

Geometrie computațională – suport de curs

Mihai-Sorin Stupariu

Sem. I, 2017-2018

Cuprins

1	Preliminarii	3
1.1	Concepte de algebră liniară, geometrie afină și euclidiană	3
1.2	Raportul unor puncte coliniare	3
1.3	Coordonate carteziene și coordonate polare	5
1.4	Testul de orientare	5
1.5	Exerciții, probleme, aplicații	6
2	Acoperiri convexe	8
2.1	Generalități	8
2.2	Algoritmi lenți (naivi)	9
2.3	Algoritmi ”clasici”	11
2.3.1	Algoritmi incremental: Graham’s scan, Jarvis’ march . .	11
2.4	Aplicație - determinarea punctelor antipodale	13
2.5	Exerciții, probleme, aplicații	14
3	Triangulări	15
3.1	Triangularea poligoanelor. Problema galeriei de artă	15
3.2	Triangularea unei mulțimi arbitrare de puncte	18
3.3	Triangulări unghiular optime	18
3.4	Exerciții, probleme, aplicații	20
4	Intersecții	22
4.1	Intersecția segmentelor (generalități)	22
4.2	Metoda liniei de baleiere	23
4.3	Alte rezultate	25
4.4	Suprapunerea straturilor tematice	25
4.5	Exerciții, probleme, aplicații	27
5	Elemente de programare liniară	29
5.1	Motivație: turnarea pieselor în matrite	29
5.2	Intersecții de semiplane	31
5.3	Dualitate	31
5.4	Programare liniară	32
5.5	Exerciții, probleme, aplicații	33

5.6	Anexa	35
6	Diagrame Voronoi	36
6.1	Generalități	36
6.2	Proprietăți	36
6.3	Diagrame Voronoi și triangulări Delaunay	37
6.4	Un algoritm eficient	38
6.5	Exerciții, probleme, aplicații	39
7	Probleme de căutare și localizare	40
7.1	Căutare ortogonală	40
7.2	Localizarea punctelor – Hărți trapezoidale	43
7.3	Aplicație: mișcarea unui robot-punct	47
7.4	Exerciții, probleme, aplicații	48
	Bibliografie	49
A	Proiecte	50

Capitolul 1

Preliminarii

1.1 Concepte de algebră liniară, geometrie afină și euclidiană

Noțiuni de algebră liniară: spațiu vectorial, vector, combinație liniară, liniar (in)dependentă, sistem de generatori, bază, reper, dimensiune a unui spațiu vectorial, componentele unui vector într-un reper, matrice de trecere între repere, repere orientate la fel (opus), reper drept (strâmb), produs scalar, norma unui vector, versorul unui vector nenul, spațiu vectorial euclidian, vectori ortogonali, bază ortonormată, reper ortonormat.

Noțiuni de geometrie afină: vectorul determinat de două puncte, combinație afină, afin (in)dependentă, acoperirea afină a unei mulțimi de puncte, dreapta determinată de două puncte distincte, reper cartezian, coordonatele unui punct într-un reper cartezian, sistem de axe asociat unui reper cartezian din \mathbb{R}^n , raportul a trei puncte coliniare (detalii în secțiunea 1.2), segmentul determinat de două puncte, mulțime convexă, închiderea (înfășurătoarea) convexă a unei mulțimi, aplicație afină (exemple: translație, omotetie, proiecție, simetrie).

Noțiuni de geometrie euclidiană: distanța dintre două puncte, reper cartezian ortonormat, izometrie, proiecție centrală.

Detalii pot fi găsite în [6], [8], [13] [14].

1.2 Raportul unor puncte coliniare

Lema 1.1 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 $P \in AB$.

Definiția 1.2 Scalarul r definit în lema 1.1 se numește **raportul** punctelor A, B, P (sau **raportul în care punctul P împarte segmentul $[AB]$**) și este notat cu $r(A, P, B)$.

Observația 1.3 În calcularea raportului, ordinea punctelor este esențială. Modul în care este definită această noțiune (mai precis ordinea în care sunt considerate punctele) diferă de la autor la autor.

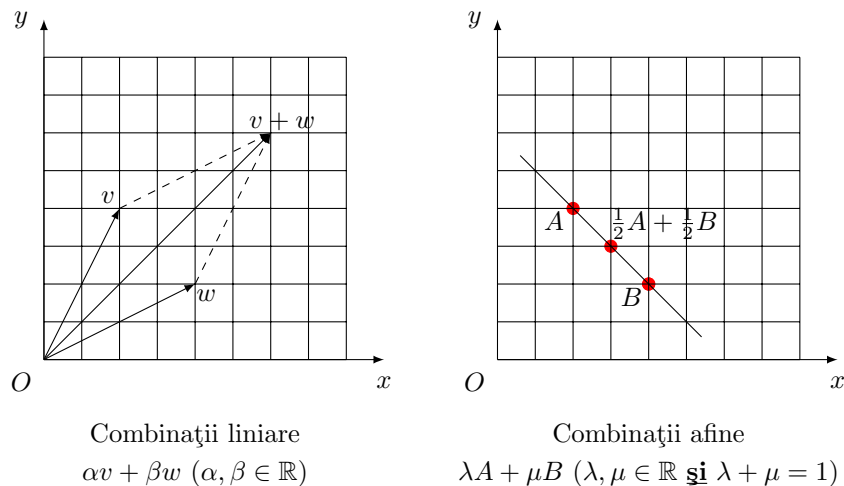


Figura 1.1: Vectori și puncte: combinații liniare și combinații afine

Exemplul 1.4 (i) 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 $r(A, C, B) = -\frac{1}{2}$, $r(B, C, A) = -2$, $r(C, A, B) = 1$, $r(C, B, A) = -2$.

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

Propoziția 1.5 Fie A, B, P trei puncte coliniare, cu $P \neq B$. Atunci:

- (i) $P = \frac{1}{r+1}A + \frac{r}{r+1}B$, unde $r = r(A, P, B)$;
- (ii) $P = (1 - \alpha)A + \alpha B$ dacă și numai dacă $r(A, P, B) = \frac{\alpha}{1-\alpha}$;
- (iii) $P = \frac{\alpha}{\alpha+\beta}A + \frac{\beta}{\alpha+\beta}B$ dacă și numai dacă $r(A, P, B) = \frac{\beta}{\alpha}$.

Observația 1.6 Fie $P \in AB \setminus \{A, B\}$. Atunci:

- (i) $r(A, P, B) > 0$ dacă și numai dacă $P \in (AB)$;
- (ii) $r(B, P, A) = \frac{1}{r(A, P, B)}$.

1.3 Coordonate carteziene și coordonate polare

Coordonate carteziene (x, y) și coordonate polare (ρ, θ) (pentru puncte din planul \mathbb{R}^2 pentru care relațiile au sens), Figura 1.2:

$$\begin{cases} x = \rho \cos \theta \\ y = \rho \sin \theta \end{cases} \quad \begin{cases} \rho = \sqrt{x^2 + y^2} \\ \theta = \arctg \frac{y}{x} \end{cases}$$

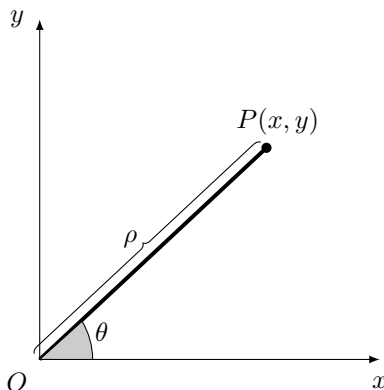


Figura 1.2: Punctul P are coordonatele carteziene (x, y) și coordonatele polare (ρ, θ) .

1.4 Testul de orientare

- Fie vectorii $v = (v_1, v_2, v_3), w = (w_1, w_2, w_3) \in \mathbf{R}^3$.
Produsul vectorial $v \times w$ se calculează dezvoltând **determinantul formal**

$$v \times w = \begin{vmatrix} v_1 & w_1 & e_1 \\ v_2 & w_2 & e_2 \\ v_3 & w_3 & e_3 \end{vmatrix}$$

- **Notăție** 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. Notăm

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

- **Lemă.** Fie P, Q, R puncte din $\mathbf{R}^2 \simeq \{x \in \mathbf{R}^3 | x_3 = 0\}$. Atunci

$$\overrightarrow{PQ} \times \overrightarrow{PR} = (0, 0, \Delta(P, Q, R)).$$

Propoziția 1.7 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

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

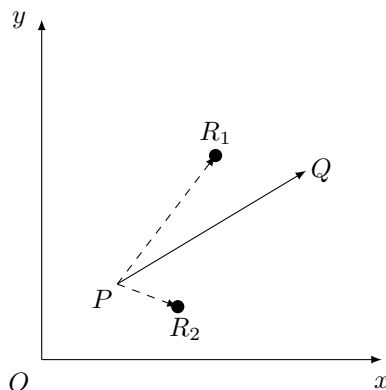


Figura 1.3: Poziția relativă a două puncte față de un vector / o muchie orientată

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 $\overrightarrow{PQ} \Leftrightarrow \Delta(P, Q, R) > 0$.

Observația 1.8 Testul de orientare se bazează pe calculul unui polinom de gradul II ($\Delta(P, Q, R)$).

Aplicații.

- dacă un punct este în dreapta / stânga unei muchii orientate;
- natura unui viraj în parcurgerea unei linii poligonale (la dreapta / la stânga);
- natura unui poligon (convex / concav);
- dacă două puncte sunt de o parte și de alta a unui segment / a unei drepte.

1.5 Exerciții, probleme, aplicații

Exercițiul 1.9 Calculați rapoartele $r(A, P, B)$, $r(B, P, A)$, $r(P, A, B)$ (stabiliți mai întâi dacă punctele sunt coliniare), pentru: (i) $A = (3, 3)$, $B = (2, 4)$, $C = (5, 1)$; (ii) $A = (1, 4, -2)$, $P = (2, 3, -1)$, $B = (4, 1, 1)$.

Exercițiul 1.10 Determinați α, β astfel ca punctele A, P, B din planul \mathbb{R}^2 , cu $A = (6, 2)$, $P = (\alpha, \beta)$, $B = (2, -2)$, să fie coliniare și $r(A, P, B) = 2$.

Exercițiul 1.11 Determinați coordonatele carteziene ale punctului M de coordonate polare $\rho = 6$; $\theta = \frac{\pi}{6}$, respectiv coordonatele polare ale punctului $N(-4, 4)$.

Exercițiul 1.12 Ordonati punctele $A = (2, 0)$, $B = (3, 2)$, $C = (4, 2)$, $D = (0, -4)$, $E = (-3, 1)$, $F = (0, 5)$, $G = (-1, -1)$, $H = (0, 4)$ (i) folosind coordonate carteziene; (ii) folosind coordonate polare.

Exercițiul 1.13 Calculați produsul vectorial $v \times w$ pentru vectorii $v = (1, -1, 0)$, $w = (-2, 1, 3)$.

Exercițiul 1.14 Fie $v, w \in \mathbb{R}^3$ doi vectori necoliniari. Folosind produsul vectorial, construiți o bază ortonormată $\{b_1, b_2\}$ a planului π generat de vectorii v și w , astfel ca b_1 să fie coliniar cu v .

Exercițiul 1.15 Fie $P = (2, 2), Q = (4, 4)$. Stabiliți, folosind testul de orientare, poziția relativă a punctelor $R_1 = (8, 8), R_2 = (6, 0), R_3 = (-2, -1)$ față de muchia orientată \overrightarrow{PQ} . Care este poziția acelorași puncte față de muchia orientată \overrightarrow{QP} ?

Exercițiul 1.16 Dați exemplul de puncte coplanare P, Q, R_1, R_2 din \mathbb{R}^3 , nesituate într-un plan de coordonate, astfel ca R_1 și R_2 să fie de o parte și de alta a segmentului $[PQ]$.

Exercițiul 1.17 Fie MNP un triunghi cu vârfurile $M = (x_M, y_M), N = (x_N, y_N), P = (x_P, y_P)$ și fie δ o dreaptă de ecuație $ax + by + c = 0$. Stabiliți și justificați care este complexitatea algebrică a calculelor pentru:

- a) a stabili dacă dreapta intersectează laturile triunghiului;
- b) a stabili dacă dreapta trece prin centrul de greutate al triunghiului.

Exercițiul 1.18 Fie ABC un triunghi cu vârfurile $A = (x_A, y_A), B = (x_B, y_B), C = (x_C, y_C)$ și fie $P = (x_P, y_P), Q = (x_Q, y_Q)$ alte două puncte, distincte, din plan. Care este complexitatea algebrică a calculelor pentru:

- a) a stabili dacă dreapta PQ este paralelă cu una dintre laturile triunghiului ABC ;
- b) a stabili dacă punctul P coincide cu centrul cercului circumscris triunghiului ABC .

Capitolul 2

Acoperiri convexe

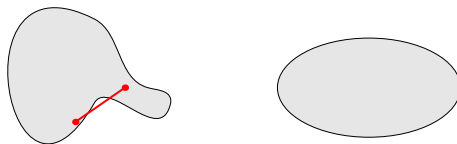
2.1 Generalități

Conceptul de mulțime convexă:

Definiția 2.1 (i) Pentru $P, Q \in \mathbf{R}^d$, segmentul $[PQ]$ este mulțimea combinațiilor convexe dintre P și Q :

$$[PQ] = \{(1-t)P + tQ | t \in [0, 1]\} = \{\alpha P + \beta Q | \alpha, \beta \in [0, 1], \alpha + \beta = 1\}.$$

(ii) O mulțime $M \subset \mathbf{R}^d$ este **convexă** dacă oricare ar fi $P, Q \in M$, segmentul $[PQ]$ este inclus în M .



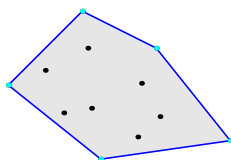
Mulțimea din stânga nu este convexă, întrucât **există** două puncte, pentru care segmentul determinat nu este inclus în mulțime (punctele cu această proprietate nu sunt unice!).

Problematizare:

Mulțimile finite cu cel puțin două elemente nu sunt convexe \rightarrow necesară **acoperirea convexă**.

Acoperire convexă a unei mulțimi finite \mathcal{P} : 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, \dots, P_n este un punct P de forma

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

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

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

- Dacă \mathcal{P} este finită, acoperirea sa convexă, $\text{Conv}(\mathcal{P})$ 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:** $d = 2$.
- **Problemă:** Cum determinăm, algoritmic, vârfurile acoperirii convexe pentru o mulțime finită \mathcal{P} din \mathbf{R}^2 (ca mulțime, ca listă)?
- **Două abordări posibile:**
 - Determinarea punctelor extreme.
 - Determinarea muchiilor frontierei acoperirii convexe.

2.2 Algoritmi lenți (naivi)

Determinarea punctelor extreme și ordonarea lor

Definiția 2.2 *Un punct M al unei mulțimi convexe S este punct extrem al lui S dacă nu există $A, B \in S$ cu $A \neq B$ astfel ca $M \in [AB]$.*

Propoziția 2.3 (Caracterizarea punctelor extreme). *Fie \mathcal{P} o mulțime finită și $\text{Conv}(\mathcal{P})$ acoperirea sa convexă. Un punct $M \in \mathcal{P}$ nu este punct extrem \Leftrightarrow este situat într-un triunghi având vârfurile în \mathcal{P} (sau în interiorul acestui triunghi), dar nu este, el însuși, vârf al triunghiului.*

Propoziția 2.4 (Ordonarea punctelor extreme). *Fie \mathcal{P} o mulțime finită și $\text{Conv}(\mathcal{P})$ acoperirea sa convexă. Ordonând punctele extreme ale lui $\text{Conv}(\mathcal{P})$ după unghiul polar (format într-un sistem de coordonate polare având originea într-un punct interior al lui $\text{Conv}(\mathcal{P})$), se obțin vârfurile consecutive ale lui $\text{Conv}(\mathcal{P})$.*

Comentarii

- Cum se stabilește dacă un punct P aparține unui triunghi ABC sau interiorului acestuia? (folosind arii, verificând dacă P situat pe laturi sau situat de aceeași parte a fiecărei laturi ca și vârful opus – ”Testul de orientare”, etc.)

- Coordonate carteziene (x, y) și coordonate polare (ρ, θ) (pentru puncte pentru care relațiile au sens):

$$\begin{cases} x = \rho \cos \theta \\ y = \rho \sin \theta \end{cases} \quad \begin{cases} \rho = \sqrt{x^2 + y^2} \\ \theta = \arctg \frac{y}{x} \end{cases}$$

- Pentru a **ordona** / **sorta** punctele nu este nevoie ca unghiurile polare să fie calculate explicit! Are loc relația $\theta(Q) > \theta(P) \Leftrightarrow Q$ este "în stânga" muchiei orientate \overrightarrow{OP} (v. "Testul de orientare").
- Dacă M_1, \dots, M_q sunt puncte extreme ale lui $\text{Conv}(\mathcal{P})$, atunci centrul de greutate $\frac{1}{q}M_1 + \dots + \frac{1}{q}M_q$ este situat în interiorul $\text{Conv}(\mathcal{P})$.

Algoritmul lent 1

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. $\mathcal{M} \leftarrow \emptyset$ /* \mathcal{M} este mulțimea punctelor extreme*/
2. **for** $P \in \mathcal{P}$
3. **do** $valid \leftarrow \text{true}$
4. **for** $(A, B, C) \in \mathcal{P} \times \mathcal{P} \times \mathcal{P}$ distincte 2×2 , distincte de P
5. **do if** P în interiorul $\triangle ABC$ sau pe laturile sale
6. **then** $valid \leftarrow \text{false}$
7. **if** $valid = \text{true}$ **then** $\mathcal{M} = \mathcal{M} \cup \{P\}$
8. **do** calculează centrul de greutate al lui \mathcal{M}
9. **do** sortează punctele din \mathcal{M} după unghiul polar, obținând lista \mathcal{L}

Comentarii

- Complexitatea: $O(n^4)$ (pașii 1-7: $O(n^4)$; pasul 8: $O(n)$; pasul 9: $O(n \log n)$).
- Complexitatea algebrică: necesare polinoame de gradul II
- Tratează corect cazurile degenerate (dacă A, B, C sunt coliniare pe frontieră, cu $C \in [AB]$, doar A și B sunt detectate ca fiind puncte extreme)!

Determinarea muchiilor frontierei

- Sunt considerate **muchiiile orientate**.
- **Q:** Cum se decide dacă o muchie orientată fixată este pe **frontieră**?
- **A:** Toate celelalte puncte sunt "în stânga" ei (v. "Testul de orientare").

Algoritmul lent 2

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** $valid \leftarrow \text{true}$
4. **for** $R \in \mathcal{P} \setminus \{P, Q\}$
5. **do if** R "în dreapta" lui \overrightarrow{PQ}
6. **then** $valid \leftarrow \text{false}$
7. **if** $valid = \text{true}$ **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*/

Comentarii

- Complexitatea: $O(n^3)$.
- Complexitatea algebrică: necesare polinoame de gradul II
- Tratarea cazurilor degenerate: poate fi adaptat. Pasul 5 trebuie rafinat:
 5. **do if** R "în dreapta" lui \overrightarrow{PQ} **or** (P, Q, R) coliniare **and** $r(P, R, Q) < 0$
 6. **then** $valid \leftarrow \text{false}$
- Robustețea: datorită erorilor de rotunjire este posibil ca algoritmul să nu returneze o listă coerentă de muchii.

2.3 Algoritmi "clasici"

2.3.1 Algoritmi incremental: Graham's scan, Jarvis' march

Graham's scan [1972]

- Punctele sunt mai întâi **sortate** (lexicografic, după unghiul polar și distanța polară) și renumerotate.
- Algoritm de tip **incremental**, punctele fiind adăugate unul câte unul la lista \mathcal{L} a frontierei acoperirii convexe. Pe parcurs, anumite puncte sunt eliminate - actualizare locală a listei vârfurilor acoperirii convexe.
- **Q:** Cum se decide dacă trei puncte sunt vârfuri consecutive ale acoperirii convexe (parcursă în sens trigonometric)?
- **A:** Se efectuează un "viraj la stânga" în punctul din mijloc.

Graham's scan (algorithm)

Input: O mulțime de puncte \mathcal{P} din \mathbb{R}^2 .

Output: O listă \mathcal{L} care conține vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Determinarea unui punct interior din $\text{Conv}(\mathcal{P})$ /* de ex. baricentrul
2. Efectuarea unei translații, punctul interior de la 1. devine originea O

3. Sortare și ordonare în raport cu unghiul polar și distanța polară, renumerotare P_1, P_2, \dots, P_n conform ordonării
4. $\mathcal{L} \leftarrow (P_1, P_2)$
5. **for** $i \leftarrow 3$ **to** n
6. **do** adaugă P_i la sfârșitul lui \mathcal{L}
7. **while** \mathcal{L} are mai mult de două puncte
 and ultimele trei nu determină un viraj la stânga
8. **do** șterge penultimul punct
9. **return** \mathcal{L}

Graham's scan, varianta lui Andrew [1979]

- Punctele sunt mai întâi **sortate** (lexicografic, după coordonatele carteziane) și renumerotate.
- Algoritmul determină două liste, reprezentând marginea **inferioară** și cea **superioară** a frontierei, pentru a le determina sunt folosite la inițializare punctele P_1, P_2 , respectiv P_n, P_{n-1} . În final, aceste liste sunt concatenate.
- Principiul: asemănător celui de la Graham's scan: punctele sunt adăugate unul câte unul la listă. Se efectuează testul de orientare pentru ultimele trei puncte și este eliminat penultimul punct, în cazul în care ultimele trei puncte nu generează un viraj la stânga.

Comentarii - Graham's scan

- Algoritm specific pentru context 2D. Nu este on-line, fiind nevoie de toate punctele.
- Complexitatea: $O(n \log n)$; spațiu: $O(n)$; complexitate algebrică: polinoame de gradul II.
- Tratarea cazurilor degenerate: corect.
- Robustețea: datorită erorilor de rotunjire este posibil ca algoritmul să returneze o listă eronată (dar coerentă) de muchii.
- Graham's scan este optim pentru "cazul cel mai nefavorabil".
- Problema sortării este transformabilă în timp liniar în problema 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.

Jarvis' march (algorithm)

Input: O mulțime de puncte necoliniare $\mathcal{P} = \{P_1, P_2, \dots, P_n\}$ 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. $k \leftarrow 1$; $\mathcal{L} \leftarrow (A_1)$; $valid \leftarrow \text{true}$
3. **while** $valid = \text{true}$
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** $valid \leftarrow \text{false}$
11. **return** \mathcal{L}

2.4 Aplicație - determinarea punctelor antipodale

Algoritm pentru determinarea punctelor antipodale

Input: Vârfurile unui poligon convex \mathcal{P} din \mathbf{R}^2 ($n \geq 3$), organizate ca o listă $= (P_0, P_1, \dots, P_n)$, gestionată cu un pointer NEXT.

Output: Afișează perechile de vârfuri antipodale ale lui \mathcal{P} .

1. $P \leftarrow P_n$; $Q \leftarrow P_0 (= \text{NEXT}[P])$
2. **while** $\mathcal{A}(\Delta(P, \text{NEXT}[P], \text{NEXT}[Q])) > \mathcal{A}(\Delta(P, \text{NEXT}[P], Q))$
3. **do** $Q \leftarrow \text{NEXT}[Q]$
4. $Q_0 \leftarrow Q$
5. **while** ($Q \neq P_0$)
6. **do** $P \leftarrow \text{NEXT}[P]$
7. PRINT(P, Q)
8. **while** $\mathcal{A}(\Delta(P, \text{NEXT}[P], \text{NEXT}[Q])) > \mathcal{A}(\Delta(P, \text{NEXT}[P], Q))$
9. **do** $Q \leftarrow \text{NEXT}[Q]$
10. **if** $((P, Q) \neq (Q_0, P_0))$ **then** PRINT(P, Q)
11. **if** $\mathcal{A}(\Delta(P, \text{NEXT}[P], \text{NEXT}[Q])) = \mathcal{A}(\Delta(P, \text{NEXT}[P], Q))$ **then**
12. **if** $((P, Q) \neq (Q_0, P_n))$ **then** PRINT($P, \text{NEXT}[Q]$)

2.5 Exerciții, probleme, aplicații

Exercițiul 2.5 Fie $\mathcal{M} = \{P_1, P_2, \dots, P_7\}$, unde $P_1 = (1, 11)$, $P_2 = (2, 7)$, $P_3 = (3, 8)$, $P_4 = (4, 10)$, $P_5 = (5, 7)$, $P_6 = (6, 7)$, $P_7 = (7, 11)$. Detaliați cum evoluează lista vârfurilor care determină marginea inferioară a frontierei acoperirii convexe a lui \mathcal{M} , obținută pe parcursul Graham's scan / Graham's scan varianta Andrew. Justificați!

Exercițiul 2.6 Considerăm punctele $A = (-6, 6)$, $B = (1, 6)$, $C = (1, -1)$, $D = (-6, 0)$, $E = (6, 0)$, $F = (3, 2)$, $G = (-4, -2)$, $H = (-1, -2)$, $I = (-2, -2)$. Precizați care este numărul maxim de elemente pe care îl conține \mathcal{L} pe parcursul parcurgerii Graham's scan, indicând explicit punctele respective din \mathcal{L} (\mathcal{L} este lista vârfurilor care determină frontiera acoperirii convexe a lui \mathcal{M} , iar punctul "intern" considerat este O). Justificați!

Exercițiul 2.7 Dați un exemplu de mulțime \mathcal{M} din planul \mathbb{R}^2 pentru care, la final, \mathcal{L}_i are 3 elemente, dar, pe parcursul algoritmului, numărul maxim de elemente al lui \mathcal{L}_i este egal cu 5 (\mathcal{L}_i este lista vârfurilor care determină marginea inferioară a frontierei acoperirii convexe a lui \mathcal{M} , obținută pe parcursul Graham's scan, varianta Andrew). Justificați!

Exercițiul 2.8 Fie punctele $P_1 = (2, 0)$, $P_2 = (0, 3)$, $P_3 = (-4, 0)$, $P_4 = (4, 2)$, $P_5 = (5, 1)$. Precizați testele care trebuie efectuate, atunci când este aplicat Jarvis' march, pentru determinarea succesorului M al "celui mai din stânga" punct și a succesorului lui M . Cum decurg testele dacă se începe cu "cel mai de jos" punct?

Exercițiul 2.9 Dați un exemplu de mulțime cu 8 elemente \mathcal{M} din planul \mathbb{R}^2 pentru care frontiera acoperirii convexe are 3 elemente și pentru care, la găsirea succesorului "celui mai din stânga" punct (se aplică Jarvis' march), toate celelalte puncte sunt testate. Justificați!

Exercițiul 2.10 Scrieți în pseudocod Graham's scan - varianta Andrew și Jarvis' march.

Exercițiul 2.11 Explicați dacă Graham's scan / Jarvis' march indică rezultatul dorit atunci când toate punctele sunt situate pe o aceeași dreaptă.

Exercițiul 2.12 Date n puncte în plan, scrieți un algoritm de complexitate $O(n \log n)$ care să determine un poligon care are toate aceste puncte ca vârfuri. Explicați cum este aplicat acest algoritm pentru punctele $P_1 = (4, 2)$, $P_2 = (7, 1)$, $P_3 = (-3, 5)$, $P_4 = (3, 6)$, $P_5 = (-4, -4)$, $P_6 = (-1, 1)$, $P_7 = (2, -6)$.

Exercițiul 2.13 Fie punctele $A = (2, 0)$, $B = (0, 1)$, $C = (0, -1)$, $D = (\lambda, 0)$. Pentru ce valori ale lui λ diametrul mulțimii $\{A, B, C, D\}$ este egal cu 2?

Exercițiul 2.14 Fie punctele $A = (1, 0)$, $B = (0, -1)$, $C = (-1, 0)$, $D = (0, 1)$, $E = (3, 3)$. Dați exemplu de dreaptă suport pentru poligonul $ABCDE$.

Exercițiul 2.15 Fie $ABCDEF$ un hexagon regulat. Explicați câte puncte antipodale are fiecare vârf al poligonului.

Exercițiul 2.16 Fie punctele $A = (3, -3)$, $B = (3, 3)$, $C = (-3, -3)$, $D = (-3, 3)$, $M = (2 - \lambda, 3 + \lambda)$, $\lambda \in \mathbb{R}$. Scrieți un algoritm care să indice numărul de puncte de pe frontiera acoperirii convexe a mulțimii $\{A, B, C, D, M\}$.

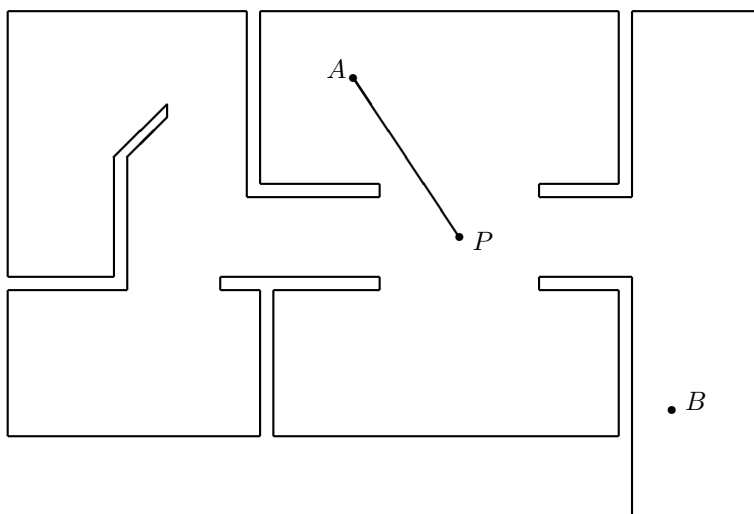
Capitolul 3

Triangulări

3.1 Triangularea poligoanelor. Problema galeriei de artă

Supravegherea unei galerii de artă

Camera din P poate supraveghea A , dar nu B .



Formalizare

- O galerie de artă poate fi interpretată (în contextul acestei probleme) ca un poligon simplu \mathcal{P} (adică un poligon fără autointersecții) având n vârfuri.
- O cameră video (vizibilitate 360°) poate fi identificată cu un punct din interiorul lui \mathcal{P} ; ea poate supraveghea acele puncte cu care poate fi unită printr-un segment inclus în interiorul poligonului.
- **Problema galeriei de artă:** câte camere video sunt necesare pentru a supraveghea o galerie de artă și unde trebuie amplasate acestea?

Numărul de camere vs. forma poligonului

- Se dorește exprimarea numărului de camere necesare pentru supraveghere în funcție de n (sau controlarea acestuia de către n).
- Pentru a supraveghea un spațiu având forma unui poligon convex, este suficientă o singură cameră.
- Numărul de camere depinde și de forma poligonului: cu cât forma este mai "complexă", cu atât numărul de camere va fi mai mare.
- **Principiu:** Poligonul considerat: descompus în triunghiuri (triangulare).

Definiții

- Fie \mathcal{P} un poligon plan.
- (i) O **diagonală** a lui \mathcal{P} este un segment ce unește două vârfuri ale acestuia și care este situat în interiorul lui \mathcal{P} .
- (ii) O **triangulare** $\mathcal{T}_{\mathcal{P}}$ a lui \mathcal{P} este o descompunere a lui \mathcal{P} în triunghiuri, dată de o mulțime maximală de diagonale ce nu se intersectează.
- **Teoremă.** *Orice poligon simplu admite o triangulare. Orice triangulare a unui poligon cu n vârfuri conține exact $n - 2$ triunghiuri.*

Rezolvarea problemei galeriei de artă

- Amplasarea camerelor se poate face în vârfurile poligonului.
- Dată o pereche $(\mathcal{P}, \mathcal{T}_{\mathcal{P}})$ se consideră o 3-colorare a acesteia: fiecărui vârf îi corespunde o culoare dintr-un set de 3 culori și pentru fiecare triunghi, cele 3 vârfuri au culori distincte.
- **Observație.** Dacă \mathcal{P} este simplu, o astfel de colorare există, deoarece graful asociat perechii $(\mathcal{P}, \mathcal{T}_{\mathcal{P}})$ este arbore.

Teorema galeriei de artă

- **Teoremă.** [Chvátal, 1975; Fisk, 1978] *Pentru un poligon cu n vârfuri, $\left\lceil \frac{n}{3} \right\rceil$ camere sunt uneori necesare și întotdeauna suficiente pentru ca fiecare punct al poligonului să fie vizibil din cel puțin una din camere.*
- Despre Teorema Galeriei de Artă: **J. O'Rourke, Art Gallery Theorems and Algorithms**

Metode de triangulare: ear cutting / clipping / trimming

- Concepte:
 - **vârf principal**,
 - **ear** (vârf / componentă de tip E) [Meisters, 1975];
 - **mouth** (vârf / componentă de tip M) [Toussaint, 1991].
- Orice vârf de tip E este convex; orice vârf de tip M este concav (reflex). Reciproc nu neapărat!
- **Teoremă.** (Two Ears Theorem [Meisters, 1975]) *Orice poligon cu cel puțin 4 vârfuri admite cel puțin două componente de tip E care nu se suprapun.*

- **Corolar.** *Orice poligon simplu admite (cel puțin) două diagonale.*
- Algoritm de triangulare bazat de metoda *ear cutting*: complexitate $O(n^2)$.
- [Link despre triangulări](#)
[Link pentru algoritmul *Ear cutting*](#)

Metode de triangulare: descompunerea în poligoane monotone

- Concept: **poligon y -monoton**
- Algoritmi de triangulare eficienți: complexitate $O(n)$ pentru poligoane y -monotone [Garey et al., 1978].
- Descompunerea unui poligon oarecare în componente y -monotone poate fi realizată cu un algoritm de complexitate $O(n \log n)$ [Lee, Preparata, 1977].
- Există și alte clase de algoritmi mai rapizi; [Chazelle, 1990]: algoritm liniar.
- [Link pentru alte abordări](#)
- [Link pentru alte abordări](#)
- Găsirea unui algoritm liniar "simplu" [Problemă în *The Open Problems Project*](#)

Triangularea poligoanelor monotone

Input: Un poligon y -monoton \mathcal{P} .

Output: O triangulare a lui \mathcal{P} .

1. Lanțul vârfurilor din partea stângă și al celor din partea dreaptă sunt unite într-un singur șir, ordonat descrescător, după y (dacă ordonata este egală, se folosește abscisa). Fie v_1, v_2, \dots, v_n șirul ordonat.
2. Inițializează o stivă vidă \mathcal{S} și inserează v_1, v_2 .
3. **for** $j = 3$ **to** $n - 1$
4. **do** **if** v_j și vârful din top al lui \mathcal{S} sunt în lanțuri diferite
5. **then** extrage toate vârfurile din \mathcal{S}
6. inserează diagonale de la v_j la vf. extrase, exceptând ultimul
7. inserează v_{j-1} și v_j în \mathcal{S}
8. **else** extrage un vârf din \mathcal{S}
9. extrage celelalte vârfuri din \mathcal{S} dacă diagonalele formate cu v_j sunt în interiorul lui \mathcal{P} ; inserează aceste diagonale; inserează înapoi ultimul vârf extras
10. inserează v_j în \mathcal{S}
11. adaugă diagonale de la v_n la vf. stivei (exceptând primul și ultimul)

3.2 Triangularea unei mulțimi arbitrare de puncte

Problematizare

- Triangularea unui poligon convex (listă ordonată de puncte (P_1, P_2, \dots, P_n)).
- Are sens să vorbim de triangulare pentru mulțimea $\{P_1, P_2, \dots, P_n\}$?
- **Comentariu:** Triangulările mulțimilor de puncte sunt esențiale în **grafica pe calculator**.
- **Definiție.** O **triangulare** a unei mulțimi \mathcal{P} din plan este o subdivizare maximală a acoperirii convexe $\text{Conv}(\mathcal{P})$ a lui \mathcal{P} cu triunghiuri ale căror vârfuri sunt elemente ale lui \mathcal{P} (fără autointersecții!)
- Trebuie făcută distincție între triangulare a unui poligon (P_1, P_2, \dots, P_n) și triangulare a mulțimii subdiacente $\{P_1, P_2, \dots, P_n\}$ (coincid dacă poligonul este convex!)

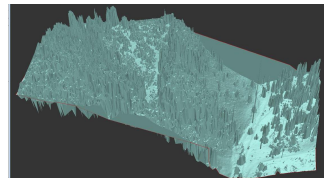
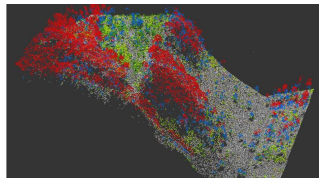
Elemente ale unei triangulări

- Dată o mulțime de puncte \mathcal{P} și o triangulare $\mathcal{T}_{\mathcal{P}}$ a sa:
vârfuri, muchii, triunghiuri.
- Legătură între aceste elemente?
- **Propoziție.** Fie \mathcal{P} o mulțime de n puncte din plan nesituate toate pe o aceeași dreaptă. Notăm cu k numărul de puncte de pe frontiera acoperirii convexe $\text{Conv}(\mathcal{P})$. Orice triangulare a lui \mathcal{P} are $(2n - k - 2)$ triunghiuri și $(3n - k - 3)$ muchii.
- **Demonstrație:** Se bazează pe formula lui Euler / numărul de incidente.
- **Exemplu:** Cazul unui poligon convex cu n vârfuri ($k = n$): $n - 2$ triunghiuri și $2n - 3$ muchii.

3.3 Triangulări unghiular optime

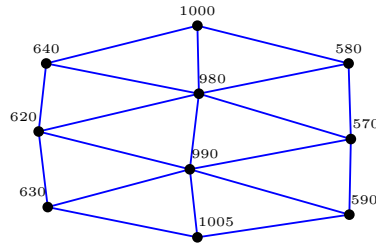
Problematizare

- **Problemă. Problemă.** Se fac măsurători ale altitudinii pentru un teren. Se dorește reprezentarea tridimensională (cât mai sugestivă). Alternativ: se dorește **generarea unui teren** pentru o aplicație.

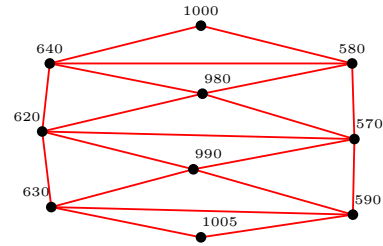


- **Problemă (reformulată).** Cum ”comparăm triangulările” unei mulțimi de puncte fixate?

- **Exemplu.** Măsurători ale altitudinii.



Triangulare 1



Triangulare 2

- **Întrebări naturale:** (i) Există o triangulare “convenabilă” a unei mulțimi de puncte? (ii) Cum poate fi determinată eficient o astfel de triangulare?

Terminologie

- Fixată: o mulțime de puncte \mathcal{P} .
- Fie \mathcal{T} o triangulare a lui \mathcal{P} cu m triunghiuri. Fie $\alpha_1, \alpha_2, \dots, \alpha_{3m}$ unghiurile lui \mathcal{T} , ordonate crescător. **Vectorul unghiurilor lui \mathcal{T} este $A(\mathcal{T}) = (\alpha_1, \alpha_2, \dots, \alpha_{3m})$.**
- **Relație de ordine pe mulțimea triangulărilor lui \mathcal{P} :** ordinea lexicografică pentru vectorii unghiurilor. Fie \mathcal{T} și \mathcal{T}' două triangulări ale lui \mathcal{P} . Atunci $A(\mathcal{T}) > A(\mathcal{T}')$ dacă $\exists i$ astfel ca $\alpha_j = \alpha'_j, \forall 1 \leq j < i$ și $\alpha_i > \alpha'_i$. ca $\alpha_j = \alpha'_j, \forall 1 \leq j < i$ și $\alpha_i > \alpha'_i$.
- **Triangulare unghiular optimă:** \mathcal{T} astfel ca $A(\mathcal{T}) \geq A(\mathcal{T}')$, pentru orice triangulare \mathcal{T}' .
- **Exemplu:** Cazul unui patrulater convex.

Muchii ilegale, triangulări legale

- Fixată: o mulțime de puncte \mathcal{P} .
- **Conceptul de muchie ilegală.** Fie $A, B, C, D \in \mathcal{P}$ fixate astfel ca $ABCD$ să fie un patrulater convex; fie $\mathcal{T}_{AC}, \mathcal{T}_{BD}$ triangulările date de diagonalele AC , respectiv BD . Muchia AC este **ilegală** dacă

$$\min A(\mathcal{T}_{AC}) < \min A(\mathcal{T}_{BD}).$$

- **Concluzie:** Muchia AC este ilegală dacă, printr-un *flip* (înlocuirea ei cu BD), cel mai mic unghi poate fi mărit (local).
- **Concluzie (reformulare):** Fie \mathcal{T} o triangulare cu o muchie ilegală e , fie \mathcal{T}' triangularea obținută din \mathcal{T} prin *flip*-ul muchiei e . Atunci $A(\mathcal{T}') > A(\mathcal{T})$.
- **Criteriu geometric** pentru a testa dacă o muchie este legală.

Triangulări unghiular optime vs. triangulări legale

- **Triangulare legală:** nu are muchii ilegale.

- O triangulare legală poate fi determinată pornind de la o triangulare arbitrară.
- **Propoziție.** Fie \mathcal{P} o mulțime de puncte din plan.
 - (i) Orice triangulare unghiular optimă este legală.
 - (ii) Dacă \mathcal{P} este în poziție generală (oricare patru puncte nu sunt conclice), atunci există o unică triangulare legală, iar aceasta este unghiular optimă.
- **Teoremă.** Fie \mathcal{P} o mulțime de n puncte din plan, în poziție generală. Triangularea unghiular optimă poate fi construită, folosind un algoritm incremental randomizat, în timp mediu $O(n \log n)$, folosind $O(n)$ memorie medie.

3.4 Exerciții, probleme, aplicații

Exercițiul 3.1 Fie \mathcal{P} poligonul dat de punctele $P_1 = (6, 0)$, $P_2 = (2, 2)$, $P_3 = (0, 7)$, $P_4 = (-2, 2)$, $P_5 = (-8, 0)$, $P_6 = (-2, -2)$, $P_7 = (0, -6)$, $P_8 = (2, -2)$. Indicați o triangulare $\mathcal{T}_{\mathcal{P}}$ a lui \mathcal{P} și construiți graful asociat perechii $(\mathcal{P}, \mathcal{T}_{\mathcal{P}})$.

Exercițiul 3.2 Aplicați metoda din demonstrația teoremei galeriei de artă, indicând o posibilă amplasare a camerelor de supraveghere în cazul poligonului $P_1P_2 \dots P_{12}$, unde $P_1 = (4, 4)$, $P_2 = (5, 6)$, $P_3 = (6, 4)$, $P_4 = (7, 4)$, $P_5 = (9, 6)$, $P_6 = (11, 6)$ iar punctele P_7, \dots, P_{12} sunt respectiv simetricele punctelor P_6, \dots, P_1 față de axa Ox .

Exercițiul 3.3 Fie poligonul $\mathcal{P} = (P_1P_2P_3P_4P_5P_6)$, unde $P_1 = (5, 0)$, $P_2 = (3, 2)$, $P_3 = (-1, 2)$, $P_4 = (-3, 0)$, $P_5 = (-1, -2)$, $P_6 = (3, -2)$. Arătați că Teorema Galeriei de Artă poate fi aplicată în două moduri diferite, așa încât în prima variantă să fie suficientă o singură cameră, iar în cea de-a doua variantă să fie necesare și suficiente două camere pentru supravegherea unei galerii având forma poligonului \mathcal{P} .

Exercițiul 3.4 Fie poligonul $\mathcal{P} = (P_1P_2 \dots P_{10})$, unde $P_1 = (0, 0)$, $P_2 = (6, 0)$, $P_3 = (6, 6)$, $P_4 = (3, 6)$, $P_5 = (3, 3)$, $P_6 = (4, 4)$, $P_7 = (4, 2)$, $P_8 = (2, 2)$, $P_9 = (2, 6)$, $P_{10} = (0, 6)$. Stabiliți natura vârfurilor lui \mathcal{P} (vârf principal sau nu / vârf convex sau concav).

Exercițiul 3.5 Dați exemplu de poligon cu 6 vârfuri care să aibă atât vârfuri convexe, cât și concave și toate să fie principale.

Exercițiul 3.6 Dați exemplu de poligon care să aibă mai multe vârfuri principale concave decât vârfuri principale convexe.

Exercițiul 3.7 Se consideră poligonul $\mathcal{P} = P_1P_2P_3P_4P_5P_6P_7P_8$ dat de punctele $P_1 = (0, 10)$, $P_2 = (1, 8)$, $P_3 = (3, 6)$, $P_4 = (7, 3)$, $P_5 = (4, 0)$, $P_6 = (6, -2)$, $P_7 = (4, -4)$, $P_8 = (-4, -1)$. Stabiliți dacă \mathcal{P} este y -monoton și, în caz afirmativ, explicați cum se aplică algoritmul liniar de triangulare.

Exercițiul 3.8 În algoritmul de triangulare a poligoanelor y -monotone au fost descrise trei cazuri. Justificați dacă, aplicând algoritmul pentru un poligon y -monoton cu cel puțin 4 laturi este necesar să apară toate aceste cazuri.

Exercițiul 3.9 Fie $n \geq 2$ un număr natural par fixat. Considerăm mulțimea $\mathcal{M} = \{A_0, \dots, A_n, B_0, \dots, B_n, C_0, \dots, C_n, D_0, \dots, D_n\}$, unde $A_i = (i, 0)$, $B_i = (0, i)$, $C_i = (i, i)$, $D_i = (n-i, i)$, pentru orice $i = 0, \dots, n$. Determinați numărul de triunghiuri și numărul de muchii al unei triangulări a lui \mathcal{M} .

Exercițiul 3.10 Dați exemplu de mulțime de puncte din \mathbb{R}^2 care să admită o triangulare având 3 triunghiuri și 7 muchii.

Exercițiul 3.11 Dați exemplu de mulțime $\mathcal{M} = \{A, B, C, D, E, F, G\}$ din \mathbb{R}^2 astfel ca \mathcal{M} să admită o triangulare ce conține 14 muchii.

Exercițiul 3.12 Dați exemplu de două mulțimi de puncte $\mathcal{M}_1, \mathcal{M}_2$ din \mathbb{R}^2 având număr diferit de puncte, dar care admit triangulări ce conțin exact 3 fețe de tip triunghi. Pentru fiecare din cele două mulțimi precizați: (i) numărul de muchii din triangulările corespunzătoare; (ii) numărul muchiilor de tip semi-dreaptă ale diagramei Voronoi asociate.

Exercițiul 3.13 Dați exemplu de mulțime \mathcal{M} cu 6 elemente din \mathbb{R}^2 care să admită o triangulare ce conține 12 muchii, iar una dintre submulțimile sale cu 4 elemente să admită o triangulare ce conține 5 muchii. Justificați alegerea făcută.

Exercițiul 3.14 Fie $ABCD$ un patrulater convex. Fie \mathcal{C} cercul circumscris triunghiului $\triangle ABC$. Demonstrați că diagonala AC este ilegală dacă și numai dacă D este în interiorul lui \mathcal{C} .

Exercițiul 3.15 Dați exemplu de patrulater convex $ABCD$ din \mathbb{R}^2 pentru care muchia AC este ilegală și aplicați criteriul numeric indicat la curs pentru exemplul ales.

Exercițiul 3.16 Fie punctele $A = (1, 1), B = (1, -1), C = (-1, -1), D = (-1, 1), E = (0, -2), M = (0, \lambda)$, unde $\lambda \in \mathbb{R}$ este un parametru real. Scrieți un algoritm care să indice numărul de triunghiuri și numărul de muchii ale unei triangulări asociate mulțimii $\{A, B, C, D, E, M\}$.

Capitolul 4

Intersecții

4.1 Intersecția segmentelor (generalități)

Motivație (Probleme geometrice în context 2D)

- **Problema 1.** Dată o listă (mulțime ordonată) de puncte $\mathcal{P} = (P_1, P_2, \dots, P_m)$, să se stabilească dacă ea reprezintă un poligon simplu (fără autointersecții).
- **Problema 2.** Date două poligoane simple \mathcal{P} și \mathcal{Q} , să se stabilească dacă se intersectează (interioarele lor se intersectează).
- **Problema 3.** Dată o mulțime $\mathcal{S} = \{s_1, \dots, s_n\}$ de n segmente închise din plan, să se determine toate perechile care se intersectează.
- **Problema 3'.** Dată o mulțime $\mathcal{S} = \{s_1, \dots, s_n\}$ de n segmente închise din plan, să se determine toate punctele de intersecție dintre ele.

Algoritmul trivial

- **Idee de lucru:** Sunt considerate toate perechile de segmente și se determină cele care se intersectează / se calculează punctele de intersecție.
- **Complexitate:**
 - timp: $O(n^2)$
 - memorie: $O(n)$
 - algebric: polinoame de gradul II (Problema 3), rapoarte de polinoame (Problema 3')
- **Comentariu:** În anumite cazuri: optim (dacă toate segmentele se intersectează).
- Algoritmi mai eficienți (*output / intersection sensitive*)?

Rezolvarea problemei în context 1D

- Ordonarea lexicografică a extremităților segmentelor / intervalelor într-o listă \mathcal{P} .
- Lista \mathcal{P} este parcursă (crescător); lista \mathcal{L} a segmentelor care conțin punctul curent din \mathcal{P} este actualizată:
 - dacă punctul curent este marginea din stânga a unui segment s , atunci s este adăugat la listă \mathcal{L}

- dacă punctul curent este marginea din dreapta a unui segment s , atunci s este șters din \mathcal{L} și se rapoartează intersecții între s și toate segmentele din \mathcal{L}
- **Teoremă.** Algoritmul are complexitate $O(n \log n + k)$ și necesită $O(n)$ memorie (k este numărul de perechi ce se intersectează).

4.2 Metoda liniei de baleiere

Un prim algoritm

- Dreapta de baleiere l : **orizontală**, astfel că toate intersecțiile situate deasupra liniei de baleiere au fost detectate.
- **Statutul (sweep line status)**: mulțimea segmentelor care intersectează l .
- **Evenimente (event points)**: capetele segmentelor \rightarrow actualizarea statutului
- **Obs. 1.** Sunt testate pentru intersecție segmente care intersectează, la un moment dat, $l \rightarrow$ (sunt testate segmentele care sunt "aproape" de-a lungul axei Oy) \rightarrow încă inefficient.
- **Obs. 2.** Dreapta de baleiere are, de fapt, o variație "discretă", nu continuă.

Un algoritm eficient [Bentley și Ottmann, 1979; Mehlhorn și Näher, 1994]

- **Idee de lucru:** ordonare (segmente, evenimente).
 - Segmentele: ordonate folosind extremitățile superioare (lexicografic, x apoi y).
 - Evenimente (puncte): (lexicografic, y apoi x).
 - Statutul: **listă** (mulțime ordonată)
- **Avantaj:** în momentul modificării statutului, sunt testate intersecțiile doar în raport cu vecinii din listă (sunt testate segmentele care sunt "aproape" de-a lungul axei Ox).
- **Fundamental:** Punctele de intersecție devin, la rândul lor, evenimente, deoarece schimbă ordinea segmentelor care le determină \rightarrow chiar dacă nu sunt determinate explicit, trebuie inserate în lista de evenimente (compararea coordonatelor poate necesita utilizarea unor polinoame de gradul V)

3 tipuri de evenimente

- Marginea superioară a unui segment
 - apare un nou segment în statutul liniei de baleiere, ce trebuie inserat corespunzător
 - testat, în raport cu vecinii, dacă au puncte de intersecție sub linia de baleiere \rightarrow vor deveni ulterior evenimente
- Marginea inferioară a unui segment

- eliminat un segment din statutul liniei de baleiere
- testare pentru segmentele vecine nou apărute
- punctele de intersecție (inserate în mod corespunzător pe parcurs)
 - segmentele care le determină trebuie ”inversate” în statutul liniei de baleiere
 - testare pentru segmentele vecine nou apărute

Implementare: structuri de date utilizate

- \mathcal{Q} coada de evenimente (*event queue*)
 - puncte, ordonate lexicografic, y apoi x ; memorată într-un arbore binar de căutare echilibrat (*balanced binary search tree*)
 - evenimentele nou detectate trebuie inserate în mod corespunzător!
 - de evaluat: complexitatea-timp (pentru o inserare, numărul de repetiții); complexitatea spațiu
- \mathcal{T} – **statut**: arbore binar de căutare echilibrat (*balanced binary search tree*)
 - pe frunze: segmente, este reținută ordinea segmentelor de la stânga la dreapta
 - în nodurile interne: segmente, privite ca elemente de ghidare
 - de evaluat: complexitatea-timp (pentru o actualizare, numărul de repetiții)

Algorithm INTERSECTII

- **Input.** O mulțime de segmente din planul \mathbb{R}^2 .
 - **Output.** Mulțimea punctelor de intersecție (explicit sau doar formal); pentru fiecare punct precizează segmentele pe care se găsește.
1. $\mathcal{Q} \leftarrow \emptyset$. Inserează extremitățile segmentelor în \mathcal{Q} ; împreună cu marginea superioară a unui segment memorează și segmentul
 2. $\mathcal{T} \leftarrow \emptyset$.
 3. **while** $\mathcal{Q} \neq \emptyset$
 4. **do** determină evenimentul p care urmează în \mathcal{Q} și îl șterge
 5. ANALIZEAZA (p)
- ANALIZEAZA (p)
1. Fie $U(p)$ mulțimea segmentelor a căror extremitate superioară este p (pentru cele orizontale este marginea din stânga) - stocată cu p în \mathcal{Q} .
 2. Determină toate segmentele care conțin p : sunt adiacente în \mathcal{T} (de ce?) Fie $D(p)$, respectiv $Int(p)$, mulțimea segmentelor care au p drept margine inferioară, respectiv conțin p în interior.
 3. **if** $U(p) \cup D(p) \cup Int(p)$ conține mai mult de un segment

4. **then** raportează p ca punct de intersecție, împreună cu segmentele din $U(p)$, $D(p)$, $Int(p)$
5. șterge segmentele din $D(p) \cup Int(p)$ din \mathcal{T}
6. inserează segmentele din $U(p) \cup Int(p)$ în \mathcal{T} (ordinea segmentelor pe frunzele lui \mathcal{T} coincide cu ordinea în care sunt intersectate de o linie de baleiere situată imediat sub p)
7. **if** $U(p) \cup Int(p) = \emptyset$
8. **then** fie s_l și s_r vecinii din stânga/dreapta ai lui p din \mathcal{T}
9. DETERMINAEVENIMENT (s_l, s_r, p)
10. **else** fie s' din $U(p) \cup Int(p)$ cel mai în stânga în \mathcal{T}
11. fie s_l vecinul din stânga al lui p
12. DETERMINAEVENIMENT (s_l, s', p)
13. fie s'' din $U(p) \cup Int(p)$ cel mai în dreapta în \mathcal{T}
14. fie s_r vecinul din dreapta al lui p
15. DETERMINAEVENIMENT (s'', s_r, p)
- DETERMINAEVENIMENT (sgt_l, sgt_r, p)
1. **if** sgt_l și sgt_r se intersectează sub linia de baleiere sau pe linia de baleiere, dar la dreapta lui p și punctul de intersecție nu este deja în \mathcal{Q}
2. **then** inserează punctul de intersecție ca eveniment în \mathcal{Q}

Rezultatul principal (intersecții de segmente) Teoremă. *Fie S o mulțime care conține n segmente din planul \mathbb{R}^2 . Toate punctele de intersecție ale segmentelor din S , împreună cu segmentele corespunzătoare, pot fi determinate în $O(n \log n + I \log n)$ timp, folosind $O(n)$ spațiu (I este numărul punctelor de intersecție).*

4.3 Alte rezultate

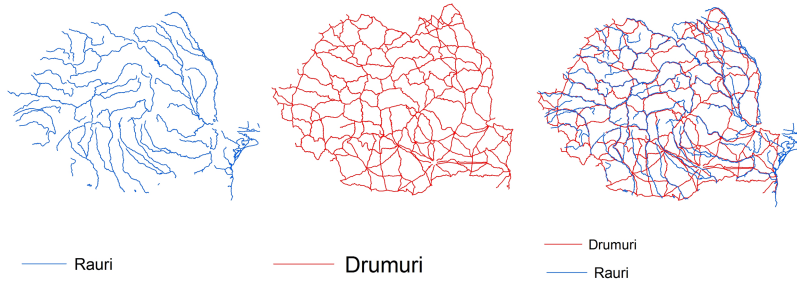
Red-blue intersections [Mairson și Stolfi, 1988; Mantler și Snoeyink, 2000]

Teoremă. *Pentru două mulțimi R și B de segmente din \mathbb{R}^2 având interioare disjuncte, perechile de segmente ce se intersectează pot fi determinate în $O(n \log n + k)$ timp și spațiu liniar, folosind predicate (polinoame) de grad cel mult I ($n = |R| + |B|$) și k este numărul perechilor ce se intersectează).*

4.4 Suprapunerea straturilor tematice

Subdiviziuni planare

- Conceptul de subdiviziune planară: vârfuri, muchii, fețe.
- **Listă dublu înlănțuită** [Müller și Preparata, 1978] (trei înregistrări: vârfuri, fețe, muchii orientate (semi-muchii)).



- **Vârf** v : coordonatele lui v în $Coordinates(v)$, pointer $IncidentEdge(v)$ spre o muchie orientată care are v ca origine
- **Față** f : pointer $OuterComponent(f)$ spre o muchie orientată corespunzătoare frontierei externe (pentru fața nemărginită este **nil**); listă $InnerComponents(f)$, care conține, pentru fiecare gol, un pointer către una dintre muchiile orientate de pe frontieră
- **Muchie orientată** \vec{e} : pointer $Origin(\vec{e})$, pointer $Twin(\vec{e})$ pointer $IncidentFace(\vec{e})$, pointer $Next(\vec{e})$, pointer $Prev(\vec{e})$.
- Oricărei subdiviziuni planare \mathcal{S} i se asociază o listă dublu înlănțuită $\mathcal{D}_{\mathcal{S}}$.

Algoritm OVERLAY ($\mathcal{S}_1, \mathcal{S}_2$)

- **Input.** Două subdiviziuni planare $\mathcal{S}_1, \mathcal{S}_2$ memorate în liste dublu înlănțuite $\mathcal{D}_{\mathcal{S}_1}, \mathcal{D}_{\mathcal{S}_2}$
 - **Output.** Overlay-ul $\mathcal{O}(\mathcal{S}_1, \mathcal{S}_2)$ dintre $\mathcal{S}_1, \mathcal{S}_2$, memorat într-o listă dublu înlănțuită $\mathcal{D}_{\mathcal{O}(\mathcal{S}_1, \mathcal{S}_2)}$
1. Copiază listele $\mathcal{D}_1, \mathcal{D}_2$ într-o nouă listă dublu înlănțuită $\mathcal{D}_{\mathcal{O}(\mathcal{S}_1, \mathcal{S}_2)}$
 2. Calculează toate intersecțiile de muchii dintre \mathcal{S}_1 și \mathcal{S}_2 cu algoritmul INTERSECTII. La fiecare eveniment, pe lângă actualizarea lui \mathcal{Q} și \mathcal{T} , efectuează:
 - Actualizează $\mathcal{D}_{\mathcal{O}(\mathcal{S}_1, \mathcal{S}_2)}$, în cazul în care evenimentul implică atât muchii ale lui \mathcal{S}_1 , cât și ale lui \mathcal{S}_2
 - Memorează noile muchii orientate adecvat
 3. Determină cicluri de frontieră din $\mathcal{O}(\mathcal{S}_1, \mathcal{S}_2)$, stabilește natura (exteriori/interiori)
 4. Construiește graful \mathcal{G} ale cărui noduri corespund ciclurilor de frontieră și ale cărui arce unesc fiecare ciclu corespunzând unui gol cu ciclul de la stânga vârfului cel mai din stânga și determină componentele conexe ale lui \mathcal{G}
 5. **for** fiecare componentă a lui \mathcal{G}
 6. **do** Fie \mathcal{C} unicul ciclu de frontieră exterioară a componentei și fie f fața mărginită a ciclului. Creează o înregistrare pentru f , setează $OuterComponent(f)$ (către una din muchiile lui \mathcal{C}), construiește lista

$InnerComponents(f)$ (pentru fiecare gol, pointer către una dintre muchiile orientate). Pentru fiecare muchie orientată, $IncidentFace()$ către înregistrarea lui f

7. Etichetează fiecare față a lui $\mathcal{O}(\mathcal{S}_1, \mathcal{S}_2)$

Rezultate principale

- **Teoremă. (Overlay-ul hărților)** Fie \mathcal{S}_1 o subdiviziune de complexitate n_1 , \mathcal{S}_2 o subdiviziune de complexitate n_2 , fie $n := n_1 + n_2$. Overlay-ul dintre \mathcal{S}_1 și \mathcal{S}_2 poate fi construit în $O(n \log n + k \log n)$, unde k este complexitatea overlay-ului.
- **Corolar. (Operații boolene)** Fie \mathcal{P}_1 un poligon cu n_1 vârfuri și \mathcal{P}_2 un poligon cu n_2 vârfuri; fie $n = n_1 + n_2$. Atunci $\mathcal{P}_1 \cup \mathcal{P}_2$, $\mathcal{P}_1 \cap \mathcal{P}_2$ și $\mathcal{P}_1 \setminus \mathcal{P}_2$ pot fi determinate în timp $O(n \log n + k \log n)$, unde k este complexitatea output-ului.

4.5 Exerciții, probleme, aplicații

Exercițiul 4.1 Fie punctele $P_1 = (4, 3), P_2 = (1, 1), P_3 = (6, 2), P_4 = (11, 8); Q_1 = (4, 5), Q_2 = (9, 9), Q_3 = (6, 7), Q_4 = (11, 0)$. Pentru fiecare $i = 1, \dots, 4$ notăm cu s_i segmentul $[P_i Q_i]$. Scrieți cum evoluează statutul liniei de baleiere, precum și evenimentele care determină modificarea sa (linia de baleiere este orizontală, iar statutul este o mulțime neordonată de segmente).

Exercițiul 4.2 Fie punctele $A_1 = (6, 1), A_2 = (3, 2), A_3 = (1, 8), A_4 = (13, 7); B_1 = (6, 6), B_2 = (11, 10), B_3 = (9, 0), B_4 = (13, -1)$. Scrieți cum evoluează statutul liniei de baleiere, precum și evenimentele care determină modificarea sa (linia de baleiere este orizontală, iar statutul este o mulțime ordonată de segmente).

Exercițiul 4.3 Fie punctele $P_1 = (4, -1), P_2 = (2, 8), P_3 = (3, 3), P_4 = (7, 0); Q_1 = (4, 11), Q_2 = (8, 2), Q_3 = (10, 10), Q_4 = (7, 4)$. Pentru fiecare $i = 1, \dots, 4$ notăm cu s_i segmentul $[P_i Q_i]$. Considerăm linia de baleiere l dată de ecuația $y = 9$. Indicați evenimentele deja eliminate din coada de evenimente \mathcal{Q} , cele rămase în \mathcal{Q} , cele care urmează să fie incluse ulterior în \mathcal{Q} și precizați statutul corespunzător lui l (statutul este o mulțime ordonată de segmente).

Exercițiul 4.4 Explicați cum poate fi parcursă frontiera unei fețe și cum pot fi găsite toate muchiile din jurul unui vârf, folosind pointerii asociați elementelor unei subdiviziuni planare.

Exercițiul 4.5 Fie \mathcal{S}_1 și \mathcal{S}_2 subdiviziunile planare date de patrulaterul $ABCD$ și triunghiul OPQ , unde $A = (-4, 0), B = (4, 0), C = (4, 2), D = (-4, 2); O = (0, 0), P = (-2, -2), Q = (2, -2)$. Explicați cum este actualizată lista de semi-muchi la construirea *overlay-ului* dintre \mathcal{S}_1 și \mathcal{S}_2 .

Exercițiul 4.6 Considerăm un dreptunghi D din interiorul căruia este scos un dreptunghi Δ . Descrieți subdiviziunea planară asociată.

Exercițiul 4.7 Considerăm trei dreptunghiuri D_1, D_2, D_3 astfel ca D_3 să fie situat în interiorul lui D_2 și D_2 în interiorul lui D_1 . Descrieți subdiviziunea planară asociată.

Exercițiul 4.8 Fie punctele $A = (4, 0)$, $B = (0, 4)$, $C = (-4, 0)$, $D = (0, -4)$. Construiți două subdiviziuni planare distincte \mathcal{S}_1 , \mathcal{S}_2 care au A, B, C, D ca vârfuri, astfel ca \mathcal{S}_1 să aibă două fețe, iar \mathcal{S}_2 să aibă trei fețe. Indicați numărul total de semimuchii pentru \mathcal{S}_1 , respectiv \mathcal{S}_2 .

Exercițiul 4.9 Fie segmentele $[AB]$, $[CD]$, $[EF]$. Stabiliți care este complexitatea algebrică a calculelor pentru ca (i) cele trei segmente să aibă același mijloc; (ii) cele trei segmente să aibă aceeași lungime.

Exercițiul 4.10 Fie punctele $A = (1, 6)$, $B = (1, 1)$, $C = (-4, 7)$, $D = (6, 7)$, $E = (1, -1)$, $F = (5, 3)$, $P = (-2, 3)$, $Q = (2 - \lambda, 3)$, unde $\lambda \in \mathbb{R}$ este un parametru. Scrieți un algoritm care să calculeze numărul de puncte de intersecție dintre segmentul $[PQ]$ și reuniunea $[AB] \cup [CD] \cup [EF]$. Algoritmul distinge între puncte interioare ale segmentelor și extremități ale acestora.

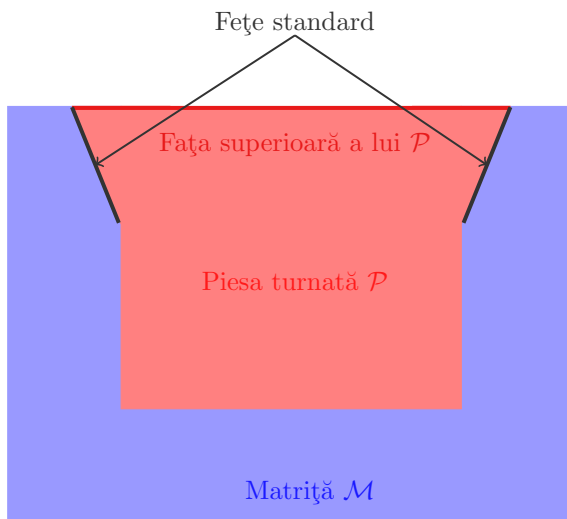
Exercițiul 4.11 Fie punctele $A = (-2, 1)$, $B = (1, 1)$, $C = (1, 5)$, $D = (5, 1)$, $E = (3 - \alpha, 3 + \alpha)$, unde α este un parametru real. Scrieți un algoritm care să determine numărul de fețe al subdiviziunii planare determinate de muchiile $[AB]$, $[BC]$, $[CA]$, $[BD]$ și $[CE]$.

Capitolul 5

Elemente de programare liniară

5.1 Motivație: turnarea pieselor în matrițe

- Turnarea pieselor în matrițe și extragerea lor fără distrugerea matriței.



- Neajunsuri: unele obiecte pot rămâne blocate în matrițe; există obiecte pentru care nu există o matriță adecvată; extragerea obiectului depinde de poziția matriței.
- **Problema studiată.** Dat un obiect, există o matriță din care să poată fi extras?
- **Convenții.**
 - Obiectele: **poliedrale**.
 - Matrițele: formate dintr-o singură piesă; fiecărui obiect \mathcal{P} îi este asociată o matriță $\mathcal{M}_{\mathcal{P}}$
 - Obiectul: extras printr-o singură translație (sau o succesiune de translații)

Terminologie și convenții

- **Alegerea orientării:** diverse orientări ale obiectului pot genera diverse matrițe.
- **Fața superioară:** prin convenție, obiectele au (cel puțin) o față superioară (este orizontală, este singura care nu este adiacentă cu matrița). Celelalte fețe: **standard**; orice față standard f a obiectului corespunde unei fețe \hat{f} a matriței.
- **Obiect care poate fi turnat (castable):** există o orientare pentru care acesta poate fi turnat și apoi extras printr-o translație (succesiune de translații): *direcție admisibilă*.
- **Convenții:** Matrița este paralelipipedică și are o cavitate corespunzătoare obiectului; fața superioară a obiectului (și a matriței) este perpendiculară cu planul Oxy .

Fundamente geometrice

- **Condiție necesară:** direcția de extragere \vec{d} trebuie să aibă componenta z pozitivă
- **În general:** o față standard \hat{f} a matriței (corespunzătoare unei fețe f a piesei) pentru care unghiul dintre normala exterioară $\vec{\nu}(f)$ la față f și \vec{d} este mai mic de 90° împiedică translația în direcția \vec{d}
- **Propoziție.** *Un poliedru \mathcal{P} poate fi extras din matrița sa $\mathcal{M}_{\mathcal{P}}$ prin translație în direcția \vec{d} dacă și numai dacă \vec{d} face un unghi de cel puțin 90° cu normala exterioară a fiecărei fețe standard a lui \mathcal{P} .*
- **Reformulare.** Dat \mathcal{P} , trebuie găsită o direcție \vec{d} astfel încât, pentru fiecare față standard f , unghiul dintre \vec{d} și $\vec{\nu}(f)$ să fie cel puțin 90° .
- **Analitic:** fiecare față definește un semiplan
- **Concluzie:** Fie \mathcal{P} un poliedru. Mulțimea direcțiilor admisibile este dată de o intersecție de semiplane.
- **Teoremă.** *Fie \mathcal{P} un poliedru cu n fețe. Se poate decide dacă \mathcal{P} reprezintă un obiect care poate fi turnat în $O(n^2)$ timp și folosind $O(n)$ spațiu. În caz afirmativ, o matriță și o direcție admisibilă în care poate fi extras \mathcal{P} este determinată cu aceeași complexitate timp.*

Exemple

1. Normalele la fețele standard sunt date de vectorii

$$(0, -1, 1), (0, 1, 1), (0, 1, 0), (0, 0, -1), (0, -1, 0).$$

2. Normalele la fețele standard sunt date de vectorii

$$(0, -1, -1), (0, 1, -1), (0, 1, 0), (0, 0, -1), (0, -1, 0).$$

5.2 Intersecții de semiplane

Formularea problemei

- Fie $\mathcal{H} = \{H_1, H_2, \dots, H_n\}$ o mulțime de semiplane din \mathbb{R}^2 ; semiplanul H_i dat de o relație de forma

$$a_i x + b_i y \leq c_i$$

- Intersecția $H_1 \cap H_2 \cap \dots \cap H_n$ este dată de un sistem de inecuații; este o mulțime poligonală convexă, mărginită de cel mult n muchii (poate fi vidă, mărginită, nemărginită,...)

Algoritm INTERSECTIISEMIPLANE (\mathcal{H})

- **Input.** O mulțime \mathcal{H} de semiplane din planul \mathbb{R}^2
 - **Output.** Regiunea poligonală convexă $\mathcal{C} = \cap_{H \in \mathcal{H}} H$
1. **if** $\text{card}(\mathcal{H}) = 1$
 2. **then** $\mathcal{C} \leftarrow H \in \mathcal{H}$
 3. **else** descompune \mathcal{H} în două mulțimi $\mathcal{H}_1, \mathcal{H}_2$ având fiecare $[n/2]$ elemente
 4. $\mathcal{C}_1 \leftarrow \text{INTERSECTIISEMIPLANE}(\mathcal{H}_1)$
 5. $\mathcal{C}_2 \leftarrow \text{INTERSECTIISEMIPLANE}(\mathcal{H}_2)$
 6. $\mathcal{C} \leftarrow \text{INTERSECTEAZAREGIUNICONVEXE}(\mathcal{C}_1, \mathcal{C}_2)$

Rezultate principale

- **Propoziție.** *Aplicând direct algoritmii de overlay, intersecția dintre două regiuni convexe (INTERSECTEAZAREGIUNICONVEXE) poate fi calculată cu complexitate-timp $O(n \log n)$; în particular algoritmul INTERSECTIISEMIPLANE are complexitate $O(n \log^2 n)$.*
- **Teoremă.** *Algoritmul INTERSECTEAZAREGIUNICONVEXE poate fi îmbunătățit, astfel încât complexitatea-timp să fie liniară.*
- **Teoremă.** *Intersecția unei mulțimi de n semiplane poate fi determinată cu complexitate-timp $O(n \log n)$ și folosind $O(n)$ memorie.*

5.3 Dualitate

Dualitate – motivație euristică

- De câte informații (numerice) este nevoie pentru a indica **un punct în plan?**: **2**
- De câte informații (numerice) este nevoie pentru a indica **o dreaptă în plan?**: **2**
- Există o modalitate naturală de a stabili o corespondență între puncte și drepte?: **DA: dualitate**
- Cum se reflectă / respectă diferite proprietăți geometrice (de exemplu incidența) prin dualitate?

Dualitate – “dicționar” concepte și configurații

Plan primal	Plan dual
Punct p	Dreaptă neverticală p^*
Dreaptă neverticală d	Punct d^*
Dreaptă determinată de două puncte	Punct de intersecție a două drepte
Punctul p deasupra dreptei d	Punctul d^* deasupra dreptei p^*
Segment	Fascicul de drepte (<i>wedge</i>)

5.4 Programare liniară

Problematizare

- **Formulare generală (în spațiul d -dimensional):**

$$\text{maximizează}(c_1x_1 + c_2x_2 + \dots + c_dx_d)$$

date constrângerile liniare (inegalități)

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \dots + a_{1d}x_d \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \dots + a_{2d}x_d \leq b_2 \\ \dots \\ a_{n1}x_1 + a_{n2}x_2 + \dots + a_{nd}x_d \leq b_n \end{cases}$$

- **Terminologie:** date de intrare, funcție obiectiv, constrângeri, regiune realizabilă (fezabilă)
- **Exemple:** probleme de programare liniară 1-dimensională, 2-dimensională.

Rezultate

- **Lemă.** (Pentru $d = 1$) *Un program liniar 1-dimensional poate fi rezolvat în timp liniar.*
- **Interpretare a cerinței de maximizare:** Maximizarea funcției obiectiv revine la a determina un punct al cărui vector de poziție are proiecția maximă de direcția dată de vectorul $\vec{c} = (c_1, c_2, \dots, c_d)$.
- Pentru o problemă de programare liniară în plan ($d = 2$) pot fi distinse patru situații: (i) o soluție unică; (ii) toate punctele de pe o muchie sunt soluții; (iii) regiunea fezabilă este nemărginită și pot fi găsite soluții de-a lungul unei semidrepte; (iv) regiunea fezabilă este vidă.

Algoritm LPMARG2D (H, \vec{c}, m_1, m_2)

- **Input.** Un program liniar ($H \cup \{m_1, m_2\}, \vec{c}$) din \mathbb{R}^2
 - **Output.** Dacă ($H \cup \{m_1, m_2\}, \vec{c}$) nu e realizabil (fezabil), raportează. În caz contrar, indică punctul cel mai mic lexicografic p care maximizează $f_{\vec{c}}(p)$.
1. $v_0 \leftarrow$ “colțul” lui c_0

2. fie h_1, h_2, \dots, h_n semiplanele din H
3. **for** $i \leftarrow 1$ **to** n
4. **do if** $v_{i-1} \in h_i$
5. **then** $v_i \leftarrow v_{i-1}$
6. **else** $v_i \leftarrow$ punctul p de pe d_i care maximizează $f_{\vec{c}}(p)$ date
constrângerile din H_i
7. **if** p nu există
8. **then** raportează “nefezabil” **end**
9. **return** v_n

Algoritm aleatoriu

- Pasul 2. este înlocuit cu:
 - 2'. Calculează o permutare arbitrară a semiplanelor, folosind o procedură adecvată.
- Algoritmul incremental LPMARG2D are complexitate-timp $O(n^2)$, iar varianta bazată pe alegerea aleatorie a semiplanelor are complexitate-timp medie $O(n)$ (n este numărul semiplanelor).

5.5 Exerciții, probleme, aplicații

Exercițiul 5.1 Considerăm două ”piese” poligonale \mathcal{P}_1 și \mathcal{P}_2 , având normalele fețelor standard date de vectorii:

$$\mathcal{P}_1 : \nu_1 = \left(\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right); \nu_2 = (1, 0); \nu_3 = (0, 1); \nu_4 = (-1, 0); \nu_5 = \left(-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\right);$$

$$\mathcal{P}_2 : \nu_1 = \left(\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right); \nu_2 = (1, 0); \nu_3 = (0, 1); \nu_4 = (-1, 0); \nu_5 = \left(-\frac{\sqrt{2}}{2}, -\frac{\sqrt{2}}{2}\right).$$

(în această ordine). Stabiliți care dintre piese poate fi extrasă din matrița asociată prin deplasare în direcția verticală (dată de $(0, 1)$). Desenați!

Exercițiul 5.2 Considerăm semiplanele S_λ, S', S'' date de inecuațiile

$$S_\lambda : x - y - \lambda \leq 0 \ (\lambda \in \mathbb{R}), \quad S' : x - 1 \geq 0, \quad S'' : y - 5 \geq 0.$$

Discutați, în funcție de λ , natura intersecției $S_\lambda \cap S' \cap S''$.

Exercițiul 5.3 Dați exemplu de cinci semiplane, dintre care trei semiplane inferioare și două superioare, astfel încât intersecția lor să fie un triunghi.

Exercițiul 5.4 Fie punctul $p = (-1, 1)$; dreapta $d : (y = 3x + 4)$. Verificați că $p \in d$ și că $d^* \in p^*$.

Exercițiul 5.5 Fie punctele $p_1 = (2, 5)$; $p_2 = (1, 6)$. Scrieți ecuația dreptei $p_1 p_2$ și detaliați (cu calcule explicite!) configurația din planul dual.

Exercițiul 5.6 Fie dreapta $d : (y = 2x + 1)$ și $p = (1, 8)$. Verificați că p este deasupra lui d și că d^* este deasupra lui p^* .

Exercițiul 5.7 (i) Fie dreapta $d : (y = 2x - 3)$. Alegeți două puncte distincte $P, Q \in d$, determinați dualele d^*, P^*, Q^* și verificați că $\{d^*\} = P^* \cap Q^*$.

(ii) Determinați duala următoarei configurații: *Fie trei drepte care trec prin același punct M ; pe fiecare dreaptă se ia câte un punct (diferit de M), astfel ca aceste puncte să fie coliniare.*

Exercițiul 5.8 Fie configurația: *trei drepte care trec prin același punct; pe fiecare dreaptă se alege un punct, diferit de punctul comun al celor trei drepte.* Descrieți configurația duală. Desenați!

Exercițiul 5.9 Dați exemplu de problemă de programare liniară pentru care regiunea fezabilă să fie un pătrat, iar optimul (maximul) să fie atins în colțul din dreapta sus.

Exercițiul 5.10 Dați exemplu de problemă de programare liniară pentru care algoritmul prezentat la curs să aibă timp total de rulare liniar.

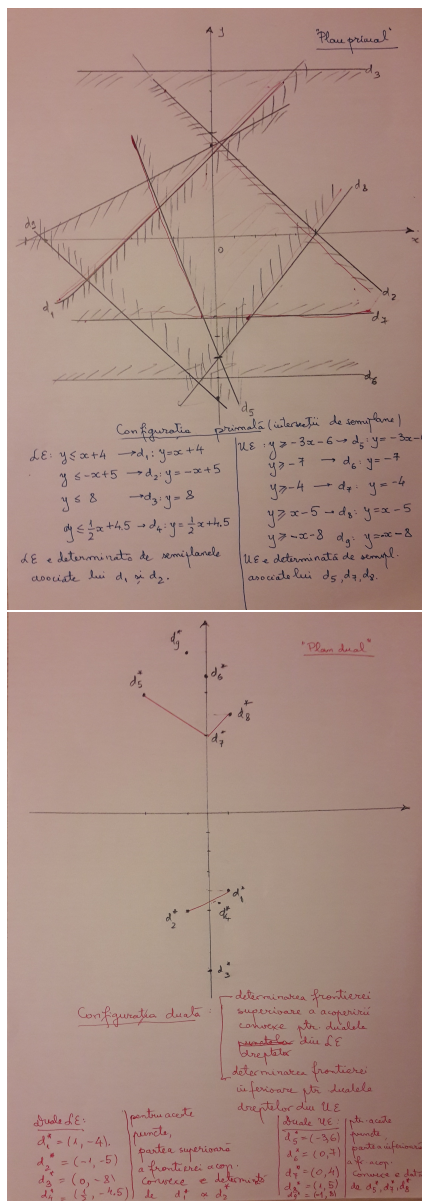
Exercițiul 5.11 Fie hiperplanele H_1, H_2, H_3, H_4 date de inecuațiile

$$H_1 : y \geq 1; \quad H_2 : y \leq 5; \quad H_3 : x \geq 0; \quad H_4 : x - y \geq \lambda,$$

unde $\lambda \in \mathbb{R}$ este un parametru. Scrieți un algoritm care să indice natura intersecției $H_1 \cap H_2 \cap H_3 \cap H_4$.

5.6 Anexa

Plan primal și plan dual: completare. Configurația din planul primal corespunde problemei intersecției de semiplane. Prin separarea în semiplane inferioare/superioare și dualitate se face transferul la două probleme distincte (marginea superioară/inferioară a unei acoperiri convexe, în care sunt considerate doar punctele corespunzătoare).



Capitolul 6

Diagrame Voronoi

6.1 Generalități

Problematizare

- Se consideră o mulțime de puncte (oficiile poștale) din plan. Care este cel mai apropiat?
- Ideea de a delimita “zone de influență” a apărut cu multă vreme în urmă (de exemplu în **lucrările lui Descartes**, dar și în **legătură cu alte probleme**; este utilizată în mod curent în varii domenii. În plus, astfel de “împărțiri” apar **în natură**.

Formalizare

- Fie $\mathcal{P} = \{P_1, P_2, \dots, P_n\}$ o mulțime de puncte din planul \mathbb{R}^2 , numite **situri**.
- **Diagrama Voronoi** a lui \mathcal{P} (notată $\text{Vor}(\mathcal{P})$) este o divizare a planului \mathbb{R}^2 în n celule $\mathcal{V}(P_1), \dots, \mathcal{V}(P_n)$ cu proprietatea că

$$P \in \mathcal{V}(P_i) \Leftrightarrow d(P, P_i) \leq d(P, P_j), \quad \forall j = 1, \dots, n.$$

- Două celule adiacente au în comun o *muchie* sau un *vârf* (punct de intersecție a muchiilor).
- **Atenție!** Vârfurile lui $\text{Vor}(\mathcal{P})$ sunt diferite de punctele din \mathcal{P} .
- Uneori, prin abuz de limbaj, este precizată doar împărțirea în muchii / vârfuri.
- Diagrame Voronoi pot fi construite pentru **diverse funcții distanță** (e.g. **distanța Manhattan**); forma celulelor depinde de **forma “cercului”** în raport cu funcția distanță respectivă.

6.2 Proprietăți

Proprietăți elementare

- Celula asociată unui punct este o intersecție de semiplane:

$$\mathcal{V}(P_i) = \bigcap_{j \neq i} h(P_i, P_j),$$

unde $h(P_i, P_j)$ este semiplanul determinat de mediatoarea segmentului $[P_i P_j]$ care conține punctul P_i .

- În particular: fiecare celulă este o mulțime convexă.
- Aplicabilitate: algoritm (lent) de determinare a diagramei Voronoi.

Structura unei diagrame Voronoi

- Fie $\mathcal{P} = \{P_1, P_2, \dots, P_n\}$ o mulțime de puncte din planul \mathbb{R}^2 .
- Dacă toate punctele sunt coliniare, atunci diagrama Voronoi asociată $\text{Vor}(\mathcal{P})$ conține $n - 1$ *drepte paralele* între ele (în particular, pentru $n \geq 3$, ea nu este conexă).
- În caz contrar, diagrama este conexă, iar muchiile sale sunt fie *segmente*, fie *semidrepte* (cui corespund acestea?).
- **Propoziție.** Fie o mulțime cu n situri. Atunci, pentru diagrama Voronoi asociată au loc inegalitățile

$$n_v \leq 2n - 5, \quad n_m \leq 3n - 6,$$

unde n_v este numărul de vârfuri ale diagramei și n_m este numărul de muchii al acesteia.

6.3 Diagrame Voronoi și triangulări Delaunay

Legătura cu triangulările legale / unghiular optime

- Construcție:
 - Mulțime de puncte \mathcal{P} în planul $\mathbb{R}^2 \implies$
 - Diagrama Voronoi $\text{Vor}(\mathcal{P}) \implies$
 - Graful dual $\mathcal{G}(\mathcal{P})$. **Noduri:** fețele diagramei Voronoi (siturile). **Arce:** dacă celulele (fețele diagramei Voronoi corespunzătoare) au o muchie comună \implies
 - Triangulare $\mathcal{T}_{\mathcal{P}}$ (numită **triangulare Delaunay**)
- **Propoziție.** Fie \mathcal{T} o triangulare a lui \mathcal{P} . Atunci \mathcal{T} este o triangulare Delaunay dacă și numai dacă pentru orice triunghi din \mathcal{T} cercul circumscris nu conține în interiorul său niciun punct al lui \mathcal{P} .
- **Teoremă.** O triangulare este legală dacă și numai dacă este o triangulare Delaunay.
- **Teoremă.** Orice triangulare unghiular optimă este o triangulare Delaunay. Orice triangulare Delaunay maximizează cel mai mic unghi, comparativ cu toate triangulările lui \mathcal{P} .
- **Întrebare:** Cum “funcționează” această construcție când punctele din \mathcal{P} sunt (de exemplu) vârfurile unui pătrat?

6.4 Un algoritm eficient

Algoritmul lui Fortune [1987]

- Complexitate: $O(n \log n)$.
- **Principiu (paradigmă):** sweep line / linie de baleiere.
- **Inconvenient:** la întâlnirea unui vârf al diagramei, linia de baleiere nu a întâlnit în că toate siturile (puncte din \mathcal{P}) care determină acest vârf!
- **Adaptare:** nu reținem informația legată de intersecția dintre linia de baleiere și diagramă, ci doar informația legată de partea diagramei care nu mai poate fi influențată de punctele situate de dincolo de linia de baleiere.
- **Concepte:**
 - beach line / linie parabolică
 - site event / eveniment de tip locație (apare un arc de parabolă)
 - circle event / eveniment de tip cerc (dispare un arc de parabolă)

Algoritmul

Input. O mulțime de situri $\mathcal{P} = \{p_1, \dots, p_n\}$ de situri în plan.

Output. Diagrama Voronoi $\text{Vor}(\mathcal{P})$ în interiorul unui *bounding box*, descrisă printr-o listă dublu înălțuită \mathcal{D} .

1. Inițializări: coada de evenimente $\mathcal{Q} \leftarrow \mathcal{P}$ (preprocesare: ordonare după y), statut (arbore balansat) $\mathcal{T} \leftarrow \emptyset$; listă dublu înălțuită $\mathcal{D} \leftarrow \emptyset$.
2. **while** $\mathcal{Q} \neq \emptyset$
3. **do** elimină evenimentul cu cel mai mare y din \mathcal{Q}
4. **if** evenimentul **ev** este un eveniment de tip sit
5. **then** $\text{PROCESSEVSIT}(p_i)$, cu $p_i = \mathbf{ev}$
6. **else** $\text{PROCESSEVCERC}(\gamma)$, cu $\gamma = \text{arc}(\mathbf{ev}) \in \mathcal{T}$
7. Nodurile interne încă prezente în \mathcal{T} corespund semidreptelor diagramei Voronoi. Consideră un *bounding box* care conține toate vârfurile diagramei Voronoi în interiorul său și leagă semidreptele de acest *bounding box*, prin actualizarea corespunzătoare a lui \mathcal{D} .
8. Traversează muchiile pentru a adăuga celulele diagramei și pointeri corespunzători.

Procedura $\text{PROCESSEVSIT}(p_i)$

1. Dacă \mathcal{T} este vidă, inserează p_i și revine, dacă nu continuă cu 2.–5.
2. Caută în \mathcal{T} arcul α situat deasupra lui p_i . Dacă frunza reprezentând α are un pointer către un eveniment de tip cerc **ev** din \mathcal{Q} , atunci **ev** este o alarmă falsă și trebuie șters.
3. Înlocuiește frunza lui \mathcal{T} care reprezintă α cu un subarbore cu trei frunze: cea din mijloc reține situl p_i și celelalte două situl p_j asociat lui α . Memorează perechile reprezentând punctele de racord în două noduri interne. Efectuează rebalansări în \mathcal{T} , dacă este necesar.

4. Generează noi înregistrări de tip semi-muchie în structura diagramei Voronoi (\mathcal{D}), pentru muchiile care separă celulele $V(p_i)$ și $V(p_j)$, corespunzând celor două noi puncte de racord.
5. Verifică tripletele de arce consecutive nou create, pentru a verifica dacă muchiile corespunzătoare punctelor de racord se întâlnesc. Dacă da, inserează evenimente de tip cerc în \mathcal{Q} și adaugă pointeri de la nodurile lui \mathcal{T} la evenimentele corespunzătoare din \mathcal{Q} .

Procedura PROCESSEVCERC (γ)

1. Șterge frunza $\gamma \in \mathcal{T}$ care corespunde arcului de cerc α care dispare. Actualizează în nodurile interne perechile care corespund punctelor de racord. Efectuează rebalansări în \mathcal{T} , dacă este necesar. Șterge toate evenimentele de tip cerc care îi corespund lui α (cu ajutorul pointerilor de la predecesorul și succesorul lui γ în \mathcal{T}).
2. Adaugă centrul cercului care determină evenimentul ca înregistrare de tip vârf în \mathcal{D} . Creează înregistrări de tip semi-muchie corespunzând noului punct de racord de pe linia parabolică și asignează pointeri corespunzători.
3. Verifică tripletele de arce consecutive nou create (care au fostii vecini ai lui α în centru), pentru a verifica dacă muchiile corespunzătoare punctelor de racord se întâlnesc. Dacă da, inserează evenimente de tip cerc în \mathcal{Q} și adaugă pointeri de la nodurile lui \mathcal{T} la evenimentele corespunzătoare din \mathcal{Q} .

Rezultate principale

- **Teoremă.** *Diagrama Voronoi a unei mulțimi de n situri poate fi determinată cu un algoritm de tip line sweep de complexitate $O(n \log n)$, folosind $O(n)$ spațiu de memorie.*
- **Teoremă.** *Triangularea Delaunay a unei mulțimi de n situri poate fi determinată cu un algoritm de tip line sweep de complexitate $O(n \log n)$, folosind $O(n)$ spațiu de memorie.*

6.5 Exerciții, probleme, aplicații

Exercițiul 6.1 Determinați, folosind metoda diagramelor Voronoi, triangularea Delaunay pentru mulțimea formată din punctele $A = (3, 5)$, $B = (6, 6)$, $C = (6, 4)$, $D = (9, 5)$ și $E = (9, 7)$.

Exercițiul 6.2 Determinați numărul de semidrepte conținute în diagrama Voronoi asociată mulțimii de puncte $\mathcal{M} = \{A_0, \dots, A_5, B_0, \dots, B_5, C_0, \dots, C_5\}$, unde $A_i = (i + 1, i + 1)$, $B_i = (-i, i)$ și $C_i = (0, i)$, pentru $i = 0, \dots, 5$.

Exercițiul 6.3 Dați exemplu de mulțimi \mathcal{M}_1 și \mathcal{M}_2 din \mathbb{R}^2 , fiecare având câte 4 puncte, astfel ca, pentru fiecare dintre ele, diagrama Voronoi asociată să conțină exact 3 semidrepte, iar diagrama Voronoi asociată lui $\mathcal{M}_1 \cup \mathcal{M}_2$ să conțină exact 6 semidrepte.

Exercițiul 6.4 Fie punctele $A_1 = (5, 1)$, $A_2 = (7, -1)$, $A_3 = (9, -1)$, $A_4 = (7, 3)$, $A_5 = (11, 1)$, $A_6 = (9, 3)$. Dați exemplu de mulțime de două puncte $\{A_7, A_8\}$ cu proprietatea că diagrama Voronoi asociată mulțimii $\{A_1, \dots, A_8\}$ are exact 4 muchii de tipul semidreaptă (explicați construcția făcută).

Exercițiul 6.5 Demonstrați că dacă punctele din mulțimea \mathcal{P} nu sunt coliniare, diagrama Voronoi $\text{Vor}(\mathcal{P})$ nu poate conține drepte.

Capitolul 7

Probleme de căutare și localizare

7.1 Căutare ortogonală

Motivație: Exemplu.

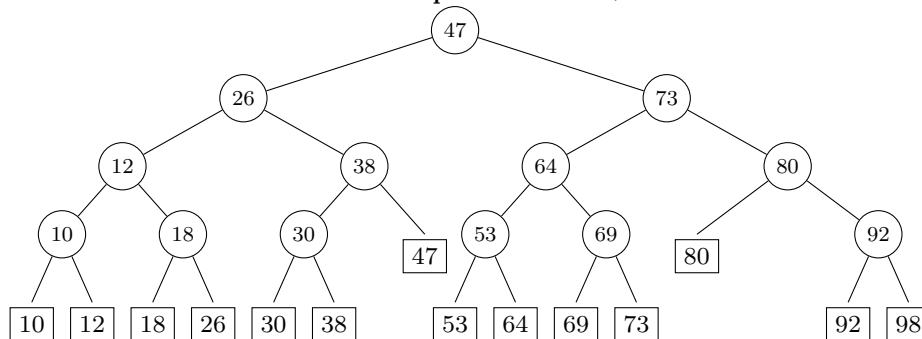
- Baza de date a unei bănci: informații numerice referitoare la clienți: data nașterii, număr de copii, venitul lunar, valoarea depozitelor, valoarea ratelor de plată, valoarea comisioanelor plătite anual, etc. → stocarea se realizează folosind puncte dintr-un spațiu numeric d -dimensional \mathbb{R}^d .
- A identifica un “grup-țintă” de clienți (de exemplu pentru lansarea unui produs), având anumite caracteristici — e.g. vârsta între 30-40 ani, 2-4 copii, un venit lunar între 3000-5000 lei, etc. revine la efectuarea căutări prin care să fie determinate punctele situate într-un “paralelipiped” d -dimensional.

Căutare 1-dimensională: formularea problemei

Cadru. Fie $M = \{a_1, a_2, \dots, a_n\}$ o mulțime de numere reale. Fie $I = [x, x'] \subset \mathbb{R}$ un interval real. Se dorește determinarea elementelor lui M situate în intervalul I .

Structura de date utilizată: Arbore binar de căutare echilibrat.

Exemplu de arbore \mathcal{T}

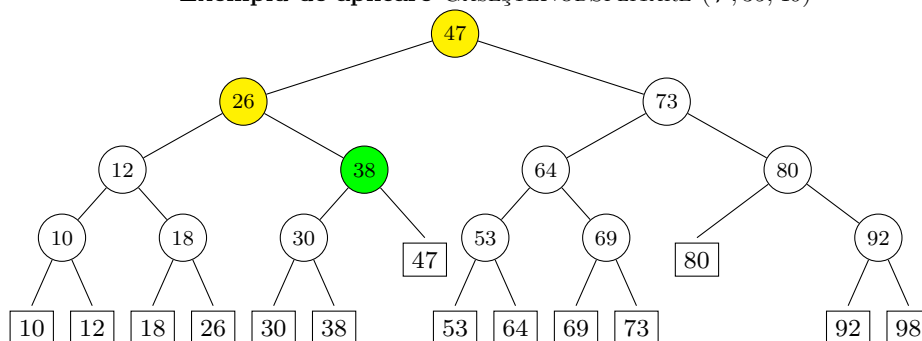


Procedura GĂSEȘTE NOD SPLITARE (\mathcal{T}, x, x')

- **Input.** Un arbore binar de căutare echilibrat \mathcal{T} , două numere reale $x < x'$.
- **Output.** Nodul v în care se realizează splitarea drumurilor către x și x' sau frunza pe care ambele drumuri se încheie.

1. $v \leftarrow \text{root}(\mathcal{T})$
2. **while** v nu este frunză **and** $(x' \leq x_v \text{ or } x \geq x_v)$
3. **do if** $x' \leq x_v$
4. **then** $v \leftarrow lc(v)$
5. **else** $v \leftarrow rc(v)$
6. **return**(v)

Exemplu de aplicare GĂSEȘTE NOD SPLITARE ($\mathcal{T}, 35, 40$)



În nodul $x_v = 47$ este ales $lc(v)$

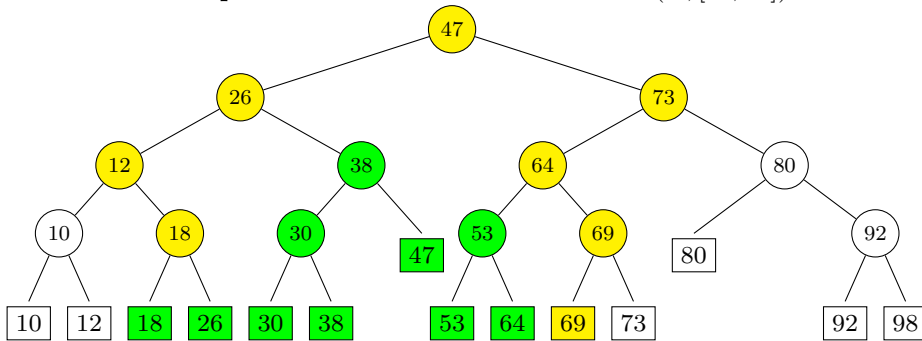
În nodul $x_v = 26$ este ales $rc(v)$

În nodul $x_v = 38$ se realizează splitarea, acest nod fiind returnat

Algoritm CĂUTARE 1-DIMENSIONALĂ ($\mathcal{T}, [x, x']$)

- **Input.** Un arbore binar de căutare echilibrat \mathcal{T} , un interval $[x, x']$.
 - **Output.** Toate elementele din \mathcal{T} aflate în intervalul $[x, x']$.
1. $v_{split} \leftarrow \text{GĂSEȘTE_NOD_SPLITARE}(\mathcal{T}, x, x')$
 2. **if** v_{split} este frunză
 3. **then** verifică dacă elementul memorat în v_{split} trebuie raportat
 4. **else** // Caută drumul spre x , raportează subarborii din dreapta
 5. $v \leftarrow lc(v_{split})$
 6. **while** v nu este o frunză
 7. **do if** $x \leq x_v$
 8. **then** RAPORTEAZĂSUBARBORE($rc(v)$)
 9. $v \leftarrow lc(v)$
 10. **else** $v \leftarrow rc(v)$
 11. Verifică dacă elementul din frunza v trebuie raportat
 - 12.-19. Efectuează pași similari pentru x'

Aplicare CĂUTARE 1-DIMENSIONALĂ ($\mathcal{T}, [18, 68]$)



Nodul de splitare este 47

Nodurile / frunzele colorate cu verde sunt raportate \rightarrow Subarborii colorati cu verde sunt raportați

Nodurile colorate cu galben sunt vizitate, fără a fi raportate

Teoremă. (Rezultatul principal - căutare 1D) Fie M o mulțime de n puncte din \mathbb{R} . Mulțimea M poate fi memorată într-un arbore binar de căutare echilibrat, folosind $O(n)$ memorie și cu timp de construcție $O(n \log n)$. Determinarea unor puncte dintr-un interval I poate fi realizată cu complexitate-timp $O(k + \log n)$, unde k este numărul de puncte din $M \cap I$.

Teoremă. (Rezultatul principal - căutare 2D) Fie M o mulțime de n puncte din planul \mathbb{R}^2 . Un arbore de intervale (range tree) pentru M necesită $O(n \log n)$ memorie și poate fi construit în timp $O(n \log n)$. Determinarea unor puncte dintr-un dreptunghi D poate fi realizată cu complexitate-timp $O(k + \log^2 n)$, unde k este numărul de puncte din $M \cap D$.

7.2 Localizarea punctelor – Hărți trapezoidale

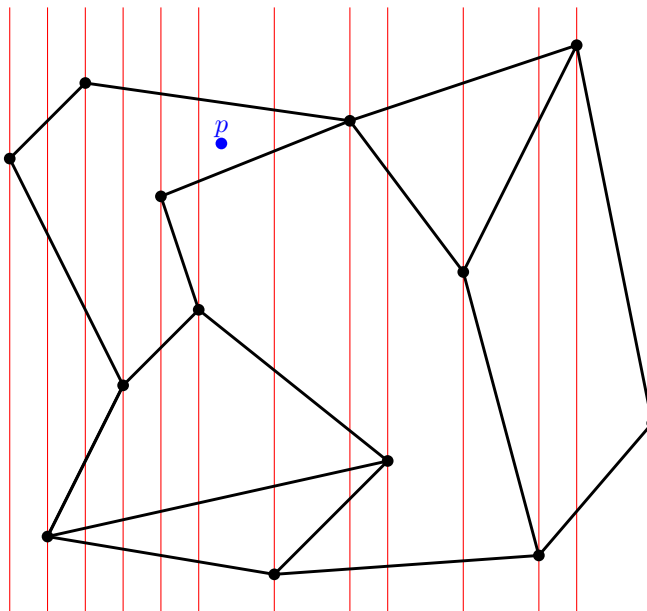
Problematizare

- Căutare cu [Google Maps](#)
- *Interogare pentru localizarea unui punct*: dată o hartă și un punct p , indicat prin coordonatele sale, să se determine regiunea hărții în care este situat p .
- *Harta*: subdiviziune planară, formată din vârfuri, (semi)muchii, fețe.
- Necesități: pre-procesare a informației; interogare rapidă.

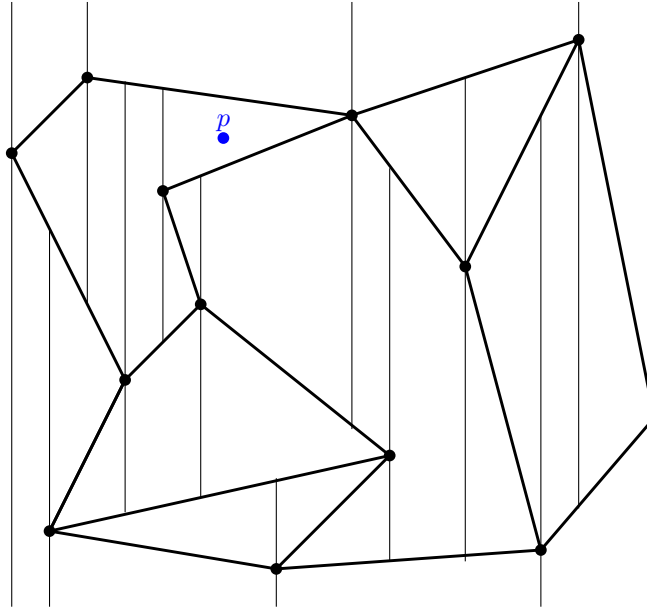
Formalizare

- Fie \mathcal{S} o subdiviziune planară cu n muchii. *Problema localizării unui punct* revine la
 - a reține informațiile referitoare la \mathcal{S} pentru a putea răspunde la interogări de tipul:
 - dat un punct p , se raportează fața f care îl conține pe p ; în cazul în care p este situat pe un segment sau coincide cu un vârf, este precizat acest lucru.
- Lucrul cu coordonate: folosirea relației de ordine!

Exemplu - rafinare folosind benzi verticale



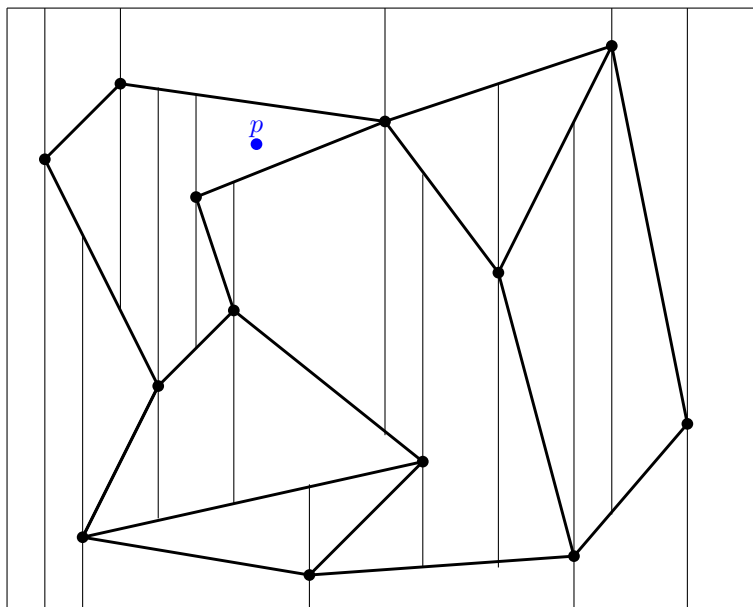
Exemplu - rafinare eficientă



Simplificări și ipoteze

- Se consideră o mulțime S de n segmente astfel ca oricare două dintre ele (i) fie nu au niciun punct comun; (ii) fie au un vârf comun.
- *Simplificare 1:* Se consideră un dreptunghi D cu laturile paralele cu axele de coordonate care include toată subdiviziunea inițială.
- *Simplificare 2:* Se presupune că nu există două vârfuri (extremități ale segmentelor din S) distincte care au aceeași coordonată x (în particular nu există segmente verticale).
- *Concluzie:* Se consideră o mulțime de n segmente S care verifică ipotezele de mai sus: *mulțime de segmente în poziție generală*. **Harta trapezoidală / descompunere verticală / descompunere cu trapeze** (*trapezoidal map*) $\mathcal{T}(S)$ a lui S este subdiviziunea indusă de S , dreptunghiul D și de extensiile verticale inferioare și superioare (concept introdus de Seidel, 1991).

Exemplu - hartă trapezoidală



Hărți trapezoidale – probleme studiate

- Descrierea obiectelor geometrice din care sunt formate – ce informații se rețin?
- Aspecte legate de complexitate?
- Structuri de date adecvate?
- Un algoritm eficient?

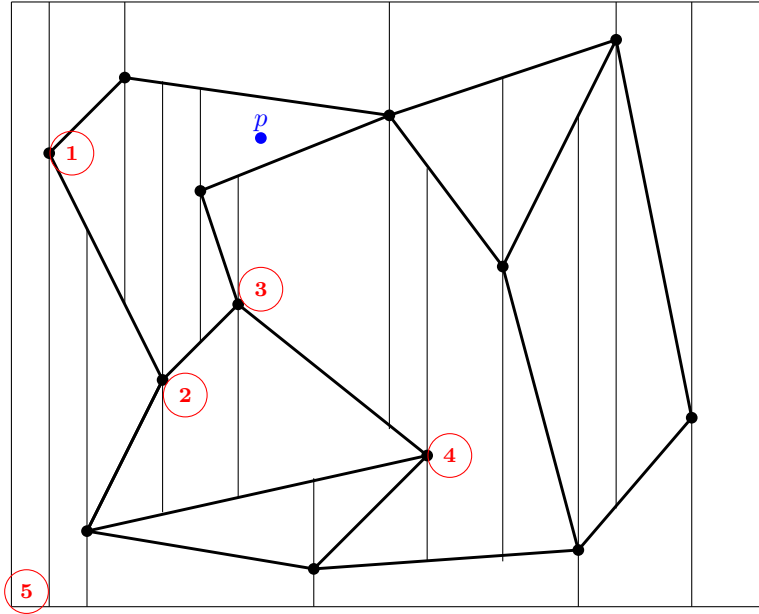
Descrierea obiectelor

- **Lema 1.** Fie S o mulțime de segmente în poziție generală. Fiecare față a unei hărți trapezoidale $\mathcal{T}(S)$ are una sau două margini verticale și exact două margini ne-verticale.

De fapt: fiecare față este un trapez, sau un dreptunghi sau un triunghi (ultimele putând fi privite drept cazuri particulare de trapeze).

- Ce informații geometrice sunt reținute pentru un trapez?
- $\mathbf{t}(T)$, $\mathbf{b}(T)$, $\mathbf{lp}(T)$, $\mathbf{rp}(T)$ determină în mod unic un trapez fixat T .
 $\mathbf{t}(T)$, $\mathbf{b}(T)$ sunt **segmente**, iar $\mathbf{lp}(T)$, $\mathbf{rp}(T)$ sunt **vârfuri** (extremități ale segmentelor)
- Există cinci cazuri posibile pentru marginea stângă \mathbf{lp} (analog pentru marginea dreaptă \mathbf{rp}).

Exemplu - hartă trapezoidală și cazuri posibile pentru marginea stângă



Complexitate și alte aspecte cantitative

- **Lema 2.** Fie S o mulțime de n segmente în poziție generală. Harta trapezoidală $\mathcal{T}(S)$ conține cel mult $6n + 4$ vârfuri și cel mult $3n + 1$ trapeze.
- **Lema 3.** Fie S o mulțime de n segmente în poziție generală. Fiecare trapez T este adiacent cu cel mult patru trapeze (cel mult un vecin stânga superior, cel mult un vecin stânga inferior, cel mult un vecin dreapta superior, cel mult un vecin dreapta inferior).

Structura de date

- Înregistrări pentru segmentele din S și vârfuri (extremitățile segmentelor).
- Înregistrări pentru trapeze: pointeri t , b , lp , rp și pointeri către cei (cel mult) patru vecini.
- **Structura de căutare:** \mathcal{D} este un graf orientat aciclic cu o singură rădăcină și exact o frunză pentru fiecare trapez din $\mathcal{T}(S)$.
- **Noduri și teste asociate:**
 - x -nod, etichetat cu o extremitate a unui segment; pentru un punct p testul asociat: *este punctul p situat la stânga sau la dreapta dreptei verticale care trece prin extremitatea memorată în acest nod?*
 - y -nod, etichetat cu un segment; pentru un punct p testul asociat: *este punctul p situat deasupra sau dedesubtul segmentului memorat în acest nod?*

Algoritm HARTATRAPEZOIDALA

- **Input.** O mulțime S de n segmente în poziție generală.
 - **Output.** Harta trapezoidală $\mathcal{T}(S)$ și o structură de căutare \mathcal{D} pentru $\mathcal{T}(S)$, într-un dreptunghi R cu laturile paralele cu axele.
1. Determină dreptunghiul D .
 2. Generează o permutare s_1, s_2, \dots, s_n a segmentelor din S .
 3. **for** $i \leftarrow 1$ **to** n
 4. **do** găsește mulțimea de trapeze T_0, T_1, \dots, T_k care intersectează segmentul s_i
 5. elimină T_0, \dots, T_k și le înlocuiește cu trapezele nou apărute
 6. elimină frunzele corespunzătoare din \mathcal{D} și creează noi frunze, actualizează \mathcal{D}

Rezultatul principal

Teoremă. Fie S o mulțime de n segmente în poziție generală. Algoritmul HARTATRAPEZOIDALA determină harta trapezoidală $\mathcal{T}(S)$ și o structură de căutare \mathcal{D} pentru $\mathcal{T}(S)$ în timp mediu $O(n \log n)$. Memoria medie ocupată de structura de căutare este $O(n)$ și pentru un punct arbitrar q timpul mediu de localizare este $O(\log n)$.

7.3 Aplicație: mișcarea unui robot-punct

Algoritm DETERMINASPATIULIBER (S)

- **Input.** O mulțime \mathcal{P} de poligoane disjuncte.
 - **Output.** O hartă trapezoidală \mathcal{C}_l a spațiului liber (pentru un robot-punct).
1. Fie S mulțimea muchiilor poligoanelor din \mathcal{P} .
 2. Determină harta trapezoidală $\mathcal{T}(S)$, folosind algoritmul HARTATRAPEZOIDALA.
 3. Elimină trapezele situate în interiorul poligoanelor și returnează subdiviziunea obținută.

Algoritm DETERMINADRUM ($\mathcal{T}(\mathcal{C}_l), \mathcal{G}_d, M_{\text{start}}, M_{\text{end}}$)

- **Input.** Harta trapezoidală $\mathcal{T}(\mathcal{C}_l)$ a spațiului liber, graful drumurilor \mathcal{G}_d , punctul de start M_{start} , punctul final M_{end} .
 - **Output.** Un drum de la M_{start} la M_{end} , dacă există. În caz contrar, algoritmul precizează că nu există un drum.
1. caută trapeze în $\mathcal{T}(\mathcal{C}_l)$ conținând M_{start} , respectiv M_{end} .
 2. **if** există Δ_{start} , respectiv Δ_{end} conținând M_{start} , respectiv M_{end}
 3. **then** fie v_{start} și v_{end} centrele $\Delta_{\text{start}}, \Delta_{\text{end}}$ (noduri din \mathcal{G}_d)

4. caută un drum în \mathcal{G}_d de la v_{start} la v_{end} folosind BFS
5. **if** există drum δ
6. **then** indică drumul $[M_{\text{start}}v_{\text{start}}] \cup \delta \cup [v_{\text{end}}M_{\text{end}}]$
7. **else** raportează că nu există drum de la M_{start} la M_{end}
8. **else** raportează că nu există drum de la M_{start} la M_{end}

Rezultatul principal

Teoremă. *Fie \mathcal{R} un robot-punct care se deplasează într-o mulțime S de obstacole poligonale, având în total n muchii. Utilizând timp mediu de pre-procesare $O(n \log n)$ pentru mulțimea S , un drum liber de coliziuni între două puncte fixate poate fi calculat (dacă există!) în timp mediu $O(n)$.*

7.4 Exerciții, probleme, aplicații

Exercițiul 7.1 Pentru arborele \mathcal{T} din secțiunea 7.1 explicați aplicarea GĂSEȘTE_NOD_SPLITARE (\mathcal{T} , 50, 81) și CĂUTARE 1-DIMENSIONALĂ (\mathcal{T} , [12, 63]).

Exercițiul 7.2 Fie punctele $A = (-2, 2)$, $B = (-1, -1)$, $C = (0, 0)$, $D = (1, -2)$, $E = (2, 1)$, $F = (3, 3)$. Descrieți arborele binar asociat; submulțimile canonice și structurile asociate nodurilor copii ai rădăcinii.

Exercițiul 7.3 Considerăm un pătrat având laturile paralele cu axele de coordonate, în interiorul căruia se află un alt pătrat, astfel ca laturile sale să facă un unghi de 30° cu axele de coordonate. Stabiliți câte trapeze are harta trapezoidală a regiunii situate între cele două pătrate. Câte dintre acestea sunt degenerate?

Exercițiul 7.4 Fie punctele $A = (1, 1)$, $B = (2, 6)$, $C = (5, 3)$, $D = (4, 7)$, $E = (8, 4)$, $F = (10, 7)$, $G = (6, 9)$, considerate în interiorul dreptunghiului R delimitat de axele de coordonate și de dreptele date de ecuațiile $x = 12$, respectiv $y = 12$.

(a) Câte trapeze are harta trapezoidală asociată subdiviziunii planare induse de triunghiul ABC și patrulaterul $DEFG$?

(b) Indicați un drum parcurs de un robot-punct de la $M_{\text{start}} = (4, 1)$ la $M_{\text{end}} = (9, 9)$, determinat de algoritmul DETERMINADRU.

Exercițiul 7.5 Considerăm două triunghiuri T_1 și T_2 (astfel ca laturile lor să fie segmente în poziție generală), în interiorul unui *bounding box* R . Câte trapeze are harta trapezoidală asociată? Depinde acest număr de poziția relativă a triunghiurilor?

Exercițiul 7.6 Considerăm pătratul \mathcal{P} delimitat de dreptele $x = \pm 10$, $y = \pm 10$ (*bounding box*) și punctele $A = (2, 0)$, $B = (0, 2)$, $C = (-2, 0)$, $D = (0, \lambda)$, cu $\lambda \in [-9, 9]$. Fie \mathcal{Q} acoperirea convexă a mulțimii $\{A, B, C, D\}$. Scrieți un algoritm care să decidă câte trapeze are harta trapezoidală a regiunii situate între pătratul \mathcal{P} și poligonul \mathcal{Q} .

Bibliografie

- [1] M. de Berg, M. van Kreveld, M. Overmars și O. Schwarzkopf, *Computational Geometry, Algorithms and Applications*, Springer, 2000.
 - [2] S. Devadoss, J. O'Rourke, *Discrete and Computational Geometry*, Princeton University Press, 2011.
 - [3] B. Gärtner, M. Hoffmann, *Computational Geometry*. Note de curs, ETH Zürich. <http://www.ti.inf.ethz.ch/ew/courses/CG13/lecture/cg-2013.pdf>.
 - [4] D. Lee, F. Preparata, *Computational Geometry - A Survey*, IEEE Transactions on Computers, **33** (1984), 1072-1101.
 - [5] F. Preparata și M. Shamos, *Computational Geometry: An Introduction*, Springer, 1985.
-
- [6] L. Bădescu, *Geometrie*, Editura Universității București, 2000.
 - [7] M. do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice Hall, 1976.
 - [8] Gh. Galbură și F. Radó, *Geometrie*, Editura Didactică și Pedagogică, București, 1979.
 - [9] A. Gray, *Modern Differential Geometry of Curves and Surfaces with Mathematica*, CRC Press, 1999.
 - [10] I. Hiriță, S. Leiko, L. Nicolescu, G. Pripoe, *Geometrie diferențială. Probleme. Aplicații*, București, 1999.
 - [11] M.I. Munteanu, *Algoritmi geometrici 2D și aplicații în CAGD*, Editura Universității "Al. I. Cuza" Iași, 2005.
 - [12] L. Nicolescu, *Curs de geometrie*, București, 2002.
 - [13] L. Ornea și A. Turtoi, *O introducere în geometrie*, Editura Theta, București, 2000.
 - [14] M.S. Stupariu, *Geometrie analitică*, București, 2008.

Anexa A

Proiecte

1. Acoperirea convexă a unui poligon arbitrar.

Input: Un poligon \mathcal{P} din \mathbb{R}^2 .

Output: Vârfurile acoperirii convexe $\text{Conv}(\mathcal{P})$ (determinate în timp liniar).
Reprezentare grafică.

2. Invarianța acoperirii convexe la transformări afine.

Input: O mulțime \mathcal{P} din \mathbb{R}^2 , o transformare afină $\varphi : \mathbb{R}^2 \rightarrow \mathbb{R}^2$.

Output: Acoperirea convexă a imaginii lui \mathcal{P} prin φ (coincide cu imaginea lui $\text{Conv}(\mathcal{P})$ prin φ). Reprezentare grafică – ilustrează cât mai sugestiv proprietatea de invarianță la transformări afine a acoperirii convexe.

3. Poziția unui punct față de un poligon convex.

Input: Un poligon convex \mathcal{P} din \mathbb{R}^2 , un punct $A \in \mathbb{R}^2$.

Output: Precizează poziția lui A față de \mathcal{P} (în interior, pe laturi, în exterior), folosind o împărțire convenabilă pe sectoare. Reprezentare grafică.

4. Poligon convex și punct exterior.

Input: Un poligon convex \mathcal{P} din \mathbb{R}^2 , un punct $A \in \mathbb{R}^2$ în exteriorul lui \mathcal{P} .

Output: Determină vârfurile acoperirii convexe $\text{Conv}(\mathcal{P} \cup \{A\})$ (ca listă ordonată, parcursă în sens trigonometric). Reprezentare grafică.

5. Poligoane cu laturi paralele.

Input: Două dreptunghiuri / poligoane convexe \mathcal{P}, \mathcal{Q} din \mathbb{R}^2 , disjuncte, având laturile paralele.

Output: Determină vârfurile acoperirii convexe $\text{Conv}(\mathcal{P} \cup \mathcal{Q})$ (ca listă ordonată, parcursă în sens trigonometric). Reprezentare grafică.

6. Cercuri.

Input: O mulțime de cercuri $\mathcal{C}_1, \dots, \mathcal{C}_q$ de rază 1, disjuncte, din planul \mathbb{R}^2 (sunt indicate centrele cercurilor), un punct $A \in \mathbb{R}^2$.

Output: Precizează poziția lui A față de $\text{Conv}(\mathcal{C}_1 \cup \dots \cup \mathcal{C}_q)$. Reprezentare grafică.

7. Triangulări ale poligoanelor – invarianța la transformări afine

Input: Un poligon \mathcal{P} din planul \mathbb{R}^2 , o transformare afină $\varphi : \mathbb{R}^2 \rightarrow \mathbb{R}^2$.

Output: Construiește o triangulare $\mathcal{T}_{\mathcal{P}}$ a lui \mathcal{P} și imaginea acesteia prin φ ca triangulare a lui $\varphi(\mathcal{P})$. Reprezentare grafică – ilustrează cât mai sugestiv modificarea triangulărilor poligoanelor după aplicarea unei transformări afine.

8. Poziția unui punct față de un poligon

Input: Un poligon \mathcal{P} din planul \mathbb{R}^2 , un punct $A \in \mathbb{R}^2$.

Output: Determină o triangulare $\mathcal{T}_{\mathcal{P}}$ a lui \mathcal{P} . Precizează poziția lui A față de \mathcal{P} (în exterior, pe laturi, în interior). În cazul în care este un punct interior, indică triunghiul din $\mathcal{T}_{\mathcal{P}}$ căruia A îi aparține.

9.* Vizibilitate

Input: Un poligon \mathcal{P} , un punct A în interiorul lui \mathcal{P} .

Output: Determină, folosind o triangulare $\mathcal{T}_{\mathcal{P}}$ a lui \mathcal{P} , regiunea lui \mathcal{P} care este vizibilă din A . Reprezentare grafică.

10. Triangulări ale mulțimilor de puncte – invarianța la transformări afine

Input: O mulțime \mathcal{M} de puncte, reprezentând vârfurile unui triunghi și puncte în interiorul acestuia.

Output: Construiește o triangulare $\mathcal{T}_{\mathcal{M}}$ a lui \mathcal{M} și imaginea acesteia prin φ ca triangulare a lui $\varphi(\mathcal{M})$. Reprezentare grafică – ilustrează cât mai sugestiv modificarea triangulărilor mulțimilor de puncte după aplicarea unei transformări afine.

11. Poziția unui punct față de o triangulare

Input: O mulțime \mathcal{M} de puncte, reprezentând vârfurile unui triunghi și puncte în interiorul acestuia, un punct $A \in \mathbb{R}^2$.

Output: Determină o triangulare $\mathcal{T}_{\mathcal{M}}$ a lui \mathcal{M} . Precizează poziția lui A față de \mathcal{M} (în exterior, pe laturi, în interior). În cazul în care este un punct interior, indică triunghiul din $\mathcal{T}_{\mathcal{M}}$ căruia A îi aparține.

12.* Echivalența triangulărilor

Input: O mulțime \mathcal{M} de puncte, două triangulări $\mathcal{T}_{\mathcal{M}}, \mathcal{T}'_{\mathcal{M}}$ ale lui \mathcal{M} .

Output: Precizează dacă cele două triangulări sunt echivalente, i.e. pot fi transformate una într-una într-un număr finit de pași, prin aplicarea unor modificări de tip "flip". Reprezentare grafică.