Algoritmi avansaţi

Seminar 1 (săpt. 1 şi 2)

1. *Fie punctele* $A = (1, 2, 3), B = (4, 5, 6) \in \mathbb{R}^3$.

a) *Fie* $C = (a, 7, 8)$. *Arătaţi că există* a *astfel ca punctele* A, B, C *să fie coliniare şi pentru* a *astfel determinat calculaţi raportul* $r(A, B, C)$.

b) *Determinaţi punctul* P *astfel ca raportul* $r(A, P, B) = 1$.

c) *Daţi exemplu de punct* Q *astfel ca* $r(A, B, Q) < 0$ *şi* $r(A, Q, B) < 0$.

Soluţie.

a) Condiţia de coliniaritate a punctelor A, B, C este echivalentă cu coliniaritatea vectorilor $\overrightarrow{AB}$ şi $\overrightarrow{BC}$. Au loc relaţiile:

$$\overrightarrow{AB} = B - A = (3, 3, 3), \quad \overrightarrow{BC} = (a - 4, 2, 2).$$

Vectorii daţi sunt proporţionali dacă şi numai dacă $a - 4 = 2$, deci $a = 6$. De fapt, dreapta AB este direcţionată de vectorul $(1, 1, 1)$ (şi de orice vector proporţional cu acesta).

În acest caz, avem $\overrightarrow{AB} = B - A = (3, 3, 3), \quad \overrightarrow{BC} = (2, 2, 2)$, deci

$$\overrightarrow{AB} = \frac{3}{2}\,\overrightarrow{BC},$$

adică $r(A, B, C) = \frac{3}{2}$ (raportul $r(A, B, C)$ este acel scalar r pentru care are loc relaţia $\overrightarrow{AB} = r\,\overrightarrow{BC}$).

b) Condiţia $r(A, P, B) = 1$ este echivalentă cu $\overrightarrow{AP} = \overrightarrow{PB}$. Punctul P care verifică această condiţie este mijlocul segmentului $[AB]$, deci $P = \frac{1}{2}A + \frac{1}{2}B = (\frac{5}{2}, \frac{7}{2}, \frac{9}{2})$.

c) Semnele rapoartelor indică faptul că (i) B nu este între A şi Q; (ii) Q nu este între A şi B. Trebuie deci ca A să fie situat între Q şi B. Un astfel de punct este $Q = (0, 1, 2)$ (l-am ales ca fiind $A - (1, 1, 1)$). Au loc relaţiile

$$\overrightarrow{AB} = (3, 3, 3), \quad \overrightarrow{BQ} = (-4, -4, -4), \quad r(A, B, Q) = -\frac{3}{4},$$

$$\overrightarrow{AQ} = (-1, -1, -1), \quad \overrightarrow{QB} = (4, 4, 4), \quad r(A, Q, B) = -\frac{1}{4},$$

deci sunt verificate cerinţele din enunţ.

2. *Fie punctele* $P = (1, -1), Q = (3, 3)$.

a) *Calculaţi valoarea determinantului care apare în testul de orientare pentru muchia orientată* $\overrightarrow{PQ}$ *şi punctul de testare* $O = (0, 0)$.

b) *Fie* $R_\alpha = (\alpha, -\alpha)$, *unde* $\alpha \in \mathbb{R}$. *Determinaţi valorile lui* α *pentru care punctul* R_α *este situat în dreapta muchiei orientate* $\overrightarrow{PQ}$.

Soluţie.

a) Conform teoriei,

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

În exemplu avem:

$$\Delta(P, Q, R) = \begin{vmatrix} 1 & 1 & 1 \\ 1 & 3 & 0 \\ -1 & 3 & 0 \end{vmatrix} = 6 \text{ (dezvoltare după ultima coloană).}$$

Se poate verifica şi pe un desen că O este la stânga muchiei orientate $\overrightarrow{PQ}$.

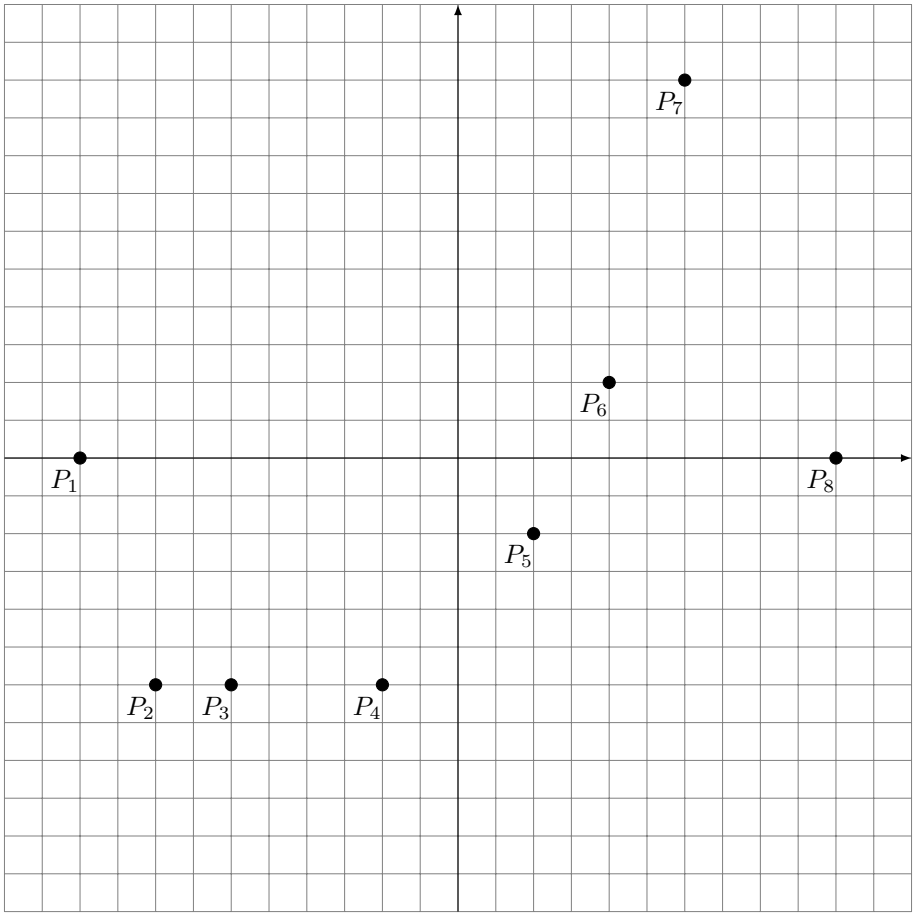
b) Calculăm, pentru un α, valoarea $\Delta(P, Q, R_\alpha)$:

$$\Delta(P, Q, R_\alpha) = \begin{vmatrix} 1 & 1 & 1 \\ 1 & 3 & \alpha \\ -1 & 3 & -\alpha \end{vmatrix} = 6(1 - \alpha).$$

Punctul R_α este situat în dreapta muchiei orientate $\overrightarrow{PQ} \Leftrightarrow \Delta(P, Q, R_\alpha) < 0 \Leftrightarrow \alpha > 1$. Acest lucru poate fi verificat şi pe desen, punctul R_α este variabil pe cea de-a doua bisectoare (de ecuaţie $x + y = 0$), iar pentru $\alpha > 1$ acest punct este situat în dreapta muchiei orientate $\overrightarrow{PQ}$.

3. *Fie* $\mathcal{M} = \{P_1, P_2, \ldots, P_9\}$, *unde* $P_1 = (-5, 0)$, $P_2 = (-4, -3)$, $P_3 = (-3, -3)$, $P_4 = (-1, -3)$, $P_5 = (1, -1)$, $P_6 = (2, 1)$, $P_7 = (3, 5)$, $P_8 = (5, 0)$. *Detaliaţi cum evoluează lista* $\mathcal{L}_i$ *a vârfurilor care determină marginea inferioară a frontierei acoperirii convexe a lui* $\mathcal{M}$, *obţinută pe parcursul Graham's scan, varianta Andrew.*

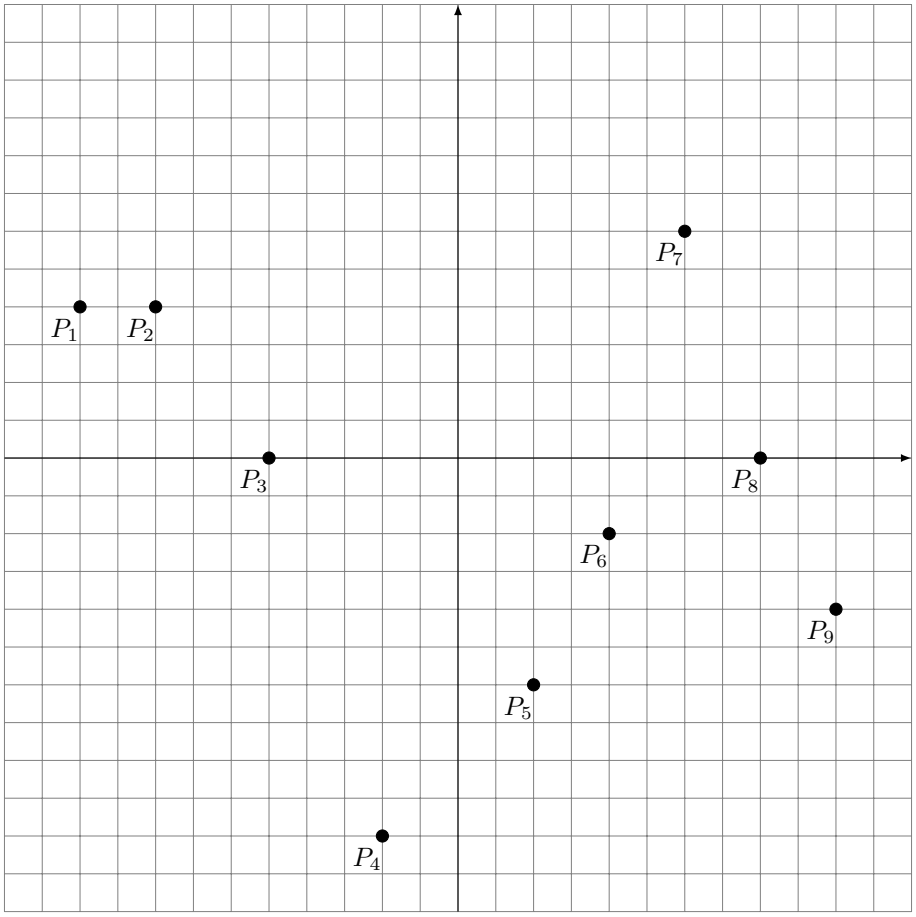
Soluţie.



Lista $\mathcal{L}_i$ evoluează astfel:
P_1P_2
$P_1P_2P_3$
$P_1P_2P_3P_4$ // este eliminat P_3, deoarece P_2, P_3, P_4 coliniare (nu viraj la stânga)
$P_1P_2P_4$
$P_1P_2P_4P_5$
$P_1P_2P_4P_5P_6$
$P_1P_2P_4P_5P_6P_7$
$P_1P_2P_4P_5P_6P_7P_8$ // punctele P_7, P_6, P_5 sunt eliminate în această ordine
$P_1P_2P_4P_8$ // lista finală ($\mathcal{L}_i$) a vârfurilor care determină marginea inferioară

4. *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 6 ($\mathcal{L}_i$ este lista vârfurilor care determină marginea inferioară a frontierei acoperirii convexe a lui $\mathcal{M}$, obţinută pe parcursul Graham's scan, varianta Andrew). Justificaţi!*

Soluţie.



Lista $\mathcal{L}_i$ are la final 3 elemente (P_1, P_4, P_9).

Numărul maxim de elemente este 6: $P_1P_4P_5P_6P_7P_8$ (la adăugarea lui P_8 în listă).

Obs. Numărul maxim de elemente *după verificări ale virajelor* este 5: $P_1P_4P_5P_6P_7$.

5. *Discutaţi un algoritm bazat pe paradigma Divide et impera pentru determinarea acoperirii convexe. Analizaţi complexitatea-timp.*

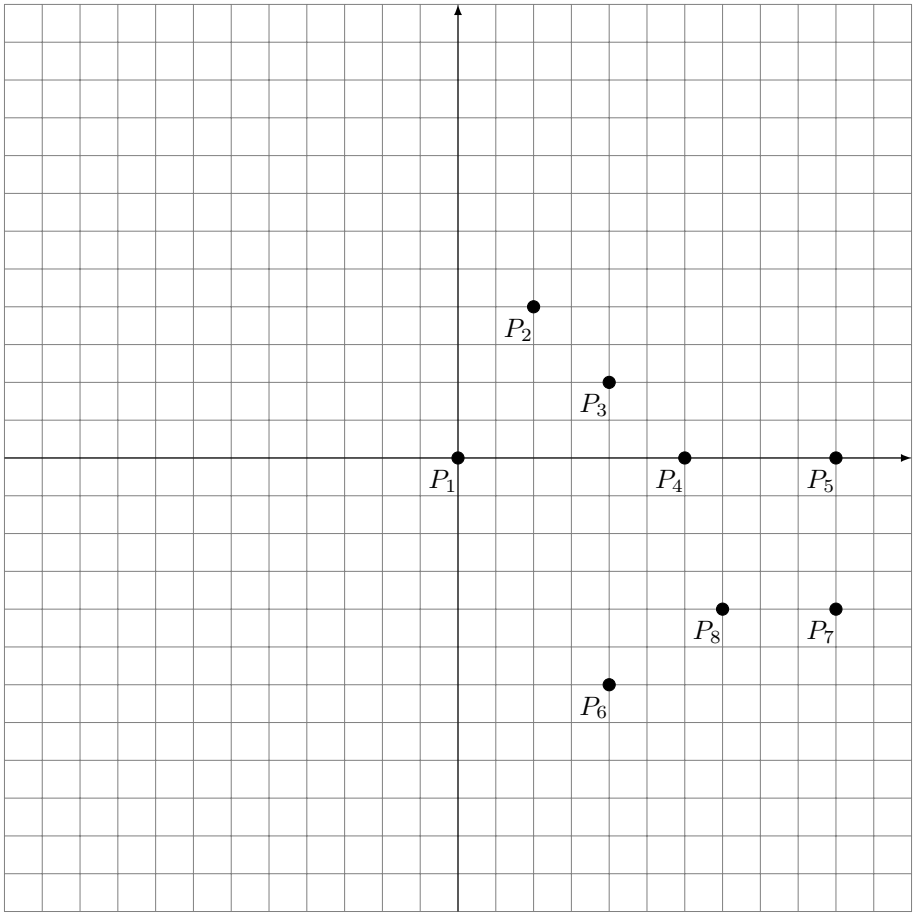
Soluţie. Complexitatea-timp este $O(n \log n)$. O descriere a algoritmului şi a analizei complexităţii poate fi găsită în survey-ul [Lee & Preparata, 1984].

Algoritmi avansaţi

Seminar 2 (săpt. 3 şi 4)

1. *Fie mulţimea* $\mathcal{P} = \{P_1, P_2, \ldots, P_7\}$*, unde* $P_1 = (0,0), P_2 = (1,2)$*,* $P_3 = (2,1), P_4 = (3,0), P_5 = (5,0), P_6 = (2,-3), P_7 = (5,-2)$*. Indicaţi testele care trebuie făcute pentru a găsi succesorul lui* P_1 *atunci când aplicăm Jarvis' march pentru a determina marginea inferioară a acoperirii convexe a lui* $\mathcal{P}$*, parcursă în sens trigonometric (drept pivot iniţial va fi considerat* P_2*).*

Soluţie.



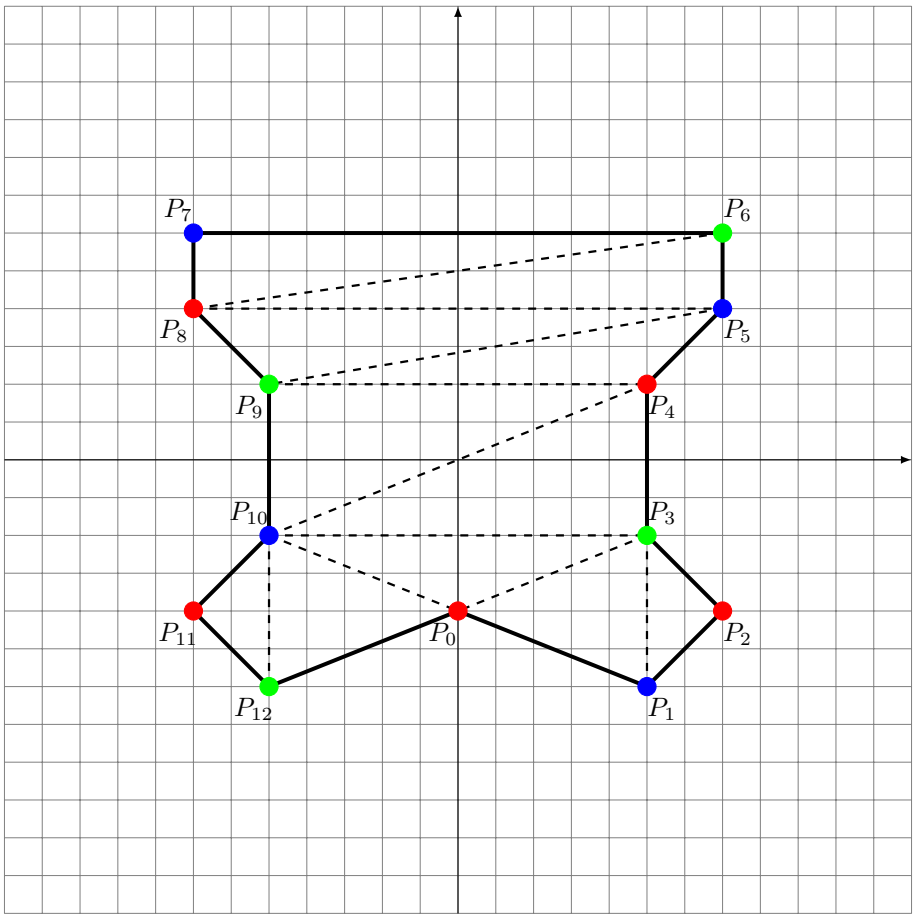
Pentru a găsi succesorul lui P_1 este adăugat pivotul P_2 (S cu notaţia din suportul de curs). Punctele sunt apoi testate, iar dacă un punct P este la

dreapta muchiei orientate P_1S, punctul P devine noul pivot.
Punctul P_3: este în dreapta muchiei P_1P_2, deci pivotul P_2 este înlocuit cu P_3.
Punctul P_4: este în dreapta muchiei P_1P_3, deci pivotul P_3 este înlocuit cu P_4.
Punctul P_5: nu este în dreapta muchiei P_1P_4, deci pivotul P_4 rămâne.
Punctul P_6: este în dreapta muchiei P_1P_4, deci pivotul P_4 este înlocuit cu P_6.
Punctul P_7: nu este în dreapta muchiei P_1P_6, deci pivotul P_6 rămâne.
Punctul P_8: nu este în dreapta muchiei P_1P_6, deci pivotul P_6 rămâne.

Au fost parcurse toate punctele. Ultimul pivot (P_6) este succesorul lui P_1 în parcurgerea frontierei acoperirii convexe a mulţimii date în sens trigonometric.

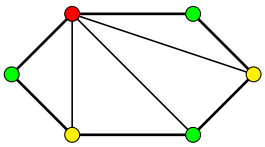
2. *Aplicaţi metoda din demonstraţia teoremei galeriei de artă, indicând o posibilă amplasare a camerelor de supraveghere în cazul poligonului* $P_0P_1P_2\dots P_{12}$, *unde* $P_0=(0,-2), P_1=(5,-6), P_2=(7,-4), P_3=(5,-2), P_4=(5,2), P_5=(7,4)$, $P_6=(7,6)$ *iar punctele* $P_7,\dots,P_{12}$ *sunt respectiv simetricele punctelor* $P_6,\dots,P_1$ *faţă de axa* Oy.

Soluţie. În figură sunt reprezentate o posibilă triangulare şi 3-colorarea asociată - există şi alte variante corecte.

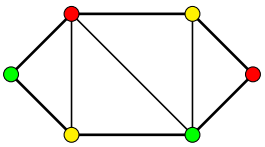


3. *Fie poligonul* $\mathcal{P} = (P_1P_2P_3P_4P_5P_6)$, *unde* $P_1 = (5, 0)$, $P_2 = (3, 2)$, $P_3 = (-1, 2)$, $P_4 = (-3, 0)$, $P_5 = (-1, -2)$, $P_6 = (3, -2)$. *Arătaţi că Teorema Galeriei de Artă poate fi aplicată în două moduri diferite, aşa încât, aplicând metoda din teoremă şi mecanismul de 3-colorare, 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* $\mathcal{P}$.

Soluţie. Poligonul este un hexagon convex, deci pentru triangularea sa vor fi folosite $3 \cdot 6 - 6 - 3 = 9$ muchii. Aceasta înseamnă că vom trasa 3 diagonale. Sunt posibile două situaţii: (a) cele trei diagonale au un vârf comun; (b) nu există un vârf comun al celor trei diagonale (acest lucru se poate demonstra trasând una dintre diagonale şi apoi raţionând inductiv - este esenţial că poligonul este un hexagon convex). În cazul (a) este suficientă o cameră, iar în cazul (b) 3-colorarea indică utilizarea a două camere.



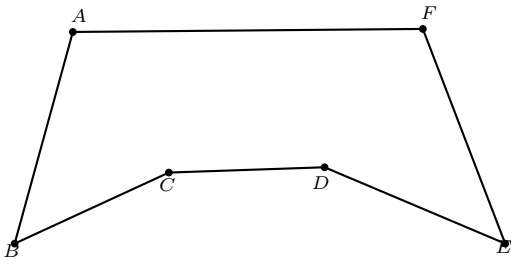
(a)



(b)

4. *Daţi exemplu de poligon cu* 6 *vârfuri care să aibă atât vârfuri convexe, cât şi concave şi toate să fie principale.*

Soluţie. În figură este desenat un poligon cu 4 vârfuri convexe şi 2 vârfuri concave. Pot fi luate în considerare şi alte variante (de exemplu cu un singur vârf concav, cu doar 3 vârfuri convexe, etc.).



5. *Fie* $\mathcal{M} = \{A_i \,|\, i = 0, \ldots, 50\} \cup \{B_i \,|\, i = 0, \ldots, 40\} \cup \{C_i \,|\, i = 0, \ldots, 30\}$*, dată de punctele* $A_i = (i + 10, 0)$*,* $i = 0, 1, \ldots, 50$*,* $B_i = (0, i + 30)$*,* $i = 0, 1, \ldots, 40$*,* $C_i = (-i, -i)$*,* $i = 0, 1, \ldots, 30$*. Determinaţi numărul de triunghiuri şi numărul de muchii ale unei triangulări a lui* $\mathcal{M}$*.*

Soluţie. Trebuie stabilite mai întâi numărul de puncte n şi numărul de puncte de pe frontiera acoperirii convexe k (atenţie la numărarea punctelor, nu trebuie numărat un punct de două ori...). Pe o schiţă se observă că sunt în total 123 de puncte (punctele din mulţimile $\{A_i \,|\, i = 0, \ldots, 50\}$, $\{B_i \,|\, i = 0, \ldots, 40\}$, respectiv $\{C_i \,|\, i = 0, \ldots, 30\}$ sunt diferite între ele). Obţinem $n = 123$, $k = 3$, apoi aplicăm formulele pentru determinarea numărului de triunghiuri, respectiv a numărului de muchii.

$$n_t = 2n - k - 2 = 241, \qquad n_m = 3n - k - 3 = 343.$$

6. *Daţi un exemplu de mulţime din* $\mathbb{R}^2$ *care să admită o triangulare având* 6 *triunghiuri şi* 11 *muchii.*

Soluţie. Fie n numărul de puncte ale unei astfel de mulţimi şi k numărul de puncte de pe frontiera acoperirii convexe. Au loc relaţiile

$$\begin{cases} 2n - k - 2 = 6 \\ 3n - k - 3 = 11 \end{cases}$$

Rezolvând acest sistem obţinem $n = 6$, $k = 4$, deci o astfel de mulţime are 6 puncte, din care 4 sunt situate pe frontiera acoperirii convexe.

Un posibil exemplu: $\{(0, 0), (5, 0), (5, 3), (0, 3), (1, 1), (3, 1)\}$.

7. *În* $\mathbb{R}^2$ *fie punctele* $P_1 = (1, 7)$*,* $P_2 = (5, 7)$*,* $P_3 = (7, 5)$*,* $P_4 = (1, 3)$*,* $P_5 = (5, 3)$*,* $P_6 = (\alpha - 1, 5)$*, cu* $\alpha \in \mathbb{R}$*. Discutaţi, în funcţie de* α*, numărul de muchii ale unei triangulări asociate mulţimii* $\{P_1, P_2, P_3, P_4, P_5, P_6\}$*.*

Soluţie. Trebuie analizată configuraţia punctelor $P_1, P_2, \ldots P_6$ şi determinate numărul n de puncte şi numărul k de puncte de pe frontiera acoperirii convexe.

Punctele $P_1P_2P_3P_4P_5$ determină un pentagon convex. Punctul P_6 descrie o dreaptă paralelă cu Ox care trece prin punctul P_3.

- Pentru $\alpha - 1 \leq 1$, adică $\alpha \in (-\infty, 2]$ punctul P_6 este situat în exteriorul sau pe laturile pentagonului $P_1P_2P_3P_4P_5$. Avem $n = 6, k = 6$, deci 4 feţe şi 9 muchii.
- Pentru $\alpha - 1 > 1$ şi $\alpha - 1 < 7$, adică $\alpha \in (2, 8)$ punctul P_6 este situat în interiorul pentagonului $P_1P_2P_3P_4P_5$. Avem $n = 6, k = 5$, deci 5 feţe şi 10 muchii.
- Pentru $\alpha - 1 = 7$, adică $\alpha \in \{8\}$ punctul P_6 coincide cu P_3. Avem $n = 5, k = 5$, deci 3 feţe şi 7 muchii.
- Pentru $\alpha - 1 > 7$, adică $\alpha \in (8, \infty)$ punctul P_6 este situat în exteriorul pentagonului $P_1P_2P_3P_4P_5$. Avem $n = 6, k = 6$, deci 4 feţe şi 9 muchii.

8. *Fie $\mathcal{G}$ un graf planar conex, v numărul de noduri, m numărul de muchii, f numărul de feţe. Se presupune că fiecare vârf are gradul ≥ 3. Demonstraţi inegalităţile*

$$v \leq \tfrac{2}{3}m, \qquad m \leq 3v - 6$$
$$m \leq 3f - 6, \quad f \leq \tfrac{2}{3}m$$
$$v \leq 2f - 4, \quad f \leq 2v - 4$$

Daţi exemplu de grafuri în care au loc egalităţi în relaţiile de mai sus.

Algoritmi avansaţi

Seminar 3 (săpt. 5 şi 6)

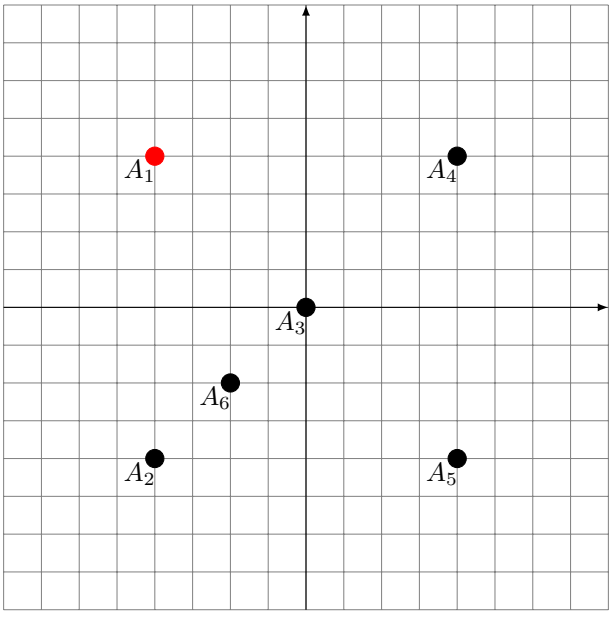
1. *Daţi exemplu de mulţime $\mathcal{M} = \{A_1, A_2, A_3, A_4, A_5, A_6\}$ din $\mathbb{R}^2$ astfel ca diagrama Voronoi asociată lui $\mathcal{M}$ să conţină exact patru semidrepte, iar diagrama Voronoi asociată lui $\mathcal{M} \setminus \{A_1\}$ să conţină exact cinci semidrepte. Justificaţi alegerea făcută.*

Soluţie. Numărul de semidrepte ale unei diagrame Voronoi este egal cu numărul de puncte de pe frontiera acoperirii convexe.

Alegem mulţimea $\mathcal{M}$ astfel ca A_1, A_2, A_3, A_4 să determine frontiera acoperirii convexe, iar pentru $\mathcal{M} \setminus \{A_1\}$ toate punctele să fie situate pe frontiera acoperirii convexe.

În figură este indicată configuraţia punctelor, însă, întrucât în enunţ se cere ca punctele să fie din $\mathbb{R}^2$, le indicăm în coordonate. În exemplul ales $A_1 = (-4, 4)$, $A_2 = (-4, -4)$, $A_3 = (0, 0)$, $A_4 = (4, 4)$, $A_5 = (4, -4)$, $A_6 = (-2, -2)$.

O altă soluţie posibilă (în care punctele A_2, A_6, A_3, A_4 nu mai sunt coliniare, este $A_1 = (-4, 4)$. $A_2 = (-4, -4)$, $A_3 = (0, 3)$, $A_4 = (4, 4)$, $A_5 = (4, -4)$, $A_6 = (-3, 0)$ (verificaţi prin desen!).



2. *a) Fie o mulţime cu n situri necoliniare. Atunci, pentru diagrama Voronoi asociată au loc inegalităţile*

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

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

b) Câte vârfuri poate avea diagrama Voronoi $\mathcal{D}$ asociată unei mulţimi cu cinci puncte din $\mathbb{R}^2$ ştiind că $\mathcal{D}$ are exact cinci semidrepte? Analizaţi toate cazurile. Este atins numărul maxim de vârfuri posibile ($n_v = 2n - 5$)? Justificaţi!

Soluţie. a) Cum punctele nu sunt coliniare, diagrama Voronoi are muchii de tip segment şi de tip semidreaptă. Se consideră un vârf suplimentar, notat v_∞, prin care, prin convenţie, trece fiecare muchie de tip semidreaptă. Se construieşte un graf $\mathcal{G}$ care are

- *vârfuri*: vârfurile diagramei Voronoi şi vârful v_∞ (deci $n_v + 1$ vârfuri);
- *muchii*: muchiile diagramei Voronoi - fiecare uneşte exact două vârfuri ale lui $\mathcal{G}$ (deci n_m muchii);
- *feţe*: în corespondenţă biunivocă cu siturile iniţiale (deci n feţe).

Graful $\mathcal{G}$ este un graf planar conex.

(i) Conform relaţiei lui Euler avem

$$(n_v + 1) - n_m + n = 2.$$

(ii) Analizăm incidenţele dintre muchii şi vârfuri.

- fiecare muchie este incidentă cu exact două vârfuri;
- fiecare vârf (inclusiv v_∞) este incident cu cel puţin trei muchii;
- numărând incidenţele în două moduri avem

$$2n_m \geq 3(n_v + 1).$$

Din (i) şi din (ii) rezultă inegalităţile dorite.

Observaţie. Pentru ca relaţiile din enunţ să fie egalităţi este necesar şi suficient ca fiecare vârf al lui $\mathcal{G}$ să fie incident cu exact trei muchii. Aceasta înseamnă să nu existe grupuri de (cel puţin) patru puncte conciclice şi pe frontiera acoperirii convexe a mulţimii de situri să fie exact trei situri.

b) Fie $\mathcal{M}$ o mulţime ca în enunţ. Diagrama Voronoi a lui $\mathcal{M}$ are cinci semidrepte, deci toate cele cinci puncte ale lui $\mathcal{M}$ sunt situate pe frontiera acoperirii convexe. În particular, conform observaţiei anterioare, nu este posibil să fie atins numărul maxim de vârfuri posibile ($n_v = 2n - 5 = 5$). Mai mult, urmând paşii din raţionament, observăm că, de fapt, $n_v \leq 3$. Sunt posibile trei situaţii:

- toate punctele sunt conciclice, $n_v = 1$,

- patru dintre puncte sunt conciclice, dar al cincilea nu este situat pe acelaşi cerc cu acestea, $n_v = 2$,
- oricare patru puncte nu sunt conciclice, $n_v = 3$.

3. *Fie punctele $O = (0,0)$, $A = (\alpha, 0)$, $B = (1,1)$, $C = (2,0)$, $D = (1,-1)$, unde $\alpha \in \mathbb{R}$ este un parametru. Discutaţi, în funcţie de α, numărul de muchii de tip semidreaptă ale diagramei Voronoi asociate mulţimii $\{O, A, B, C, D\}$.*

Soluţie. Numărul de muchii de tip semidreaptă este egal cu numărul de puncte de pe frontiera acoperirii convexe (le notăm cu m).

Punctul A este variabil pe axa Ox. Punctele O, B, C, D determină un pătrat şi trebuie doar să stabilim, în funcţie de α, poziţia lui A faţă de acest pătrat. Distingem următoarele cazuri:

- $\alpha < 0$: punctul A este situat în exteriorul pătratului $OBCD$ şi avem $m = 4$ (punctele de frontieră sunt A, B, C, D);
- $\alpha = 0$: avem $A = O$, deci $m = 4$;
- $0 < \alpha < 2$: punctul A este situat în interiorul pătratului $OBCD$ şi avem $m = 4$ (punctele de frontieră sunt O, B, C, D);
- $\alpha = 2$: avem $A = C$, deci $m = 4$;
- $\alpha > 2$: punctul A este situat în exteriorul pătratului $OBCD$ şi avem $m = 4$ (punctele de frontieră sunt O, B, D, A).

4. *(i) Fie punctul $A = (1,2)$. Alegeţi două drepte distincte d, g care trec prin A, determinaţi dualele A^*, d^*, g^* şi verificaţi că A^* este dreapta determinată de punctele d^* şi g^*.*

(ii) Determinaţi duala următoarei configuraţii: Fie patru drepte care trec printr-un acelaşi punct M. Se aleg două dintre ele; pe fiecare din aceste două drepte se consideră câte un punct diferit de M şi se consideră dreapta determinată de cele două puncte. *Desenaţi ambele configuraţii. Completaţi configuraţia iniţială (adăugând puncte/drepte) astfel încât să obţineţi o configuraţie autoduală (i.e. configuraţia duală să aibă aceleaşi elemente geometrice şi aceleaşi incidenţe ca cea iniţială).*

Soluţie. a) Conform teoriei, pentru un punct $p = (p_x, p_y)$ se defineşte dreapta $p^* : (y = p_x x - p_y)$ (duala lui p), iar pentru o dreaptă neverticală $d : (y = mx + n)$, se defineşte punctul $d^* = (m, -n)$ (dualul lui d).

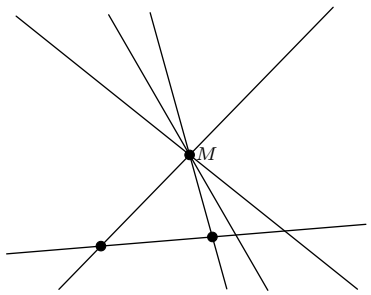
Alegem dreptele $d : (y = 2x)$ şi $g : (y = -x + 3)$. Avem:

$$A^* : (y = x - 2), \quad d^* = (2,0), \quad g^* = (-1,-3).$$

Se verifică imediat că punctele d^* şi g^* determină dreapta A^*.

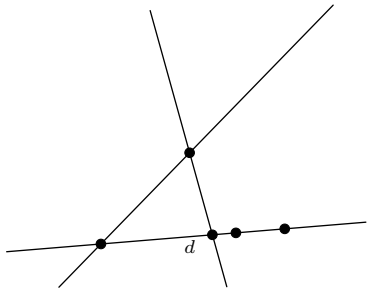
b) Configuraţia iniţială:

Fie patru drepte care trec printr-un acelaşi punct M. Se aleg două dintre ele; pe fiecare din aceste două drepte se consideră câte un punct diferit de M şi se consideră dreapta determinată de cele două puncte.



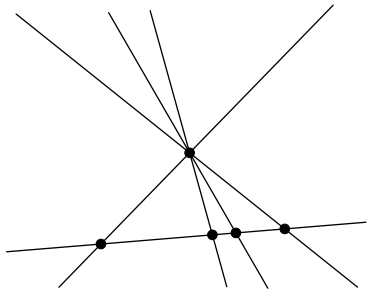
Configuraţia duală:

Fie patru puncte situate pe o aceeaşi dreaptă d. Se aleg două dintre ele; prin fiecare din aceste două puncte se consideră câte o dreaptă diferită de d şi se consideră punctul de intersecţie al acestor două drepte.



Configuraţia completată (astfel ca să fie autoduală):

Fie patru drepte care trec printr-un acelaşi punct. Pe fiecare dreaptă se alege câte un punct diferit de punctul comun, astfel ca aceste patru puncte să fie coloniare şi se consideră dreapta determinată de aceste patru puncte.



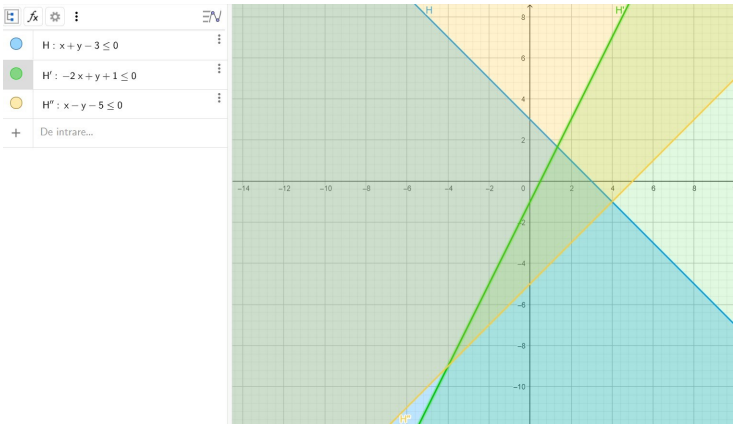
5. *a) Fie semiplanele* $H : x + y - 3 \leq 0$ *şi* $H' : -2x + y + 1 \leq 0$. *Daţi exemplu de semiplan* H'' *astfel ca intersecţia* $H \cap H' \cap H''$ *să fie un triunghi dreptunghic.*

b) Fie semiplanele H_1, H_2, H_3, H_4 *date de inecuaţiile*

$$H_1 : -y + 1 \leq 0; \qquad H_2 : y - 5 \leq 0; \qquad H_3 : -x \leq 0; \qquad H_4 : x - y + a \leq 0,$$

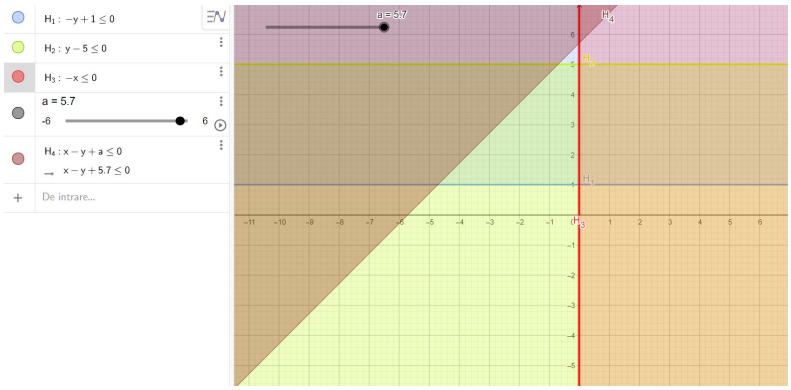
unde $a \in \mathbb{R}$ *este un parametru. Discutaţi, în funcţie de parametrul* a*, natura intersecţiei* $H_1 \cap H_2 \cap H_3 \cap H_4$.

Soluţie. a) Alegem semimplanul $H'' : x - y - 5 \leq 0$. Dreapta suport $x - y - 5 = 0$ este perpendiculară pe dreapta suport a lui H, $x + y - 3 = 0$. Intersecţia semiplanelor este un triunghi (cf. figură, se pot determina vârfurile acestuia), deci H'' verifică cerinţele din enunţ.

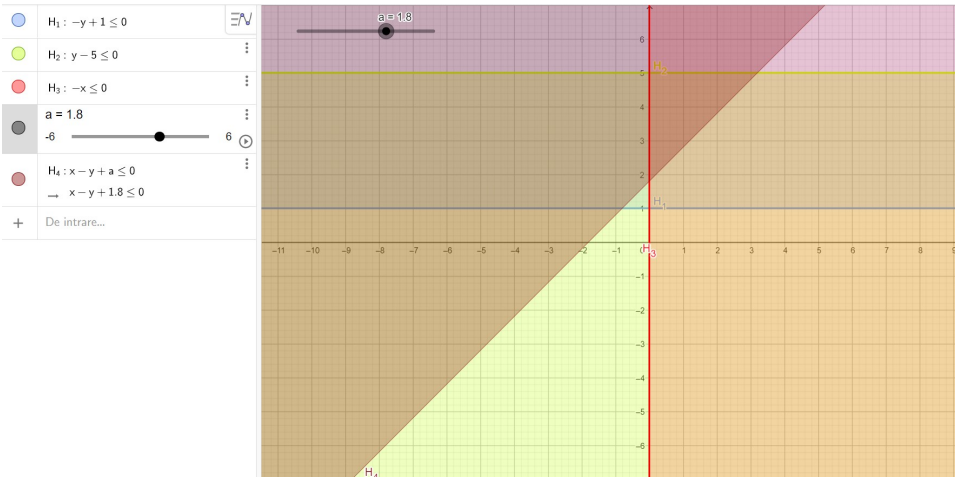


b) Intersecţia $H_1 \cap H_2 \cap H_3$ este o mulţime convexă nemărginită delimitată de dreptele $y = 1$, $y = 5$, respectiv $x = 0$. Vârfurile relevante sunt punctele $(0, 1)$ şi $(0, 5)$.

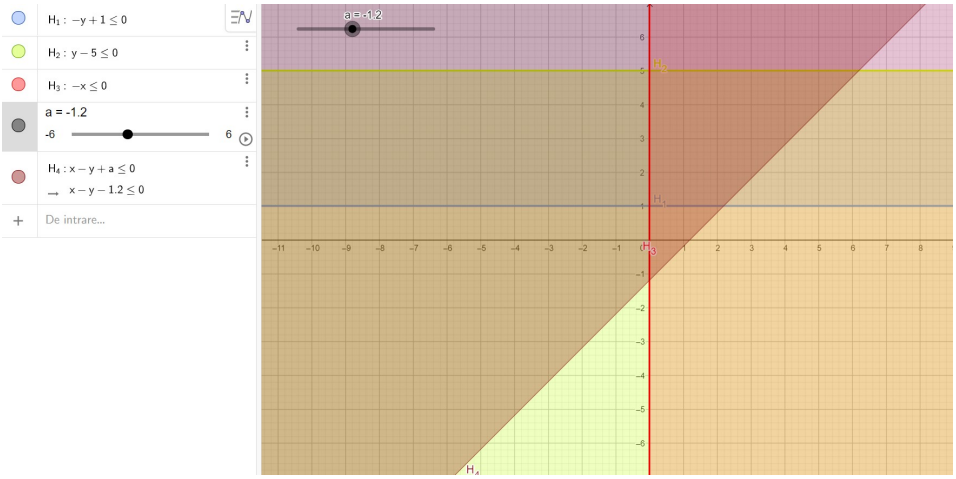
- Pentru $a > 5$ se obţine $0 \geq x - y + a > x - y + 5 \geq 0$ (deoarece $x \geq 0$ şi $-y + 5 \geq 0$), contradicţie. În acest caz intersecţia este mulţimea vidă.



- Pentru $a = 5$ se obţine punctul $(0, 5)$.
- Pentru $1 \leq a < 5$ se obţine un triunghi având unul dintre vârfuri $(0, 5)$, un vârf la intersecţia dintre dreptele $x = 0$ şi $x - y + a = 0$ şi un vârf la intersecţia dintre dreptele $y = 5$ şi $x - y + a = 0$. Pentru $a = 1$ punctul $(0, 1)$ este vârf al acestui triunghi.



- Pentru $a < 1$ se obţine un trapez. Laturile sale au ca drepte suport exact dreptele suport ale celor patru semiplane.



6. *Scrieţi inecuaţiile semiplanelor corespunzătoare şi studiaţi intersecţia acestora, dacă normalele exterioare ale feţelor standard sunt coliniare cu vectorii*

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

Soluţie. Conform teoriei, dacă normala exterioară a unei feţe f este coliniară cu un vector $\vec{v}_f = (a,b,c)$, semiplanul corespunzător feţei f este

$$ax + by + c \leq 0.$$

Se ajunge la inegalităţile

$$\begin{aligned} y - 1 &\leq 0 \\ y &\leq 0 \\ -1 &\leq 0 \\ -y &\leq 0 \\ -y - 1 &\leq 0 \end{aligned}$$

Acest sistem este compatibil, admiţând soluţia $y = 0$.

Notă. Ca în exemplul de la curs, am făcut abstracţie de componenta x.

7. ***(Suplimentar)*** *Demonstraţi că arborele parţial de cost minim al lui $\mathcal{P}$ este un subgraf al triangulării Delaunay.*

Soluţie. D. Mount, *Computational Geometry*, pag. 75.

Algoritmi Avansați
Seminar 4 (săpt. 7 și 8)

Problema rucsacului

Descrierea problemei. Se dă o mulțime de obiecte $O = \{ o_1, o_2, \dots, o_n \}$. Pentru fiecare obiect o_i știm cât este **valoarea** obiectului (pe care o vom nota val_i), respectiv **greutatea** obiectului (pe care o vom nota g_i).

Scopul este să găsim o submulțime de obiecte $S \subseteq O$ astfel încât suma greutăților obiectelor din S să nu depășească o capacitate C a unui rucsac (fixată de la început), iar valoarea totală a acestora să fie maximă.

Formularea matematică. Să se găsească $S \subseteq O$ care să maximizeze

$$\sum_{o_i \in S} val_i$$

cu condiția că

$$\sum_{o_i \in S} g_i \leq C$$

Observație. Toate submulțimile S care respectă condiția de mai sus se vor numi **soluții candidat**. Dintre acestea, soluția (sau soluțiile) care maximizează valoarea totală se va numi **soluția optimă**.

Varianta fracționară

Elementele din S pot fi fracțiuni din obiecte; cu alte cuvinte, pentru fiecare obiect o_i reținem și procentul din el pe care îl luăm în considerare, $p_i \in [0, 1]$. Fiecare obiect din soluție contribuie la valoarea totală cu $p_i \cdot val_i$, respectiv este contorizat spre greutatea totală cu $p_i \cdot g_i$.

Formularea matematică devine: să se găsească $S \subseteq O$ care să maximizeze $\sum_{o_i \in S} p_i \cdot val_i$ cu condiția că $\sum_{o_i \in S} g_i \leq C$.

Complexitatea pe care ar trebui să o aibă soluția voastră: $\mathcal{O}(n \log n)$

Varianta discretă

Elementele din S pot fi doar obiecte luate cu totul; cu alte cuvinte, pentru fiecare obiect o_i decidem dacă îl includem sau nu.

Complexitatea pe care ar trebui să o aibă soluția voastră: $\mathcal{O}(n \cdot C)$

Alte probleme pseudo-polinomiale

Submulțime de elemente cu suma dată

Se dă o mulțime de n numere naturale și un număr natural $M < 10000$. Să se determine, dacă există, o submulțime a mulțimii date de sumă egală cu M.

Exemplu. Pentru $n = 6$, mulțimea $\{ 12, 1, 3, 4, 5, 7 \}$ și $M = 14$, o soluție este submulțimea $\{ 3, 4, 7 \}$.

Complexitatea pe care ar trebui să o aibă soluția voastră: $\mathcal{O}(n \cdot M)$

Împărțirea cadourilor

Moș Crăciun a poposit la bradul a doi frați, unde și-a golit sacul. Când s-au trezit, frații au intrat într-o mare dilemă: cum își vor împărți ei cadourile? Știind că fiecare cadou are o valoare cuprinsă între 1 și 100 și că sunt maxim 100 de cadouri, scrieți un program care să determine sumele cadourilor fraților precum și modul de împărțire, astfel încât sumele obținute să fie cât de apropiate posibil.

Exemplu. Pentru 7 cadouri cu valorile $\{ 28, 7, 11, 8, 9, 7, 27 \}$, sumele sunt 48 și 49, o împărțire a cadourilor fiind: $\{ 28, 11, 9 \}$, respectiv $\{ 7, 8, 7, 27 \}$.

Complexitatea pe care ar trebui să o aibă soluția voastră: $\mathcal{O}(n \cdot S)$, unde S reprezintă valoarea totală a cadourilor.

Alte probleme de programare dinamică

Generalizarea problemei spectacolelor

Se dau n activități, pentru fiecare având: **timpul de început**, **timpul de sfârșit** și **profitul** asociat desfășurării activității (adică n intervale de numere reale, fiecare cu o pondere asociată). Să se determine o submulțime

de activități compatibile (intervale disjuncte două câte două) care au profitul total maxim. Se vor afișa profitul total și activitățile selectate.

Exemplu. Pentru $n = 4$ și activitățile:

- $A_1 = [1, 3]$ cu profitul $p_1 = 1$
- $A_2 = [2, 6]$ cu profit $p_2 = 8$
- $A_3 = [4, 7]$ cu profit $p_3 = 2$
- $A_4 = [10, 15]$ cu profit $p_4 = 5$

O soluție optimă ar fi formată din activitățile A_2 și A_4 (adică intervalele $[2, 6]$ și $[10, 15]$), cu profitul total 13.

Complexitatea pe care ar trebui să o aibă soluția voastră: încercați să găsiți o soluție în $\mathcal{O}(n \log n)$; dacă nu reușiți, încadrați-vă în cel mult $\mathcal{O}(n^2)$.

Generalizarea problemei planificării activităților

Se dă o listă de n activități. Pentru fiecare activitate cunoaștem **profitul** (p_i), **termenul limită** până la care trebuie efectuată (t_i) și **durata** acesteia (l_i). Trebuie să găsim o submulțime *ordonată* (o sublistă) de activități pentru a obține un profit total maxim.

Exemplu. Pentru $n = 4$ și activitățile:

- A_1 cu $p_1 = 3$, $t_1 = 5$, $l_1 = 3$
- A_2 cu $p_2 = 2$, $t_2 = 2$, $l_2 = 1$
- A_3 cu $p_3 = 3$, $t_3 = 2$, $l_3 = 2$
- A_4 cu $p_4 = 5$, $t_4 = 4$, $l_4 = 3$

O soluție optimă se obține dacă planificăm activitățile în ordinea A_2, A_4, profitul fiind 7.

Observație. Problema discretă a rucsacului poate fi privită ca un caz particular al acestei probleme (obiectele sunt activități de durată g_i, profit c_i și termen limită C).

Complexitatea pe care ar trebui să o aibă soluția voastră: $\mathcal{O}(n \cdot T + n \log n)$, unde $T = \max \{ t_i \}$.

Seminar 5

Ştefan Popescu

April 2023

Vouchere de Vacanță

Cerință

(10p) George a intrat în concediu, şi ca atare doreşte să consume cât mai mult din pachetul său de vouchere de vacanţă. El şi-a făcut o listă cu n locaţii unde poate cheltui din suma alocată. Ştiind că George nu are de gând să cheltuiască din banii proprii decât pe transportul între locaţii şi că el se poate deplasa între oricare două locaţii, ajutaţi-l pe George să găsească o listă de locaţii pe care să le viziteze astfel încât să îşi folosească voucherele cât mai eficient.

Cerinţe:

Se cere un algoritm 1/2-aproximativ care rezolvă problema lui George în timp eficient din punct de vedere al complexităţii timp. (pseudocod + justificare)

Algoritmul va avea ca output lista de locaţii şi suma totală ce va fi cheltuită.

Notaţii şi indicaţii:

- Notăm cu S suma de bani pe care o are George sub formă de vouchere.
- Locaţiile sunt etichetate cu numerele $1, 2, ..., n$. Costurile celor n locaţii sunt notate cu $C_1, C_2, ..., C_n$
- Suma de bani S este alocată pe un card, nu sub formă de tichete. Din cont se poate extrage orice sumă disponibilă.
- Se lucrează cu numere întregi.

Soluție

Problema este în esență o variantă a *Problemei Rucsacului* în care valoarea și greutatea pentru fiecare obiect sunt egale. Se poate folosi algoritmul din curs, care este 1/2 aproximativ.

Camioane și Transporturi

Cerință

(15p) Avem n colete de transportat, fiecare având greutatea de $w_1, w_2, ..., w_n$. Pentru a le transporta, putem folosi un număr de camioane, fiecare având capacitatea de transport G. presupunem că $w \leq G$, pentru orice i. Ne dorim sa minimizăm numărul de camioane folosite. Considerăm următorul plan de încărcare a camioanelor:

Odată ce avem la dispoziție un camion pentru a fi încărcat, iterăm prin mulțimea coletelor, încărcându-le în camion, până când dăm peste primul colet ce nu mai încape. În acel moment considerat că am terminat de încărcat camionul curent și trecem la următorul camion, prima dată încărcând coletul care nu a mai încăput în cel precedent.

Cerințe:

a) Arătați, printr-un exemplu simplu, că metoda de mai sus nu furnizează soluția optimă. **(5p)**

b) Arătați totuși că soluția de mai sus este un algoritm 2-aproximativ pentru problema noastră. **(10p)**

Notații:

n - numărul de colete

G - capacitatea unui camion

W - Lista cu greutățile coletelor în ordinea în care sosesc spre a fi procesate.

OPT – soluția optimă care există: Numărul minim de camioane ce poate fi folosit pentru a transporta cele n colete de greutate totală G

ALG – soluția oferită de algoritmul nostru: numărul de camioane folosite conform algoritmului

$G[i]$ - greutatea cu care este încărcat camionul i în configurația dată de algoritmul nostru. Evident $G[i] \leq G$.

Soluție

a) Un exemplu ar putea fi $n = 3, G = 100, W = \{30, 80, 60\}$; soluțiile obținute sunt: $ALG = 3, OPT = 2$.

b) Presupunem că soluția dată de algoritm folosește $2p$ camioane. Vom arăta că soluția optimă folosește cel puțin p camioane:

Din algoritm se poate observa că $G[2i-1] + G[2i] \geq G, \forall i \in \{1, 2, ..., p\}$. Deci pentru fiecare pereche de camioane consecutive folosite de algoritmul nostru, o abordare optima ar folosi cel puțin 1 camion. Cum algoritmul nostru folosește exact p perechi de camioane, o abordare optimă ar folosi cel puțin p camioane.

Dacă soluția noastră folosește $2p + 1$ camioane:

Din analiza precedentă am putut observa că primele p perechi de camioane transportă o încărcătură totală de minim $p * G$. Pentru această încărcătură sunt necesare cel puțin p camioane. Însă mai este nevoie de un camion în plus pentru a transporta și $G[2p+1]$, pentru un total de minim $p+1$ camioane într-o configurație optimă, atunci când algoritmul nostru folosește $2p + 1$ camioane.

Cadourile lui Moș Crăciun

Cerință

(30p) O clasă de elevi este formată din m copii. La această clasă Moș Crăciun aduce n cadouri, fiecare dintre ele având o valoare notată $val_1, val_2, ..., val_n$. Moșul, grăbit fiind, lasă task-ul împărțirii cadourilor pe umerii profesorului de informatică. Acesta își alege următorul obiectiv pentru împărțirea cadourilor: sa maximizeze valoarea cadourilor primite de către copilul care primește cel mai puțin. (Altfel spus, copilul cel mai "vitregit" în urma împărțirii să primească totuși cadouri de o valoare cât mai bună / Să existe un "spread" cât mai bun al cadourilor). În timp ce se gândește la o soluție de implementare, profesorul își amintește de la cursurile de algoritmică din facultate că astfel de probleme de optim sunt NP-hard! Fiind cuprins de groază (dar și de o nostalgie pentru cursurile de algoritmică din facultate) el vă cere vouă ajutorul.

Notații:

OPT – soluția optimă care există: Valoarea totală a cadourilor primite de copilul cel mai "vitregit" în o configurație optima

ALG – soluția oferită de algoritmul vostru: Valoarea totală a cadourilor primite de copilul cel mai "vitregit" în urma algoritmului vostru

$C[k]$ - lista cadourilor primite de către copilul k la finalul algoritmului vostru

$W(K)$ – valoarea totala a cadourilor primite de către copilul k la finalul algoritmului vostru

$val(i)$ – valoarea cadoului i

$Val = \sum_{1 \le i \le n} val(i)$ – valoarea totală a cadourilor

Copiii vor fi indexați folosind variabile de forma k, q, k', q', etc... Cadourile vor fi indexate folosind variabile de forma i, j, i', j', etc....

Restricții: Considerăm că $n \ge m$ și că $val(i) \le \frac{Val}{2m}$ pentru oricare cadou i.

Cerințe:

a) Descrieți un algoritm $\frac{1}{2}$ aproximativ pentru problema cadourilor in complexitate $\mathcal{O}(n \log m)$ **(10p)**

b) Fie k acel copil pentru care $ALG = W(K)$. Fie i ultimul cadou primit de un copil oarecare q ($q \neq k$). Care este relația între $W(K)$ și $W(Q) - val(i)$? Justificaţi. **(5p)**

c) Pe baza punctului b) arătați că $ALG \ge \frac{Val}{2m}$ **(5p)**

d) Demonstrați că algoritmul descris la punctul a) este $\frac{1}{2}$ aproximativ **(5p)**

e) Dați un exemplu format din minimum 2 copii și 4 cadouri pentru care algoritmul vostru nu găsește soluția optima. Spuneți care este soluția optima. Spuneți care este soluția dată de algoritmul vostru. **(5p)**

Soluție

a) **for** $k \in [1 : m]$ **do**: *// initializare*

$C[k] \leftarrow \emptyset$; $W(K) \leftarrow 0$;

for $i \in [1 : n]$ **do**: *// pentru fiecare cadou $'i'$*; $\mathcal{O}(n)$

$q \leftarrow k$ cu proprietatea ca W(K) - minima; *//alegem pe cel mai "vitregit" copil;* $\mathcal{O}(\log m)$ *cu minheap sau priority queue*

$C[q] \leftarrow C[q] \cup \{i\}$ *//adaugam cadoul $'i'$ in lista de cadouri ale lui $'q'$*

$W(Q) \leftarrow W(Q) + val(i)$ *//se actualizează valoarea totala a cadourilor lui $'q'$.*

b) Deoarece cadoul i este atribuit copilului q, înseamnă că, la acel moment, q avea cadourile de valoare totală cea mai mică, deci în mod evident mai mică decât cea a lui k. Deci avem relația $W(K) \ge W(Q) - val(i)$. Aceasta poate fi rescrisă ca $W(K) + val(i) \ge W(Q)$

c) Fie k acel copil pentru care $ALG = W(K)$. Vom presupune prin absurd că $W(K) < \frac{Val}{2m}$. Deoarece $W(K) + val(i) \ge W(Q)$ și $val(i) \le \frac{Val}{2m}$ avem:

$$\begin{aligned} Val &= \sum_{1 \le q \le m} W(Q) = W(K) + \sum_{1 \le q \le m}^{q \neq k} W(Q) \\ &\le W(K) + (m-1)(W(K) + \frac{Val}{2m}) \\ &\le m * W(K) + (m-1) * \frac{Val}{2m} \\ &< m * W(K) + \frac{Val}{2} \\ &< m * \frac{Val}{2m} + \frac{Val}{2} \\ &< \frac{Val}{2} + \frac{Val}{2} \\ &< Val(!) \end{aligned}$$

d) Știm că $OPT \leq \frac{Val}{m}$ (cazul unei egalități ar implica un "spread" perfect). Am demonstrat la punctul *c*) că $ALG \geq \frac{Val}{2m}$. Rezultă deci că $2 * ALG \geq \frac{Val}{m} \geq OPT$; Iar în final avem relația dorită: $\mathbf{ALG \geq \frac{1}{2} * OPT}$

e) * $m = 2$, $n = 4$; Cadourile au valorile: $2, 3, 1, 6$. [1] Soluția optimă implică faptul că primul copil primește cadoul 4 iar cel de-al doilea cadourile $1, 2, 3$. În acest caz ambii copii primesc cadouri în valoare de 6 unități. Algoritmul nostru va rezulta în alocarea primului copil cadourile $1, 3$, celui de-al doilea copil cadoul 2 iar cadoul 4 oricăruia dintre cei doi. În orice caz, copilul cel mai vitregit primește cadouri în valoare de doar 3 unități.

[1] Acest exemplu are o scăpare: nu respectă restricția $val(i) \leq \frac{Val}{2m}$

Algoritmi Avansați
Seminar 6

Gabriel Majeri

1 Introducere

În acest seminar vom discuta câteva tipuri de algoritmi care nu se încadrează în tiparele obișnuite de rezolvare ale problemelor, dar care pot fi extrem de utili și versatili în unele situații.

2 Algoritmi probabiliști

O metodă de a rezolva mai eficient unele probleme dificile din punct de vedere computațional este să ne folosim de *șansă*. Astfel obținem algoritmi „probabiliști" / „randomizați" [1].

Un scurt istoric

În cadrul Proiectului Manhattan, cercetătorii Stanislaw Ulam și John von Neumann aveau nevoie să rezolve niște probleme foarte complicate legate de difuzia neutronilor. Așa că au venit cu ideea să modeleze pe calculator o simulare a reacțiilor care au loc în nucleul unei arme de fisiune nucleară, ca să înțeleagă ce se întâmplă fără să mai fie nevoie de un experiment fizic. Astfel a fost creată și folosită pentru prima dată *metoda Monte Carlo* [2, 3]: modelarea pe calculator a unei probleme teoretice pentru a o rezolva numeric.

Generarea numerelor aleatoare

Primul pas în implementarea unui algoritm probabilist este să avem acces la o sursă de numere aleatoare [4].

Calculatoarele digitale obișnuite sunt mașinării *deterministe*, care nu pot produce numere cu adevărat aleatoare. Un calculator poate folosi un **generator de numere pseudo-aleatoare** (PRNG [5]) pentru a genera un șir de numere care aparent nu au nicio legătură unul cu celălalt, plecând de la un număr dat (denumit *seed*); dar dacă numărul inițial este dezvăluit, se pot determina

toate numerele care urmează. Din acest motiv, dacă aveți nevoie să generați valori aleatoare într-un context sensibil (e.g. în criptografie), recomandarea este să utilizați un **generator de numerele pseudo-aleatoare sigur din punct de vedere criptografic** (CSPRNG [6]).

2.1 Algoritmi Monte Carlo

Un algoritm de tip Monte Carlo se execută pentru un anumit număr de pași și ne oferă o soluție care este doar *probabil* corectă; cu cât lăsăm algoritmul să ruleze mai mult, cu atât ne apropiem de soluția optimă sau o aproximăm mai bine [7].

Un exemplu de astfel de algoritm este un algoritm bazat pe programarea genetică; cu cât lăsăm să ruleze mai mult programul, cu atât soluțiile obținute sunt din ce în ce mai bune.

2.2 Algoritmi Las Vegas

Un algoritm de tip Las Vegas s-ar putea să nu obțină o soluție validă după o execuție, și să fie nevoie să-l rulăm de mai multe ori. Dar în momentul în care a obținut o soluție, aceasta este sigur cea corectă [8].

Un exemplu de algoritm de acest tip este cel care rezolvă problema așezării a 8 regine pe tabla de șah alegând aleator unde să pună o regină pe următorul rând; există riscul ca o soluție pe care a început să o construiască să nu fie corectă și să fie obligat să o ia de la capăt, dar în momentul în care a ajuns la final, soluția este sigur bună.

Vedere de ansamblu

Tabelul de mai jos compară cele două familii principale de algoritmi probabiliști, arătând cum una este „duala” celeilalte:

	Timp de execuție	Corectitudine rezultat
Monte Carlo	Determinist	Probabilistă
Las Vegas	Probabilist	Deterministă

Exerciții

1. Dacă aveți la dispoziție o funcție `random()` care generează un număr real în intervalul $[0, 1]$ (cu o distribuție uniformă), cum puteți obține un număr real în intervalul $[a, b]$ (cu o distribuție uniformă pe $[a, b]$)?

Soluție: Fie x un număr din intervalul $[0, 1]$, pe care l-am obținut prin funcția `random()`. Înmulțindu-l cu $b - a$, obținem un nou număr $(b - a)x$, care se află în intervalul $[0, b - a]$. Adunând valoarea lui a la acesta, obținem $a + (b - a)x$, care este în intervalul $[a, b]$.

Această formulă este utilă în limbajele care nu oferă în biblioteca standard o funcție pentru a genera un număr aleator dintr-un interval arbitrar $[a, b]$, ci doar din $[0, 1]$, cum se întâmplă în cazul limbajului JavaScript. □

2. Dacă aveți la dispoziție o funcție `biased_random()` care generează aleator 0 sau 1, dar cu o probabilitate inegală p ($\neq 50\%$), puteți să definiți o funcție care să vă genereze 0 sau 1 cu probabilitate 50%–50%?

Soluție: Da, putem construi o astfel de funcție.

Observația de la care plecăm este că, dacă apelăm de două ori funcția `biased_random()`, singurele valori pe care le putem obține 00, 01, 10 sau 11. Calculând probabilitatea pentru fiecare dintre aceste situații, obținem

X	00	01	10	11
$\mathbb{P}(X)$	$(1-p)^2$	$p(1-p)$	$(1-p)p$	p^2

unde am notat cu p probabilitatea ca un apel al funcției `biased_random()` să returneze valoarea 1. Există o simetrie în tabel: $p(1-p) = (1-p)p$; deci $\mathbb{P}(01) = \mathbb{P}(10)$.

```
def unbiased_random():
    while True:
        x = biased_random()
        y = biased_random()

        # Suntem în cazul 00 sau 11, care nu ne ajută
        if x == y:
            continue
        else:
            if x == 0:
                # Cazul 01
                return 0
            else:
                # Cazul 10
                return 1
```

Remarcăm că această implementare este un *algoritm probabilist de tip Las Vegas*; rezultatul obținut respetă întotdeauna cerința (este 0 sau 1 cu pro-

babilitate 50%–50%), dar nu putem spune de dinainte de câte ori trebuie să se execute blocul `while` înainte de a obține un rezultat valid. □

3. În acest exercițiu ne propunem să construim un algoritm probabilist care să ne ajute să determinăm valoarea lui π.

1. Care este aria cercului unitate?
2. Care este aria pătratului definit de punctele $(-1,-1)$, $(1,-1)$, $(1,1)$ și $(-1,1)$?
3. Dacă aleg aleator un punct în pătratul descris mai sus, care este probabilitatea ca el să fie în interiorul cercului unitate?
4. Dacă am un punct $(x,y) \in \mathbb{R}^2$, cum pot verifica dacă acesta se află în interiorul cercului unitate?
5. Avem următorul algoritm: luăm aleator perechi de numere din $[-1,1]$ (adică puncte din pătratul descris mai sus). Dacă punctul descris de această pereche de numere este în interiorul cercului unitate îl contorizăm, altfel îl ignorăm. La final, împărțim numărul de puncte care au căzut în interiorul cercului la numărul total de puncte luate în considerare. Ce valoare aproximează acest raport?
6. Ce fel de algoritm este cel descris anterior? Monte Carlo sau Las Vegas?

Soluție:

1. Aria cercului unitate $A_\bigcirc = \pi \cdot 1^2 = \pi$.
2. Laturile definite de acele puncte au toate lungimea $|1-(-1)| = 2$, deci pătratul corespunzător are aria $A_\square = 2^2 = 4$.
3. Având în vedere că cercul unitate este conținut în pătratul descris anterior, probabilitatea în acest caz va fi raportul dintre aria cercului și aria pătratului, $\mathbb{P}(X) = A_\bigcirc / A_\square = \pi/4$.
4. Cel mai ușor este să folosim formula pentru distanța Euclidiană față de origine, $d = \sqrt{x^2+y^2}$, și să verificăm că $d \leq 1$.
5. Acest raport aproximează probabilitatea ca un punct aleator din pătratul descris anterior să fie în cercul unitate. Cu alte cuvinte, dacă n este numărul de puncte din eșantion, avem că

$$\lim_{n\to+\infty} \frac{\text{nr. puncte în cercul unitate}}{\text{nr. total de puncte}} = \frac{A_\bigcirc}{A_\square} = \frac{\pi}{4}.$$

Dacă înmulțim rezultatul cu 4, avem o metodă probabilistă de a calcula numeric valoarea lui π.

6. Algoritmul este unul de tip Monte Carlo. Cu cât rulăm pentru mai mult timp algoritmul și eșantionăm mai multe puncte, cu atât crește precizia rezultatului nostru.

□

3 Programare liniară și algoritmul simplex

Unele dintre cele mai simple tipuri de probleme de optimizare (și cele mai des întâlnite în practică) sunt cele de *programare liniară*[1]. Din fericire, pentru acestea avem o interpretare vizuală destul de clară (cel puțin, în cazul 2D) și un algoritm eficient de rezolvare (metoda simplex).

Pentru a înțelege cum arată aceste probleme și cum le putem rezolva, vă recomand să vă uitați la [9], la capitolul 13 din [10] (notele de curs sunt disponibile aici), la această lecție [11] dintr-un curs de la MIT sau la acest video [12].

[1] În acest context, la fel ca în cazul „programării dinamice”, termenul de „programare” se referă la o metodă tabelară de rezolvare a unei probleme, nu neapărat la programare/dezvoltare software.

Referințe

[1] David Rydeheard, *Probabilistic (Randomized) algorithms*, URL: http://www.cs.man.ac.uk/~david/courses/advalgorithms/probabilistic.pdf.

[2] Roger Eckhardt, „Stan Ulam, John von Neumann, and the Monte Carlo method", în *Los Alamos Science* (15 1987), pp. 131–137, URL: https://permalink.lanl.gov/object/tr?what=info:lanl-repo/lareport/LA-UR-88-9068.

[3] Wikipedia contributors, *Monte Carlo method*, URL: https://en.wikipedia.org/wiki/Monte_Carlo_method.

[4] Wikipedia contributors, *Random number generation*, URL: https://en.wikipedia.org/wiki/Random_number_generation.

[5] Wikipedia contributors, *Pseudorandom number generator*, URL: https://en.wikipedia.org/wiki/Pseudorandom_number_generator.

[6] Wikipedia contributors, *Cryptographically-secure pseudorandom number generator*, URL: https://en.wikipedia.org/wiki/Cryptographically-secure_pseudorandom_number_generator.

[7] Wikipedia contributors, *Monte Carlo algorithm*, URL: https://en.wikipedia.org/wiki/Monte_Carlo_algorithm.

[8] Wikipedia contributors, *Las Vegas algorithm*, URL: https://en.wikipedia.org/wiki/Las_Vegas_algorithm.

[9] Trefor Bazett, *Intro to Linear Programming and the Simplex Method*, URL: https://www.youtube.com/watch?v=K7TL5NMlKIk.

[10] Georgia Tech, *Computability, Complexity & Algorithms*, URL: https://www.udacity.com/course/computability-complexity-algorithms--ud061.

[11] MIT OpenCourseWare, *15. Linear Programming: LP, reductions, Simplex*, URL: https://www.youtube.com/watch?v=WwMz2fJwUCg.

[12] The Organic Chemistry Tutor, *Linear Programming*, URL: https://www.youtube.com/watch?v=Bzzqx1F23a8.