Geometrie computaţională – suport de curs

Mihai-Sorin Stupariu

Sem. I, 2016-2017

Cuprins

Capitolul 1

Preliminarii

1.1 Concepte de algebră liniară, geometrie afină şi euclidiană

Noţiuni de algebră liniară: spaţiu vectorial, vector, combinaţie liniară, liniar (in)dependenţă, sistem de generatori, bază, reper, dimensiune a unui spaţiu 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, spaţiu vectorial euclidian, vectori ortogonali, bază ortonormată, reper ortonormat.

Noţiuni de geometrie afină: vectorul determinat de două puncte, combinaţie afină, afin (in)dependenţă, acoperirea afină a unei mulţimi 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 secţiunea 1.2), segmentul determinat de două puncte, mulţime convexă, închiderea (înfăşurătoarea) convexă a unei mulţimi, aplicaţie afină (exemple: translaţie, omotetie, proiecţie, simetrie).

Noţiuni de geometrie euclidiană: distanţa dintre două puncte, reper cartezian ortonormat, izometrie, proiecţie centrală.

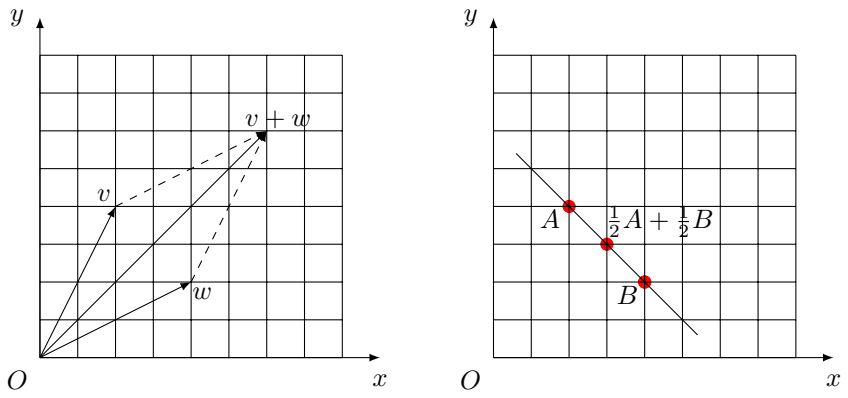
Detalii pot fi găsite în [6], [8], [13] [14].

1.2 Raportul unor puncte coliniare

Lema 1.1 Fie A şi B două puncte distincte în $\mathbb{R}^n$. Pentru orice punct $P \in AB$, $P \neq B$ există un unic scalar $r \in \mathbb{R}\backslash\{-1\}$ astfel ca $\overrightarrow{AP} = r\,\overrightarrow{PB}$. Reciproc, fiecărui scalar $r \in \mathbb{R} \setminus \{-1\}$, îi corespunde un unic punct $P \in AB$.

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

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



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

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

Figura 1.1: Vectori şi puncte: combinaţii liniare şi combinaţii 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 şi 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 $\mathbb{R}^n$ şi $M = \frac{1}{2}A + \frac{1}{2}B$. Atunci $r(A, M, B) = 1$, $r(M, A, B) = -\frac{1}{2}$.

Propoziţia 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 şi numai dacă $r(A, P, B) = \frac{\alpha}{1-\alpha}$;

(iii) $P = \frac{\alpha}{\alpha+\beta}A + \frac{\beta}{\alpha+\beta}B$ dacă şi numai dacă $r(A, P, B) = \frac{\beta}{\alpha}$.

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

(i) $r(A, P, B) > 0$ dacă şi numai dacă $P \in (AB)$;

(ii) $r(B, P, A) = \frac{1}{r(A,P,B)}$.

1.3 Coordonate carteziene şi coordonate polare

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

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

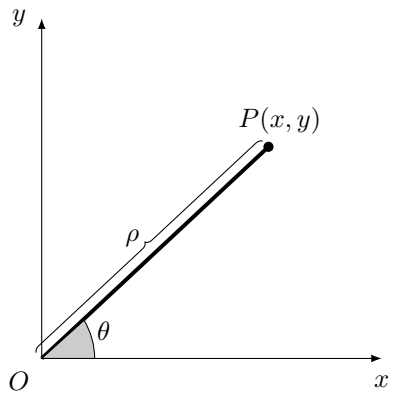


Figura 1.2: Punctul P are coordonatele carteziene (x, y) şi coordonatele polare (ρ, θ).

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

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

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

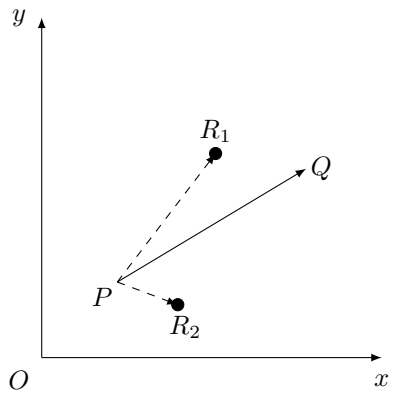


Figura 1.3: Poziţia relativă a două puncte faţă de un vector / o muchie orientată

Atunci R este situat:

(i) pe dreapta PQ $\Leftrightarrow \Delta(P,Q,R) = 0$ *("ecuaţia dreptei");*

(ii) "în dreapta" segmentului orientat $\overrightarrow{PQ} \Leftrightarrow \Delta(P,Q,R) < 0$*;*

(iii) "în stânga" segmentului orientat $\overrightarrow{PQ} \Leftrightarrow \Delta(P,Q,R) > 0$.

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

Aplicaţii.

- 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 şi de alta a unui segment / a unei drepte.

1.5 Exerciţii, probleme, aplicaţii

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

Exerciţiul 1.10 Determinaţi 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 şi $r(A,P,B) = 2$.

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

Exerciţiul 1.12 Ordonaţi 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.

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

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

Exerciţiul 1.15 Fie $P = (2, 2), Q = (4, 4)$. Stabiliţi, folosind testul de orientare, poziţia relativă a punctelor $R_1 = (8, 8), R_2 = (6, 0), R_3 = (-2, -1)$ faţă de muchia orientată $\overrightarrow{PQ}$. Care este poziţia aceloraşi puncte faţă de muchia orientată $\overrightarrow{QP}$?

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

Capitolul 2

Acoperiri convexe

2.1 Generalităţi

Conceptul de mulţime convexă:

Definiţia 2.1 *(i) Pentru $P, Q \in \mathbf{R}^m$, segmentul $[PQ]$ este mulţimea combinaţiilor convexe dintre P şi Q:*

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

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

Problematizare:
Mulţimile finite cu cel puţin două elemente nu sunt convexe $\longrightarrow$ necesară **acoperirea convexă**.

Acoperire convexă a unei mulţimi finite $\mathcal{P}$: caracterizări echivalente

- Cea mai "mică" (în sensul incluziunii) mulţime convexă care conţine $\mathcal{P}$.
- Intersecţia tuturor mulţimilor convexe care conţin $\mathcal{P}$.
- Mulţimea tuturor combinaţiilor convexe ale punctelor din $\mathcal{P}$. O **combinaţie convexă** a punctelor $P_1, P_2, \ldots, P_n$ este un punct P de forma

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

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

- Dacă $\mathcal{P}$ este finită, acoperirea sa convexă, Conv($\mathcal{P}$) este un **politop convex**.
- Cazuri particulare: $m = 1$ (segment); $m = 2$ (poligon); $m = 3$ (poliedru).
- Cazul $m = 1$: acoperirea convexă este un segment; algoritmic: parcurgere a punctelor (complexitate $O(n)$).
- **În continuare:** $m = 2$.
- **Problemă:** Cum determinăm, algoritmic, vârfurile acoperirii convexe (ca mulţime, ca listă)?

2.2 Algoritmi lenţi (naivi)

Determinarea punctelor extreme şi ordonarea lor

Definiţia 2.2 *Un punct M al unei mulţimi convexe $\mathcal{S}$ este* **punct extrem** *al lui $\mathcal{S}$ dacă* **nu** *există $A, B \in \mathcal{S}$ astfel ca $M \in [AB]$.*

Propoziţia 2.3 (Caracterizarea punctelor extreme). *Fie $\mathcal{P}$ o mulţime finită şi* $\mathrm{Conv}(\mathcal{P})$ *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 însuşi, vârf al triunghiului.*

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

Comentarii

- Cum se stabileşte dacă un punct P aparţine unui triunghi ABC sau interiorului acestuia? (folosind arii, verificând dacă P situat pe laturi sau situat de aceeaşi parte a fiecărei laturi ca şi vârful opus − "Testul de orientare", etc.)
- Coordonate carteziene (x, y) şi coordonate polare (ρ, θ) (pentru puncte pentru care relaţiile au sens):

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

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

Algoritmul lent 1

Input: O mulţime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.

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

1. $\mathcal{M} \leftarrow \emptyset$ /*$\mathcal{M}$ este mulţimea punctelor extreme*/
2. **for** $P \in \mathcal{P}$
3. **do** *valid* $\leftarrow$ 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 ΔABC sau pe laturile sale
6. **then** *valid* $\leftarrow$ false
7. **if** *valid*=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, obţinând lista $\mathcal{L}$

Comentarii

- Complexitatea: $O(n^4)$ (paşii 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 şi B sunt detectate ca fiind puncte extreme)!

Determinarea muchiilor frontierei

- Sunt considerate **muchiile 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 mulţime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conţinte vârfurile ce determină frontiera acoperirii convexe, parcursă în sensul trigonometric.

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

- Robusteţea: datorită erorilor de rotunjire este posibil ca algoritmul să nu returneze o listă coerentă de muchii.

2.3 Algoritmi "clasici"

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

Graham's scan [1972]

- Punctele sunt mai întâi **sortate** (lexicografic, după unghiul polar şi distanţa polară) şi 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 (algoritm)
Input: O mulţime de puncte $\mathcal{P}$ din $\mathbf{R}^2$.
Output: O listă $\mathcal{L}$ care conţine vârfurile ce determină frontiera acoperirii convexe, parcursă în sens trigonometric.

1. Determinarea unui punct interior din $\mathrm{Conv}(\mathcal{P})$ /* de ex. baricentrul
2. Efectuarea unei translaţii, punctul interior de la 1. devine originea O
3. Sortare şi ordonare în raport cu unghiul polar şi distanţa polară, renumerotare $P_1, P_2, \ldots, P_n$ conform ordonării
4. $\mathcal{L} \leftarrow (P_1, P_2)$
5. **for** $i \leftarrow 3$ **to** n
6. **do** adaugă P_i la sfârşitul lui $\mathcal{L}$
7. **while** $\mathcal{L}$ are mai mult de două puncte
   and ultimele trei nu determină un viraj la stânga
8. **do** şterge penultimul punct
9. **return** $\mathcal{L}$

Graham's scan, varianta lui Andrew [1979]

- Punctele sunt mai întâi **sortate** (lexicografic, după coordonatele carteziene) şi renumerotate.
- Algoritmul determină două liste, reprezentând marginea **inferioară** şi cea **superioară** a frontierei, pentru a le determina sunt folosite la iniţializare 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. Seefectuează testul de orientare pentru ultimele trei puncte şi 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: $O(n \log n)$; spaţiu: $O(n)$; complexitate algebrică: polinoame de gradul II.
- Tratarea cazurilor degenerate: corect.
- Robusteţea: 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.
- Iniţializare: 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 (algoritm)

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

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

2.4 Exerciţii, probleme, aplicaţii

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

Exerciţiul 2.6 Considerăm punctele $A = (-6, 6)$, $B = (1, 6)$, $C = (1, -1)$, $D = (-6, 0)$, $E = (6, 0)$, $F = (3, 2)$, $G = (-4, -2)$, $H = (-1, -2)$, $I = (-2, -2)$. Precizaţi care este numărul maxim de elemente pe care îl conţine $\mathcal{L}$ 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). Justificaţi!

Exerciţiul 2.7 Daţi un exemplu de mulţime $\mathcal{M}$ 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 $\mathcal{M}$, obţinută pe parcursul Graham's scan, varianta Andrew). Justificaţi!

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

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

Exerciţiul 2.10 Scrieţi în pseudocod Graham's scan - varianta Andrew şi Jarvis' march.

Exerciţiul 2.11 Explicaţi dacă Graham's scan / Jarvis' march indică rezultatul dorit atunci când toate punctele sunt situate pe o aceeaşi dreaptă.

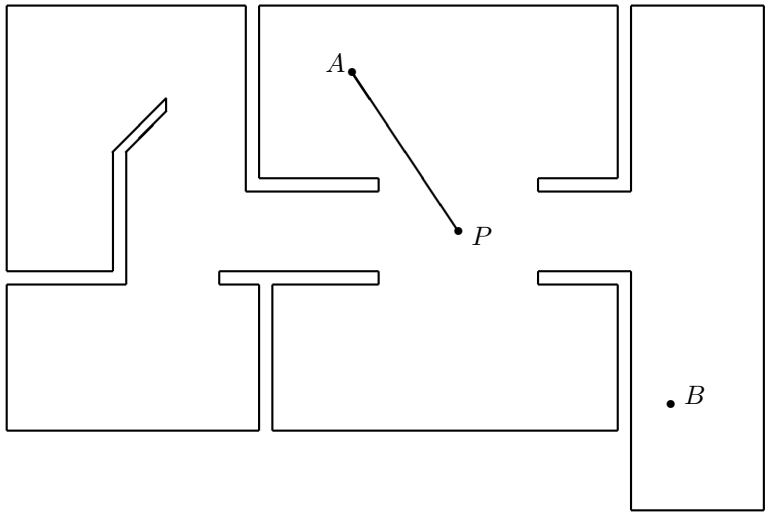
Exerciţiul 2.12 Date n puncte în plan, scrieţi un algoritm de complexitate $O(n \log n)$ care să determine un poligon care are toate aceste puncte ca vârfuri. Explicaţi cum este aplicat acest algoritm pentru punctele $P_1 = (4, 2)$, $P_2 = (7, 1)$, $P_3 = (-3, 5)$, $P_4 = (3, 6)$, $P_5 = (-4, -4)$, $P_6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

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 autointersecţii) având n vârfuri.
- O cameră video (vizibilitate 360^0) 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 şi unde trebuie amplasate acestea?*

Numărul de camere vs. forma poligonului

– Se doreşte exprimarea numărului de camere necesare pentru supraveghere în funcţie de n (sau controlarea acestuia de către n).

– Pentru a supraveghea un spaţiu având forma unui poligon convex, este suficientă o singură cameră.

– Numărul de camere depinde şi 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).

Definiţii

– Fie $\mathcal{P}$ un poligon plan.

– (i) O **diagonală** a lui $\mathcal{P}$ este un segment ce uneşte două vârfuri ale acestuia şi care este situat în interiorul lui $\mathcal{P}$.

– (ii) O **triangulare** $\mathcal{T}_P$ a lui $\mathcal{P}$ este o descompunere a lui $\mathcal{P}$ în triunghiuri, dată de o mulţime maximală de diagonale ce nu se intersectează.

– **Teoremă.** *Orice poligon simplu admite o triangulare. Orice triangulare a unui poligon cu n vârfuri conţine exact $n-2$ triunghiuri.*

Rezovlarea problemei galeriei de artă

– Amplasarea camerelor se poate face în vârfurile poligonului.

– Dată o pereche $(\mathcal{P}, \mathcal{T}_P)$ se consideră o 3-colorare a acesteia: fiecărui vârf îi corespunde o culoare dintr-un set de 3 culori şi pentru fiecare triunghi, cele 3 vârfuri au culori distincte.

– **Observaţie.** Dacă $\mathcal{P}$ este simplu, o astfel de colorare există, deoarece graful asociat perechii $(\mathcal{P}, \mathcal{T}_P)$ este arbore.

Teorema galeriei de artă

– **Teoremă.** [Chvátal, 1975; Fisk, 1978] *Pentru un poligon cu n vârfuri, $\left\lfloor \frac{n}{3} \right\rfloor$ camere sunt* **uneori necesare** *şi întotdeauna* **suficiente** *pentru ca fiecare punct al poligonului să fie vizibil din cel puţin una din camere.*

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 puţin 4 vârfuri admite cel puţin două componente de tip E care nu se suprapun.*

– **Corolar.** *Orice poligon simplu admite (cel puţin) două diagonale.*

– Algoritmul 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 eficienţi: complexitate $O(n)$ pentru poligoane y-monotone [Garey et al., 1978].

– Descompunerea unui poligon oarecare in componente y-monotone poate fi realizată cu un algoritm de complexitate $O(n \log n)$ [Lee, Preparata, 1977].

– Există şi alte clase de algoritmi mai rapizi; [Chazelle, 1990]: algoritm liniar.

– Link pentru alte abordări

Triangularea poligoanelor monotone

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

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

1. Lanţul vârfurilor din partea stângă şi al celor din partea dreaptă sunt unite într-un singur şir, ordonat descrescător, dupa y (dacă ordonata este egală, se foloseşte abscisa). Fie $v_1, v_2, \ldots, v_n$ şirul ordonat.
2. Iniţializează o stivă vidă $\mathcal{S}$ şi inserează v_1, v_2.
3. **for** $j = 3$ **to** $n - 1$
4. **do** **if** v_j şi vârful din top al lui $\mathcal{S}$ sunt în lanţuri 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} şi 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 şi ultimul)

3.2 Triangularea unei mulţimi arbitrare de puncte

Problematizare

- Triangularea unui poligon convex (listă ordonată de puncte $(P_1, P_2, \ldots, P_n)$.
- Are sens să vorbim de triangulare pentru mulţimea $\{P_1, P_2, \ldots, P_n\}$?
- **Definiţie.** O **triangulare** a unei mulţimi $\mathcal{P}$ 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 autointersecţii!)
- Trebuie făcută distincţie între triangulare a unui poligon $(P_1, P_2, \ldots, P_n)$ şi triangulare a mulţimii subdiacente $\{P_1, P_2, \ldots, P_n\}$ (coincid dacă poligonul este convex!)

Elemente ale unei triangulări

- Dată o mulţime de puncte $\mathcal{P}$ şi o triangulare $\mathcal{T}_P$ a sa:
  vârfuri, muchii, triunghiuri.
- Legătură între aceste elemente?
- **Propoziţie.** *Fie $\mathcal{P}$ o mulţime de n puncte din plan nesituate toate pe o aceeaşi 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 şi $(3n-k-3)$ muchii.*
- **Demonstraţie:** Se bazează pe formula lui Euler.
- **Exemplu:** Cazul unui poligon convex.

3.3 Triangulări unghiular optime

Problematizare

- **Problemă.** Se fac măsurători ale altitidinii pentru un teren. Se doreşte reprezentarea tridimensională (cât mai sugestivă).
- **Problemă (reformulată).** Cum "comparăm triangulările" unei mulţimi de puncte fixate?
- **Exemplu.** Cazul unui patrulater convex.

Terminologie

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

Muchii ilegale, triangulări legale

- Fixată: o mulţime 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 obţinută 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.
- **Propoziţie.** *Fie $\mathcal{P}$ o mulţime de puncte din plan.*

  (i) *Orice triangulare unghiular optimă este legală.*

  (ii) *Dacă $\mathcal{P}$ este în poziţie generală (oricare patru puncte nu sunt conciclice), atunci există o unică triangulare legală, iar aceasta este unghiular optimă.*

3.4 Exerciţii, probleme, aplicaţii

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

Exerciţiul 3.2 Aplicaţi metoda din demonstraţia 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 $P_6, \dots, P_1$ faţă de axa Ox.

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

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

Exerciţiul 3.5 Fie $n \geq 2$ un număr natural par fixat. Considerăm mulţimea $\mathcal{M} = \{A_0, \ldots, A_n, B_0, \ldots, B_n, C_0, \ldots, C_n, D_0, \ldots, D_n\}$, unde $A_i = (i, 0)$, $B_i = (0, i)$, $C_i = (i, i)$, $D_i = (n-i, i)$, pentru orice $i = 0, \ldots, n$. Determinaţi numărul de triunghiuri şi numărul de muchii al unei triangulări a lui $\mathcal{M}$.

Exerciţiul 3.6 Daţi exemplu de poligon care să aibă mai multe vârfuri principale concave decât vârfuri principale convexe.

Exerciţiul 3.7 Daţi exemplu de mulţime de puncte din $\mathbf{R}^2$ care să admită o triangulare având 3 triunghiuri şi 7 muchii.

Exerciţiul 3.8 Daţi exemplu de mulţime $\mathcal{M} = \{A, B, C, D, E, F, G\}$ din $\mathbf{R}^2$ astfel ca $\mathcal{M}$ să admită o triangulare ce conţine 14 muchii.

Exerciţiul 3.9 Fie $ABCD$ un patrulater convex. Fie $\mathcal{C}$ cercul circumscris triunghiului ΔABC. Demonstraţi că diagonala AC este ilegală dacă şi numai dacă D este în interiorul lui $\mathcal{C}$.

Capitolul 4

Intersecţii

4.1 Intersecţia segmentelor (generalităţi)

Motivaţie (Probleme geometrice în context 2D)

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

Algoritmul trivial

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

Rezolvarea problemei în context 1D

- Ordonarea lexicografică a extremităţilor segmentelor / intervalelor într-o listă $\mathcal{P}$.
- Lista $\mathcal{P}$ este parcursă (crescător); lista $\mathcal{L}$ a segmentelor care conţin 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 şters din $\mathcal{L}$ şi se rapoartează intersecţii între s şi toate segmentele din $\mathcal{L}$

– **Teoremă.** Algoritmul are complexitate $O(n \log n + k)$ şi necesită $O(n)$ memorie (k este numărul de perechi ce se intersectează).

4.2 Metoda liniei de baleiere

Un prim algoritm

– Linia de baleire l: **orizontală**, astfel că toate intersecţiile situate deasupra liniei de baleiere au fost detectate.

– **Statutul (sweep line status)**: mulţimea segmentelor care intersectează l.

– **Evenimente (event points)**: capetele segmentelor $\longrightarrow$ actualizarea statutului

– **Obs. 1.** Sunt testate pentru intersecţie segmente care intersectează, la un moment dat, $l \longrightarrow$ (sunt testate segmentele care sunt "aproape" de-a lungul axei Oy) $\longrightarrow$ încă ineficient.

– **Obs. 2.** Linia de baleiere are, de fapt, o variaţie "discretă", nu continuă.

–

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

– **Idee de lucru:** ordonare (segmente, evenimente).

  - Segmentele: ordonate folosind extremităţile superioare (lexicografic, x apoi y).
  - Evenimente (puncte): (lexicografic, y apoi x).
  - Statutul: **listă** (mulţime ordonată)

– **Avantaj:** în momentul modificării statutului, sunt testate intersecţiile doar în raport cu vecinii din listă (sunt testate segmentele care sunt "aproape" de-a lungul axei Ox).

– **Fundamental:** Punctele de intersecţie devin, la rândul lor, evenimente, deoarece schimbă ordinea segmenetelor care le determină $\longrightarrow$ 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 intersecţie sub linia de baleiere $\longrightarrow$ 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 intersecţie (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 eveniment (event queue): puncte, ordonate lexicografic, y apoi x**

– **$\mathcal{T}$ arbore binar de căutare echilibrat (balanced binary search tree): segmente $\longrightarrow$ este reţinută ordinea segmentelor**

Algoritm INTERSECTII

– **Input. O mulţime de segmente din planul $\mathbb{R}^2$.**

– **Output. Mulţimea punctelor de intersecţie (explicit sau doar formal); pentru fiecare punct precizează segmentele pe care se găseşte.**

1. **$\mathcal{Q} \leftarrow \emptyset$. Inserează extremităţile segmentelor în $\mathcal{Q}$; împreună cu marginea superioară a unui segment memorează şi segmentul**

2. **$\mathcal{T} \leftarrow \emptyset$.**

3. **while $\mathcal{Q} \neq \emptyset$**

4. **do determină evenimentul p care urmează în $\mathcal{Q}$ şi îl şterge**

5. ANALIZEAZA **(p)**

ANALIZEAZA (p)

1. Fie $U(p)$ mulţimea 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 conţin p: sunt adiacente în $\mathcal{T}$ (de ce?) Fie $D(p)$, respectiv $Int(p)$, mulţimea segmentelor care au p drept margine inferioară, respectiv conţin p în interior.
3. **if** $U(p) \cup D(p) \cup Int(p)$ conţine mai mult de un segment
4. **then** raportează p ca punct de intersecţie, împreună cu segmentele din $U(p)$, $D(p)$, $Int(p)$
5. şterge 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 şi 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 şi sgt_r se intersectează sub linia de baleiere sau pe linia de baleiere, dar la dreapta lui p şi punctul de intersecţie nu este deja în $\mathcal{Q}$
2. **then** inserează punctul de intersecţie ca eveniment în $\mathcal{Q}$

Rezultatul principal (intersecţii de segmente)

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

4.3 Alte rezultate

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

Teoremă. *Pentru două mulţimi R şi 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 şi spaţiu liniar, folosind predicate (polinoame) de grad cel mult II ($n = |R| + |B|$) şi k este numărul perechilor ce se intersectează).*

4.4 Suprapunerea straturilor tematice

Subdiviziuni planare

- Conceptul de subdiviziune planară: vârfuri, muchii, feţe.
- **Listă dublu înlănţuită** [Müller şi Preparata, 1978] (trei înregistrări: vârfuri, feţe, muchii orientate (semi-muchii)).
  - **Vârf** v: coordonatele lui v în $Coordinates(v)$, pointer $IncidentEdge(v)$ spre o muchie orientată care are v ca origine
  - **Faţă** f: pointer $OuterComponent(f)$ spre o muchie orientată corespunzătoare frontierei externe (pentru faţa nemărginită este **nil**); listă $InnerComponents(f)$, care conţine, pentru fiecare gol, un pointer către una dintre muchiile orientate de pe frontieră
  - **Muchie orientată** $\overrightarrow{e}$: pointer $Origin(\overrightarrow{e})$, pointer $Twin(\overrightarrow{e})$ pointer $IncidentFace(\overrightarrow{e})$, pointer $Next(\overrightarrow{e})$, pointer $Prev(\overrightarrow{e})$.

– Oricărei subdiviziuni planare $\mathcal{S}$ i se asociază o listă dublu înlănţuită $\mathcal{D}_{\mathcal{S}}$.

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

– **Input.** Două subdiviziuni planare $\mathcal{S}_1$, $\mathcal{S}_2$ memorate în liste dublu înlănţuite $\mathcal{D}_{\mathcal{S}_1}, \mathcal{D}_{\mathcal{S}_2}$

– **Output.** Overlay-ul $\mathcal{O}(\mathcal{S}_1, \mathcal{S}_2)$ dintre $\mathcal{S}_1, \mathcal{S}_2$, memorat într-o listă dublu înlănţuită $\mathcal{D}_{\mathcal{O}(\mathcal{S}_1,\mathcal{S}_2)}$

1. Copiază listele $\mathcal{D}_1, \mathcal{D}_2$ într-o nouă listă dublu înlănţuită $\mathcal{D}_{\mathcal{O}(\mathcal{S}_1,\mathcal{S}_2)}$
2. Calculează toate intersecţiile de muchii dintre $\mathcal{S}_1$ şi $\mathcal{S}_2$ cu algoritmul INTERSECTII. La fiecare eveniment, pe lângă actualizarea lui $\mathcal{Q}$ şi $\mathcal{T}$, 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 şi ale lui $\mathcal{S}_2$
   – Memorează noile muchii orientate adecvat
3. Determină ciclii de frontieră din $\mathcal{O}(\mathcal{S}_1, \mathcal{S}_2)$, stabileşte natura (exteriori/interiori)
4. Construieşte graful $\mathcal{G}$ ale cărui noduri corespund ciclilor de frontieră şi ale cărui arce unesc fiecare ciclu corespunzând unui gol cu ciclul de la stânga vârfului cel mai din stânga şi 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 şi fie f faţa mărginită a ciclului. Creează o înregistrare pentru f, setează $OuterComponent(f)$ (către una din muchiile lui $\mathcal{C}$), construieşte lista $InnerComponents(f)$ (pentru fiecare gol, pointer către una dintre muchiile orientate). Pentru fiecare muchie orientată, $IncidentFace()$ către înregistrarea lui f
7. Etichetează fiecare faţă a lui $\mathcal{O}(\mathcal{S}_1, \mathcal{S}_2)$

Rezultate principale

– **Teoremă. (Overlay-ul hărţilor)** *Fie S_1 o subdiviziune de complexitate n_1, S_2 o subdiviziune de complexitate n_2, fie $n := n_1 + n_2$. Overlay-ul dintre S_1 şi S_2 poate fi construit în $O(n \log n + k \log n)$, unde k este complexitatea overlay-ului.*

– **Corolar. (Operaţii boolene)** *Fie $\mathcal{P}_1$ un poligon cu n_1 vârfuri şi $\mathcal{P}_2$ un poligon cu n_2 vârfuri; fie $n = n_1 + n_2$. Atunci $\mathcal{P}_1 \cup \mathcal{P}_2$, $\mathcal{P}_1 \cap \mathcal{P}_2$ şi $\mathcal{P}_1 \setminus \mathcal{P}_2$ pot fi determinate în timp $O(n \log n + k \log n)$, unde k este complexitatea output-ului.*

4.5 Exerciţii, probleme, aplicaţii

Exerciţiul 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 $i = 1, \ldots, 4$ notăm cu s_i segmentul $[P_i Q_i]$. Scrieţi cum evoluează statutul liniei de baleiere, precum şi evenimentele care determină modificarea sa (linia de baleiere este orizontală, iar statutul este o mulţime neordonată de segmente).

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

Exerciţiul 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 $i = 1, \ldots, 4$ notăm cu s_i segmentul $[P_iQ_i]$. Considerăm linia de baleiere l dată de ecuaţia $y = 9$. Indicaţi evenimentele deja eliminate din coada de evenimente $\mathcal{Q}$, cele rămase în $\mathcal{Q}$, cele care urmează să fie incluse ulterior în $\mathcal{Q}$ şi precizaţi statutul corespunzător lui l (statutul este o mulţime ordonată de segmente).

Exerciţiul 4.4 Explicaţi cum poate fi parcursă frontiera unei feţe şi cum pot fi găsite toate muchiile din jurul unui vârf, folosind pointerii asociaţi elementelor unei subdiviziuni planare.

Exerciţiul 4.5 Considerăm un dreptunghi D din interiorul căruia este scos un dreptunghi Δ. Descrieţi subdiviziunea planară asociată.

Exerciţiul 4.6 Fie punctele $A = (4,0)$, $B = (0,4)$, $C = (-4,0)$, $D = (0,-4)$. Construiţi 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 feţe, iar $\mathcal{S}_2$ să aibă trei feţe. Indicaţi numărul total de semimuchii pentru $\mathcal{S}_1$, respectiv $\mathcal{S}_2$.

Capitolul 5

Elemente de programare liniară

5.1 Motivaţie: turnarea pieselor în matriţe

– Turnarea pieselor în matriţe şi extragerea lor fără distrugerea matriţei.

– Neajunsuri: unele obiecte pot rămâne blocate în matriţe; există obiecte pentru care nu există o matriţă adecvată; extragerea obiectului depinde de poziţia matriţei.

– **Problema studiată.** Dat un obiect, există o matriţă din care să poată fi extras?

– **Convenţii.**

  – Obiectele: **poliedrale**.

  – Matriţele: formate dintr-o singură piesă; fiecărui obiect $\mathcal{P}$ îi este asociată o matriţă $\mathcal{M}_{\mathcal{P}}$

  – Obiectul: extras printr-o singură translaţie (sau o succesiune de translaţii)

Terminologie şi convenţii

– **Alegerea orientării:** diverse orientări ale obiectului pot genera diverse matriţe.

– **Faţa superioară:** prin convenţie, obiectele au (cel puţin) o faţa superioară (este orizontală, este singura care nu este adiacentă cu matriţa). Celelalte feţe: **standard**; orice faţă standard f a obiectului corespunde unei feţe $\hat{f}$ a matriţei.

– **Obiect care poate fi turnat (*castable*):** există o orientare pentru care acesta poate fi turnat şi apoi extras printr-o translaţie (succesiune de translaţii): *direcţie admisibilă*.

– **Convenţii:** Matriţa este paralelipipedică şi are o cavitate corespunzătoare obiectului; faţa superioară a obiectului (şi a matriţei) este perpendiculară cu planul Oxy.

Fundamente geometrice

– **Condiţie necesară:** direcţia de extragere $\vec{d}$ trebuie să aibă componenta z pozitivă

– **În general:** o faţă $\hat{f}$ a matriţei pentru care unghiul dintre normala exterioară $\vec{\nu}(f)$ la faţă f şi $\vec{d}$ este mai mic de 90° împiedică translaţia în direcţia $\vec{d}$

– **Propoziţie.** *Un poliedru $\mathcal{P}$ poate fi extras din matriţa sa $\mathcal{M}_\mathcal{P}$ prin translaţie în direcţia $\vec{d}$ dacă şi numai dacă $\vec{d}$ face un unghi de cel puţin 90° cu normala exterioară a fiecărei feţe standard a lui $\mathcal{P}$.*

– **Reformulare.** Dat $\mathcal{P}$, trebuie găsită o direcţie $\vec{d}$ astfel încât, pentru fiecare faţă standard f, unghiul dintre $\vec{d}$ şi $\vec{\nu}(f)$ să fie cel puţin 90°.

– **Analitic:** fiecare faţă defineşte un semiplan

– **Concluzie:** Fie $\mathcal{P}$ un poliedru. Mulţimea direcţiilor admisibile este dată de o intersecţie de semiplane.

– **Teoremă.** Fie $\mathcal{P}$ un poliedru cu n feţe. Se poate decide dacă $\mathcal{P}$ reprezintă un obiect care poate fi turnat în $O(n^2)$ timp şi folosind $O(n)$ spaţiu. În caz afirmativ, o matriţă şi o direcţie admisibiă în care poate fi extras $\mathcal{P}$ este determinată cu aceeaşi complexitate timp.

5.2 Intersecţii de semiplane

Formularea problemei

– Fie $\mathcal{H} = \{H_1, H_2, \ldots, H_n\}$ o mulţime de semiplane din $\mathbb{R}^2$; semiplanul H_i dat de o relaţie de forma

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

– Intersecţia $H_1 \cap H_2 \cap \ldots \cap H_n$ este dată de un sistem de inecuaţii; este o mulţime poligonală convexă, mărginită de cel mult n muchii (poate fividă, mărginită, nemărginită,...)

Algoritm IntersectiiSemiplane $(\mathcal{H})$

– **Input.** O mulţime $\mathcal{H}$ 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 mulţimi $\mathcal{H}_1$, $\mathcal{H}_2$ având fiecare $[n/2]$ elemente
4. $\mathcal{C}_1 \leftarrow$ IntersectiiSemiplane $(\mathcal{H}_1)$
5. $\mathcal{C}_2 \leftarrow$ IntersectiiSemiplane $(\mathcal{H}_2)$
6. $\mathcal{C} \leftarrow$ IntersecteazaRegiuniConvexe $(\mathcal{C}_1, \mathcal{C}_2)$

Rezultate principale

– **Propoziţie.** *Aplicând direct algoritmii de overlay, intersecţia 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 îmbunătăţit, astfel încât complexitatea-timp să fie liniară.*

– **Teoremă.** *Intersecţia unei mulţimi de n semiplane poate fi determinată cu complexitate-timp $O(n \log n)$ şi folosind $O(n)$ memorie.*

5.3 Programare liniară

Problematizare

– **Formulare generală (în spaţiul d-dimensional):**

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

date constrângerile liniare (inegalităţi)

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

– **Terminologie:** date de intrare, funcţie 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 cerinţei de maximizare:** Maximizarea funcţiei obiectiv revine la a determina un punct al cărui vector de poziţie are proiecţia maximă de direcţia dată de vectorul $\vec{c} = (c_1, c_2, \ldots, c_d)$.

– Pentru o problemă de programare liniară în plan ($d = 2$) pot fi distinse patru situaţii: (i) o soluţie unică; (ii) toate punctele de pe o muchie sunt soluţii; (iii) regiunea fezabilă este nemărginită şi pot fi găsite soluţii 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. $v_0 \leftarrow$ "colţul" lui c_0
2. fie $h_1, h_2, \ldots, h_n$ 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).

5.4 Exerciţii, probleme, aplicaţii

Exerciţiul 5.1 Considerăm două "piese" poligonale $\mathcal{P}_1$ şi $\mathcal{P}_2$, având normalele feţelor standard date de vectorii:

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

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

(în această ordine). Stabiliţi care dintre piese poate fi extrasă din matriţa asociată prin deplasare în direcţia verticală (dată de $(0,1)$. Desenaţi!

Exerciţiul 5.2 Considerăm semiplanele S_λ, S', S'' date de inecuaţiile

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

Discutaţi, în funcţie de λ, natura intersecţiei $S_\lambda \cap S' \cap S''$.

Exerciţiul 5.3 Fie punctul $p = (-1,1)$; dreapta $d : (y = 3x + 4)$. Verificaţi că $p \in d$ şi că $d^* \in p^*$.

Exerciţiul 5.4 Fie punctele $p_1 = (2,5)$; $p_2 = (1,6)$. Scrieţi ecuaţia dreaptei p_1p_2 şi detaliaţi (cu calcule explicite!) configuraţia din planul dual.

Exerciţiul 5.5 Fie dreapta $d : (y = 2x + 1)$ şi $p = (1,8)$. Verificaţi că p este deasupra lui d şi că d^* este deasupra lui p^*.

Exerciţiul 5.6 Fie configuraţia: *trei drepte care trec prin acelaşi punct; pe fiecare dreaptă se alege un punct, diferit de punctul comun al celor trei drepte.* Descrieţi configuraţia dual ă. Desenaţi!

Capitolul 6

Probleme de localizare

6.1 Hărţi trapezoidale

Algoritm HartaTrapezoidala

- **Input.** O mulţime S de n segmente în poziţie generală.
- **Output.** Harta trapezoidală $\mathcal{T}(S)$ şi o structură de căutare $\mathcal{D}$ pentru $\mathcal{T}(S)$, într-un dreptunghi R cu laturile paralele cu axele.

1. Determină dreptunghiul R.
2. Generează o permutare $s_1, s_2, \ldots, s_n$ a segmentelor din S.
3. **for** $i \leftarrow 1$ **to** n
4. **do** găseşte mulţimea de trapeze $\Delta_0, \Delta_1, \ldots, \Delta_k$ care intersectează segmentul s_i
5. elimină $\Delta_0, \ldots, \Delta_k$ şi le înlocuieşte cu trapezele nou apărute
6. elimină frunzele corespunzătoare din $\mathcal{D}$ şi creează noi frunze, actualizează $\mathcal{D}$

Rezultatul principal

Teoremă. *Fie S o mulţime de n segmente în poziţie generală. Algoritmul* HartaTrapezoidala *determină harta trapezoidală $\mathcal{T}(S)$ şi 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 $O(n)$ şi pentru un punct arbitrar q timpul mediu de localizare este $O(\log n)$.*

6.2 Aplicaţie: mişcarea unui robot-punct

Algoritm DeterminaSpatiuLiber **(S)**

- **Input.** O mulţime $\mathcal{P}$ de poligoane disjuncte.
- **Output.** O hartă trapezoidală $\mathcal{C}_l$ a spaţiului liber (pentru un robot-punct).

1. Fie S mulţimea muchiilor poligoanelor din $\mathcal{P}$.
2. Determină harta trapezoidală $\mathcal{T}(S)$, folosind algoritmul HartaTrapezoidala.

3. Elimină trapezele situate în interiorul poligoanelor şi returnează subdiviziunea obţinuta.

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 spaţiului 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 $\mathcal{T}(\mathcal{C}_l)$ conţinând M_{start}, respectiv M_{end}.
2. **if** există Δ_{start}, respectiv Δ_{end} conţinând M_{start}, respectiv M_{end}
3. **then** fie v_{start} şi v_{end} centrele Δ_{start}, Δ_{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 mulţime S de obstacole poligonale, având în total n muchii. Utilizând timp mediu de preprocesare $O(n \log n)$ pentru mulţimea S, un drum liber de coliziuni între două puncte fixate poate fi calculat (dacă există!) în timp mediu $O(n)$.*

6.3 Exerciţii, probleme, aplicaţii

Exerciţiul 6.1 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. Stabiliţi câte trapeze are harta trapezoidală a regiunii situate între cele două pătrate. Câte dintre acestea sunt degenerate?

Exerciţiul 6.2 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 şi de dreptele date de ecuaţiile $x = 12$, respectiv $y = 12$.

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

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

Exerciţiul 6.3 Considerăm două triunghiuri T_1 şi T_2 (astfel ca laturile lor să fie segmente în poziţie generală), în interiorul unui *bounding box* R. Câte trapeze are harta trapezoidală asociată? Depinde acest număr de poziţia relativă a triunghiurilor?

Capitolul 7

Diagrame Voronoi

7.1 Generalităţi

Problema oficiilor poştale

– Se consideră o mulţime de puncte (oficiile poştale) din plan. Care este cel mai apropiat?

Formalizare

– Fie $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ o mulţime de puncte din planul $\mathbb{R}^2$.

– **Diagrama Voronoi** a lui $\mathcal{P}$ (notată $\text{Vor}(\mathcal{P})$) este o divizare a planului $\mathbb{R}^2$ în n celule $\mathcal{V}(P_1), \ldots, \mathcal{V}(P_n)$ cu proprietatea că

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

– Două celule adiacente au în comun o *muchie* sau un *vârf* (punct de intersecţie a muchiilor).

– **Atenţie!** Vârfurile lui $\text{Vor}(\mathcal{P})$ sunt diferite de punctele din $\mathcal{P}$.

– Uneori, prin abuz de limbaj, este precizată doar împărţirea în muchii / vârfuri.

7.2 Proprietăţi

Proprietăţi elementare

– Celula asociată unui punct este o intersecţie 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 conţine punctul P_i.

– În particular: fiecare celulă este o mulţime convexă.

– Aplicabilitate: algoritm (lent) de determinare a diagramei Voronoi.

– **Comentariu.** Forma celulelor depinde de funcţia distanţă aleasă.

Structura unei diagrame Voronoi

- Fie $\mathcal{P} = \{P_1, P_2, \ldots, P_n\}$ o mulţime de puncte din planul $\mathbb{R}^2$.
- Dacă toate punctele sunt coliniare, atunci diagrama Voronoi asociată $\text{Vor}(\mathcal{P})$ conţine $n-1$ *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?).
- Legătură între numărul de vârfuri, respectiv de muchii şi numărul de puncte ($n_v \leq 2n-5,\ n_m \leq 3n-6$).

7.3 Diagrame Voronoi şi triangulări Delaunay

Legătura cu triangulările unghiular optime

- Mulţime de puncte $\mathcal{P}$
- Diagrama Voronoi $\text{Vor}(\mathcal{P})$
- Graful dual $\mathcal{G}(\mathcal{P})$
- Triangulare $\mathcal{T}_\mathcal{P}$ (triangulare Delaunay)
- **Teoremă.** *Triangularea Delaunay este unghiular optimă.*
- **Întrebare:** Ce se poate spune despre această construcţie în cazul în care punctele mulţimii $\mathcal{P}$ sunt (de exemplu) vârfurile unui pătrat?

7.4 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 reţinem informaţia legată de intersecţia dintre linia de baleiere şi diagramă, ci doar informaţia legată de partea diagramei care <u>nu</u> mai poate fi influenţată de punctele situate de dincolo de linia de baleiere.
- **Concepte:**
  - beach line / linie parabolică
  - site event / eveniment de tip locaţie (apare un arc de parabolă)
  - circle event / eveniment de tip cerc (dispare un arc de parabolă)

Rezultate principale

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

7.5 Exerciţii, probleme, aplicaţii

Exerciţiul 7.1 Determinaţi, folosind metoda diagramelor Voronoi, triangularea Delaunay pentru mulţimea formată din punctele $A = (3, 5)$, $B = (6, 6)$, $C = (6, 4)$, $D = (9, 5)$ şi $E = (9, 7)$.

Exerciţiul 7.2 Determinaţi numărul de semidrepte conţinute în diagrama Voronoi asociată mulţimii de puncte $\mathcal{M} = \{A_0, \ldots, A_5, B_0, \ldots, B_5, C_0, \ldots, C_5\}$, unde $A_i = (i + 1, i + 1)$, $B_i = (-i, i)$ şi $C_i = (0, i)$, pentru $i = 0, \ldots, 5$.

Exerciţiul 7.3 Daţi exemplu de mulţimi $\mathcal{M}_1$ şi $\mathcal{M}_2$ din $\mathbb{R}^2$, fiecare având câte 4 puncte, astfel ca, pentru fiecare dintre ele, diagrama Voronoi asociată să conţină exact 3 semidrepte, iar diagrama Voronoi asociată lui $\mathcal{M}_1 \cup \mathcal{M}_2$ să conţină exact 6 semidrepte.

Exerciţiul 7.4 Fie punctele $A_1 = (5, 1), A_2 = (7, -1), A_3 = (9, -1), A_4 = (7, 3)$, $A_5 = (11, 1), A_6 = (9, 3)$. Daţi exemplu de mulţime de două puncte $\{A_7, A_8\}$ cu proprietatea că diagrama Voronoi asociată mulţimii $\{A_1, \ldots, A_8\}$ are exact 4 muchii de tipul semidreaptă (explicaţi construcţia făcută).

Exerciţiul 7.5 Demonstraţi că dacă punctele din mulţimea $\mathcal{P}$ nu sunt coliniare, diagrama Voronoi $\mathrm{Vor}(\mathcal{P})$ nu poate conţine drepte.

Bibliografie

[1] **M. de Berg, M. van Kreveld, M. Overmars şi 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 şi M. Shamos, *Computational Geometry: An Introduction*, Springer, 1985.

---

[6] L. Bădescu, *Geometrie*, Editura Universităţii Bucureşti, 2000.

[7] M. do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice Hall, 1976.

[8] Gh. Galbură şi F. Radó, *Geometrie*, Editura Didactică şi Pedagogică, Bucureşti, 1979.

[9] A. Gray, *Modern Differential Geometry of Curves and Surfaces with* `Mathematica`, CRC Press, 1999.

[10] I. Hirică, S. Leiko, L. Nicolescu, G. Pripoae, *Geometrie diferenţială. Probleme. Aplicaţii*, Bucureşti, 1999.

[11] M.I. Munteanu, *Algoritmi geometrici 2D şi aplicaţii în CAGD*, Editura Universităţii "Al. I. Cuza" Iaşi, 2005.

[12] L. Nicolescu, *Curs de geometrie*, Bucureşti, 2002.

[13] L. Ornea şi A. Turtoi, *O introducere în geometrie*, Editura Theta, Bucureşti, 2000.

[14] M.S. Stupariu, *Geometrie analitică*, Bucureşti, 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. Invarianţa acoperirii convexe la transformări afine.

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

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 invarianţă la transformări afine a acoperirii convexe.

3. Poziţia unui punct faţă de un poligon convex.

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

Output: Precizează poziţia lui A faţă de $\mathcal{P}$ (în interior, pe laturi, în exterior), folosind o împărţire convenabilă pe sectoare. Reprezentare grafică.

4. Poligon convex şi 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 mulţime de cercuri $\mathcal{C}_1, \ldots, \mathcal{C}_q$ de rază 1, disjuncte, din planul $\mathbb{R}^2$ (sunt indicate centrele cercurilor), un punct $A \in \mathbb{R}^2$.

Output: Precizează poziţia lui A faţă de $\text{Conv}(\mathcal{C}_1 \cup \ldots \cup \mathcal{C}_q)$. Reprezentare grafică.

7. Triangulări ale poligoanelor − invarianţa la transformări afine

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

Output: Construieşte o triangulare $\mathcal{T}_\mathcal{P}$ a lui $\mathcal{P}$ şi 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. Poziţia unui punct faţă 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 poziţia lui A faţă 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 aparţine.

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 mulţimilor de puncte − invarianţa la transformări afine

Input: O mulţime $\mathcal{M}$ de puncte, reprezentând vârfurile unui triunghi şi puncte în interiorul acestuia.

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

11. Poziţia unui punct faţă de o triangulare

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

Output: Determină o triangulare $\mathcal{T}_\mathcal{M}$ a lui $\mathcal{M}$. Precizează poziţia lui A faţă 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 aparţine.

12.* Echivalenţa triangulărilor

Input: O mulţime $\mathcal{M}$ 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 intr-alta într-un număr finit de paşi, prin aplicarea unor modificări de tip "flip". Reprezentare grafică.