

# ***Algoritmai ir duomenų struktūros***

13 paskaita

2022-05-18

# ***Svarbi informacija***

- Gegužės 18 d. paskaita skirta kursui užbaigti.
- Gegužės 25 d. paskaita dedikuota Jūsų pasirinktų temų pristatymams ir gražiausių 3D modelių išrinkimui.
- Atsiskaityti praktines užduotis galima iki gegužės galo.
- **Egzaminas birželio 20 d. 12 val. nuotoliniu būdu.**
- **Egzamino trukmė – 2 val.**

# ***13 paskaitos tikslas***

- Apžvelgti programavimo užduotis:
  - C9 – Look-and-say seka,
  - C7 – Šachmatų varžybų tvarkaraščio generavimas,
  - B15 – Deloné trianguliacija.
- Apžvelgti kitus geometrinius uždavinius, jų sprendimo algoritmus ir naujas programavimo užduotis:
  - C12 – plokštumos atkarpų susikirtimo patikrinimas,
  - C13 – erdvės trikampių susikirtimo patikrinimas.
- Pradėti kurso kartojimą.

# ***Look-and-say seka***

1, 11, 21, 1211, 111221, 312211, 13112221, 1113213211, ...

- 1 = vienas vienetas  $\rightarrow$  11,
- 11 = du vienetai  $\rightarrow$  21,
- 21 = vienas dvejetas, vienas vienetas  $\rightarrow$  1211,
- 1211 = vienas vienetas, vienas dvejetas, du vienetai  $\rightarrow$  111221, ...

## ***Apibendrintoji look-and-say seka***

- Pirmasis sekos narys – natūralusis skaičius, pavyzdžiui:
- 2, 12, 1112, 3112, 132112, 1113122112, 311311222112, ...
- 13, 1113, 3113, 132113, 1113122113, 311311222113, ...

# ***Šachmatų varžybų tvarkaraščio sudarymo algoritmas***

- Varžybose dalyvauja  $n > 1$  žaidėjų.
- Žaidėjai poromis turi sužaisti po vieną kartą.
- Jei žaidėjų skaičius  $2k + 1$ , tai kiekvienas žaidėjas turi  $k$  partijų sužaisti balta ir  $k$  partijų juoda spalva (be to, kažkuris žaidėjas turi palaukti, kol kiti suloš vieną ratą).
- Jei žaidėjų skaičius  $2k$ , tai pirmoji pusė žaidėjų sulošia  $k$  partijų balta ir  $k - 1$  partijų juoda spalva.

# Šachmatų varžybų tvarkaraščio sudarymo algoritmas (1)

- Tarkime varžybose dalyvauja 7 (arba 8) žaidėjai.
- Tikslas – sudaryti tokį tvarkaraštį:

1 ratas:	1 - (8)	2 - 7	3 - 6	4 - 5
2 ratas:	(8) - 5	6 - 4	7 - 3	1 - 2
3 ratas:	2 - (8)	3 - 1	4 - 7	5 - 6
4 ratas:	(8) - 6	7 - 5	1 - 4	2 - 3
5 ratas:	3 - (8)	4 - 2	5 - 1	6 - 7
6 ratas:	(8) - 7	1 - 6	2 - 5	3 - 4
7 ratas:	4 - (8)	5 - 3	6 - 2	7 - 1

# Šachmatų varžybų tvarkaraščio sudarymo algoritmas (2)

- Iš pradžių sudarome tuščią 7 ratų lentelę.
- Jei žaidėjų skaičius  $2k$  arba  $2k - 1$ , reikia  $k - 1$  ratų.

1 ratas:	-	-	-	-
2 ratas:	-	-	-	-
3 ratas:	-	-	-	-
4 ratas:	-	-	-	-
5 ratas:	-	-	-	-
6 ratas:	-	-	-	-
7 ratas:	-	-	-	-

# Šachmatų varžybų tvarkaraščio sudarymo algoritmas (3)

- Į pirmąjį stulpelį įrašome aštuntąjį (2k-tąjį) žaidėją.
- Jei tokio žaidėjo nėra, jo oponentas tą ratą laisvas.

1 ratas:	- (8)	-	-	-
2 ratas:	(8) -	-	-	-
3 ratas:	- (8)	-	-	-
4 ratas:	(8) -	-	-	-
5 ratas:	- (8)	-	-	-
6 ratas:	(8) -	-	-	-
7 ratas:	- (8)	-	-	-



# Šachmatų varžybų tvarkaraščio sudarymo algoritmas (4)

- Į likusias laisvas vietas prie kiekvienos lentos, žaidėjų numerius surašome tokia tvarka: 1, 2, 3, 4, 5, 6, 7, 1, 2,...

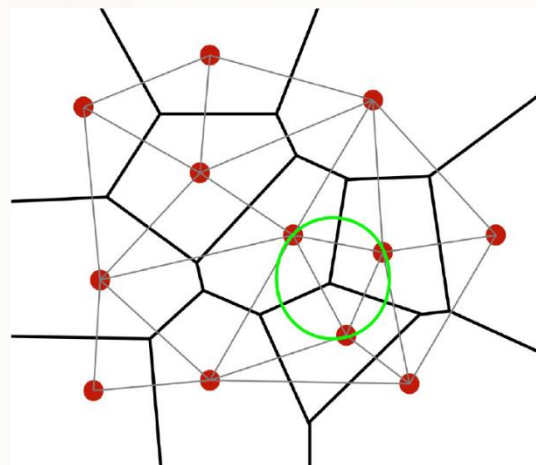
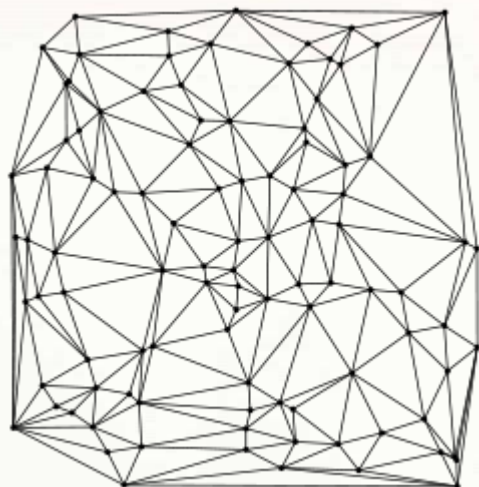
1 ratas:	1 - (8)	2 -	3 -	4 -
2 ratas:	(8) - 5	6 -	7 -	1 -
3 ratas:	2 - (8)	3 -	4 -	5 -
4 ratas:	(8) - 6	7 -	1 -	2 -
5 ratas:	3 - (8)	4 -	5 -	6 -
6 ratas:	(8) - 7	1 -	2 -	3 -
7 ratas:	4 - (8)	5 -	6 -	7 -

# Šachmatų varžybų tvarkaraščio sudarymo algoritmas (5)

- Į likusias laisvas vietas prie kiekvienos lentos, žaidėjų numerius surašome tokia tvarka: 7, 6, 5, 4, 3, 2, 1, 7, 6,...

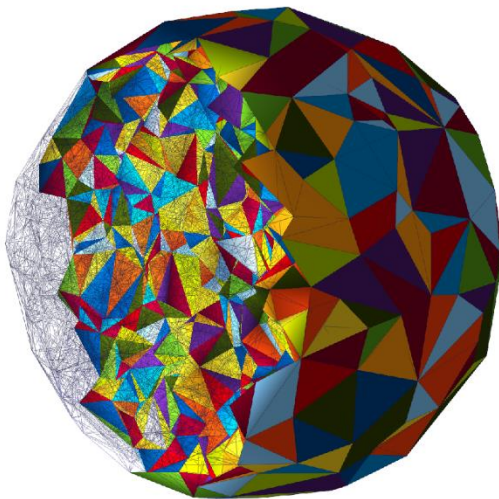
1 ratas:	1 - (8)	2 - 7	3 - 6	4 - 5
2 ratas:	(8) - 5	6 - 4	7 - 3	1 - 2
3 ratas:	2 - (8)	3 - 1	4 - 7	5 - 6
4 ratas:	(8) - 6	7 - 5	1 - 4	2 - 3
5 ratas:	3 - (8)	4 - 2	5 - 1	6 - 7
6 ratas:	(8) - 7	1 - 6	2 - 5	3 - 4
7 ratas:	4 - (8)	5 - 3	6 - 2	7 - 1

# ***Delonė trianguliacija (1)***



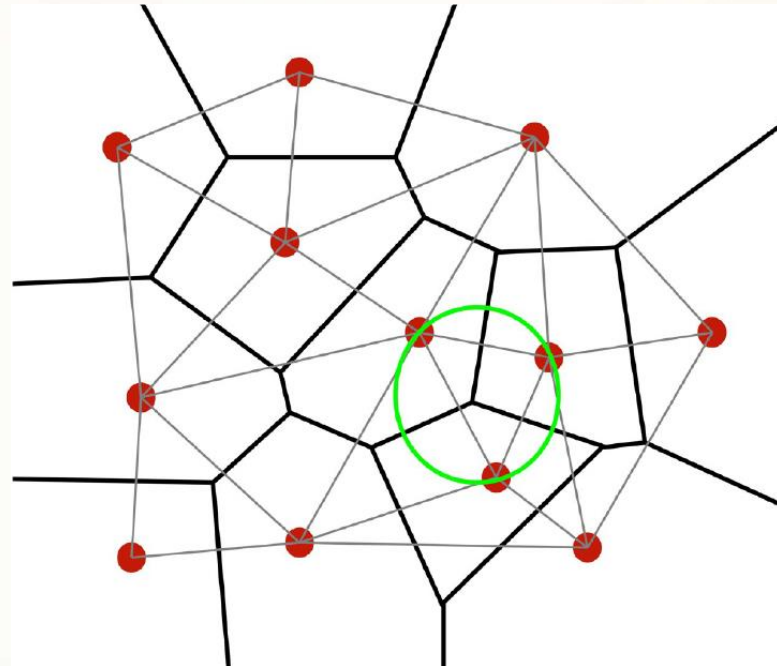
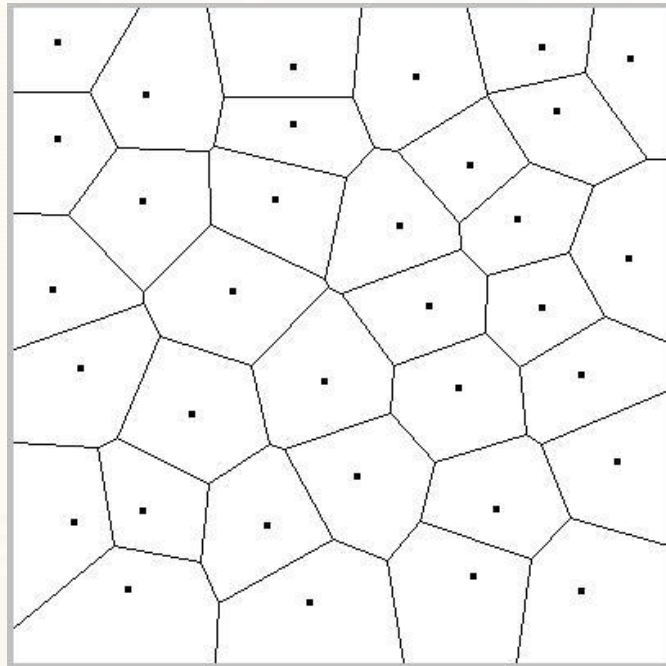
- Delonė trianguliacija vadinamas toks plokštumos taškų aibės  $P$  trejetų susiejimas į trikampius, kad apie kiekvieną tokį trikampį apibrėžto apskritimo viduje nėra nė vieno kito tokio plokštumos taško, kuris priklausytų aibei  $P$ .

## ***Delonė trianguliacija (2)***



- Trimačiu atveju Delonė trianguliacija vadinamas toks erdvės taškų aibės  $Q$  ketvertų susiejimas į tetraedrus, kad apie kiekvieną tokį tetraedrą apibrėžtos sferos viduje nėra nė vieno kito tokio erdvės taško, kuris priklausytų aibei  $Q$ .
- Kaip matome, Delonė trianguliacijos uždavinys gali būti apibendrintas  $n$ -mačiu atveju.

# Voronojaus diagrama



- Plokštumos atveju, Voronojaus diagrama padalina plokštumą į ląsteles (sritis). Gretimų 2 ląstelių sankirtos atkarpa statmena tiesei, einančiai per ląstelių centrus ir vienodai nutolusi nuo šių ląstelių centrų.
- Delonė trianguliacija yra duali Voronojaus diagramai. Kodėl?

# ***Delonė trianguliacija taikoma***

- Miestų planavime, kartografijoje.
- Mobilųjų telefonų retransliacijos stočių išdėstymo planavime.
- Efektyviai artimiausiojo kaimyno paieškai.
- Paviršių aproksimavimui (kompiuterinė geometrija, atpažinimas).
- Baigtinių elementų metodu formuojant gardeles, (diferencialinės lygtys).
- Vizualizacijoje (kompiuterinė rega, geometrija).
- Daugiamačių duomenų struktūrų sudarymui (kompiuterija).

# Delonė trianguliacijos realizavimo algoritmai (1)

- Siekiant patikrinti, ar plokštumos taškas  $D(D_x, D_y)$  yra viduje apskritimo, nubrėžto per taškus  $A(A_x, A_y)$ ,  $B(B_x, B_y)$  ir  $C(C_x, C_y)$ , išdėstytus prieš laikrodžio rodyklę, užtenka patikrinti sąlygą:

$$f(A, B, C, D) = \begin{vmatrix} A_x & A_y & A_x^2 + A_y^2 & 1 \\ B_x & B_y & B_x^2 + B_y^2 & 1 \\ C_x & C_y & C_x^2 + C_y^2 & 1 \\ D_x & D_y & D_x^2 + D_y^2 & 1 \end{vmatrix} > 0.$$

- Jei ši sąlyga nėra teisinga, trikampis ABC įtraukiamas į formuojamą Delonė trikampių tinklą.

# ***Delonė trianguliacijos realizavimo algoritmai (2)***

**Tiesioginis Delonė trianguliacijos algoritmas:**

$P = \{P[1], \dots, P[n]\}$  – plokštumos taškų aibė,

$P[i] = [P[i][1], P[i][2], i = 1, \dots, n,$

$D$  – formuojamas Delonė trikampių tinklas.

**DelaunayTr(P, n)**

$D \leftarrow \{\}$

for  $i \leftarrow 0$  to  $n - 2$  do

    for  $j \leftarrow i + 1$  to  $n - 1$  do

        for  $k \leftarrow j + 1$  to  $n$  do

            for  $q \leftarrow 0$  to  $n$  do

                if  $f(P[i], P[j], P[k], P[q]) \leq 0$

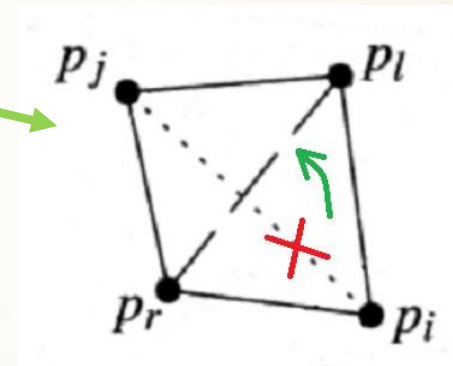
                    then  $D \leftarrow D \cup [P[i], P[j], P[k]]$

Algoritmo sudėtingumas  $O(n^4)$ , kodėl?



# Delonė trianguliacijos realizavimas efektyvesniais algoritmais

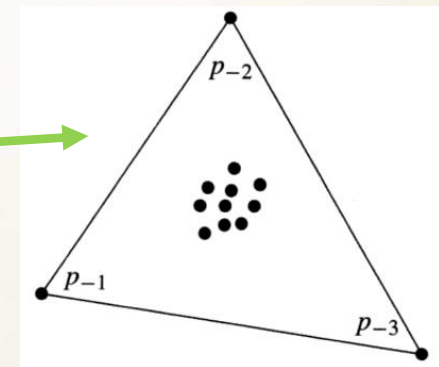
- Keitimo algoritmas (esamo trikampių tinklo korekcija).



- Atsitiktinio įterpimo algoritmas.

- „Skaldyk ir valdyk“ algoritmas.

- Gaubiančiojo trikampio algoritmas.

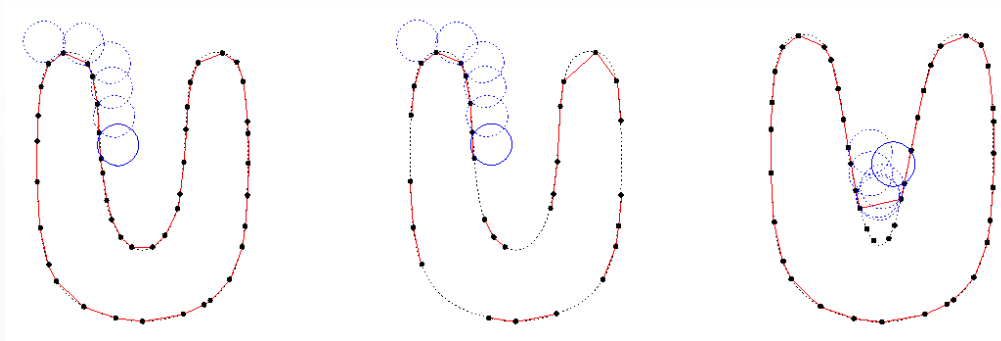


- Daugiau informacijos:

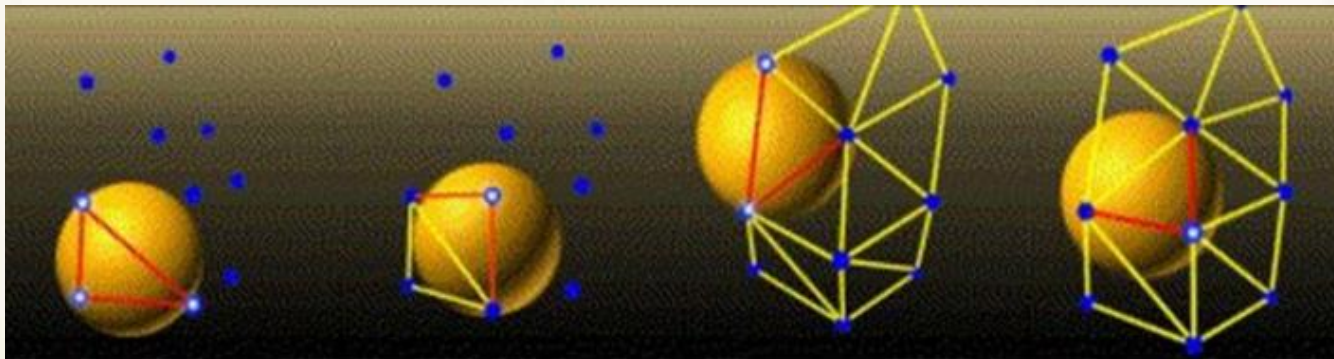
<https://kedras.mif.vu.lt/bastys/academic/ATE/delaunay/delaunay.html>

# *Paviršiaus trikampių tinklo formavimas*

- Ball pivoting (sferų sukimosi) algoritmas:



2D atveju



3D atveju

- <https://www.youtube.com/watch?v=ZqYWRVJ3bpA>

# ***Kiti geometriniai uždaviniai***

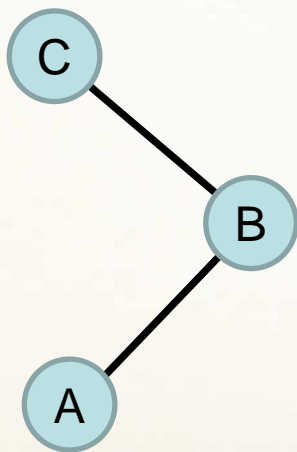
- Ar 3 plokštumos taškai išsidėsto prieš laikrodžio rodyklę?
- Ar plokštumos atkarpos kertasi? (C12)
- Ar erdvės trikampiai kertasi? (C13)
- Ar plokštumos taškas yra daugiakampio viduje?
- Iškilajo apvalkalo uždavinys.

Detalesnė informacija:

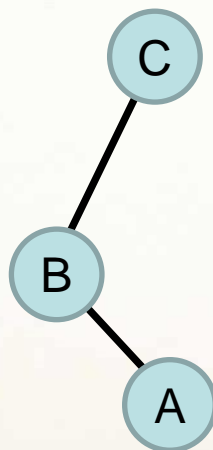
<https://www.cs.princeton.edu/~rs/AlgsDS07/16Geometric.pdf>

# *Prieš ar pagal laikrodžio rodyklę?*

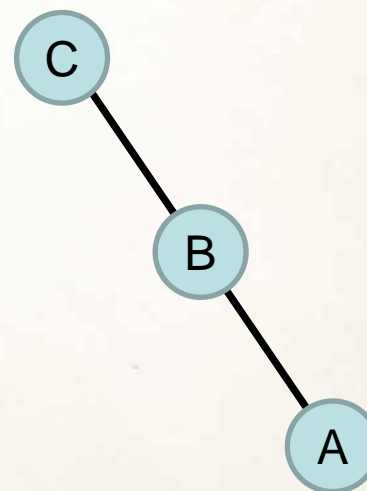
- Ar 3 plokštumos taškų seka [A, B, C] eina prieš laikrodžio rodyklę?



Taip



Ne



???

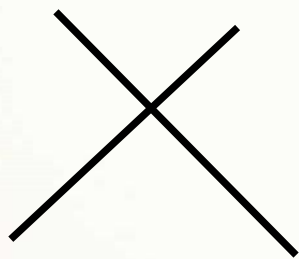
# *Prieš ar pagal laikrodžio rodyklę?*

- Tegu  $A(x_A, y_A)$ ,  $B(x_B, y_B)$ ,  $C(x_C, y_C)$  ir

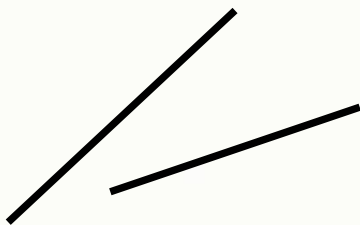
$$f(A, B, C) = \begin{vmatrix} x_A & y_A & 1 \\ x_B & y_B & 1 \\ x_C & y_C & 1 \end{vmatrix}.$$

- Jei  $f(A, B, C) > 0$ , tai taškai  $A, B, C$  išsidėsto prieš laikrodžio rodyklę.
- Jei  $f(A, B, C) < 0$ , tai taškai  $A, B, C$  išsidėsto pagal laikrodžio rodyklę.
- Jei  $f(A, B, C) = 0$ , tai taškai  $A, B, C$  yra vienoje tiesėje.

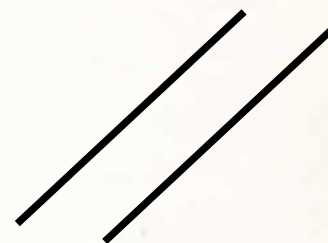
# Ar plokštumos atkarpos kertasi?



Taip



Ne

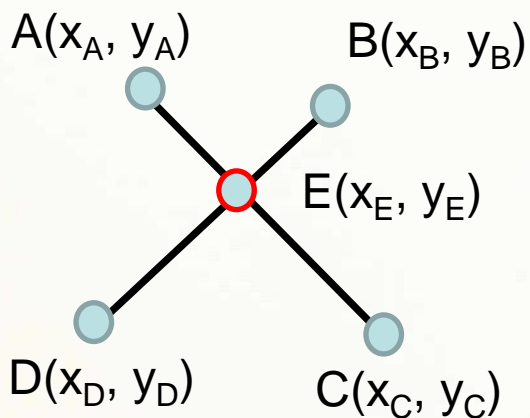


Ne

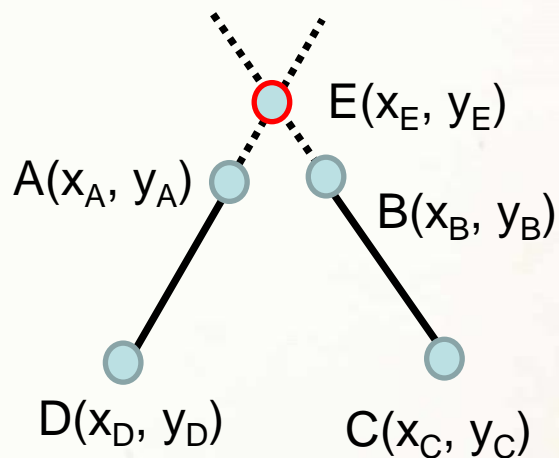
## Atkarpų susikirtimo patikrinimo idėja.

- 1) Rasti pratęstų atkarpų **susikirtimo tašką** įvertinus atskirus atvejus:
  - a) atkarpos lygiagrečios, bet ne vienoje tiesėje (niekada nesikirs).
  - b) atkarpos vienoje tiesėje (susikirtimo patikrinimas elementarus).
- 2) Jei to taško koordinatės priklauso atitinkamų atkarpų intervalų sankirtoms, atkarpos kertasi.

# Ar plokštumos atkarpos kertasi?



$$x_E \in [x_D, x_B] \cap [x_C, x_A],$$
$$y_E \in [y_D, y_B] \cap [y_C, y_A].$$

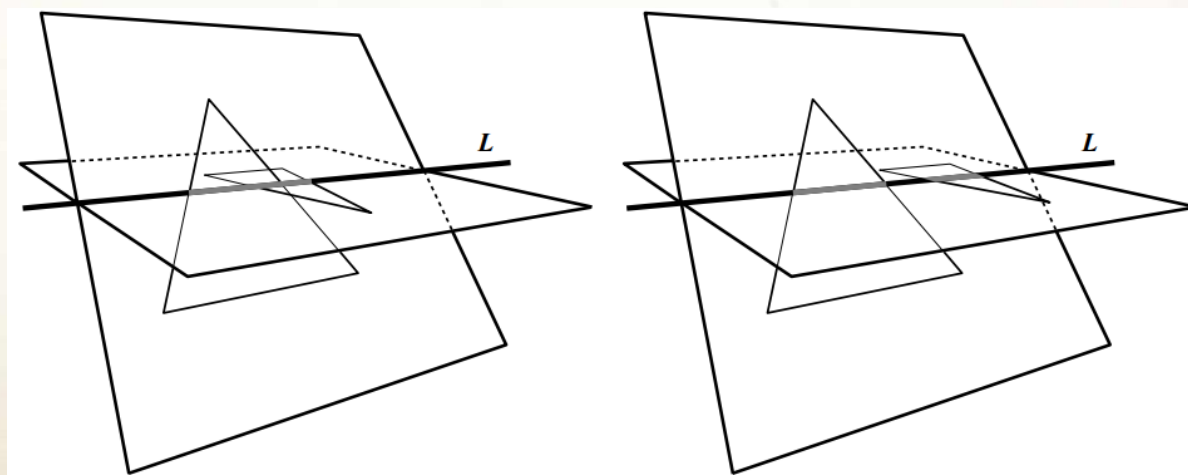


$$x_E \notin [x_D, x_A] \cap [x_C, x_B],$$
$$y_E \notin [y_D, y_A] \cap [y_C, y_B].$$

# Ar erdvės trikampiai kertasi?

**Erdvės trikampių susikirtimo patikrinimo idėja.**

- 1) Apskaičiuoti trikampių plokštumų lygtis įvertinus atskirus atvejus:
  - a) Trikampių plokštumos lygiagrečios, bet ne vienoje plokštumoje (trikampiai niekada nesikirs).
  - b) Trikampių plokštumos vienoje plokštumoje (susikirtimo patikrinimas susiveda į atkarpų susikirtimo patikrinimo uždavinį).
- 2) Išvesti trikampių plokštumų susikirtimo tiesės  $L$  lygtį ir ją suprojektuoti tokioje Dekarto ašyje, kurios atžvilgiu mažiausias kampas su tiese  $L$ .

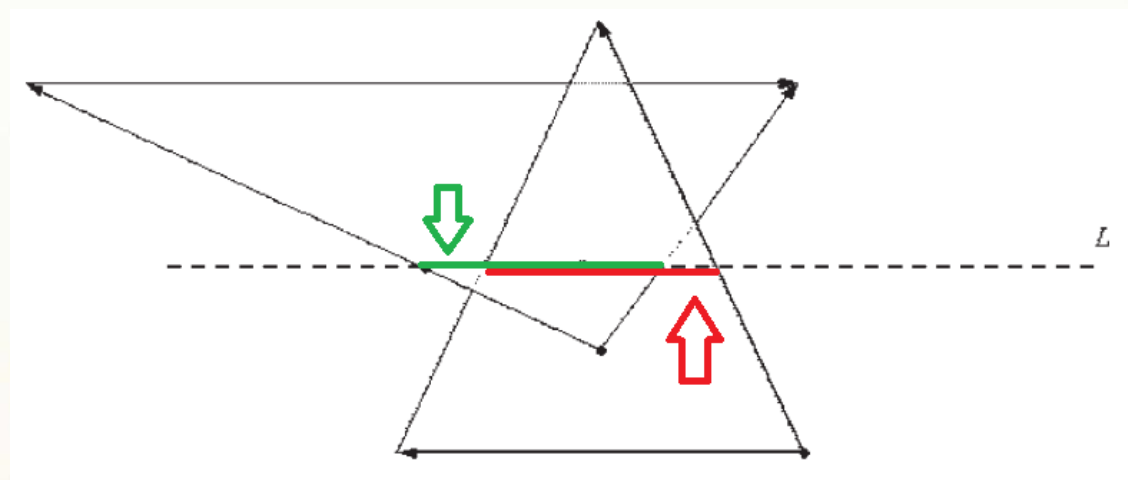




# Ar erdvės trikampiai kertasi?

Erdvės trikampių susikirtimo patikrinimo idėja.

3) Suprojektuoti trikampiams priklausančias atkarpas tiesėje  $L$ :



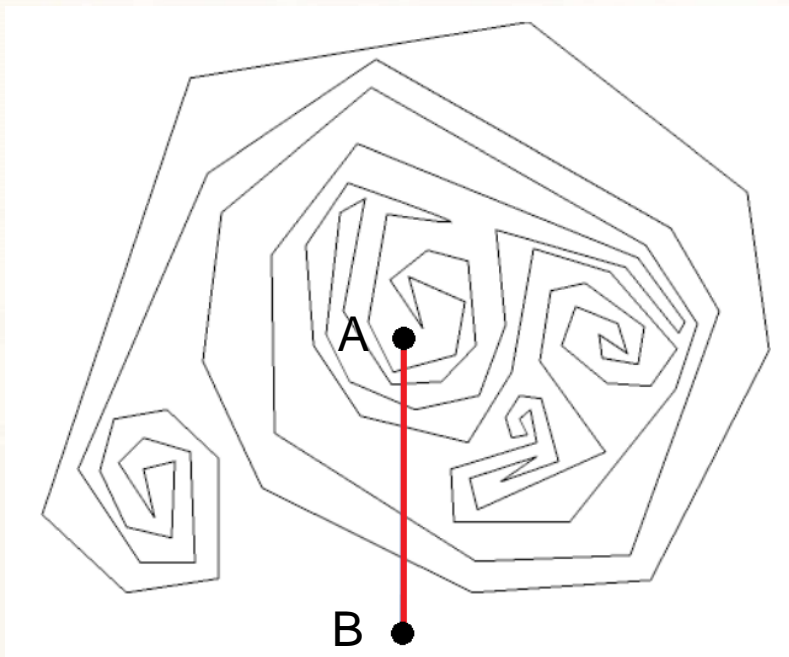
4) Įvertinti, ar šių atkarpų sankirta nėra tuščioji aibė.

Daugiau informacijos:

<https://web.stanford.edu/class/cs277/resources/papers/Moller1997b.pdf>

<http://webee.technion.ac.il/~ayellet/Ps/TroppTalShimshoni.pdf>

# ***Daugiakampio viduje ar išorėje?***



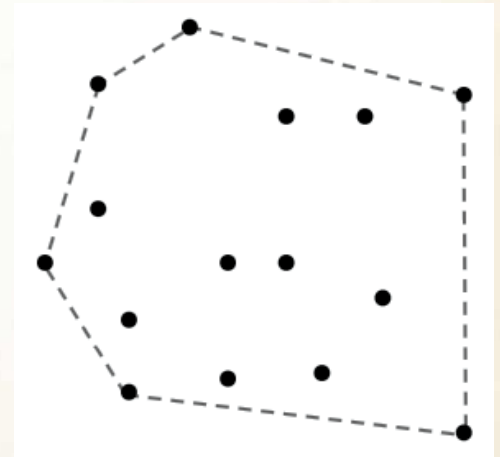
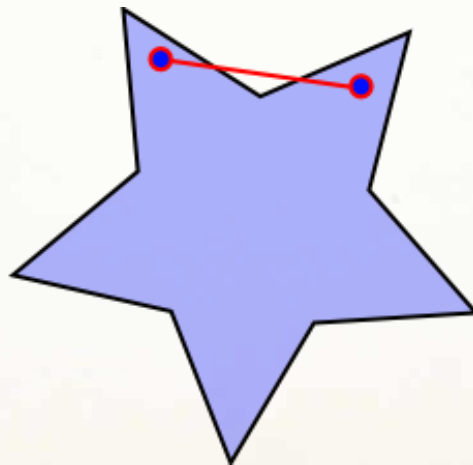
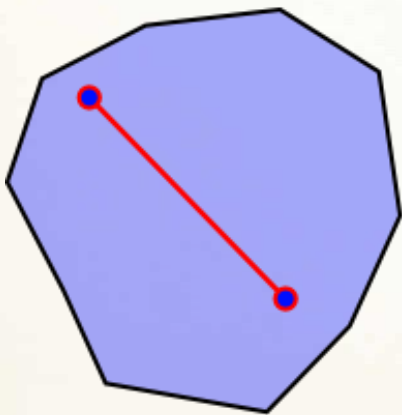
- Uždaras ir savęs nekertantis daugiakampis plokštumą dalija į 2 dalis: vidų ir išorę.
- Kertant daugiakampio kraštinę pereinama arba iš daugiakampio vidaus į išorę arba iš išorės į daugiakampio vidų.
- Kaip tuo pasinautodi?

**Ar duotas taškas A daugiakampio viduje? Uždavinio sprendimo idėja:**

- 1) Apskaičiuoti daugiakampio išorės tašką B.
- 2) Taškus A ir B sujungti atkarpa.
- 3) Apskaičiuoti, kiek kartų atkarpa AB kerta daugiakampį.
- 4) Jei tų kirtimų skaičius – nelyginis, taškas B yra daugiakampio viduje.

# Iškilojo apvalkalo uždavinys

Jei bet kuriuos 2 daugiakampio vidaus taškus jungianti atkarpa priklauso daugiakampiui, daugiakampis yra iškilasis, priešingu atveju – neiškilasis.

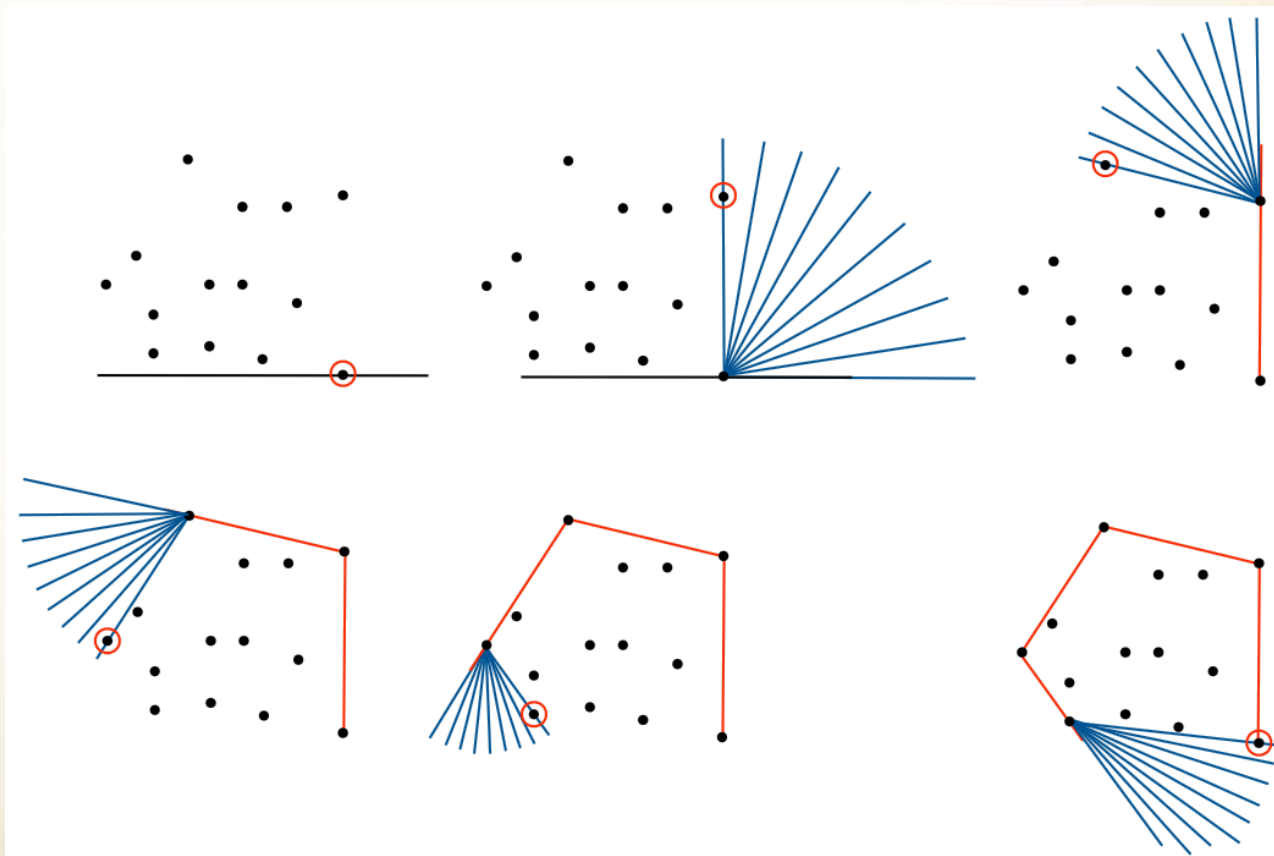


Mažiausio ploto iškilasis daugiakampis, kuriam priklauso visi duotosios aibės taškai, vadinamas iškiliuoju apvalkalu.

# Iškilojo apvalkalo sudarymas

Algoritmo idėja (tegu įvesties taškų aibė  $Q$ ):

- 1) Iš aibės  $Q$  pasirinkti pradžios tašką, kurio mažiausia  $y$  koordinatė.
- 2) Iš to taško  $X$  ašies kryptimi išvesti spindulį  $r$  ir jį sukti prieš laikrodžio rodyklę.
- 3) Spinduliui  $r$  kaskart „atsitrenkus“ į kitą aibės  $Q$  tašką, į jį perkelti sukamą spindulį  $r$ .
- 4) Algoritmas stabdomas, kai spindulys  $r$  „atsitrenkia“ į tašką su mažiausia  $y$  koordinate.
- 5) Spindulio  $r$  paliestų taškų sąrašas atitinka iškiląjį apvalkalą.



# ***Daugiau geometrinų algoritmų***

**Stenfordo universitete dėstomo dalyko „Geometry Processing Algorithms“ paskaitų skaidrės:**

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/01\\_Introduction.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/01_Introduction.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/02\\_Basics.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/02_Basics.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/02\\_Mesh\\_Data\\_Structures.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/02_Mesh_Data_Structures.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/02\\_Open\\_Mesh.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/02_Open_Mesh.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/03\\_Surface\\_Reconstruction.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/03_Surface_Reconstruction.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/04\\_Surface\\_Reconstruction.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/04_Surface_Reconstruction.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/05\\_Diff\\_Geo.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/05_Diff_Geo.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/06\\_smoothing.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/06_smoothing.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/07\\_Linear%20solvers.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/07_Linear%20solvers.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/08\\_Simplification.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/08_Simplification.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/09\\_Progressive\\_Meshes.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/09_Progressive_Meshes.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/10\\_Subdivision.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/10_Subdivision.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/11\\_shape\\_matching.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/11_shape_matching.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/12\\_Parameterization1.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/12_Parameterization1.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/13\\_Parameterization2.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/13_Parameterization2.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/14\\_Remeshing1.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/14_Remeshing1.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/15\\_Remeshing2.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/15_Remeshing2.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/16\\_spectral\\_methods1.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/16_spectral_methods1.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/17\\_spectral\\_methods2.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/17_spectral_methods2.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/18\\_Deformation\\_1.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/18_Deformation_1.pdf)

[http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/19\\_Deformation\\_2.pdf](http://graphics.stanford.edu/courses/cs468-10-fall/LectureSlides/19_Deformation_2.pdf)

# ***Egzamino temos (1)***

1. Algoritmo užrašymo būdai:
  - a. Blokinė schema;
  - b. Pseudokodas.
2. Algoritmo realizavimo metodai:
  - a. Indukcija;
  - b. Rekursija.
3. Klasikiniai algoritmai:
  - a. Euklido algoritmas;
  - b. Pirminių (ir pirminių dvynių) skaičių paieškos algoritmai;
  - c. Skaičiavimo sistemos keitimo algoritmas.
4. Paieškos algoritmai:
  - a. Nuoseklioji (tiesinė) paieška;
  - b. Dvejetainė paieška;
  - c. Paieška medžiuose.
5. Algoritmo sudėtingumas.

## ***Egzamino temos (2)***

### **6. Tiesinės ir dinaminės duomenų struktūros:**

- a. Tiesinis sąrašas;
- b. Tiesinis abipusis sąrašas;
- c. Dėklas;
- d. Abipusis dėklas;
- e. Eilutė;
- f. Ciklinė eilutė.

### **7. Abstraktus duomenų tipas:**

- a. Matrica;
- b. Aibė;
- c. Medžiai;
- d. Heap struktūra;
- e. Prioritetinė eilutė.

# ***Egzamino temos (3)***

## **8. Rikiavimo algoritmai:**

- a. Išrinkimo algoritmas;
- b. Burbuliuko algoritmas;
- c. Įterpimo algoritmas;
- d. Rikiavimas Šelo metodu;
- e. Spartaus rikiavimo algoritmas;
- f. Sąlajos rikiavimas;
- g. Išorinis rikiavimas;
- h. Piramidės rikiavimas;
- i. Skaitmeninis rikiavimas.

## **9. Grafai, digrafai, multigrafai ir medžiai:**

- a. Gretimumo matrica;
- b. Incidentumo matrica;
- c. Priuferio kodas;
- d. Medžio centras;
- e. Hierarchinis medžių vaizdavimas;
- f. Radialinis medžių vaizdavimas.



# ***Egzamino temos (4)***

10. Medžių tipai ir medžių paieškos algoritmai:
- a. Dvejetainis medis;
  - b. Dvejetainis paieškos medis;
  - c. Medžio apėjimo būdai;
  - d. AVL medžiai;
  - e. Fibonačio medžiai;
  - f. Raudoni–juodi medžiai;
  - g. 2–3 medžiai;
  - h. 2–3–4 medžiai;
  - i. B–medžiai.

# ***Egzamino temos (5)***

## 11. Grafų paieškos algoritmai:

- a. Paieška į plotį;
- b. Paieška į gylį;
- c. Kruskalio algoritmas;
- d. Primo algoritmas;
- e. Dijkstros algoritmas;
- f. Belmano–Fordo algoritmas;
- g. Lėtas trumpiausių takų paieškos algoritmas;
- h. Greitas trumpiausių takų paieškos algoritmas;
- i. Floido–Varšalo algoritmas;
- j. Fordo ir Fulkersono metodas;
- k. Edmondso–Karmo algoritmas;
- l. Priešsraučio stūmimo algoritmas.

***Ačiū už dėmesį.***