei algoritmilor* [Lee & Preparata, 1984]
- Problemă (exemplu):



Cum poate fi deplasat discul din A în B fără a atinge obstacolele?

Introducere

- Algoritmii geometrici sunt legați de Geometria Computațională: *complexitatea computațională a problemelor geometrice în contextul analizei algoritmilor* [Lee & Preparata, 1984]
- Problemă (exemplu):

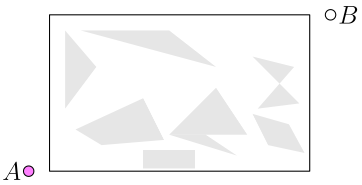


Cum poate fi deplasat discul din A în B fără a atinge obstacolele?

- Complexitatea: memorie, timp, calcule

Introducere

- Algoritmii geometrici sunt legați de Geometria Computațională: *complexitatea computațională a problemelor geometrice în contextul analizei algoritmilor* [Lee & Preparata, 1984]
- Problemă (exemplu):

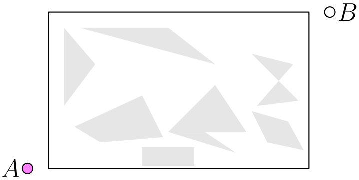


Cum poate fi deplasat discul din A în B fără a atinge obstacolele?

- Complexitatea: memorie, timp, calcule
- Probleme abordate: acoperiri convexe, proximitate, intersecții, căutare, etc.

Introducere

- Algoritmii geometrici sunt legați de Geometria Computațională: *complexitatea computațională a problemelor geometrice în contextul analizei algoritmilor* [Lee & Preparata, 1984]
- Problemă (exemplu):



Cum poate fi deplasat discul din A în B fără a atinge obstacolele?

- Complexitatea: memorie, timp, calcule
- Probleme abordate: acoperiri convexe, proximitate, intersecții, căutare, etc.
- Tehnici utilizate: construcții incrementale, divide et impera, plane-sweep, transformări geometrice, etc.

Introducere

- Note istorice:

Introducere

- Note istorice:
  - Ideea de construcție geometrică realizată într-un număr finit de pași: Elementele lui Euclid

Introducere

- Note istorice:
  - Ideea de construcție geometrică realizată într-un număr finit de pași: Elementele lui Euclid
  - Abordare numerică - sistem de coordonate în plan: Descartes (sec. XVII)

Introducere

- Note istorice:
  - Ideea de construcție geometrică realizată într-un număr finit de pași: Elementele lui Euclid
  - Abordare numerică - sistem de coordonate în plan: Descartes (sec. XVII)
  - "Simplicitatea" construcțiilor geometrice: Lemoine ($\sim$ 1900)

Introducere

- Note istorice:
  - Ideea de construcție geometrică realizată într-un număr finit de pași: Elementele lui Euclid
  - Abordare numerică - sistem de coordonate în plan: Descartes (sec. XVII)
  - "Simplicitatea" construcțiilor geometrice: Lemoine ($\sim$ 1900)
  - a doua jumătate a sec. XX: formularea / rezolvarea unor probleme de GC; în 1975 este folosit prima dată termenul de *Computational Geometry* (M.I. Shamos, "Geometric Complexity", Proc. 7th ACM Annual Symposium on Theory of Computing)

Introducere

- Note istorice:
  - Ideea de construcție geometrică realizată într-un număr finit de pași: Elementele lui Euclid
  - Abordare numerică - sistem de coordonate în plan: Descartes (sec. XVII)
  - "Simplicitatea" construcțiilor geometrice: Lemoine ($\sim$ 1900)
  - a doua jumătate a sec. XX: formularea / rezolvarea unor probleme de GC; în 1975 este folosit prima dată termenul de *Computational Geometry* (M.I. Shamos, "Geometric Complexity", Proc. 7th ACM Annual Symposium on Theory of Computing)
- Domenii de aplicabilitate: grafică pe calculator, pattern recognition, robotică, statistică, gestionarea bazelor de date numerice, cercetări operaționale

Bibliografie

- M. de Berg, M. van Kreveld, M. Overmars si O. Schwarzkopf, *Computational Geometry, Algorithms and Applications*, Springer, 2008.
- F. Preparata si M. Shamos, *Computational Geometry: An Introduction*, Springer, 1985.
- S. Devadoss, J. O'Rourke, *Discrete and Computational Geometry*, Princeton University Press, 2011.
  (Site: http://cs.smith.edu/˜orourke/DCG/)

Bibliografie

- M. de Berg, M. van Kreveld, M. Overmars si O. Schwarzkopf, *Computational Geometry, Algorithms and Applications*, Springer, 2008.
- F. Preparata si M. Shamos, *Computational Geometry: An Introduction*, Springer, 1985.
- S. Devadoss, J. O'Rourke, *Discrete and Computational Geometry*, Princeton University Press, 2011.
  (Site: http://cs.smith.edu/~orourke/DCG/)
- D. Lee, F. Preparata, *Computational Geometry - A Survey*, IEEE Transactions on Computers, **33** (1984), 1072-1101.
- B. Gärtner, M. Hoffmann, Computational Geometry. Lecture Notes, ETH Zürich, 2013.
- J. Goodman, J. O'Rourke, C. Tóth (eds.) Handbook of Discrete and Computational Geometry, 2017.

Criterii numerice – poziția relativă a unor puncte

- Stabilirea unor relații între puncte / ordonare

Criterii numerice – poziția relativă a unor puncte

- Stabilirea unor relații între puncte / ordonare
- **Context 1D**

Criterii numerice – poziția relativă a unor puncte

- Stabilirea unor relații între puncte / ordonare
- **Context 1D**
  - ordonare (**relativ la un sistem de coordonate**)

Criterii numerice — poziția relativă a unor puncte

- Stabilirea unor relații între puncte / ordonare
- **Context 1D**
  - ordonare (**relativ la un sistem de coordonate**)
  - raport (**independent de alegerea unui sistem de coordonate**)

Criterii numerice – poziția relativă a unor puncte

- Stabilirea unor relații între puncte / ordonare
- **Context 1D**
  - ordonare (**relativ la un sistem de coordonate**)
  - raport (**independent de alegerea unui sistem de coordonate**)
- **Context 2D**

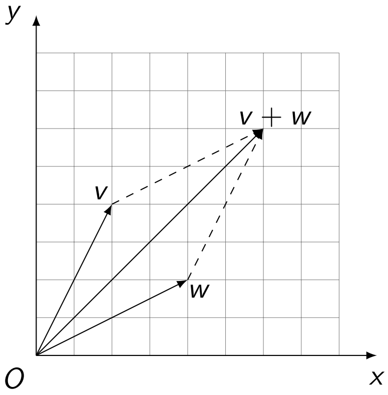
Criterii numerice – poziția relativă a unor puncte

- Stabilirea unor relații între puncte / ordonare
- **Context 1D**
  - ordonare (**relativ la un sistem de coordonate**)
  - raport (**independent de alegerea unui sistem de coordonate**)
- **Context 2D**
  - ordonare (**relativ la un sistem de coordonate** – posibile alegeri: coordonate carteziene, coordonate polare)

Criterii numerice – poziția relativă a unor puncte

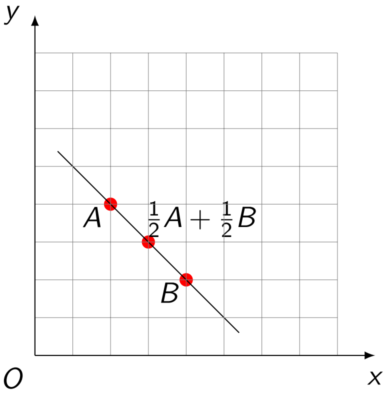
- Stabilirea unor relații între puncte / ordonare
- **Context 1D**
  - ordonare (**relativ la un sistem de coordonate**)
  - raport (**independent de alegerea unui sistem de coordonate**)
- **Context 2D**
  - ordonare (**relativ la un sistem de coordonate** – posibile alegeri: coordonate carteziene, coordonate polare)
  - testul de orientare (**independent de alegerea unui sistem cartezian de coordonate**)

Vectori și puncte



Combinații liniare

$\alpha v + \beta w \ (\alpha, \beta \in \mathbb{R})$



Combinații afine

$\lambda A + \mu B \ (\lambda, \mu \in \mathbb{R}$ **și** $\lambda + \mu = 1)$

Conceptul de raport

- **Lemă** Fie A și B două puncte distincte în $\mathbb{R}^n$. Pentru orice punct $P \in AB$, $P \neq B$ există un unic scalar $r \in \mathbb{R} \setminus \{-1\}$ astfel ca $\overrightarrow{AP} = r\ \overrightarrow{PB}$. Reciproc, fiecărui scalar $r \in \mathbb{R} \setminus \{-1\}$, îi corespunde un unic punct $P \in AB$.

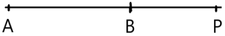
Conceptul de raport

- **Lemă** Fie A și B două puncte distincte în $\mathbb{R}^n$. Pentru orice punct $P \in AB$, $P \neq B$ există un unic scalar $r \in \mathbb{R} \setminus \{-1\}$ astfel ca $\overrightarrow{AP} = r\ \overrightarrow{PB}$. Reciproc, fiecărui scalar $r \in \mathbb{R} \setminus \{-1\}$, îi corespunde un unic punct $P \in AB$.
- **Definiție** Scalarul r definit în lema anterioară se numește **raportul** punctelor A, B, P (sau **raportul în care punctul P împarte segmentul** $[AB]$) și este notat cu $r(A, P, B)$.

Conceptul de raport

- **Lemă** Fie A și B două puncte distincte în $\mathbb{R}^n$. Pentru orice punct $P \in AB$, $P \neq B$ există un unic scalar $r \in \mathbb{R} \setminus \{-1\}$ astfel ca $\overrightarrow{AP} = r\ \overrightarrow{PB}$. Reciproc, fiecărui scalar $r \in \mathbb{R} \setminus \{-1\}$, îi corespunde un unic punct $P \in AB$.
- **Definiție** Scalarul r definit în lema anterioară se numește **raportul** punctelor A, B, P (sau **raportul în care punctul** P **împarte segmentul** $[AB]$) și este notat cu $r(A, P, B)$.

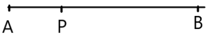
$$\overrightarrow{AP} = r\ \overrightarrow{PB},\ r > 0$$



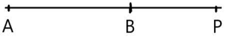
$$\overrightarrow{AP} = r\ \overrightarrow{PB},\ r < 0$$

Conceptul de raport

- **Lemă** Fie A și B două puncte distincte în $\mathbb{R}^n$. Pentru orice punct $P \in AB$, $P \neq B$ există un unic scalar $r \in \mathbb{R} \setminus \{-1\}$ astfel ca $\overrightarrow{AP} = r\ \overrightarrow{PB}$. Reciproc, fiecărui scalar $r \in \mathbb{R} \setminus \{-1\}$, îi corespunde un unic punct $P \in AB$.
- **Definiție** Scalarul r definit în lema anterioară se numește **raportul** punctelor A, B, P (sau **raportul în care punctul** P **împarte segmentul** $[AB]$) și este notat cu $r(A, P, B)$.



$\overrightarrow{AP} = r\ \overrightarrow{PB}, r > 0$



$\overrightarrow{AP} = r\ \overrightarrow{PB}, r < 0$

- **Observație importantă.** În calcularea raportului, <u>ordinea</u> punctelor este esențială. Modul în care este definită această noțiune (mai precis ordinea în care sunt considerate punctele) diferă de la autor la autor.

Raport- exemple

(i) În $\mathbb{R}^2$ considerăm punctele $A = (1, 1), B = (2, 2), C = (7, 7)$. Determinăm raportul $r(A, B, C)$.

$r = ?$ a.î. $\overrightarrow{AB} = r\,\overrightarrow{BC}$

$\overrightarrow{AB} = B - A = (x_B - x_A,\ y_B - y_A) = (1, 1)$

$\overrightarrow{BC} = C - B = (x_C - x_B,\ y_C - y_B) = (5, 5)$

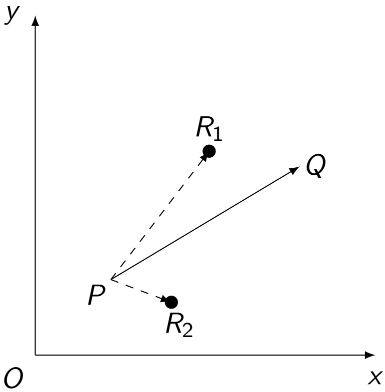
$$\overrightarrow{AB} = \underbrace{\frac{1}{5}}_{r(A,B,C)} \cdot \overrightarrow{BC}$$

Raport- exemple

(ii) În $\mathbb{R}^3$ considerăm punctele $A = (1, 2, 3)$, $B = (2, 1, -1)$, $C = (0, 3, 7)$. Atunci punctele A, B, C sunt coliniare și avem $r(A, C, B) = -\frac{1}{2}$, $r(B, C, A) = -2$, $r(C, A, B) = 1$, $r(C, B, A) = -2$.

(iii) Fie A, B două puncte din $\mathbb{R}^n$ și $M = \frac{1}{2}A + \frac{1}{2}B$. Atunci $r(A, M, B) = 1$, $r(M, A, B) = -\frac{1}{2}$.

Testul de orientare - motivație



Poziția relativă a două puncte față de un vector / o muchie orientată

Enunț principal

- **Propoziție.** *Fie* $P=(p_1,p_2), Q=(q_1,q_2)$ *două puncte distincte din planul* $\mathbf{R}^2$, *fie* $R=(r_1,r_2)$ *un punct arbitrar și*

$$\Delta(P,Q,R)=\begin{vmatrix} 1 & 1 & 1 \\ p_1 & q_1 & r_1 \\ p_2 & q_2 & r_2 \end{vmatrix}.$$

Atunci R *este situat:*
(i) pe dreapta $PQ \Leftrightarrow \Delta(P,Q,R)=0$ *("ecuația dreptei");*
(ii) "în dreapta" segmentului orientat $\overrightarrow{PQ} \Leftrightarrow \Delta(P,Q,R)<0$*;*
(iii) "în stânga" segmentului orientat $\overrightarrow{PQ} \Leftrightarrow \Delta(P,Q,R)>0$*.*

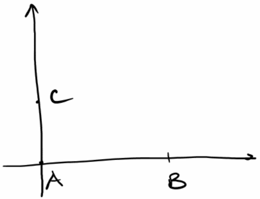
Enunț principal

- **Propoziție.** *Fie* $P = (p_1, p_2), Q = (q_1, q_2)$ *două puncte distincte din planul* $\mathbf{R}^2$, *fie* $R = (r_1, r_2)$ *un punct arbitrar și*

$$\Delta(P, Q, R) = \begin{vmatrix} 1 & 1 & 1 \\ p_1 & q_1 & r_1 \\ p_2 & q_2 & r_2 \end{vmatrix}.$$

Atunci R *este situat:*
(i) pe dreapta $PQ \Leftrightarrow \Delta(P, Q, R) = 0$ *("ecuația dreptei");*
(ii) "în dreapta" segmentului orientat $\overrightarrow{PQ} \Leftrightarrow \Delta(P, Q, R) < 0$*;*
(iii) "în stânga" segmentului orientat $\overrightarrow{PQ} \Leftrightarrow \Delta(P, Q, R) > 0$*.*
- **Obs.** Testul de orientare se bazează pe calculul unui polinom de gradul II ($\Delta(P, Q, R)$).

Testul de orientare - exemplu



$A = (0,0)$

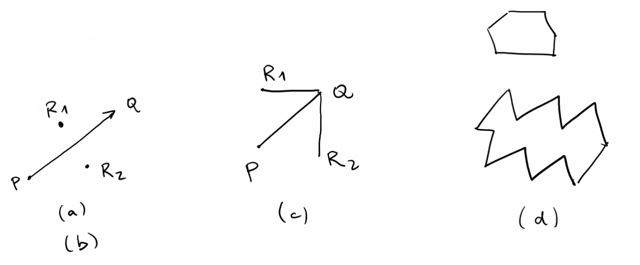
$B = (1,0)$

$C = (0,1)$

$$\Delta(A,B,C) = \begin{vmatrix} 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{vmatrix} = 1 > 0$$

deci cf. criteriului C este în stânga muchiei orientate $\overrightarrow{AB}$

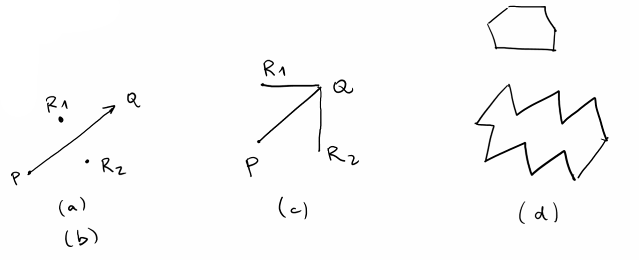
Aplicații



(a) dacă un punct este în dreapta / stânga unei muchii orientate;

(b) dacă două puncte sunt de o parte și de alta a unui segment / a unei drepte;

Aplicații

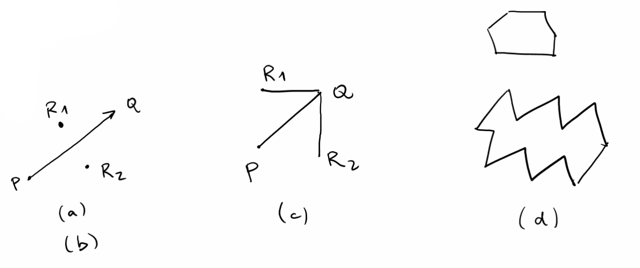


(a) dacă un punct este în dreapta / stânga unei muchii orientate;

(b) dacă două puncte sunt de o parte și de alta a unui segment / a unei drepte;

(c) natura unui viraj în parcurgerea unei linii poligonale (la dreapta / la stânga);

Aplicații



(a) dacă un punct este în dreapta / stânga unei muchii orientate;

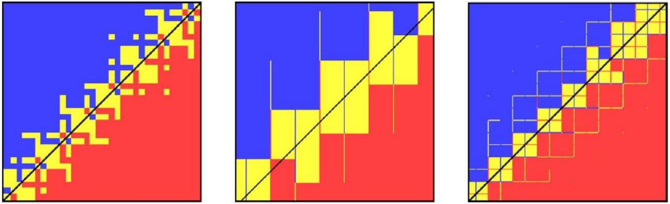
(b) dacă două puncte sunt de o parte și de alta a unui segment / a unei drepte;

(c) natura unui viraj în parcurgerea unei linii poligonale (la dreapta / la stânga);

(d) natura unui poligon (convex / concav).

Limitări - robustețe și erori de rotunjire

L. Kettner et al. / Computational Geometry 40 (2008) 61–78 65



(a) $p: \begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$, $q: \begin{pmatrix} 12 \\ 12 \end{pmatrix}$, $r: \begin{pmatrix} 24 \\ 24 \end{pmatrix}$

(b) $\begin{pmatrix} 0.50000000000002531 \\ 0.5000000000000171 \end{pmatrix}$, $\begin{pmatrix} 17.300000000000001 \\ 17.300000000000001 \end{pmatrix}$, $\begin{pmatrix} 24.00000000000005 \\ 24.0000000000000517765 \end{pmatrix}$

(c) $\begin{pmatrix} 0.5 \\ 0.5 \end{pmatrix}$, $\begin{pmatrix} 8.8000000000000007 \\ 8.8000000000000007 \end{pmatrix}$, $\begin{pmatrix} 12.1 \\ 12.1 \end{pmatrix}$

Fig. 2. The weird geometry of the float-orientation predicate: The figure shows the results of *float_orient*$(p_x + Xu_x, p_y + Yu_y, q, r)$ for $0 \leqslant X, Y \leqslant 255$, where $u_x = u_y = 2^{-53}$ is the increment between adjacent floating-point numbers in the considered range. The result is color coded: Yellow (red, blue, resp.) pixels represent collinear (negative, positive, resp.) orientation. The line through q and r is shown in black.

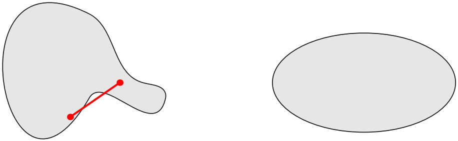
Sursa: Kettner et al, *Classroom examples of robustness problems in geometric computations*, 2008

Mulțimi convexe: generalități

- **Conceptul de mulțime convexă:**
  O mulțime $M \subset \mathbf{R}^m$ este convexă dacă oricare ar fi $p, q \in M$, segmentul $[pq]$ este inclus în M.

Mulțimi convexe: generalități

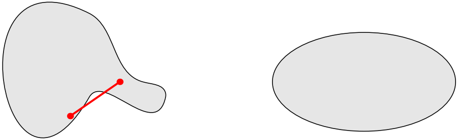
- **Conceptul de mulțime convexă:**
  O mulțime $M \subset \mathbf{R}^m$ este convexă dacă oricare ar fi $p, q \in M$, segmentul $[pq]$ este inclus în M.



Mulțimea din stânga nu este convexă, întrucât **există** două puncte, pentru care segmentul determinat nu este inclus în mulțime (punctele cu această proprietate nu sunt unice!).

Mulțimi convexe: generalități

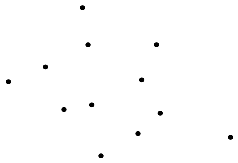
- **Conceptul de mulțime convexă:**
  O mulțime $M \subset \mathbf{R}^m$ este convexă dacă oricare ar fi $p, q \in M$, segmentul $[pq]$ este inclus în M.



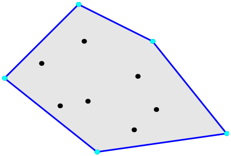
Mulțimea din stânga nu este convexă, întrucât **există** două puncte, pentru care segmentul determinat nu este inclus în mulțime (punctele cu această proprietate nu sunt unice!).

- **Problematizare:**
  Mulțimile finite cu cel puțin două elemente nu sunt convexe $\longrightarrow$ necesară **acoperirea convexă**.

Acoperire convexă a unei mulțimi (finite) $\mathcal{P}$: concept

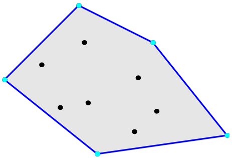


Acoperire convexă a unei mulțimi finite $\mathcal{P}$: concept



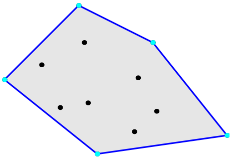
- Caracterizări echivalente:

Acoperire convexă a unei mulțimi finite $\mathcal{P}$: concept



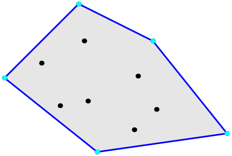
- Caracterizări echivalente:
  - Cea mai "mică" (în sensul incluziunii) mulțime convexă care conține $\mathcal{P}$.

Acoperire convexă a unei mulțimi finite $\mathcal{P}$: concept



- Caracterizări echivalente:
  - Cea mai "mică" (în sensul incluziunii) mulțime convexă care conține $\mathcal{P}$.
  - Intersecția tuturor mulțimilor convexe care conțin $\mathcal{P}$.

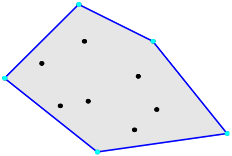
Acoperire convexă a unei mulțimi finite $\mathcal{P}$: concept



- Caracterizări echivalente:
  - Cea mai "mică" (în sensul incluziunii) mulțime convexă care conține $\mathcal{P}$.
  - Intersecția tuturor mulțimilor convexe care conțin $\mathcal{P}$.
  - Mulțimea tuturor combinațiilor convexe ale punctelor din $\mathcal{P}$. O **combinație convexă** a punctelor $P_1, P_2, \ldots, P_n$ este un punct P de forma

$$P = \alpha_1 P_1 + \ldots + \alpha_n P_n, \quad \alpha_1, \ldots, \alpha_n \in [0, 1],\ \alpha_1 + \ldots + \alpha_n = 1.$$

Acoperire convexă a unei mulțimi finite $\mathcal{P}$: concept



- Caracterizări echivalente:
  - Cea mai "mică" (în sensul incluziunii) mulțime convexă care conține $\mathcal{P}$.
  - Intersecția tuturor mulțimilor convexe care conțin $\mathcal{P}$.
  - Mulțimea tuturor combinațiilor convexe ale punctelor din $\mathcal{P}$. O **combinație convexă** a punctelor $P_1, P_2, \ldots, P_n$ este un punct P de forma

$$P = \alpha_1 P_1 + \ldots + \alpha_n P_n, \quad \alpha_1, \ldots, \alpha_n \in [0,1], \; \alpha_1 + \ldots + \alpha_n = 1.$$

- **Problematizare:** Aceste caracterizări echivalente nu conduc la un algoritm de determinare a acoperirii convexe.

Acoperire convexă a unei mulțimi finite $\mathcal{P}$: problematizare

- Dacă $\mathcal{P} \subset \mathbb{R}^d$ este finită, acoperirea sa convexă, $\mathrm{Conv}(\mathcal{P})$ este un **politop convex**.

Acoperire convexă a unei mulțimi finite $\mathcal{P}$: problematizare

- Dacă $\mathcal{P} \subset \mathbb{R}^d$ este finită, acoperirea sa convexă, $\mathrm{Conv}(\mathcal{P})$ este un **politop convex**.
- Cazuri particulare: $d = 1$ (segment); $d = 2$ (poligon); $d = 3$ (poliedru).

Acoperire convexă a unei mulțimi finite $\mathcal{P}$: problematizare

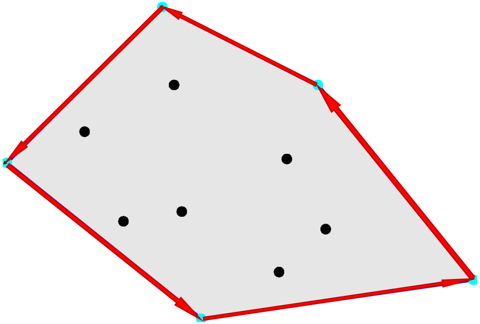
- Dacă $\mathcal{P} \subset \mathbb{R}^d$ este finită, acoperirea sa convexă, $\mathrm{Conv}(\mathcal{P})$ este un **politop convex**.
- Cazuri particulare: $d = 1$ (segment); $d = 2$ (poligon); $d = 3$ (poliedru).
- Cazul $d = 1$: acoperirea convexă este un segment; algoritmic: parcurgere a punctelor (complexitate $O(n)$).

Acoperire convexă a unei mulțimi finite $\mathcal{P}$: problematizare

- Dacă $\mathcal{P} \subset \mathbb{R}^d$ este finită, acoperirea sa convexă, $\mathrm{Conv}(\mathcal{P})$ este un **politop convex**.
- Cazuri particulare: $d = 1$ (segment); $d = 2$ (poligon); $d = 3$ (poliedru).
- Cazul $d = 1$: acoperirea convexă este un segment; algoritmic: parcurgere a punctelor (complexitate $O(n)$).
- În continuare: $d = 2$.

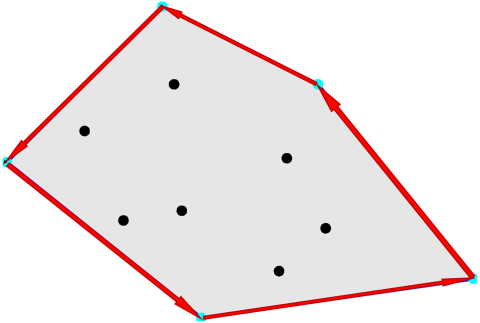
Acoperire convexă a unei mulțimi finite $\mathcal{P}$ (practic)

- De fapt, dacă $\mathcal{P}$ este finită, acoperirea sa convexă, $\mathrm{Conv}(\mathcal{P})$ este un poligon convex.



Acoperire convexă a unei mulțimi finite $\mathcal{P}$ (practic)

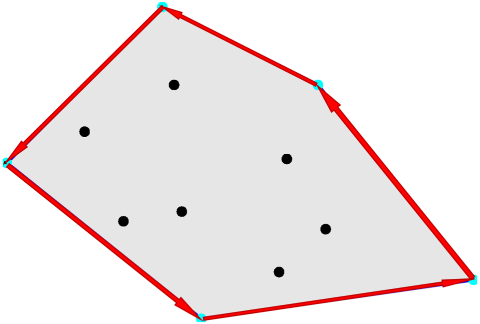
- De fapt, dacă $\mathcal{P}$ este finită, acoperirea sa convexă, $\mathrm{Conv}(\mathcal{P})$ este un poligon convex.



- **Problemă:**
  Cum determinăm, algoritmic, vârfurile acestui poligon?

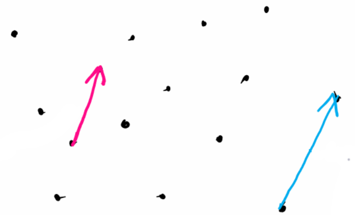
Acoperire convexă a unei mulțimi finite $\mathcal{P}$ (practic)

- De fapt, dacă $\mathcal{P}$ este finită, acoperirea sa convexă, $\mathrm{Conv}(\mathcal{P})$ este un poligon convex.



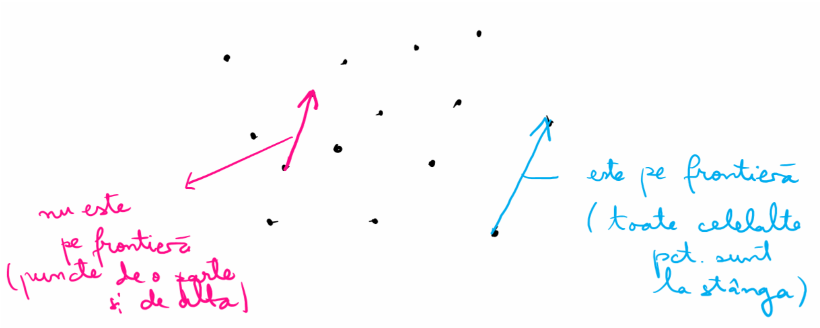
- **Problemă:**
  Cum determinăm, algoritmic, vârfurile acestui poligon?
- **Convenție:** Sensul de parcurgere a frontierei este cel trigonometric.

Un algoritm "lent": idee de lucru



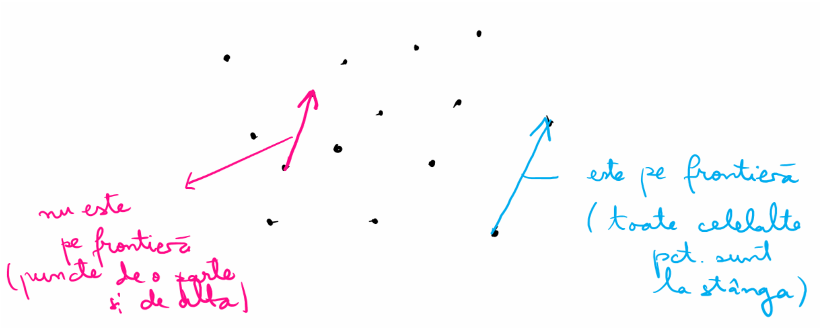
Sunt considerate **muchiile orientate**.

Un algoritm "lent": idee de lucru



Sunt considerate **muchiile orientate**.

Un algoritm "lent": idee de lucru



Sunt considerate **muchiile orientate**.

- **Q:** Cum se decide dacă o muchie orientată fixată este pe **frontieră**?
- **A:** Toate celelalte puncte sunt "în stanga" ei (v. "testul de orientare").

Un algoritm lent

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sensul trigonometric.

1. $E \leftarrow \emptyset$, $\mathcal{L} \leftarrow \emptyset$ /*E este lista muchiilor orientate*/

Un algoritm lent

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sensul trigonometric.

1. $E \leftarrow \emptyset$, $\mathcal{L} \leftarrow \emptyset$ /*E este lista muchiilor orientate*/
2. **for** $(P, Q) \in \mathcal{P} \times \mathcal{P}$ cu $P \neq Q$

Un algoritm lent

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sensul trigonometric.

1. $E \leftarrow \emptyset$, $\mathcal{L} \leftarrow \emptyset$ /*E este lista muchiilor orientate*/
2. **for** $(P, Q) \in \mathcal{P} \times \mathcal{P}$ cu $P \neq Q$
3. **do** *valid* $\leftarrow$ true

Un algoritm lent

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sensul trigonometric.

1. $E \leftarrow \emptyset$, $\mathcal{L} \leftarrow \emptyset$ /*E este lista muchiilor orientate*/
2. **for** $(P, Q) \in \mathcal{P} \times \mathcal{P}$ cu $P \neq Q$
3. **do** *valid* $\leftarrow$ true
4. **for** $R \in \mathcal{P} \setminus \{P, Q\}$

Un algoritm lent

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sensul trigonometric.

1. $E \leftarrow \emptyset$, $\mathcal{L} \leftarrow \emptyset$ /*E este lista muchiilor orientate*/
2. **for** $(P, Q) \in \mathcal{P} \times \mathcal{P}$ cu $P \neq Q$
3.   **do** *valid* $\leftarrow$ true
4.   **for** $R \in \mathcal{P} \setminus \{P, Q\}$
5.   **do if** R "în dreapta" lui $\overrightarrow{PQ}$

Un algoritm lent

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sensul trigonometric.

1. $E \leftarrow \emptyset$, $\mathcal{L} \leftarrow \emptyset$ /*E este lista muchiilor orientate*/
2. **for** $(P, Q) \in \mathcal{P} \times \mathcal{P}$ cu $P \neq Q$
3.   **do** *valid* $\leftarrow$ true
4.   **for** $R \in \mathcal{P} \setminus \{P, Q\}$
5.   **do if** R "în dreapta" lui $\overrightarrow{PQ}$
6.   **then** *valid* $\leftarrow$ false

Un algoritm lent

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sensul trigonometric.

1. $E \leftarrow \emptyset$, $\mathcal{L} \leftarrow \emptyset$ /*E este lista muchiilor orientate*/
2. **for** $(P, Q) \in \mathcal{P} \times \mathcal{P}$ cu $P \neq Q$
3.   **do** *valid* $\leftarrow$ true
4.   **for** $R \in \mathcal{P} \setminus \{P, Q\}$
5.   **do if** R "în dreapta" lui $\overrightarrow{PQ}$
6.   **then** *valid* $\leftarrow$ false
7.   **if** *valid*=true **then** $E = E \cup \{\overrightarrow{PQ}\}$

Un algoritm lent

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sensul trigonometric.

1. $E \leftarrow \emptyset$, $\mathcal{L} \leftarrow \emptyset$ /*E este lista muchiilor orientate*/
2. **for** $(P, Q) \in \mathcal{P} \times \mathcal{P}$ cu $P \neq Q$
3.   **do** *valid* $\leftarrow$ true
4.   **for** $R \in \mathcal{P} \setminus \{P, Q\}$
5.   **do if** R "în dreapta" lui $\overrightarrow{PQ}$
6.   **then** *valid* $\leftarrow$ false
7.   **if** *valid*=true **then** $E = E \cup \{\overrightarrow{PQ}\}$
8. din E se construiește lista $\mathcal{L}$ a vârfurilor acoperirii convexe /*este necesar ca E să fie **coerentă***/

Algoritmul "lent": comentarii

- Complexitatea: $O(n^3)$

Algoritmul "lent": comentarii

- Complexitatea: $O(n^3)$
- Complexitate algebrică: polinoame de gradul II

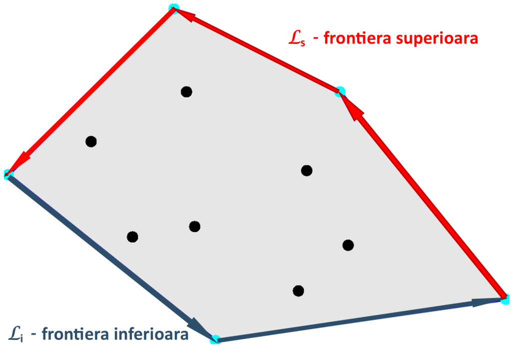
Algoritmul "lent": comentarii

- Complexitatea: $O(n^3)$
- Complexitate algebrică: polinoame de gradul II
- Tratarea cazurilor degenerate: poate fi adaptat.

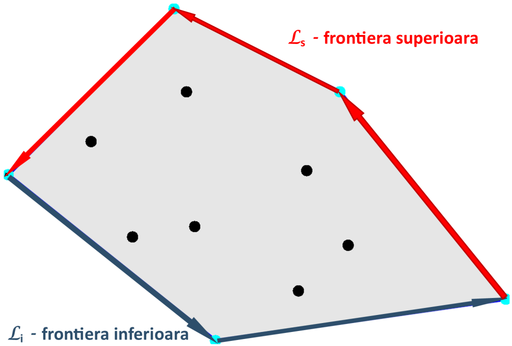
Algoritmul "lent": comentarii

- Complexitatea: $O(n^3)$
- Complexitate algebrică: polinoame de gradul II
- Tratarea cazurilor degenerate: poate fi adaptat.
- Robustețea: datorită erorilor de rotunjire este posibil ca algoritmul să nu returneze o listă coerentă de muchii.

Graham's scan, varianta Andrew: idee de lucru

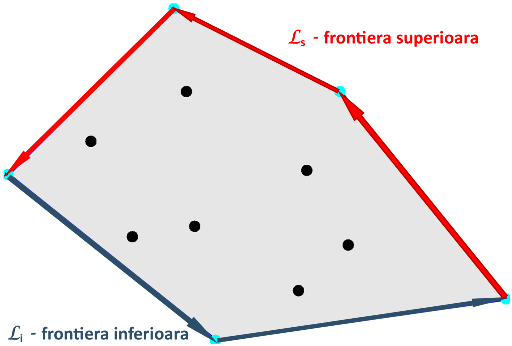


Graham's scan, varianta Andrew: idee de lucru



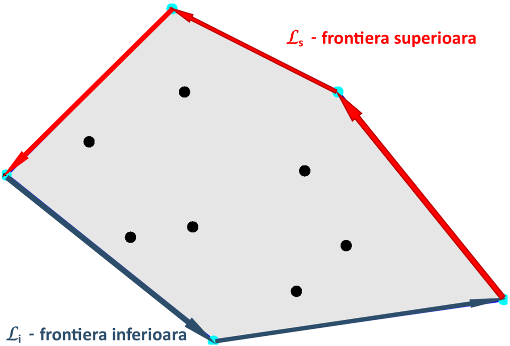
- Punctele sunt mai întâi sortate și renumerotate **lexicografic**.

Graham's scan, varianta Andrew: idee de lucru



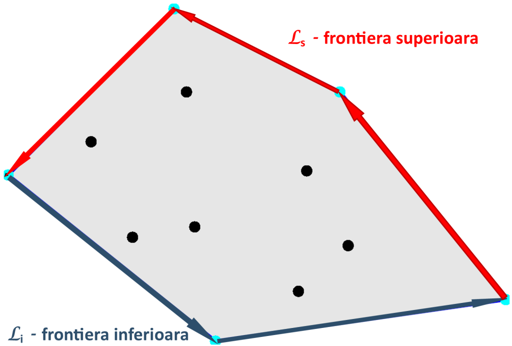
- Punctele sunt mai întâi sortate și renumerotate **lexicografic**.
- Algoritmul este de tip **incremental**, punctele fiind adăugate unul câte unul, fiind apoi eliminate anumite puncte.

Graham's scan, varianta Andrew: idee de lucru

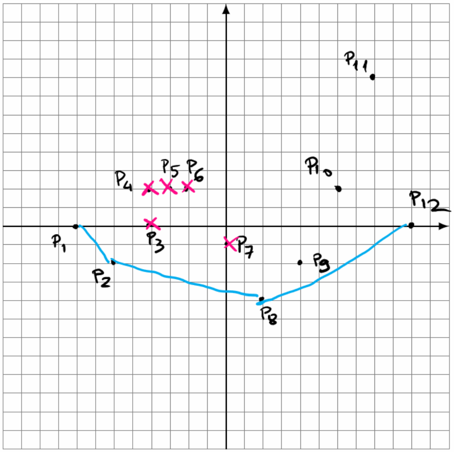


- Punctele sunt mai întâi sortate și renumerotate **lexicografic**.
- Algoritmul este de tip **incremental**, punctele fiind adăugate unul câte unul, fiind apoi eliminate anumite puncte.
- **Q:** Cum se decide dacă trei puncte sunt vârfuri consecutive ale acoperirii convexe?

Graham's scan, varianta Andrew: idee de lucru



- Punctele sunt mai întâi sortate și renumerotate **lexicografic**.
- Algoritmul este de tip **incremental**, punctele fiind adăugate unul câte unul, fiind apoi eliminate anumite puncte.
- **Q:** Cum se decide dacă trei puncte sunt vârfuri consecutive ale acoperirii convexe?
- **A:** Se efectuează un "viraj la stânga" în punctul din mijloc.



Cum evoluează $\mathcal{L}_i$ pe parcursul Graham's scan - varianta Andrew.

$P_1 P_2$

$P_1 P_2 P_3$

$P_1 P_2 P_3 \cancel{P_4} P_5$ viraj la dreapta în P_4

$P_1 P_2 \cancel{P_3} \cancel{P_5} P_6$ viraj la dreapta în P_5

$P_2, P_3 P_6$ coliniare

$P_1 P_2 \cancel{P_6} P_7$ viraj dr. P_6

$P_1 P_2 \cancel{P_7} P_8$ viraj dr. P_7

$P_1 P_2 P_8 \cancel{P_9} \cancel{P_{10}} \cancel{P_{11}} P_{12}$ viraje dr. P_{11}, P_{10}, P_9

$\Downarrow$

$\boxed{P_1 P_2 P_8 P_{12}}$ frontiera inferioară

$P_1 = (-8, 0)$; $P_2 = (-6, -2)$; $P_3 = (-4, 0)$;
$P_4 = (-4, 2)$; $P_5 = (-3, 2)$; $P_6 = (-2, 2)$;
$P_7 = (0, -1)$; $P_8 = (2, -4)$; $P_9 = (4, -2)$
$P_{10} = (6, 2)$; $P_{11} = (8, 8)$, $P_{12} = (10, 0)$

Graham's scan, varianta Andrew (algoritm)

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Sortare lexicografică, renumerotare $P_1, P_2, \ldots, P_n$ conform ordonării

Graham's scan, varianta Andrew (algoritm)

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Sortare lexicografică, renumerotare $P_1, P_2, \ldots, P_n$ conform ordonării
2. $\mathcal{L} \leftarrow (P_1, P_2)$

Graham's scan, varianta Andrew (algoritm)

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Sortare lexicografică, renumerotare $P_1, P_2, \ldots, P_n$ conform ordonării
2. $\mathcal{L} \leftarrow (P_1, P_2)$
3. **for** $i \leftarrow 3$ **to** n

Graham's scan, varianta Andrew (algoritm)

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Sortare lexicografică, renumerotare $P_1, P_2, \ldots, P_n$ conform ordonării
2. $\mathcal{L} \leftarrow (P_1, P_2)$
3. **for** $i \leftarrow 3$ **to** n
4. **do** adaugă P_i la sfârșitul lui $\mathcal{L}$

Graham's scan, varianta Andrew (algoritm)

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Sortare lexicografică, renumerotare $P_1, P_2, \ldots, P_n$ conform ordonării
2. $\mathcal{L} \leftarrow (P_1, P_2)$
3. **for** $i \leftarrow 3$ **to** n
4.   **do** adaugă P_i la sfârșitul lui $\mathcal{L}$
5.   **while** $\mathcal{L}$ are mai mult de două puncte
        and ultimele trei nu determină un viraj la stânga

Graham's scan, varianta Andrew (algoritm)

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Sortare lexicografică, renumerotare $P_1, P_2, \ldots, P_n$ conform ordonării
2. $\mathcal{L} \leftarrow (P_1, P_2)$
3. **for** $i \leftarrow 3$ **to** n
4. **do** adaugă P_i la sfârșitul lui $\mathcal{L}$
5. **while** $\mathcal{L}$ are mai mult de două puncte
   and ultimele trei nu determină un viraj la stânga
6. **do** șterge penultimul punct

Graham's scan, varianta Andrew (algoritm)

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Sortare lexicografică, renumerotare $P_1, P_2, \ldots, P_n$ conform ordonării
2. $\mathcal{L} \leftarrow (P_1, P_2)$
3. **for** $i \leftarrow 3$ **to** n
4.   **do** adaugă P_i la sfârșitul lui $\mathcal{L}$
5.   **while** $\mathcal{L}$ are mai mult de două puncte
           and ultimele trei nu determină un viraj la stânga
6.   **do** șterge penultimul punct
7. **return** $\mathcal{L}_i$

Graham's scan, varianta Andrew (algoritm)

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Sortare lexicografică, renumerotare $P_1, P_2, \ldots, P_n$ conform ordonării
2. $\mathcal{L} \leftarrow (P_1, P_2)$
3. **for** $i \leftarrow 3$ **to** n
4. **do** adaugă P_i la sfârșitul lui $\mathcal{L}$
5. **while** $\mathcal{L}$ are mai mult de două puncte
   and ultimele trei nu determină un viraj la stânga
6. **do** șterge penultimul punct
7. **return** $\mathcal{L}_i$
8. Parcurge pași analogi pentru a determina $\mathcal{L}_s$

Graham's scan, varianta Andrew (algoritm)

Input: O mulțime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Sortare lexicografică, renumerotare $P_1, P_2, \ldots, P_n$ conform ordonării
2. $\mathcal{L} \leftarrow (P_1, P_2)$
3. **for** $i \leftarrow 3$ **to** n
4. **do** adaugă P_i la sfârșitul lui $\mathcal{L}$
5. **while** $\mathcal{L}$ are mai mult de două puncte
   and ultimele trei <u>nu</u> determină un viraj la stânga
6. **do** șterge penultimul punct
7. **return** $\mathcal{L}_i$
8. Parcurge pași analogi pentru a determina $\mathcal{L}_s$
9. Concatenează $\mathcal{L}_i$ și $\mathcal{L}_s$

Graham's scan, varianta Andrew: comentarii

- Complexitatea: $O(n \log n)$.

Graham's scan, varianta Andrew: comentarii

- Complexitatea: $O(n \log n)$.
- Tratarea cazurilor degenerate: corect.

Graham's scan, varianta Andrew: comentarii

- Complexitatea: $O(n \log n)$.
- Tratarea cazurilor degenerate: corect.
- Robustețea: datorită erorilor de rotunjire este posibil ca algoritmul să returneze o listă eronată (dar coerentă) de muchii.

Jarvis' march / Jarvis' wrap [1973]

- Algoritm de tip **incremental**. Nu necesită sortare prealabilă.

Jarvis' march / Jarvis' wrap [1973]

- Algoritm de tip **incremental**. Nu necesită sortare prealabilă.
- Inițializare: un punct care este sigur un vârf al acoperirii convexe (e.g. punctul cel mai de jos / din stânga / stânga jos).

Jarvis' march / Jarvis' wrap [1973]

- Algoritm de tip **incremental**. Nu necesită sortare prealabilă.
- Inițializare: un punct care este sigur un vârf al acoperirii convexe (e.g. punctul cel mai de jos / din stânga / stânga jos).
- Lista se actualizează prin determinarea succesorului: "cel mai la dreapta" punct.

Jarvis' march / Jarvis' wrap [1973]

- Algoritm de tip **incremental**. Nu necesită sortare prealabilă.
- Inițializare: un punct care este sigur un vârf al acoperirii convexe (e.g. punctul cel mai de jos / din stânga / stânga jos).
- Lista se actualizează prin determinarea succesorului: "cel mai la dreapta" punct.
- Implementare: două abordări (i) ordonare; (ii) testul de orientare.

Jarvis' march / Jarvis' wrap [1973]

- Algoritm de tip **incremental**. Nu necesită sortare prealabilă.
- Inițializare: un punct care este sigur un vârf al acoperirii convexe (e.g. punctul cel mai de jos / din stânga / stânga jos).
- Lista se actualizează prin determinarea succesorului: "cel mai la dreapta" punct.
- Implementare: două abordări (i) ordonare; (ii) testul de orientare.
- Complexitate: $O(hn)$, unde h este numărul punctelor de pe frontiera acoperirii convexe.

Jarvis' march / Jarvis' wrap [1973]

- Algoritm de tip **incremental**. Nu necesită sortare prealabilă.
- Inițializare: un punct care este sigur un vârf al acoperirii convexe (e.g. punctul cel mai de jos / din stânga / stânga jos).
- Lista se actualizează prin determinarea succesorului: "cel mai la dreapta" punct.
- Implementare: două abordări (i) ordonare; (ii) testul de orientare.
- Complexitate: $O(hn)$, unde h este numărul punctelor de pe frontiera acoperirii convexe.
- Algoritmul lui Chan "combină" ideile celor doi algoritmi, ajungând la complexitatea-timp $O(n \log h)$.

Jarvis' march (algoritm)

Input: O mulțime de puncte necoliniare $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ din $\mathbf{R}^2$ ($n \geq 3$).
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Determinarea unui punct din $\mathcal{P}$ care aparține frontierei (de exemplu cel mai mic, folosind ordinea lexicografică); acest punct este notat cu A_1.

Jarvis' march (algoritm)

Input: O mulţime de puncte necoliniare $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ din $\mathbf{R}^2$ ($n \geq 3$).
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Determinarea unui punct din $\mathcal{P}$ care aparține frontierei (de exemplu cel mai mic, folosind ordinea lexicografică); acest punct este notat cu A_1.
2. $k \leftarrow 1$; $\mathcal{L} \leftarrow (A_1)$; *valid* $\leftarrow$ true

Jarvis' march (algoritm)

Input: O mulțime de puncte necoliniare $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ din $\mathbf{R}^2$ ($n \geq 3$).
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Determinarea unui punct din $\mathcal{P}$ care aparține frontierei (de exemplu cel mai mic, folosind ordinea lexicografică); acest punct este notat cu A_1.
2. $k \leftarrow 1$; $\mathcal{L} \leftarrow (A_1)$; *valid* $\leftarrow$ true
3. **while** *valid*$=$ true

Jarvis' march (algoritm)

Input: O mulțime de puncte necoliniare $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ din $\mathbf{R}^2$ ($n \geq 3$).
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Determinarea unui punct din $\mathcal{P}$ care aparține frontierei (de exemplu cel mai mic, folosind ordinea lexicografică); acest punct este notat cu A_1.
2. $k \leftarrow 1$; $\mathcal{L} \leftarrow (A_1)$; *valid* $\leftarrow$ true
3. **while** *valid*= true
4.   **do** alege un pivot arbitrar $S \in \mathcal{P}$, diferit de A_k

Jarvis' march (algoritm)

Input: O mulțime de puncte necoliniare $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ din $\mathbf{R}^2$ ($n \geq 3$).
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Determinarea unui punct din $\mathcal{P}$ care aparține frontierei (de exemplu cel mai mic, folosind ordinea lexicografică); acest punct este notat cu A_1.
2. $k \leftarrow 1$; $\mathcal{L} \leftarrow (A_1)$; *valid* $\leftarrow$ true
3. **while** *valid*= true
4.   **do** alege un pivot arbitrar $S \in \mathcal{P}$, diferit de A_k
5.   **for** $i \leftarrow 1$ **to** n

Jarvis' march (algoritm)

Input: O mulțime de puncte necoliniare $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ din $\mathbf{R}^2$ ($n \geq 3$).
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Determinarea unui punct din $\mathcal{P}$ care aparține frontierei (de exemplu cel mai mic, folosind ordinea lexicografică); acest punct este notat cu A_1.
2. $k \leftarrow 1$; $\mathcal{L} \leftarrow (A_1)$; *valid* $\leftarrow$ true
3. **while** *valid*= true
4.   **do** alege un pivot arbitrar $S \in \mathcal{P}$, diferit de A_k
5.   **for** $i \leftarrow 1$ **to** n
6.   **do if** P_i este la dreapta muchiei orientate $A_k S$

Jarvis' march (algoritm)

Input: O mulțime de puncte necoliniare $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ din $\mathbf{R}^2$ ($n \geq 3$).
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Determinarea unui punct din $\mathcal{P}$ care aparține frontierei (de exemplu cel mai mic, folosind ordinea lexicografică); acest punct este notat cu A_1.
2. $k \leftarrow 1$; $\mathcal{L} \leftarrow (A_1)$; *valid* $\leftarrow$ true
3. **while** *valid*= true
4.   **do** alege un pivot arbitrar $S \in \mathcal{P}$, diferit de A_k
5.   **for** $i \leftarrow 1$ **to** n
6.   **do if** P_i este la dreapta muchiei orientate A_kS
7.   **then** $S \leftarrow P_i$

Jarvis' march (algoritm)

Input: O mulțime de puncte necoliniare $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ din $\mathbf{R}^2$ $(n \geq 3)$.
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Determinarea unui punct din $\mathcal{P}$ care aparține frontierei (de exemplu cel mai mic, folosind ordinea lexicografică); acest punct este notat cu A_1.
2. $k \leftarrow 1$; $\mathcal{L} \leftarrow (A_1)$; *valid* $\leftarrow$ true
3. **while** *valid*$=$ true
4.   **do** alege un pivot arbitrar $S \in \mathcal{P}$, diferit de A_k
5.   **for** $i \leftarrow 1$ **to** n
6.   **do if** P_i este la dreapta muchiei orientate $A_k S$
7.   **then** $S \leftarrow P_i$
8.   **if** $S \neq A_1$

Jarvis' march (algoritm)

Input: O mulțime de puncte necoliniare $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ din $\mathbf{R}^2$ ($n \geq 3$).
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Determinarea unui punct din $\mathcal{P}$ care aparține frontierei (de exemplu cel mai mic, folosind ordinea lexicografică); acest punct este notat cu A_1.
2. $k \leftarrow 1$; $\mathcal{L} \leftarrow (A_1)$; *valid* $\leftarrow$ true
3. **while** *valid*= true
4.   **do** alege un pivot arbitrar $S \in \mathcal{P}$, diferit de A_k
5.   **for** $i \leftarrow 1$ **to** n
6.   **do if** P_i este la dreapta muchiei orientate $A_k S$
7.   **then** $S \leftarrow P_i$
8.   **if** $S \neq A_1$
9.   **then** $k \leftarrow k + 1$;
   $A_k = S$
   adaugă A_k la $\mathcal{L}$

Jarvis' march (algoritm)

Input: O mulțime de puncte necoliniare $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ din $\mathbf{R}^2$ ($n \geq 3$).
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Determinarea unui punct din $\mathcal{P}$ care aparține frontierei (de exemplu cel mai mic, folosind ordinea lexicografică); acest punct este notat cu A_1.
2. $k \leftarrow 1$; $\mathcal{L} \leftarrow (A_1)$; *valid* $\leftarrow$ true
3. **while** *valid* $=$ true
4.   **do** alege un pivot arbitrar $S \in \mathcal{P}$, diferit de A_k
5.   **for** $i \leftarrow 1$ **to** n
6.   **do if** P_i este la dreapta muchiei orientate $A_k S$
7.   **then** $S \leftarrow P_i$
8.   **if** $S \neq A_1$
9.   **then** $k \leftarrow k + 1$;
                   $A_k = S$
                   adaugă A_k la $\mathcal{L}$
10.   **else** *valid* $\leftarrow$ false

Jarvis' march (algoritm)

Input: O mulțime de puncte necoliniare $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ din $\mathbf{R}^2$ ($n \geq 3$).
Output: O listă $\mathcal{L}$ care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Determinarea unui punct din $\mathcal{P}$ care aparține frontierei (de exemplu cel mai mic, folosind ordinea lexicografică); acest punct este notat cu A_1.
2. $k \leftarrow 1$; $\mathcal{L} \leftarrow (A_1)$; *valid* $\leftarrow$ true
3. **while** *valid*= true
4.   **do** alege un pivot arbitrar $S \in \mathcal{P}$, diferit de A_k
5.   **for** $i \leftarrow 1$ **to** n
6.   **do if** P_i este la dreapta muchiei orientate $A_k S$
7.   **then** $S \leftarrow P_i$
8.   **if** $S \neq A_1$
9.   **then** $k \leftarrow k + 1$;
               $A_k = S$
               adaugă A_k la $\mathcal{L}$
10.   **else** *valid* $\leftarrow$ false
11. **return** $\mathcal{L}$

Alte direcții de lucru

- **Aplicații:** grafică pe calculator, robotică, GIS, recunoașterea formelor, gestionarea bazelor de date multi-dimensionale, etc.

Alte direcții de lucru

- **Aplicații:** grafică pe calculator, robotică, GIS, recunoașterea formelor, gestionarea bazelor de date multi-dimensionale, etc.
- Pot fi stabilite legături cu algoritmi studiați în alt context. De exemplu: *Traveling Salesman Problem*, în context euclidian (costurile sunt date de distanțele dintre puncte). În acest caz, ordinea în care nodurile de pe frontieră apar în traseul optim coincide cu ordinea în care acestea apar în parcurgerea frontierei acoperirii convexe - exemplu.

Alte direcții de lucru

- **Aplicații:** grafică pe calculator, robotică, GIS, recunoașterea formelor, gestionarea bazelor de date multi-dimensionale, etc.
- Pot fi stabilite legături cu algoritmi studiați în alt context. De exemplu: *Traveling Salesman Problem*, în context euclidian (costurile sunt date de distanțele dintre puncte). În acest caz, ordinea în care nodurile de pe frontieră apar în traseul optim coincide cu ordinea în care acestea apar în parcurgerea frontierei acoperirii convexe - exemplu.
- Algoritmi pentru spații euclidiene de dimensiune $m \geq 3$.

Alte direcții de lucru

- **Aplicații:** grafică pe calculator, robotică, GIS, recunoașterea formelor, gestionarea bazelor de date multi-dimensionale, etc.
- Pot fi stabilite legături cu algoritmi studiați în alt context. De exemplu: *Traveling Salesman Problem*, în context euclidian (costurile sunt date de distanțele dintre puncte). În acest caz, ordinea în care nodurile de pe frontieră apar în traseul optim coincide cu ordinea în care acestea apar în parcurgerea frontierei acoperirii convexe - exemplu.
- Algoritmi pentru spații euclidiene de dimensiune $m \geq 3$.
- Algoritmi eficienți pentru determinarea acoperirii convexe pentru vârfurile unui poligon arbitrar.

Alte direcții de lucru

- **Aplicații:** grafică pe calculator, robotică, GIS, recunoașterea formelor, gestionarea bazelor de date multi-dimensionale, etc.
- Pot fi stabilite legături cu algoritmi studiați în alt context. De exemplu: *Traveling Salesman Problem*, în context euclidian (costurile sunt date de distanțele dintre puncte). În acest caz, ordinea în care nodurile de pe frontieră apar în traseul optim coincide cu ordinea în care acestea apar în parcurgerea frontierei acoperirii convexe - exemplu.
- Algoritmi pentru spații euclidiene de dimensiune $m \geq 3$.
- Algoritmi eficienți pentru determinarea acoperirii convexe pentru vârfurile unui poligon arbitrar.
- Algoritmi dinamici (on-line, real-time, convex hull maintenance).