VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Baigiamasis bakalauro darbas

# Grafų vizualizacijos įrankių analizė

(Analysis of graph visualization tools)

Atliko: 4 kurso, 2 grupės studentas
Julian Dzisevič (parašas)

Darbo vadovas:
lekt. Irmantas Radavičius (parašas)

Recenzentas:
Lekt., Dr., Gintaras Skersys (parašas)

Vilnius,
2017

# Turinys

Įvadas ....3
1. Įrankių vertinimo kriterijai ....5
2. Grafų piešiniai ....5
2.1. Paprastas grafo piešinio modelis ....5
2.2. Piešinio atpažinimas ....6
2.2.1. Tiesė iš trijų pikselių ....7
2.2.2. Tris pikseliai sudaro statų kampą ....7
2.2.3. Atstumas tarp tiriamosios viršūnės ir tiesės, jungiančios jos kaimynes ....7
2.2.4. Viršūnių aibių šalinimas sukuriant centrinę viršūnę ....9
2.2.5. Šalinimas, remiantis sudarytomis aibėmis ....10
3. Grafų vizualizacijos įrankių sugeneruotų piešinių atpažinimo programa. ....11
3.1. Atpažinimas iš nuotraukos ....11
3.2. Apie įrankį ....12
3.2.1. Reader() ....12
3.2.2. createVerticesList() ....13
3.2.3. findNeighbours() ....13
3.2.4. eliminateInlineVertices() ....13
3.2.5. eliminateOnAngleVertices() ....14
3.2.6. eliminateByDistances() ....14
3.2.7. deleteStickedVerticesAndLeaveOneInCenter() ....14
3.2.8. removeLeftVerticesInTheMiddle() ....14
3.2.9. createOutput() ....15
3.3. Viršūnės trynimas ....15
3.3.1. Šalinimo funkcija, kai trinama (2.2.2) minėta viršūnė. ....16
3.3.2. Viršūnių aibės (2.2.4) šalinimas ir centrinės viršūnės kūrimas ....16
4. Grafų vizualizacijos įrankiai ....17
4.1. Cytoscape ....17
4.2. Gephi ....17
4.3. Tulip ....17
5. Gebėjimas atpažinti paprasto grafo piešinį ....18
5.1. Pirmojo duomenų rinkinio atpažinimas. ....18
5.2. Antrojo duomenų rinkinio atpažinimas. ....22
6. Išvados ....24
7. Summary ....25
8. Literatūros šaltiniai ....26
9. Priedai ....27

# Įvadas

Grafų teorijos atsiradimas XVIII amžiuje yra susijęs su matematiniais galvosūkiais, todėl gana ilgai į grafų mokslą buvo žiūrima kaip į „nerimtą“ temą, kurios „taikomoji“ vertė susijusi tik su lošimais ir pramogomis. XX amžiaus pradžioje grafai patraukė topologų dėmesį, todėl pirmoje praeito amžiaus pusėje grafų teorija buvo laikoma sudėtingos šiuolaikinės matematikos šakos, kuria domisi tik siauras specialistų ratas – topologijos dalimi. Vėliau paaiškėjo, kad grafų teorija labai naudinga, sprendžiant daugelį svarbių praktikos klausimų, iš kurių čia verta paminėti vadinamuosius „transporto uždavinius“ (racionaliausios krovinių pervežimo sistemos transporto tinkle planavimo uždaviniai), uždavinius susijusius su elektros tinklais, bei grafų teorijos taikymą tokiose srityse kaip ekonomika, psichologija ir biologija [Bis05].

Technologijoms besivystant atsiranda vis didesnės galimybės bei poreikiai kaupti duomenis bei juos analizuoti. Daryti išvadas iš sukauptų duomenų būtų labai sunku neturint tam skirtų priemonių, pavyzdžiui, programų, kurias pasitelkus duomenys yra pateikiami vizualiu, žmogui suprantamu formatu. Būdų vaizduoti duomenis yra nemažai, tačiau (vieni geba suprantamiau ar informatyviau perteikti specializuotus duomenis nei kiti) kiekvienas jų turi savo stipriąsias puses vaizduojant tam tikrus duomenis. Vienas minėtųjų būdų vizualizuoti duomenis yra vizualizavimas grafais. Yra keletas priežasčių, kodėl grafai yra galingas vizualizacijos įrankis:

- Grafai yra paprastas modelis, kuris gali būti nesunkiai pritaikytas daugelyje sričių. Bet kokie duomenys, kurie turi sąryšį tarpusavyje, gali būti modeliuojami grafais. Žiniatinklis (angl. World Wide Web) yra puikus pavyzdys, kuriame internetiniai puslapiai gali būti atvaizduojami viršūnėmis, o sąryšiai tarp viršūnių gali būti atvaizduojami saitais (angl. hyperlink). Yra daug kitų pavyzdžių, tokių kaip ryšiai tarp žmonių socialiniuose tinkluose, gyvūnų rūšių medžiai, kompiuterio failinė sistema ir t.t.
- Grafas yra abstrakti sąvoka, turinti savitą taikymo sritį. Po kelių šimtmečių vystymosi, grafų teorijos sritis šiuo metu turi tvirtą pagrindą bei visą rinkinį galingų priemonių, nepriklausomų algoritmų sukurtų tam, kad efektyviai kurti bei apdoroti grafus.
- Vaizdinio suvokimo sistema yra stipriausia iš žmonių turimų jutiminių sistemų. Žmonių smegenyse, 70% receptorių ir 40% žievelės (angl. cortex) yra naudojama regėjimo procesui [Spe01]. Be to, yra įrodyta, jog žmonėms lengviau sekasi „atpažinimo“, o ne „įsiminimo“ užduotys [DF98]. Pastarieji teiginiai įrodo, jog tekstinės išraiškos yra mažiau efektyvios lyginant su vaizdinėmis (grafu, piešiniu, grafikais) išraiškomis, kai yra siekiama analizuoti didelius duomenų srautus. Naudojant grafus, kaip vizualizacijos priemonę dideliems duomenims vizualizuoti, galima supaprastinti žmonių darbą žinant, jog vaizdinė informacija yra labiau intuityvi ir artima žmonėms. [Dzi17]

Programų, generuojančių grafų piešinius iš joms pateiktų duomenų yra gana nemažai, tačiau nėra aišku ar visos programos generuoja vienodai gerus grafo piešinius. Kadangi, neretai stengiamasi automatizuoti kuo daugiau procesų, generuoti kuo paprastesnius bei lengviau atpažįstamus piešinius yra aktualu. Neretai ne vien žmonės dirba su grafų piešiniais, tačiau nėra aišku, kaip programos atpažįsta kitų įrankių sugeneruotus grafus.

Tam, kad išbandyti grafų programas, pradžioje buvo ieškoma populiariausių būtent grafų vizualizacijos programų, tam, kad galima būtų jų generuojamus piešinius atpažinti bei palyginti tarpusavyje.

Taigi, šio **darbo tikslas** palyginti grafų vizualizacijos įrankius, vadovaujantis sukurtos grafų piešinių atpažinimo įrankio rezultatais, kaip įvestį teikiant vizualizacijos programos sugeneruotą piešinį.

Tam, kad pasiekti minėtąjį išsikeltą tikslą, reikia įgyvendinti šiuos uždavinius:

- Išanalizuoti grafo atpažinimo iš nuotraukos procesą.
- Gero piešinio kriterijų suformulavimas.
- Automatinio grafo piešinio paieška/kūrimas.
- Įrankio vertinimo metodikos pasiūlymas.

# 1. Įrankių vertinimo kriterijai

Šiais laikais, plati interneto prieiga pasauliniu mastu leidžia sujungti didelį žmonių skaičių. Remiantis socialinio tinklo Facebook statistika[1], 2012 metais buvo apie 500 milijonų aktyvių vartotojų, o iki 2016 rugsėjo 30d. svetainė jau turėjo apie 1,79 milijardo aktyvių vartotojų, kurie prisijungia bent kartą per mėnesį. Kaip rašoma Gartnerio tyrime [DS13], toks spartus augimas didina socialinių tinklų, kurie gali būti vaizduojami grafais, analizės metodų poreikį, skirtų padėti suprasti tuos sąryšius, juose vyraujančias tendencijas, aptikti bendruomenes bei ištirti jų evoliuciją, pritaikant gautus rezultatus tokiose srityse, kaip rinkodara, prekyba ar kt. Dėl šitos priežasties, pradėjo daugėti programų, skirtų socialinių tinklų analizei (*angl*. SNA). Šie įrankiai labai palengvina analizę duomenis naudotojui pateikdamos grafiškai. Jos apskaičiuoja skirtingus rodiklius, kurie padeda lengviau suprasti tinklo struktūrą, sąryšius tarp veikėjų (viršūnių), o taip pat, konkrečių veikėjų tikslų išdėstymą (*angl*. layout). Minėtieji įrankiai taip pat įgalina ir palyginimą tarp kelių grafų [Dzi17].

Šio darbo rėmuose, programinės įrangos palyginimas bus vykdomas vertinant kiekvienos iš viename iš ateinančių skyrių pristatytų naudotų programų sugeneruotus grafų piešinius. Pagrindinis vertinimas bus atliekamas atsižvelgiant į tai, kaip juos pavyko atpažinti su suprogramuotu paprastų grafų atpažinimo iš nuotraukos įrankiu.

Norint įvertinti gebėjimą atpažinti grafo piešinį, vizualizacijos įrankio sugeneruotas paprastas piešinys yra perduodamas grafų atpažinimo įrankiui, pagal kurio išvestį galima spręsti ar vizualizacijos įrankis sugebėjo sugeneruoti paprastą piešinį ar ne. Tuo pačiu yra tikrinamas ir atpažinimo įrankio veikimas.

# 2. Grafų piešiniai

## 2.1. Paprastas grafo piešinio modelis

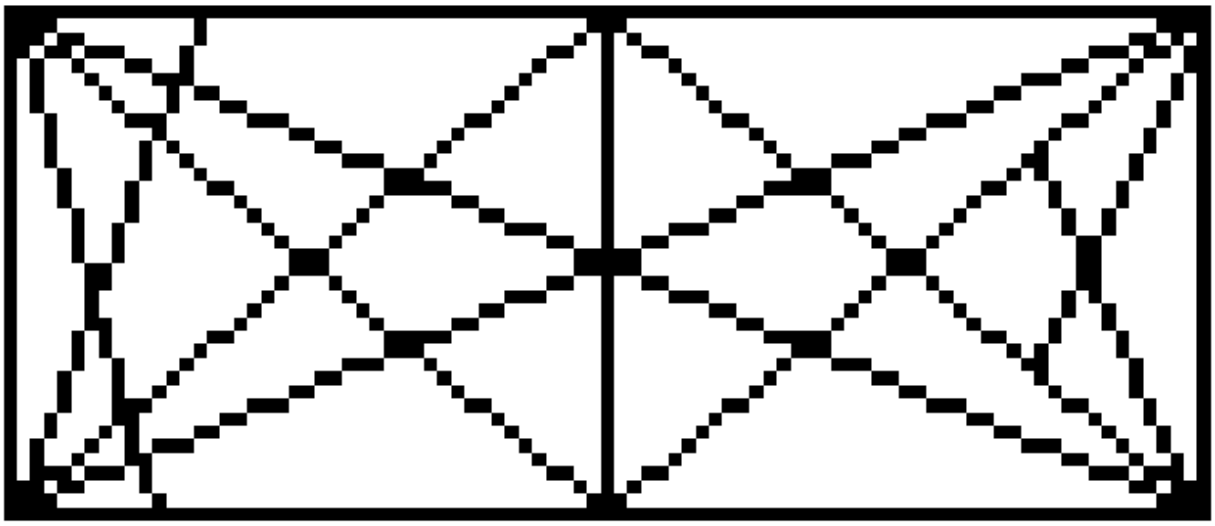
Grafai yra tradicinė bei galinga priemonė, kurios pagalba žmogus geba pavaizduoti duomenų rinkinius bei ryšius tarp jų. Jie turi stebėtinai ilgą istoriją, kur ilgus metus padėjo žmonėms bendrauti, suprasti juos supantį pasaulį bei spręsti specifines mokslo problemas. Grafų ištakos siekia viduramžius [Par09]. Pagrinde, grafų piešimas padeda išspręsti tokias specifines vizualizavimo problemas, kaip grafų, tinklų ar kitų panašių struktūrų, kurių ypatybė yra sąryšiai tarp esybių. Tai prideda geometrinės bei grafinės informacijos, viršūnėms priskiriant konkrečias koordinates ir pagal jas jungiant briaunas. Neretai piešiniams yra pridedama papildomų grafinių elementų, kaip pavyzdžiui, skirtingos viršūnių ir briaunų formos, spalvos, papildomos lentelės su duomenimis, tam, kad piešinys būtų lengviau išskaitomas jį analizuojančiam žmogui [ABB+13], tačiau, jeigu piešinį norima atpažinti su atpažinimo programa, papildomi grafiniai elementai yra pertekliniai ir gali sukelti sunkumų atpažįstant piešinį, kadangi atpažįstant svarbiausia yra paprastumas.

Norint sugebėti atpažinti grafą iš nuotraukos, piešinys turi būti pakankamai paprastas ir tvarkingas, kad atpažinimo programa gebėtų atpažinti, tai, kas yra pavaizduota. Šiame darbe laikysime, jog paprastas grafo piešinys yra toks (Pav. 1), kuriame:

- Kiekviena viršūnė vieno pikselio atstumu neturi nei vienos kaimynės.
- Nėra briaunų susikirtimų, arba briaunų susikirtimuose yra viršūnė.
- Briaunos linijos storis yra lygiai vieno pikselio.
- Briauna, einanti ne vertikaliai ar horizontaliai, turi „lūžti“ taip, kaip pavaizduota (Pav. 1).

[1] http://newsroom.fb.com/company-info/

- Yra laikoma, kad viršūnė yra briaunos galuose arba ten, kur susikerta briaunos.
- Viršūnės nėra išryškintos (spalva, forma, dydžiu).
- Briaunos neturi posūkių ir yra nupieštos tik tiesiomis linijomis tarp viršūnių.



*Pav. 1 Paprastas grafo piešinys. Briaunų susidūrimų vietose yra viršūnės.*

## 2.2. Piešinio atpažinimas

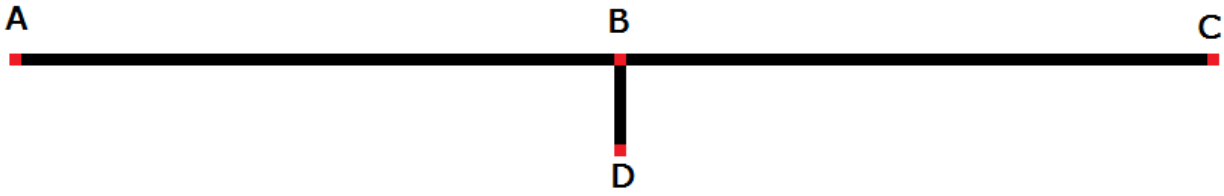
Žinant, kas yra paprastas piešinys ir norint jį atpažinti, galima įsivaizduoti, kaip gali būti vykdomas piešinio atpažinimas. Grafo piešinio atpažinimo tikslas yra atskirti, kur piešinyje yra pavaizduotos viršūnės, o kur briaunos bei išgauti visą įmanomą ir svarbią papildomą informaciją, kuri gali būti esminė tolimesniam grafo piešinio tyrimui, pavyzdžiui, kiekvienos viršūnės koordinates *x* ir *y* ašyse, taip pat, kaimyninių viršūnių sąrašą. Pastarasis sąrašas yra reikalingas tam, kad galima būtų atgaminti, kurios viršūnės yra tarpusavyje sujungtos briaunomis.

Kadangi, šiame darbe yra laikoma, kad briaunos maksimalus toris yra 1 pikselis, šio darbo rėmuose yra analizuojama ir dirbama su atskirais analizuojamo piešinio pikseliais. Yra laikoma, kad visi grafo nuotraukos pikseliai yra vienos iš dviejų spalvų – baltos (RGB[255,255,255]) arba juodos (RGB[0,0,0]). Viršūnių atpažinimas yra vykdomas atmetimo būdu, kitaip tariant, pradžioje visi pikseliai, kurie yra juodi yra laikomi viršūnėmis, o balta spalva yra laikoma fonu ir ji tiesiog ignoruojama.

Yra laikoma, kad viršūnė (B) negali būti ant briaunos, jungiančios kitas dvi viršūnes (A, C) vidurio (Pav. 2), nebent pirmosios viršūnės laipsnis[2] yra didesnis už dvejetą (*deg(B) > 2*) (Pav. 3).


*Pav. 2 Negalimas B viršūnės padėjimo atvejis*

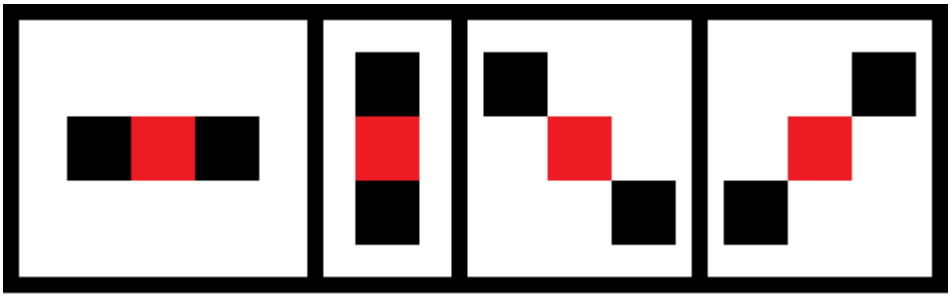
[2] Viršūnės laipsnis – su keliomis kitomis viršūnėmis minėtoji viršūnė turi jas jungiančių briaunų. Laipsnis žymimas *deg*(V).




*Pav. 3 Galimas B viršūnės padėjimo atvejis*

### 2.2.1. Tiesė iš trijų pikselių

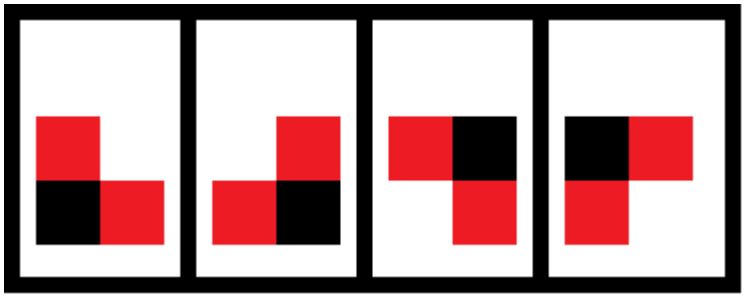
Piešinio analizė yra vykdoma pradedant tikrinimu ar vidurinė viršūnė turi dvi kaimynes, su kuriomis sudarytų tiesią liniją. Jeigu minėtoji sąlyga yra tenkinama, kaip pavaizduota (Pav. 4), tai raudonai pažymėta viršūnė yra trinama, o juodos viršūnės yra pažymimos, kaip viena kitos kaimynės.



*Pav. 4 Atvejai, kai raudona viršūnė yra trinama, o juodos padaromos kaimynėmis*

### 2.2.2. Tris pikseliai sudaro statų kampą

Pašalinus visas viršūnes, kurios atitinka (Pav. 4) pavaizduotus atvejus, yra ieškoma viršūnių, kurios su savo dvejomis kaimynėmis sudaro statųjį kampą (Pav. 5). Radus tokią viršūnę, pašalinamos tos viršūnės kaimynės. Taip yra daroma dėl to, kad yra akivaizdu, jog stačiuose kampuose tikrai bus vienintelė viršūnė, kadangi, kaip buvo minėta, pagal formuluojamą sąlygą, negali būti dvi ar daugiau viršūnių tarpusavyje vieno pikselio atstumu.

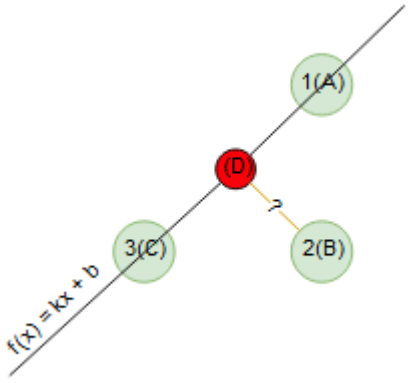


*Pav. 5 Atvejai, kai juodos viršūnės kaimynės yra trinamos.*

### 2.2.3. Atstumas tarp tiriamosios viršūnės ir tiesės, jungiančios jos kaimynes

Tolimesniam tikrųjų piešinio viršūnių nuo briaunų taškų atskyrimui yra naudojama truputį geometrijos. Yra paimama viršūnė, turinti dvi kaimynes, šį kartą, nebesvarbu, kokiu atstumu tarpusavyje. Yra sprendžiama ar pašalinti antrąją (2(B)) viršūnę (Pav. 6). Tikrinimas ar galima pašalinti antrąją viršūnę remiasi atstumo ieškojimu tarp tiesės, jungiančios pirmąją ir trečiąją

viršūnes, ir antrosios viršūnės tame taške, kai abi tiesės yra statmenos viena kitai. Piešinyje pavaizduotas raudonas apskritimas žymi tašką, kuriame pirmąją ir trečiąją viršūnes jungianti briauna bei ieškoma briauna, išeinanti iš antrosios viršūnės yra statmenos.




*Pav. 6 Atstumo (BD) paieška tarp tiriamosios (B) ir statmenos jai tiesės, jungiančios viršūnės (B) kaimynines (A) ir (C).*

Tam, kad surasti klaustuku pažymėtą atstumą, mums reikia sužinoti D taško koordinates. Surasti tašką D galima tiesės lygtį, einančios tarp pirmos ir antros viršūnių. Tiesės lygtis:

$$f(x) = kx + b$$

Tam, kad surastume krypties koeficientą k bei konstantą b, reikia išspręsti lygčių sistemą:

$$\begin{cases} y_1 = k_{13}x_1 + b_{13} \\ y_3 = k_{13}x_3 + b_{13} \end{cases}$$

Iš čia gauname, jog

$$k_{13} = \frac{y_3 - y_1}{x_3 - x_1};\ b_{13} = \frac{x_3y_1 - x_1y_3}{x_3 - x_1}$$

Tiesės yra statmenos tada, kai ($k_{13}$ yra tiesės lygties koeficientas, einančios per pirmą bei trečią viršūnes, o $k_2$ yra tiesės lygties koeficientas, einančios per antrą viršūnę ir sudarančios statųjį kampą su prieš tai minėtąja viršūne)

$$k_{13}k_2 = -1$$

Įsistatę vertes ir išsprendę lygtį gauname, kad ieškomos tiesės krypties koeficientas k bei konstanta b yra lygi:

$$k_2 = \frac{x_1 - x_3}{y_3 - y_1}; b_2 = \frac{(x_3 - x_1)x_2}{y_3 - y_1} + y_2$$

Tam, kad gauti dviejų tiriamų tiesių susikirtimo taško koordinates, skaičiuojame lygybę:

$$k_{13}x' + b_{13} = k_2x' + b_2$$

Taigi, $x'$ir $y'$ yra lygūs:

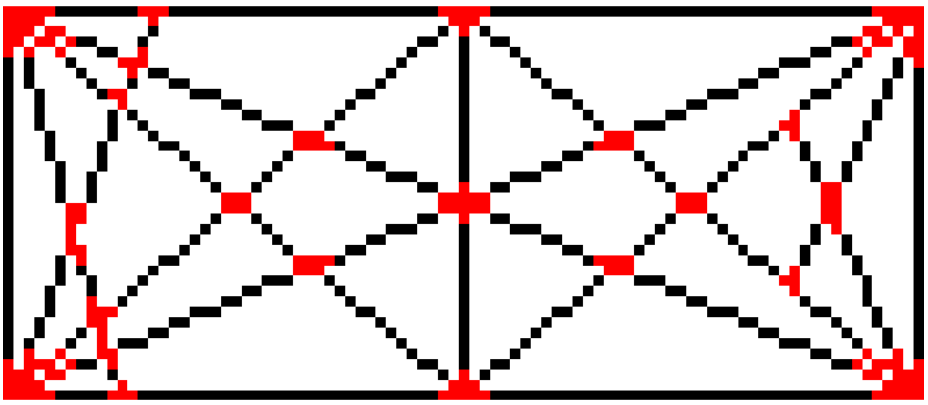
$$x' = \frac{b_2 - b_{13}}{k_{13} - k_2}; y' = k_2x' + b_2$$

Galiausiai, yra skaičiuojamas atstumas tarp D ir B taškų:

$$DB = \sqrt{(x' - x_2)^2 + (y' - y_2)^2}$$

Jeigu *DB* atstumas viršija vienetą (vieną pikselį), tada antroji (*B*) viršūnė yra trinama iš viršūnių sąrašo. Tokiu atveju, jeigu antroji viršūnė buvo ištrinta, išvardinti veiksmai yra toliau atliekami rekursiškai, kur tikrinamąja viršūne B tampa prieš tai buvusi pirmoji A viršūnė.

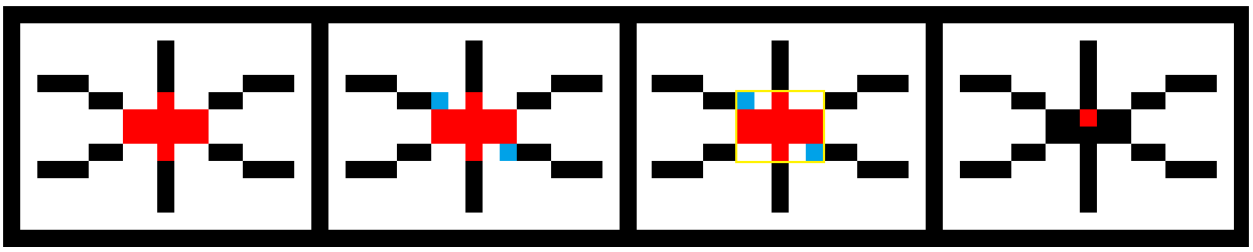
Po atliktų veiksmų, (Pav. 1) piešinys turėtų atrodyti taip:



*Pav. 7 Grafas iš (Pav. 1) po trijų viršūnių šalinimo operacijų.*

### 2.2.4. Viršūnių aibių šalinimas sukuriant centrinę viršūnę

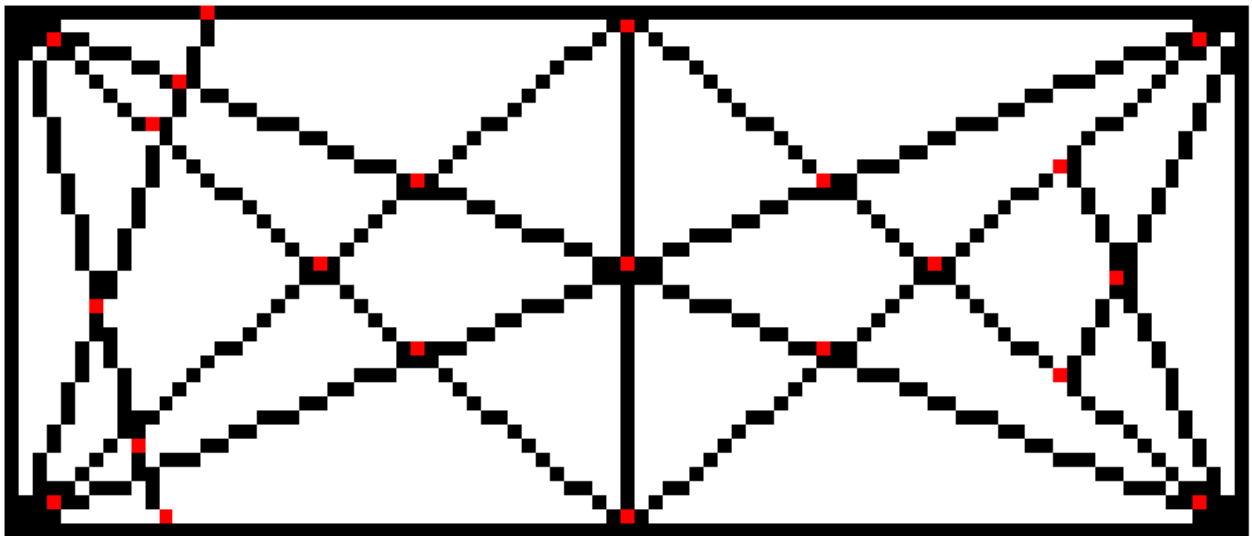
Norint pašalinti likusias viršūnes, kurios yra susigrūdusios viena prie kitos, reikia surasti, atskirus plotus, kuriuose viršūnės yra išsidėsčiusios viena prie kitos. Galima imti (Pav. 7) centre pavaizduotą viršūnių aibę (Pav. 8 (a)). Tam, kad išskirti vieną centrinę viršūnę, nagrinėjamoje viršūnių aibėje yra ieškoma mažiausias bei didžiausias koordinačių *x* ir *y* vertes turinčių viršūnių. Radus vertes, jos yra išsaugomos (Pav. 8 (b)) ir sudaromas plotas (Pav. 8 (c)), iš kurio bus šalinamos viršūnės. Viršūnės, priklausančios nagrinėjamai aibei, kurios buvo ištrintos iš viršūnių sąrašo, yra išsaugomos (Pav. 11) pavaizduotame atvejo sprendimui. Pašalinus viršūnes, minėtojo ploto centre yra sukuriama nauja viršūnė (Pav. 8 (d)).



a) b) c) d)

*Pav. 8 Viršūnių šalinimas paliekant tik centrinę.*

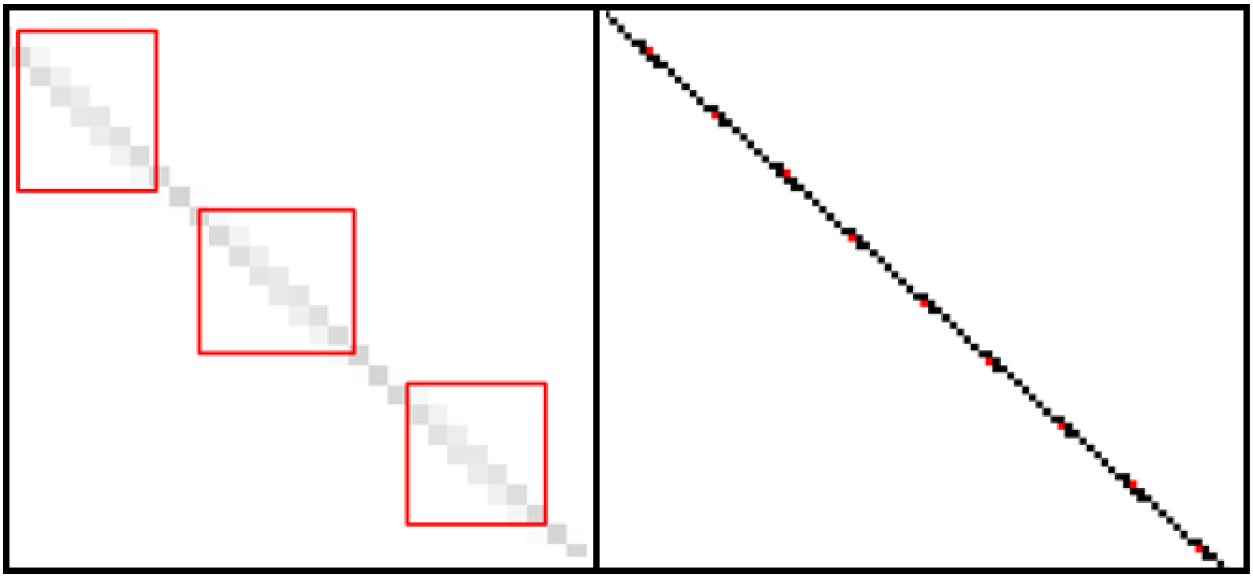
Galiausiai, atlikus visas prieš tai aprašytus veiksmus susijusius su grafo viršūnių šalinimu, yra gaunamas galutinis paprasto grafo piešinys, kuriame yra pavaizduotos visos grafe esančios viršūnės (Pav. 9).



***Pav. 9*** *Galutinis atpažinto grafo piešinys. Raudoni taškai žymi atpažintas viršūnes.*

### 2.2.5. Šalinimas, remiantis sudarytomis aibėmis

Testuojant programą, kurioje įgyvendintos prieš tai paminėtos funkcijos susijusios su grafo viršūnių tvarkymo veiksmais, buvo pastebėta, jog grafų vizualizacijos programos generuoja kiek kitokias linijas nei ranka piešiant paprastą piešinį. Skirtumas yra tame, jog vizualizacijos programos briaunos linijos lūžį piešia nudažant lūžio pikselius skirtingos spalvos juodos spalvos atspalviais, o ne palieka ryškų (Pav. 10 (a)), todėl bandant atpažinti piešinį, briaunos lūžio vietose (priklausomai nuo grafo dydžio ir lūžio kampo) briauna yra atpažįstama kaip dviejų pikselių storio ir jai yra taikoma funkcija, kuri ištrina viršūnių aibę ir palieka centrinę viršūnę. Tokiu būdu, ant briaunos lieka nemažai papildomų viršūnių, kurios nėra tikros (Pav. 10 (b)).

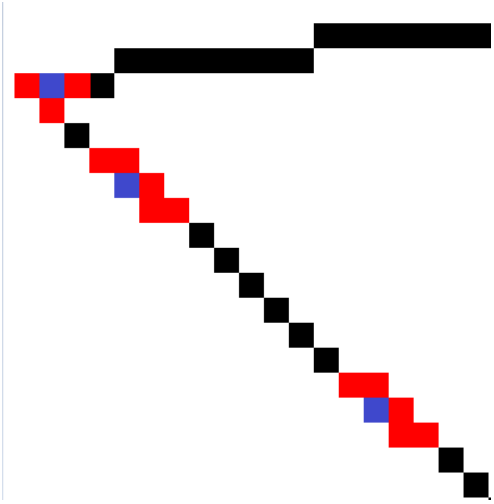


a) b)

***Pav. 10*** *a) Programos sugeneruoto piešinio briaunos lūžis. b) Atpažinta sugeneruoto piešinio briauna.*

Tam, kad pašalinti nenumatytas, nereikalingas viršūnes (Pav. 10 (a)) buvo nuspręsta sugalvoti dar vieną šalinimo būdą, kurį taikant yra tikrinamos visos likusios dvi kaimynes turinčios viršūnės pagal tai ar bent viena iš tiriamosios viršūnės kaimynių yra viršūnių, esančių tarpusavyje 1 pikselio

atstumu, aibės centru. Jeigu tokių, tarkime, yra dvi, tikrinama yra vidurinė viršūnė su kiekviena viršūne iš pirmosios viršūnių aibės, kur pirmoji tiriamosios viršūnės kaimynė yra aibės centru bei antrosios viršūnių aibės, kur antroji tikrinamosios viršūnės kaimynė yra aibės centru (Pav. 11).



*Pav. 11 Juoda – briaunos pikseliai, raudona – viršūnių aibė, esančių tarpusavyje 1 pikselio atstumu, mėlyna – viršūnių aibių centrai.*

Jeigu buvo nagrinėjamas tvarkingas ir sąlygas atitinkantis grafo piešinys bei visi minėtieji būdai buvo pritaikyti nuskaitytam piešiniui, tada iš likusių viršūnių turėtume rezultate turėti visas tikrąsias pradiniame piešinyje nupieštas.

# 3. Grafų vizualizacijos įrankių sugeneruotų piešinių atpažinimo programa.

Grafo atpažinimas iš nuotraukos yra sudėtingas uždavinys, kuris, šiuo atveju, yra naudojamas, grafų vizualizacijos įrankių sugeneruotų piešinių ar pačių įrankių veikimo vertinimui. Minėtojo uždavinio sprendimą galima automatizuoti, naudojant automatinį grafo piešinių atpažinimą. Renkant duomenis šiam darbui, buvo rasta panašios krypties darbų susijusių su grafų atpažinimu [ABB+13], tačiau funkcionalaus ir darbo formatą atitinkančio įrankio rasti nepavyko, gebančio atpažinti grafą iš nuotraukos, todėl kaip vieną iš būdų vertinti vizualizacijos įrankių generuojamus piešinius, buvo nuspręsta parašyti ir naudoti programą, kuri gebėtų atpažinti paprastą grafą iš jo piešinio ***.**png formatu.

## 3.1. Atpažinimas iš nuotraukos

Atpažinimu iš nuotraukos (angl. Image recognision) yra vadinamas procesas, kurio metu yra siekiama išskirti visą įmanomą naudingą informaciją iš bandomos atpažinti nuotraukos, pavyzdžiui, visą nuotraukos turinį. Nuotraukos atpažinimas yra natūralus ir be galo svarbus pačiam žmogui, kad pažinti bei suprasti jį supantį pasaulį [Web05], tačiau vis labiau yra žengiama su dirbtiniu atpažinimu iš nuotraukos, kurį vykdo mašina. Vykdant atpažinimo procesą, labai svarbu yra išskirti duomenis, kurie iš tikrųjų yra svarbūs bandomoje atpažinti nuotraukoje, o kurie yra pašaliniai [Web04].

Nuotraukų atpažinimas naudojamas gana plačiai, pavyzdžiui paieškos varikliuose, kaip Google ar Bing galima ieškoti panašių rezultatų pagal įkeltą nuotrauką. Pirma nuotrauka yra

išanalizuojama, atpažįstamas pagrindiniai objektai ir tada ieškoma duomenų bazėse panašių nuotraukų [Web06].

Šio darbo rėmuose yra apsiribojama atpažinimu grafo piešinio iš nuotraukos, sugeneruotos grafo vizualizacijos įrankiais, pagal apibrėžtus paprasto piešinio kriterijus (2.1). Kaip buvo minėta, programa geba atpažinti 1 pikselio storio briaunas Tikslas ateityje gali būti plečiamas iki grafo atpažinimo, kur briaunos storis neturės įtakos grafo atpažinimui, neapsiribojant vien tik programomis. Išplėtus tikslą iki minėtojo, galima bus tirti, tarkime, ranka ant lapo pieštų ir atskanuotų grafų piešinius iš nuotraukų.

Kaip buvo minėta, šiame darbe nagrinėjame ar programos, generuojančios grafų piešinius geba sugeneruoti paprastą piešinį pagal šio darbo rėmuose nustatytus kriterijus. Atpažinimas vykdomas grafo piešinių atpažinimo įrankiu, apie kurį plačiau pasakojama ateinančiame poskyryje.

## 3.2.Apie įrankį

Grafų atpažinimo iš nuotraukos įrankis yra parašytas naudojantis JAVA kalba. Programa gauna vieną įvestį – nuotraukos, kurioje yra pavaizduotas grafas, pavadinimą. Nuotrauka yra atidaroma ir nuskaitomi nuotraukoje esančių pikselių RGB kodai. Programa veikia atmetinėjimo principu, tai yra, visos vietos, kur gali būti viršūnė, yra įtraukiamos į viršūnių sąrašą, o vėlesniuose žingsniuose yra einama per sudarytą viršūnių sąrašą ir atmetinėjami tie taškai, kurie tikrai nėra viršūnėmis. Pradžioje visiems juodiems pikseliams yra priskiriama vieneto reikšmė, o baltiems priskiriama nulio reikšmė. Taip pat, nuskaičius vienetą, viršūnei yra priskiriamas unikalus numeris, pagal kurį yra lengviau identifikuojama viršūnė, su ja atliekant veiksmus.

Iš esmės, programos veikimo principas yra paskirstytas į 8 etapus:

1. Pradinio failo nuskaitymas ir pavertimas visų pikselių baltais (0) arba juodais (1), jeigu juodas pikselis, pridėjimas jo koordinačių prie viršūnių sąrašo.
2. Visoms viršūnėms jų kaimynių priskyrimas.
3. Viršūnės šalinimas, kai ji turi dvi kaimynes ir su jomis sudaro tiesią liniją (vertikalią, horizontalią ar įstrižą).
4. Viršūnės šalinimas, kai tiriamoji turi dvi kaimynes ir su jomis sudaro statų kampą (2.2.2).
5. Vidurinės viršūnės šalinimas, kai atstumas iki tiesės, jungiančios dvi vidurinės kaimynes ir statmenos jai tiesės išeinančios iš antros viršūnės, yra mažesnis už vieną.
6. Viršūnių aibės šalinimas ir naujos viršūnės įvedimas aibės centre, kai yra daugiau nei viena viršūnė tarpusavyje esančių mažiau nei 1 pikselio atstumu.
7. Viršūnės, turinčios dvi kaimynes šalinimas tikrinant ar bent viena iš kaimynių buvo viršūnių aibės centru. Jeigu taip, tai yra tikrinama viršūnė su dviem kaimynėmis pagal 4 punktą.
8. Rezultatų išvedimas.

### 3.2.1. Reader()

Failo nuskaitymas yra vykdomas atskiros klasės *Reader* konstruktoriuje, kuriam kaip parametrai yra perduodami norimo atpažint grafo nuotraukos pavadinimas bei atpažinimo tolerancija (pagal nutylėjimą 239) Tolerancija – šviesiausio bandomo atpažinti pikselio vienos iš RGB spalvos kodų reikšmė, kadangi visų spalvų vertės yra vienodos. Šis skaičius reiškia šviesiausio nuskaityto pikselio spalvos kodą. Kadangi vizualizacijos įrankiuose briaunos atvaizdavimo spalva yra nustatoma kaip juoda, tad briaunų lūžiuose pikseliai bus kažkurios spalvos nuo juodos iki baltos. Kitaip tariant, raudonos, žalios ir mėlynos spalvos kiekis bus vienodas, todėl pakanka nurodyti tik vieną iš spalvų tolerancijos rėžių. Šiuo atveju yra žiūrima

pagal raudonos spalvos kiekį pikselyje. Kuo tolerancijos skaičius didesnis, tuo šviesesni pikseliai yra žymimi kaip viršūnės.

### 3.2.2. createVerticesList()

Pirmajame etape yra nuskaitomi pradiniai duomenys, vykdant funkciją *createVerticesList()*. Visi juodos spalvos pikseliai yra laikomi viršūnėmis. Iš pat pradžių yra suformuojamas sąrašas viršūnių, kuriame kiekviena iš jų turi savo unikalų numerį, x ir y koordinates bei kaimynių sąrašą.

### 3.2.3. findNeighbours()

Antrajame etape jau turint suformuotą visų viršūnių sąrašą, yra pradedama duomenų analizė kviečiant *findNeighbours()* funkciją – sukamas ciklas nuo pradinės koordinatės (0,0) iki nuotraukos rėmelio dydžio, kuris yra gaunamas kaip paties paveikslėlio papildomas laukelis, ir ieškoma visų einamosios viršūnės kaimynių kiekvienai viršūnei iš sudaryto sąrašo. Paieška yra vykdoma paeiliui aštuoniomis kryptimis (Lentelė 1) Kaip buvo minėta, programa pradžioje laiko visus nebaltus pikselius viršūnėmis ir vėliau šalina taškus, kurie nėra viršūnės, todėl pagrindinis programos atliekamas veiksmas yra tikrųjų viršūnių paiešką ir taškų, kurie nėra viršūnėmis trynimas.

*Lentelė 1 Tikrinamos einamosios viršūnės koordinatės.*

| | | |
|---|---|---|
| y-1, x-1 | y-1, x | y-1, x+1 |
| y, x-1 | Einamoji | y, x+1 |
| y+1, x-1 | y+1, x | y+1, x+1 |

Radus viršūnės kaimynę, pastaroji yra pridedama į pradinės viršūnės papildomą laukelį *neighbours* (liet. Kaimynės), kuris yra *ArrayList* tipo. Toks sprendimas buvo priimtas dėl to, kad nereikia iš anksto nurodyti kokio dydžio bus masyvas, o taip pat yra paprastos viršūnių pridėjimo bei ištrynimo iš sąrašo operacijos. Po išvardintų veiksmų, jau yra sudarytas sąrašas viršūnių, kur kiekviena iš jų turi savo kaimynių sąrašą. Tolimesni žingsniai tikrina tam tikrus kriterijus ir pagal juos šalina viršūnes, neatitinkančias prieš tai minėtų sąlygų.

### 3.2.4. eliminateInlineVertices()

Turint visus reikiamus duomenis analizei, programa pradeda ieškoti nereikalingų viršūnių, kviesdama funkciją *eliminateInlineVertices(i),* kur *i* yra skaičius nuo 0 iki viršūnių skaičiaus sąraše. Pačioje pradžioje yra tikrinamos viršūnės, turinčios dvi kaimynes, kurios kartu su einamąja viršūne sudaro vertikalią, horizontalią arba įstrižą tiesę (Pav. 4), kadangi pagal susitarimą yra aišku, jog tiesėje, kurią niekas nekerta viršūnės nebus. Jeigu tokia viršūnė yra randama, ji yra pašalinama iš viršūnių sąrašo. Ši paieška yra atliekama rekursiškai į gyli. Radus kriterijus atitikusią viršūnę ir ją pašalinus, einama prie kairiosios kaimynės ir atliekami tie patys žingsniai, kaip ir su prieš tai buvusią. Šie veiksmai yra atliekami tol, kol prieinama viršūnė turinti ne dvi kaimynes arba tos dvi kaimynes su einamąja nesukuria vertikalios, horizontalios arba įstrižos tiesės. Apie viršūnių trynimą ir su tuo iškylančiomis problemomis yra kalbama ateinančiame poskyryje.

### 3.2.5. eliminateOnAngleVertices()

Tęsiant viršūnių šalinimą, yra kviečiama funkcija *eliminateOnAngleVertices(),* kuri vykdo tikrinimą ar einamoji viršūnė turi lygiai dvi kaimynes, su kuriomis sudarytų statų kampą (Pav. 5). Radus tokią viršūnę, yra peiliui siunčiamos pirmoji ir antroji tiriamosios viršūnės kaimynės į trynimo funkciją.

### 3.2.6. eliminateByDistances()

Toliau nagrinėjant piešinį yra taikomas tikrinimas, aprašytas (2.2.3) poskyryje. Paieška programoje yra realizuota iš viršūnių sąrašo paimant pirmąją neizoliuota viršūnę, kuri turi ne mažiau nei vieną kaimynę, antroji viršūnė yra imama pirmu numeriu einanti viršūnė pirmosios kaimynių sąraše, o trečioji iš antrosios viršūnės sąrašo, tačiau yra tikrinama ar pirmoji ir trečioji viršūnė nėra viena ir ta pati. Taip pat, svarbu, kad pirmoji ir trečioji viršūnės nebūtų kaimynės tarpusavyje. Jeigu visos sąlygos yra tenkinamos, visos trys viršūnės yra siunčiamos į funkciją *eliminateByDistances(first, second, third),* kurioje yra sprendžiama ar pašalinti antrąją (vidurinę) viršūnę. Papildomas apribojimas yra antrosios viršūnės kaimynių skaičiui, kuris turi būti lygus dviem. Jeigu kaimynių skaičius viršija du, tai reiškia, kad arba tiriamasis taškas iš tikrųjų yra viršūnė ir negali būti ištrintas, arba toje vietoje yra briaunų susikirtimas. Šitos situacijos nagrinėjimą paliekame tolimesniam laikui. Tikrinimas ar galima pašalinti antrąją viršūnę remiasi atstumo ieškojimu tarp tiesės, jungiančios pirmąją ir trečiąją viršūnes, ir antrosios viršūnės (Pav. 6). Jeigu iškvietus funkciją *checkSecond(first, second, third)* paaiškėja, kad piešinyje klaustuku pažymėtas atstumas yra mažesnis nei vienetas, tada antroji viršūnė trinama, o pirmoji su trečiąja ir trečioji su pirmąja yra padaromos kaimynėmis.

### 3.2.7. deleteStickedVerticesAndLeaveOneInCenter()

Prieš paskutinė su viršūnių atrinkimu susijusi funkcija yra pavadinta *deleteStickedVerticesAndLeaveOneInCenter()*. Apie veikimo principą detaliau yra aprašyta (2.2.4) poskyryje. Funkcijos viduje veikti pradedama nuo viršūnių, tarpusavyje esančių ne didesniu nei vieno pikselio atstumu aibės sudarymu, kuri yra išsaugoma antrojo perduodamo kintamojo sąraše *makeListOfStickedNeighbours(current, nearVertices)*. Turint aibės sąrašą, sukuriama centrinė aibės viršūnė *center,* kuriai priskiriamos visos aibėje esančių viršūnių kaimynes, o kaimynėms priskiriama centrinė viršūnė. Vėliau, sukant ciklą, visos viršūnės, priklausančios einamajai aibei yra šalinamos ir panaikinami sąryšiai tarp jų su kaimynais. Galiausiai, viršūnė *center* pridedama kaip nauja į viršūnių sąrašą. Pati aibė yra išsaugoma sąraše *area,* kuris bus naudojamas toliau aprašysimoje funkcijoje.

### 3.2.8. removeLeftVerticesInTheMiddle()

Galiausiai, yra kviečiama yra funkcija *removeLeftVerticesInTheMiddle(),* kurią įvykdžius grafo atpažinimas yra baigtas. Funkcijos viduje yra einama per viršūnių sąrašą ir ieškoma likusiųjų viršūnių su dvejomis kaimynėmis. Turint einamąją viršūnę *current* yra tikrinamos dvi jos kaimynes *first* ir *second*, ar bent viena iš jų yra viršūnių aibės centru. Tokią radus, yra žiūrima, kurį iš trijų atvejų esama situacija atitinka. Atvejai yra:

- Abi kaimynės yra viršūnių aibių centrais;
- Viršūnių aibės centras yra *first*;
- Viršūnių aibės centras yra *second*;

Ketvirtas atvejis, kai nei viena kaimynė nėra viršūnių aibės centru nėra nagrinėjama dėl to, kad toks atvejis turi būti atpažintas paties pirmojo tikrinimo metu (2.2.1).

Situacijai atitikus bent vieną iš išvardintų aukščiau atvejų, yra sukamas ciklas, kuriame su kiekviena iteracija yra paimama nauja viršūnė *newFirst* arba *newSecond* iš aibės, kur *first* arba *second* yra centrinė. Radus tokią, yra kviečiama tikrinimo funkcija, aprašyta (2.2.3) skyriuje, pavadinimu *checkSecond(newFirst, current, newSecond, first, second)*. Jeigu *current* turi tik vieną kaimyną, esantį viršūnių aibės centru, tada į *checkSecond* funkciją yra perduodamas pats kintamasis. Pavyzdžiui, jeigu viršūnių aibės centru yra tik *first*, tai funkcijos kvietimas atrodytų taip - *checkSecond(newFirst, current, second, first, second)*. Funkcija tikrina atstumą tarp antrojo ir tiesės einančios per pirmąjį ir trečiąjį funkcijai perduotus kintamuosius. Jeigu atstumas yra mažesnis už vieną pikselį, current yra trinama, o viršūnių aibių centrai yra padaromi kaimynais.

Tokiu būdu patikrinus visas viršūnes yra gaunamas galutinis sąrašas viršūnių, kuris ir yra laikomas atpažintu grafu.

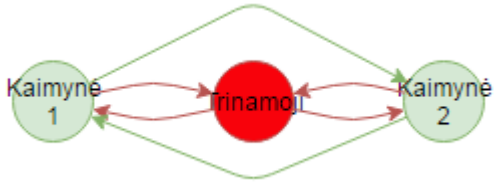
#### 3.2.9. createOutput()

Galiausiai, programa savo darbą baigia po to, kai išveda gautus duomenis. Rezultatai yra išvedami iškvietus funkciją *createOutput()*, kuri suformuoja dvi išvestis:

1. Sugeneruojant tokią pačią grafo vizualizacijos nuotrauką *.png* formatu, kaip ir ta, kuri buvo įvesta. Naujoje sugeneruotoje nuotraukoje taškuose, kuriose buvo atpažintos viršūnės, yra piešiamas raudonas taškas vietoj juodo.
2. Išvedant esminę informaciją į tekstinį failą. Faile yra išsaugomas nuskaitytas piešinys su viršūnių numeriais bei pilnas briaunų sąrašas, žymimas kurias viršūnių numeriais, kurios buvo tarpusavyje sujungtos. Kaip buvo minėta prieš tai, balta spalva yra žymima nuliu, o kiti skaičiai yra viršūnių unikalūs numeriai.

### 3.3. Viršūnės trynimas

Grafų piešinių vertinimo įrankyje viršūnės šalinimas šiek tiek skiriasi nuo to, kuris yra naudojamas grafų teorijoje ir labiau yra panašus į viršūnės sutapatinimą. Grafų teorijoje ištrynus viršūnę kartu su ja yra trinamos ir visos jai incidentiškos briaunos, tuo tarpu šiuo atveju, trinant viršūnę, privalu kaimynėmis padaryti trinamosios viršūnės kaimynes, taip pat, kaip ir sutapatinant viršūnę. Programos rėmuose, viršūnių trynimas yra vykdomas atliekant trynimo funkciją *removeSecond()*, išskyrus atvejį, kai šalinamos yra viršūnės iš atpažintos aibės (3.2.7). Su *removeSecond()* metodu viršūnė yra trinama panaikinant sąryšius su kaimynėmis ir sukuriant naujus sąryšius tarp trinamosios kaimynių (Pav. 12). Sukurti sąryšį tarp *Kaimynė 1* ir *Kaimynė 2* reiškia į *Kaimynė 1* sąrašą *Neighbours* pridėti *Kaimynę 2* ir atvirkščiai. Automatiškai, panaikinti sąryšį, reiškia pašalinti viršūnę iš *Neighbours* sąrašo.




*Pav. 12 Viršūnės šalinimas sujungiant kaimynes. Raudonai pažymėtos briaunos (sąryšiai) ir viršūnė yra trinamos, Žalios briaunos (sąryšiai) sukuriamos.*

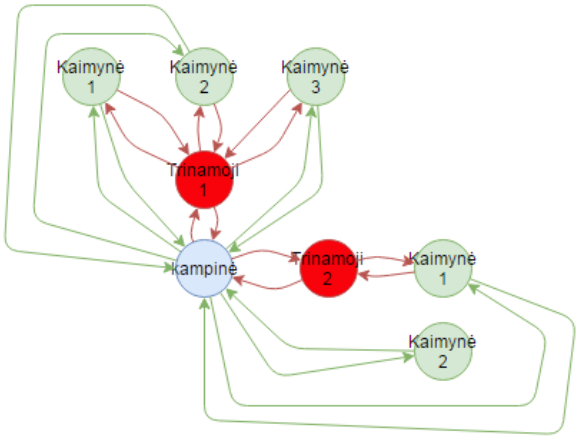
Minėtasis viršūnių šalinimo iš sąrašo metodas yra kviečiamas trijose funkcijose:

- Iš (3.2.4) tiesiogiai kviečiant trynimo funkciją, perduodant tikrinamąją viršūnę ir dvi jos kaimynes.
- Iš (3.2.6) yra kviečiama funkcija *checkSecond(neighbour1, toDelete, neighbour2)*, kurioje yra tikrinamas atstumas tarp *toDelete* ir tiesės, jugiančios *neighbour1* ir *neighbour2* yra mažesnis už 1, yra kviečiama *removeSecond()* funkcija.
- (3.2.8) situacija trynimo atžvilgiu yra identiška (3.2.6), tik checkSecond yra tokiu pačiu pavadinimu, bet skirtingu priimamų parametrų skaičiumi, naudojantis programavimo kalbos JAVA suteikiama galimybę panaudoti metodų perkrovimo galimybe (angl. Method overload) [Web01]. Šiuo atveju, checkSecond() priima 5 parametrus, tačiau veikimo principas yra identiškas prieš tai aprašytam, o konkrečiau (3.2.8).

Toliau išvardintos funkcijos šalina viršūnes šiek tiek kitokiu būdu.

### 3.3.1. Šalinimo funkcija, kai trinama (2.2.2) minėta viršūnė.

Kadangi, (3.2.5) situacijoje problema atsiranda tuomet, kai trinamoji viršūnė turi daugiau nei dvi kaimynes ir kyla klausimas, kurias gi kaimynes sujungti tarpusavyje, o iš kurių kaimynių sąrašo trinamąją viršūnę tiesiog pašalinti. Radus situaciją atitinkančias viršūnes, yra su kiekviena *Tikrinamąja* viršūne kviečiama funkcija *removeAndSwitch()*, kurioje atliekami veiksmai, pavaizduoti (Pav. 13) – *Tikrinamosios* viršūnės *Kaimynės* yra padaromos kaimynėmis su *kampine* viršūne, o *Tikrinamoji* yra pašalinama. Ši funkcija yra atliekama su kiekviena *Tikrinamąja* viršūne.




*Pav. 13 Dviejų viršūnių pašalinimas, esant (2.2.2) situacijai. Raudoni sąryšiai yra pašalinami, o žali sukuriami.*

### 3.3.2. Viršūnių aibės (2.2.4) šalinimas ir centrinės viršūnės kūrimas

Turint aibę viršūnių, atpažintą (3.2.7) būdu, viršūnių šalinimas vyksta ieškant nepriklausančių gretimų viršūnių aibei kaimynių tų viršūnių, kurios yra gretimų viršūnių aibėje. Suradus tokią viršūnę, iš jos kaimynių sąrašo yra pašalinama viršūnė, priklausanti gretimų viršūnių aibei ir pridedama tos aibės naujai sukurta centrinė viršūnė. Baigus atpažinimą, visos aibei priklausančios viršūnės yra ištrinamos iš viršūnių sąrašo.

# 4. Grafų vizualizacijos įrankiai

Šiame skyriuje bus pristatomi šio darbo tyrimo objektas – programos, skirtos grafų vizualizacijai. Bakalauriniam baigiamajam darbui buvo pasirinkta analizuoti sukurtus įrankius grafų arba tiesiog duomenų vizualizavimui didelių grafų bei paprastų grafų piešinių vizualizacijos požiūriu. Įrankiai buvo atrinkti tikrinant, ar jie atitinka šio darbo formatą – ar jų implementuotas funkcionalumas leidžia koreguoti galutinį grafo piešinį, taip pat, ar jie turi grafinę vartotojo sąsają, suderinamumą su testine aplinka. Nemaža dalis kitų programų, tokių kaip Pajek, Neo4j, GraphViz ir kt., buvo išbandyta, tačiau buvo atsisakyta jas įtraukti dėl šio darbo apimties, ar kitų prieš tai išvardintų priežasčių. Įrankių paieška buvo vykdoma įvairiuose internetinėse svetainėse susijusiose su duomenų analize [Web02][Web03], taip pat forumuose susijusiuose su grafų ar duomenų vizualizavimu ir analize bei moksliniuose straipsniuose [CLZ+10].

## 4.1. Cytoscape

Šio darbo rėmuose yra naudojama Cytoscape 3.4.0 versija. Cytoscape yra atviro kodo programinė įranga, skirta pagrinde biologinių tinklų, tačiau tinka ir socialinių tinklų integracijai, vizualizacijai bei analizei, sukurtas liepos mėn. 2002 metais. Įrankis naudoja java, todėl norint naudotis, teks įsidiegti Java Runtime Environment, kas yra ne visada gerai, turint omenyje java saugumo spragas [Zah15].

Cystoscape yra pakankamai suprantamas įrankis naudojimuisi netgi nedaug patirties su grafų vizualizacija ar SNA turinčiam žmogui. Visi punktai yra išdėstyti pakankamai intuityviai, yra galimybė išsaugoti gautą vaizdą skirtingais formatais, reguliuoti plotą skirta grafui. Yra laisvai pasiekiama ir gali būti įdiegta Windows, Linux bei MacOS X operacinėse sistemomose.

## 4.2. Gephi

Šio darbo rėmuose yra naudojama Gephi 0.9.1 versija. Gephi yra atviro kodo, nemokama programinė įranga, skirta tinklų vaizdavimui bet manipuliavimui, sukurta liepos mėn. 2010 metais. Programa yra suderinama su Windows, Linux bei MacOS X operacinėmis sistemomis.

Sukurti bei implementuoti įrankio viduje moduliai gali filtruoti, importuoti, vizualizuoti, manipuliuoti bei eksportuoti skirtingų tipų tinklus. Vizualizacijos modulis grafo vaizdinės išraiškos sukūrimui realiu laiku naudoja specialų 3D renderinimo (angl. rendering) variklį. Šis būdas gauti norimo grafo vizualizaciją, priklausomai nuo tiriamojo objekto dydžio, sunaudoja nemažai kompiuterio vaizdo plokštės, o procesorių palieką laisvą kitiems įrankio skaičiavimams. Yra teigiama [MAH12], jog praktinis didžiausio įrankio Gephi leistino grafo apribojimai yra 50 tūkstančių viršūnių bei 1 milijonas briaunų dėl to, kad ji yra parašyta naudojantis išskirstyto (lygiagretaus) skaičiavimo būdu, todėl geriau dirba aplinkoje, kurioje procesoriai turi didesnį branduolių skaičių. Yra implementuota nemažai funkcionalumo, skirto lengviau ir paprasčiau analizuoti vaizduojamus duomenis, pavyzdžiui, viršūnių stiliaus, dydžio ir spalvos ar net pačios viršūnės keitimas nuotrauka, briaunų storio ir spalvos keitimas, platus spektras išdėstymo algoritmų, kuriuos naudojant yra daug nustatymų, kuriuos vartotojas gali laisvai keisti realiu laiku. Yra galimybė leisti kelis algoritmus vienu metu, neblokuojant vartotojo sąsajos ar algoritmų tarpusavyje. Tekstinis modulis gali rodyti arba paslėpti pasirinktos duomenų skilties žymas vizualizacijos metu. Algoritmas Label Adjust gali būti naudojamas, norint, kad žymės tarpusavyje nepersidengtų.

## 4.3. Tulip

Šio darbo rėmuose yra naudojama Tulip 4.10 versija. Šis įrankis buvo kurtas kaip informacijos vizualizacijos karkasas (*angl.* framework), skirtas analizuoti bei vaizduoti duomenis su ryšiais,

tačiau galiausiai kūrėjai nusprendė sukurti ir versiją žmonėms neturintiems programavimo žinių, kitaip tariant su vaizdine vartotojo sąsaja. Šis įrankis gali būti leidžiamas trijose platformose – Windows, Linux bei MaxOS X [Dzi17].

# 5. Gebėjimas atpažinti paprasto grafo piešinį

Šiame skyriuje bus siekiama įvertinti grafų vizualizacijos įrankius bei jų generuojamus piešinius, tenkinančius paprasto piešinio apibrėžimą paminėtą anksčiau. Bandymai bei vertinimas bus atliekami remiantis sukurto grafų atpažinimo iš nuotraukos programos gautais rezultatais bandant atpažinti grafų vizualizacijos įrankių sugeneruotus piešinius.

Svarbu yra pastebėti, kad paprasto piešinio vienas iš ribojimų teigia, kad maksimali leistina briaunos storio linija yra 1 pikselio. Automatiškai, kad galima būtų vertinti įrankį, būtinai turi būti galimybė pasirinkti linijos storį. Taip pat, atpažinimo įrankis nesugeba atskirti jokio tipo papildomų žymėjimo elementų, pavyzdžiui, bet kokio viršūnių žymėjimo ar stilizavimo, briaunų ar viršūnių pavadinimų (angl. label), todėl tiriamosios programos pagalba generuojant paprastą grafo piešinį, svarbu yra gebėjimas nustatyti norimo generuoti piešinio parametrus taip, kad jie tenkintų šio darbo rėmuose apibrėžto paprasto piešinio kriterijus. Visos prieš tai pristatytos programos tenkina galutinio grafo piešinio koregavimo galimybes.

Testavimas su visais duomenų rinkiniais buvo atliekamas naudojant vienodus grafų piešinių nustatymus programose. Taip pat, buvo užtikrinami vienodi nustatymai ir tarp skirtingų programų. Nustatymai piešiniui, kurį grafų atpažinimo įrankis gali atpažinti yra šie:

- Panaikinti viršūnes ir bet kokią informaciją susijusią su jomis.
- Briaunos storį nustatyti 0,01 dydžio, kadangi nustačius 1px, piešinyje yra piešiama storesnė linija dėl briaunos lūžių.
- Balta (RGB[255,255,255]) fono spalva.
- Juoda (RGB[0,0,0]) briaunų spalva.

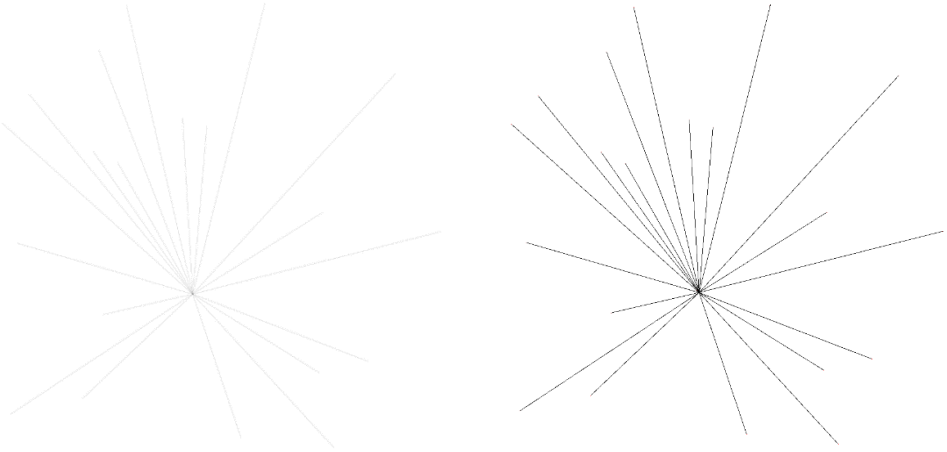
## 5.1. Pirmojo duomenų rinkinio atpažinimas.

Pirmuoju duomenų rinkiniu buvo pasirinkti žvaigždės tipo duomenys, kur centrinė viršūnė yra kaimynė su visomis kitomis grafo viršūnėmis (Pav. 14). Šis išdėstymas yra parankus todėl, kad žvaigždės tipo grafas pilnai atitinka keliamus paprasto grafo reikalavimus, tik vienoje vietoje turi tarpusavyje esančių ne toliau nei per vieną pikselį viršūnių aibę, nebent programa prideda briaunos lūžio taške papildomų pikselių.

Pirmąjį duomenų rinkinį sudaro 21 viršūnė bei 20 briaunų tarp jų. Duomenys yra sugeneruoti atsitiktinai ir jokios prasmės neturi.

**Gephi**

Grafų vizualizacijos įrankio Gephi piešinį grafų atpažinimo programai pavyko atpažinti

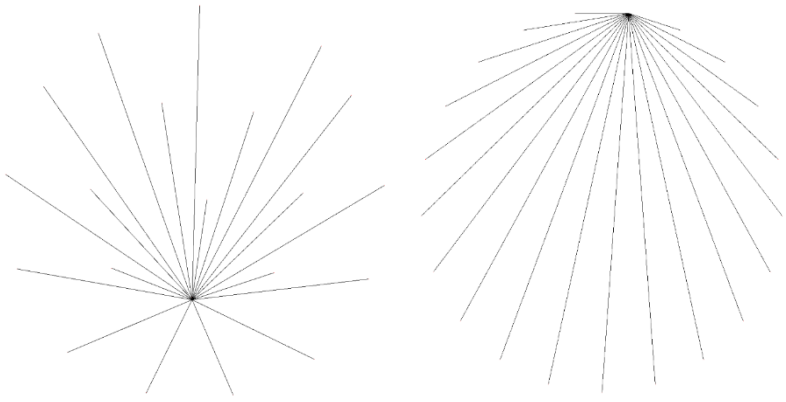


*a)* *b)*

*Pav. 14 a) Gephi įrankio sugeneruotas grafo piešinys; b) Grafų atpažinimo įrankio sugeneruotas piešinys, netaikant išdėstymo algoritmo.*

nesunkiai ir visiškai tiksliai. Pirmojo duomenų rinkinio atpažinimo metu įrankis atpažino 21 višūnę ir 20 briaunų. Naudota tolerancija buvo 239.

Grafas buvo sėkmingai atpažintas ir Gephi programoje pritaikius skirtingus viršūnių išdėstymo algoritmus, tarp kurių Fruchterman Reingold ir Apskirtas (angl. circular) (Pav. 15).

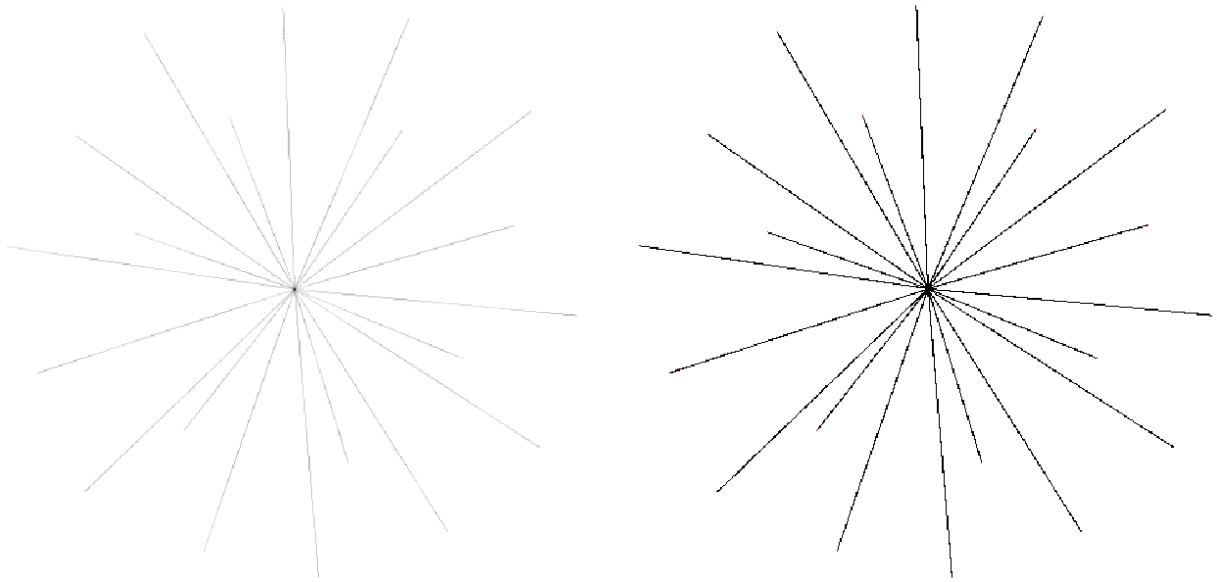


*a)* *b)*

*Pav. 15 a)(Pav. 14(b)) grafas po Fruchterman Reingold viršūnių išdėstymo algoritmo pritaikymo ir atpažinimo; b) Tas pats grafas po Circular layout pritaikymo ir atpažinimo.*

**Cytoscape**

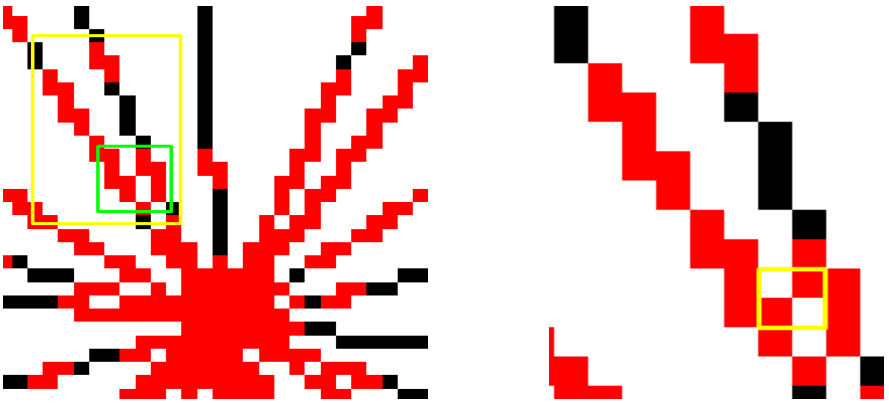
Bandant atpažinti programos Cytoscape sugeneruotą grafo piešinį, kuris, kaip galima įtarti, šiek tiek skyrėsi nuo Gephi piešinio, grafas nebuvo visiškai sėkmingai atpažintas. Programa atpažino 22 viršūnes ir 21 briauną.



*a)* *b)*

*Pav. 17 a) Cytoscape įrankio sugeneruotas grafo piešinys; b) Grafų atpažinimo įrankio sugeneruotas piešinys.*

Bandant išsiaiškinti, kodėl taip įvyko, buvo surastas nenumatytas atvejis atpažinimo programoje. Klaidingai atpažįstama yra tada, kai nagrinėjama geltonai pažymėta atkarpa (Pav. 16 (a)). Žaliai pažymėtame plote vykdant viršūnių aibės aptikimo ir pašalinimo funkciją, bereikalingai įtraukiama yra ir kitos briaunos dalis(Pav. 16 (b)). Rezultate gaunasi, kad centrinis taškas jungia ne vieną briauną, o dvi. Automatiškai, centrinei briaunai kaimynės yra įrašomos ne tik iš kairesniosios briaunos, bet ir iš dešiniosios. Kaimynių skaičius vietoj turėjusio būti du, tampa trimis.

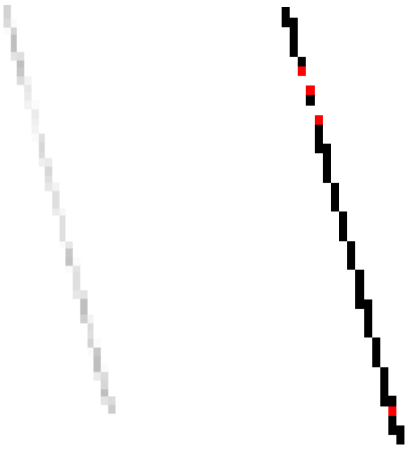


*a)* *b)*

*Pav. 16 Cytoscape piešinio atpažinimo klaidos vieta. Piešinys darytas atlikus 2.2.3 funkciją.*

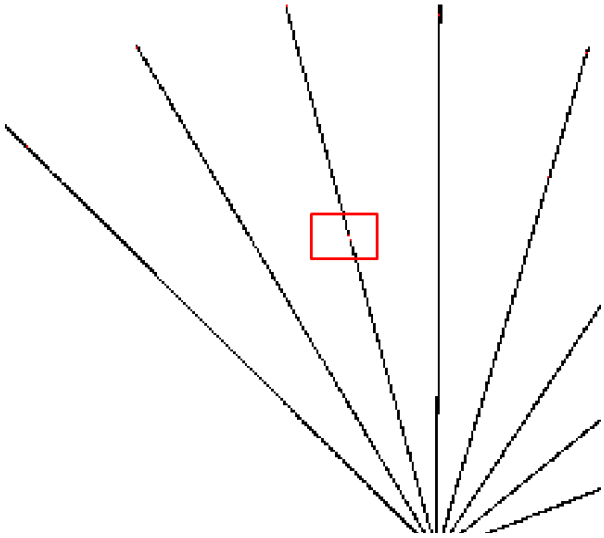
Dėl šitos priežasties, po Viršūnių aibių šalinimas sukuriant centrinę viršūnę funkcijos, bandant vykdyti Šalinimas, remiantis sudarytomis aibėmis funkciją, viršūnė yra praleidžiama, kadangi jos laipsnis yra didesnis už 2.

Bandant išvengti šios problemos pakeičiant viršūnių išdėstymą, buvo pastebėta, jog Cytoscape įrankio kuriamos grafo vizualizacijoje atvaizduotose keliose grafo briaunos lūžiuose, pikseliai buvo nupiešti labai neryškiai. Dėl šios priežasties, grafų piešinių atpažinimo įrankis nesugebėjo atpažinti visos briaunos. Keliose vietose, kuriose programos nupiešta briauna „lūžo“ ir pikseliai buvo per šviesūs(*Pav. 19*(a)), dėl šios priežasties briauna nebuvo korektiškai atpažinta, kaip pavaizduota (*Pav. 19*(b)).



*a)* *b)*

*Pav. 19 a) Cytosacpe programos sugeneruoto grafo piešinio briaunos fragmentas; b) (a) minimo fragmento atpažinimas.*

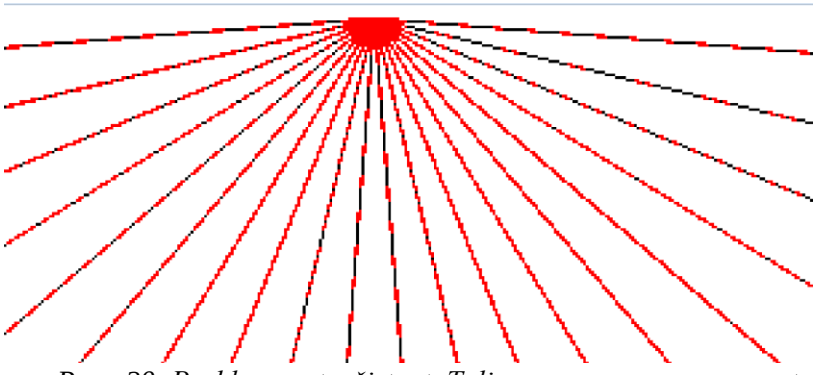


*Pav. 18 Problema, kylanti atpažįstant Cytoscape sugeneruotą grafą, jam pritaikius apskirtą išdėstymo algoritmą*

Ta pati minėtoji problema dėl briaunos fragmento neatpažinimo buvo ir toliau pastebima bandant atpažinti Cytoscape generuotus piešinius, taikant skirtingus išdėstymo algoritmus, pavyzdžiui, apskirtas. Tie patys rezultatai gaudavosi ir didinant atpažįstamos spalvos toleranciją, tačiau vietomis vis tiek atsirasdavo per šviesių pikselių. Atpažinimo rezultatai nepasikeitė tolerancijos skaičių pakėlus nuo 239 iki 246. Tęsiant tolerancijos padidinimą iki pakankamai didelio, didžioji grafo dalis buvo atpažįstama (2.2.4) funkcijos metu ir rodmenys buvo visiškai netikslūs.

**Tulip**

Bandant iš nuotraukos atpažinti programos Tulip sugeneruotą grafo piešinį iš karto kilo keblumas, kadangi dar prieš vykdant procedūrą (3.2.7) grafas atpažintas buvo su keliomis briaunomis, kurios yra 2 pikselių storio (Pav. 20), o sumažinus toleranciją nuo 96 iki 95, briaunos atrodė panašiai, kaip ir prieš tai (Pav. 19), dėl to skaičius atpažintų viršūnių buvo labai daug.



*Pav. 20 Problema atpažįstant Tulip programos sugeneruota grafo piešinį dėl pikselių spalvų*

Bandant programą su kitokiais duomenimis, išdėstymais ar viršūnių laipsniais atpažinimas nesikeitė. Dėl šios priežasties, ši programa testuojama nebuvo.

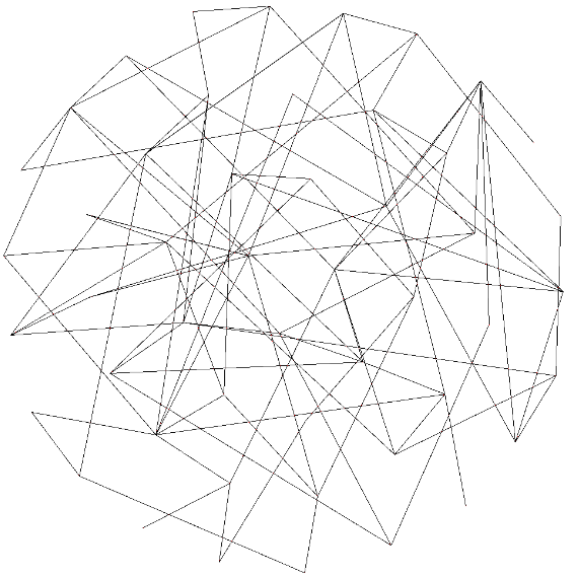
## 5.2. Antrojo duomenų rinkinio atpažinimas.

Antrasis duomenų rinkinys buvo automatiškai sugeneruotas, naudojantis Tulip programoje implementuotu funkcionalumu. Generuojant duomenis, buvo nustatyti grafo apribojimai dydžiui bei briaunoms, kad viršūnė neturėtų briaunos su pačia savimi. Buvo sugeneruotas grafas, turintis 50 viršūnių ir 100 briaunų. Nenaudojant išdėstymų algoritmo, atpažinti piešinį būtų neįmanoma, kadangi iš pradžių atsiranda labai daug briaunų persidengimų, todėl buvo naudoti išdėstymo algoritmai. Nepavyko visoms programoms panaudoti vienodų algoritmų, kadangi, pavyzdžiui Cytoscape turi labai nedidelį implementuotų algoritmų kiekį.

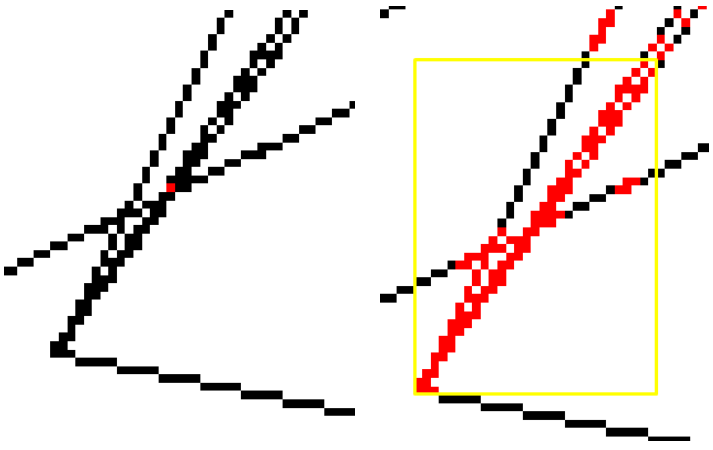
### Gephi

Įkėlus duomenų failą į įrankį, iš karto buvo pritaikytas Fruchterman Reingold grafo išdėstymo algoritmas (Pav. 22). Minėtosios programos piešinį atpažinti, naudojantis grafų piešinių atpažinimo įrankiu sekėsi neblogai. Iki pilno atpažinimo pritrūko atpažinti visus briaunų susikirtimus (Pav. 21(a)). To padaryti nepavyko, dėl (2.2.4) vykdytos funkcijos, kurios metu yra sudaromas geltonai pažymėtas plotas (Pav. 21(b)), iš kurio yra pašalinamos visos viršūnės ir užpiešiama viena centrinė viršūnė. Jeigu į minėtąjį plotą įeina ir viršūnės, esančios kituose briaunų susikirtimuose, jos irgi yra pašalinamos.

Viso, iš piešinio buvo atpažintos 303 viršūnės ir 609 briaunos. Skaičiai taip drastiškai išaugo dėl labai didelio briaunų susikirtimų skaičiaus esančio sugeneruotame grafo piešinyje.



*Pav. 22 Antrojo duomenų rinkinio vizualizacija programa Gephi, pritaikius Fruchterman Reingold išdėstymo algoritmą*



*a)* *b)*

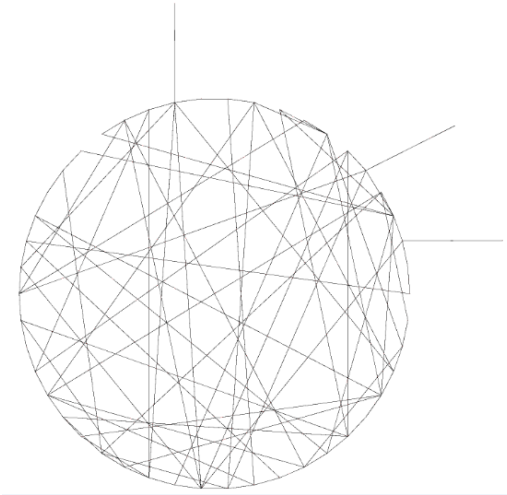
*Pav. 21 a)Vieta, kur vietoj kelių viršūnių palikta tik viena; b)Viršūnių aibė, kurios viduryje buvo sukurta nauja viršūnė.*

### Cytoscape

Atidarius antrąjį duomenų failą su Cytoscape, buvo pritaikytas Grid grafo viršūnių išdėstymo algoritmas (Pav. 23). Bandant atpažinti piešinį, buvo susidurta su (Pav. 18) pavaizduota problema, dėl to atpažintų viršūnių skaičius gerokai išaugo, kaip ir briaunų skaičius. Padidinus toleranciją,

kaip ir prieš tai, susidurta su problema, jog paimama per didelė viršūnių aibė, kuri yra šalinama ir rezultatai gaunasi netikslūs.

Galiausiai, pavyko atpažinti 476 viršūnių ir 888 briaunų.



*Pav. 23 Antrojo duomenų rinkinio pavaizduoti naudojant Cytoscape su apskirtu viršūnių išdėstymu.*

# 6. Išvados

Taigi, vizualizacija yra labai svarbi ir plati sritis, kuri dar nėra pilnai išnagrinėta. Vizualizuoti galima skirtingus duomenis ir skirtingais būdais. Norint matyti pačius objektus bei jų tarpusavio sąryšius, neišvengsime vizualizavimo grafais būdo, kadangi remiantis atliktais tyrimais, tai vienas labiausiai intuityvių, natūralų ir žmogui suprantamų piešimo būdų. Norint atlikti gerą vizualizaciją, teks išsirinkti vieną įrankį, kuriuo pagalba grafo piešiniu bus paverčiami pradiniai norimi vizualizuoti duomenys.

Skirtingi vizualizacijos įrankiai turi savo stipriąsias bei silpnąsias puses, atsižvelgiant į patogumą, implementuotą funkcionalumą, išnaudojamų resursų kiekį ar kt. [Dzi17], tačiau svarbu yra žinoti, kuri programa generuoja lengviausiai atpažįstamus grafų piešinius.

Šio darbo rėmuose, buvo apibrėžtos būtinosios ir pakankamos sąlygos piešiniui, kad jis tenkintų suformuluotus paprasto piešinio kriterijus (2.1). Remiantis išvardintais kriterijais buvo išnagrinėtas grafo atpažinimo iš nuotraukos procesas. Taip pat, neradus grafus iš nuotraukų atpažįstančio įrankio, buvo parašytas šį uždavinį sprendžianti programa bei išmėginta praktiškai su grafų vizualizacijos įrankių sugeneruotais piešiniais, siekiant palyginti grafų vizualizacijos įrankius.

Kaip paaiškėjo šio darbo metu, bandytos programos veikia bei atvaizduoja duomenis skirtingai. Tai tapo akivaizdu, bandant atpažinti sugeneruotus piešinius todėl, kad buvo pastebėti neesminiai skirtumai piešiant grafo briaunas. Esant briaunos „lūžiui“ programa Cytoscape piešia šviesesnius pikselius nei Gephi, dėl šios priežasties atsiranda papildomų keblumų, bandant atpažinti piešinį.

Remiantis atlikto tyrimo, atlikto su grafų piešinių atpažinimo įrankiu ir programų sugeneruotais grafų piešiniais, rezultatais, galima daryti išvadą, jog lengviau yra atpažinti grafų piešinius, sugeneruotus su programa Gephi.

# 7. Summary

Visualization of graphs is very important and wide area, which is not yet fully explored. Data can be visualized in many different ways. To see the objects and their inter-relationships, the most handy is to visualize data with graphs, whereas according to studies it is one of the most intuitive, natural and human-readable drawing technique. In order to make a good visualization, one should have to choose a tool for the transformation of data to visualize to the final drawing of a graph.

Different visualization tools has its own strong and weak points, depending on the comfort, functionality implemented, exploitation of resources quantities or others. [Dzi17], but it is important to know which program generates easily recognizable graph drawings for human as well as machine.

Within the framework of this paper, required and sufficient conditions were defined for a design which satisfies the criteria which was formulated for a graph drawing to be a simple drawing (2.1). Based on the criteria listed above, detection of image process was analyzed. Also, without an agreement on counts from the photo-aware tool that was written in this program and the task of dealing with the tried and tested in practice graph visualization tools generated drawings in order to compare the graph visualization tools to each other. Also, in the range of this paper a graph recognition from photo tool was not found, the program that manage to recognize a simple graph was created and tested in practice as an input to recognition program given a picture of a graph, generated by graph visualization program in order to compare visualization programs.

As it turned out this work, all of the tested the programs work and presents the input data as a graph in different ways. This became noticeable when attempting to identify the graph drawings because of a minor differences were seen in the way programs draw a graph edges. At the point where edge "breaks" Cytoscape program draws lighter pixels than Gephi, for this reason, there are additional problems in an attempt to recognize the pattern.

According to the survey, conducted with the graph drawing recognition tool and the drawings, generated with graph visualization tools, results, it can be concluded that it is easier to recognize graph drawings generated by the program Gephi.

# 8. Literatūros šaltiniai


[Bis05] H. Bisikirskaitė, „*Kai kurios briauninių grafų savybės*“, 2005. URL: http://www.moku.lt/wp-content/uploads/2015/05/briaunines-grafos.pdf

[Dzi17] J. Dzisevič, „*Grafų vizualizacijos įrankių analizė*“, 2017

[ABB+13] Auer C., Bachmaier C., Brandenburg F.J., Gleißner A., Reislhuber J. (2013) „*Optical Graph Recognition. In: Didimo“* W., Patrignani M. (eds) Graph Drawing. GD 2012. Lecture Notes in Computer Science, vol 7704. Springer, Berlin, Heidelberg

[Web01] https://docs.oracle.com/javase/tutorial/java/javaOO/index.html, 2015

[Web02] https://www.quora.com/What-software-exists-for-visualizing-and-analyzing-large-networks

[Web03] URL: http://www.kdnuggets.com/2015/06/top-30-social-network-analysis-visualization-tools.html

[CLZ+10] D. Combe, Ch. Largeron, E. Egyed-Zsigmond, M. Géry. A comparative study of social network analysis tools, 2010. URL: http://wic.litislab.fr/2010/pdf/Combe_WIVE10.pdf Dydis: 180KB

[Zah15] A. Zaharia. Why are Java’s Vulnerabilities One of the Biggest Security Holes on Your Computer?, 2015. URL: https://heimdalsecurity.com/blog/java-biggest-security-hole-your-computer/

[MAH12] Md. Maksudul Alam, S. M. Arifuzzaman, Md. Hasanuzzaman Bhuiyan. Large Scale Network Visualization With Gephi, 2012. URL: https://vtechworks.lib.vt.edu/bitstream/handle/10919/19088/FinalReportProjCINETViz.pdf?sequence=1 Dydis: 1,6MB

[Web04] URL: https://www.quora.com/What-is-image-recognition

[Web05] URL: http://whatis.techtarget.com/definition/image-recognition

[Web06] URL: https://www.quora.com/Why-is-image-recognition-important

# 9. Priedai

1. Kompaktinis diskas, prisegtas prie darbo.