

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS FUNDAMENTINIŲ MOKSLŲ FAKULTETAS INFORMACINIŲ TECHNOLOGIJŲ KATEDRA

**„Trumpiausio kelio radimas važiuojant dviračiais“**

#### Operacijų tyrimo ir taikymo kursinis darbas

Darbą atliko: DGTfm-15 studentai, Tomaš Tatul ir Ernestas Uscila

Darbą tikrino: Lekt. Aleksandr Igumenov

Vilnius, 2016

Turinys

[Įvadas 3](#_Toc468388196)

[Djikstra algoritmas 3](#_Toc468388197)

[Praktinė dalis 7](#_Toc468388198)

[Rezultatai 10](#_Toc468388199)

[Išvados 11](#_Toc468388200)

[Literatūros sąrašas 12](#_Toc468388201)

# Įvadas

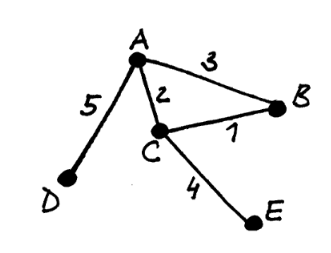
Šio darbo tikslas yra pademonstruoti kaip galima optimizuoti kelio pasirinkimą pasinaudojant tam tikrais algoritmais. Dažniausiai tokie algoritmai naudojami surasti trumpiausius arba ilgiausius maršrutus įvairiems vartotojams. Šiuo konkrečiu atveju buvo pasirinkta dviratė transporto priemonė – važiavimas dviračiais iš vieno taško į kitą tašką per tam tikrus tarpinius taškus.

Šiame darbe užduotis susideda iš to, kad du dviratininkai susilažino, kad pirmasis (toliau D1) gali aplenkti antrąjį (toliau D2), važiuojant nuo pradinio taško iki galutinio, kuomet D2 jau yra įveikęs dalį atstumo. Šiam uždaviniui spręsti reikia pasinaudoti trumpiausio kelio radimo algoritmais ir nustatyti ar D1 gali aplenkti D2. Tam reikia realizuoti ne tik algoritmą , bet ir sudarytį matematinį modelį, kuris padėtų pasirinktam algoritmui padarytų reikiamus skaičiavimus. Taip pat reikia padaryti programa, kuri realiai leistų vartotojui pasirinkti D2 pradinę vietą, atvaizduoti grafiškai žemėlapį ir galimus tarpinius taškus. Taip pat turi būti galimybė vartotojui kelio atkarpoms nurodyti atitinkamus parametrus, kurie pasunkina arba palengvina atkarpos sudėtingumą. Galiausiai visą apskaičiuota ir gautą rezultatą reikia atvaizduoti programoje.

# Djikstra algoritmas

Šiam uždaviniui spręsti bus naudojamas Djikstra trumpiausio kelio algoritmas. Dijkstros algoritmas arba Deikstros algoritmas – Edgar Dijkstra sukurtas algoritmas randantis trumpiausius kelius nuo vienos viršūnės iki kitų svoriniame grafe su neneigiamais svoriais [2].

Šis algoritmas yra iš svorinių grafų srities. Grafas, kurio visoms briaunoms (lankams) yra priskirti dydžiai (svoriai), vadinamas svoriniu (žr. 1pav).



1pav. Svorinio grafo pavyzdys

Dažnai tikslinga grafo briaunai (arba lankui) priskirti kokį nors dydį. Pavyzdžiui, jei grafu modeliuojama vietovės žemėlapį (viršūnėmis – miestus, o briaunomis – kelius), tai briaunoms galima priskirti tų kelių ilgius [1].

Tarkime yra svorinis grafas G, kurio briaunos (u, v) svoris reiškia atstumą tarp viršūnių u ir v. Kelio svoriniame grafe ilgiu vadinti galima visų kelią sudarančių briaunų svorių sumą. Nagrinėsime svorinį grafą G, kurio briaunos (u, v) neneigiamas svoris reiškia atstumą tarp viršūnių u ir v [1].

Dijkstros algoritmas iš duotosios viršūnės p randa trumpiausius kelius iki visų svorinio grafo viršūnių. Algoritmas skirsto viršūnes į dvi aibes: tų, iki kurių trumpiausi keliai (ir atstumai) jau žinomi (jas vadinsime prijungtomis), ir visų kitų. Pradžioje nežinomas trumpiausias kelias nė iki vienos viršūnės, išskyrus pradinę p, tad pažymima, kad atstumai iki šių viršūnių yra begaliniai. Atstumas (nuo pradinės) iki pradinės viršūnės jau žinomas – jis lygus nuliui [1].

Kiekvienu žingsniu algoritmas suranda dar neprijungtą viršūnę, iki kurios atstumas yra mažiausias (pirmu algoritmo žingsniu tai pradinė viršūnė p, kadangi iki visų kitų viršūnių atstumai yra begaliniai). Pasirinktoji viršūnė prijungiama, o tuomet atnaujinama informacija apie visas neprijungtas jos kaimynes: galbūt kelias iki šios viršūnės dar nebuvo rastas, o jei buvo – tai galbūt kelias, einantis per ką tik prijungtąją viršūnę iki šios kaimynės, yra trumpesnis už iki šiol rastąjį [1].

Taigi pirmuoju algoritmo žingsniu prijungiama pradinė viršūnė p. Antruoju – artimiausia p kaimynė. Kiekvienu žingsniu prijungiamų viršūnių atstumai sudaro nemažėjančią seką, kadangi visąlaik bandoma prijungti kuo artimesnes viršūnes. Šie samprotavimai intuityviai pagrindžia algoritmo teisingumą. Prijungdami viršūnę, galime būti tikri, jog rastasis atstumas yra trumpiausias, kadangi visi kiti, vėliau atrasti, trumpiausi atstumai bus tik ilgesni už šį [1].

Kadangi ieškoma trumpiausių kelių, o ne tik jų ilgių, kiekvienai viršūnei išsaugoma jos pirminė viršūnė (tai viršūnė, iš kurios į ją ateinama einant trumpiausiu keliu). Kol kelias iki viršūnės nerastas, jos pirminė viršūnė yra neapibrėžta. Atnaujinant atstumą iki viršūnės, kartu pažymima, iš kurios viršūnės į ją ateinama. Algoritmo vykdymo metu kiekvienos viršūnės pirminė viršūnė (kaip ir trumpiausias rastas atstumas) gali ne kartą pasikeisti. Dijkstros algoritmo vykdymas konkrečiame grafe, kai ieškomi trumpiausi keliai iš viršūnės a iki kitų grafo viršūnių [1].

Djikstros algoritmo iliustracija [1]:

|  |  |
| --- | --- |
| http://inf.knyga.nmakademija.lt/_images/67_lin_dijkstra1.png | Pradinė situacija: trumpiausio kelio iki viršūnės a (pasirinktosios pradinės viršūnės) ilgis lygus 0, o iki kitų viršūnių – nežinomas; |
| http://inf.knyga.nmakademija.lt/_images/67_lin_dijkstra2.png | Viršūnė a turi dvi kaimynes b ir c; iki šių viršūnių rasti trumpesni keliai |
| http://inf.knyga.nmakademija.lt/_images/67_lin_dijkstra3.png | Iš neprijungtų viršūnių išrenkama ta, iki kurios atstumas trumpiausias (viršūnę b); trumpesnio kelio iki b rasti negalima, ji prijungiama; peržiūrimos neprijungtos b kaimynės c ir d ir pastebima, kad iki šių abiejų viršūnių rasti trumpesni keliai per viršūnę b: iki viršūnės d kelias anksčiau nebuvo rastas, o iki viršūnės c buvo rastas tiesioginis kelias iš a; tačiau naujasis kelias per viršūnę b yra trumpesnis |
| Dijkstros algoritmo iliustracija |  |
| http://inf.knyga.nmakademija.lt/_images/67_lin_dijkstra5.png |  |
| http://inf.knyga.nmakademija.lt/_images/67_lin_dijkstra6.png | Baigus vykdyti Dijkstros algoritmą visos viršūnės yra prijungtos (t. y. visos yra pasiekiamos iš pradinės viršūnės) ir žinomi trumpiausi atstumai iki jų: trumpiausio kelio iki viršūnės b ilgis lygus 3, iki c – 4, iki d – 6, iki e – 8. |

Šio algoritmo sudėtingumas yra O(n^2), kur n – grafo viršūnių skaičius. Pasitelkus sudėtingesnes duomenų struktūras, Dijkstros algoritmą galima pagreitinti iki O((n + b) \log n) (čia b – grafo briaunų skaičius). Pastarasis sudėtingumas yra kur kas geresnis retuose (turinčiuose nedaug briaunų) grafuose [1].

# Praktinė dalis

Matematinio modelio pradiniai duomenys:

* žemėlapio situacija (konkreti vieta)
* pradinis taškas
* galutinis taškas
* D2 buvimo vietos taškas, kuomet pradedama judėti nuo pradinio taško (su sąlyga, kad varžovas pradėjo važiuoti anksčiau ir yra įveikęs atstumą)
* varžovo įveiktas atstumas nuo pradinio taško
* varžovo likęs kelias iki galutinio taško
* varžovo buvimo vietos sąlygos (žr. papildomos sąlygos)

Papildomos sąlygos:

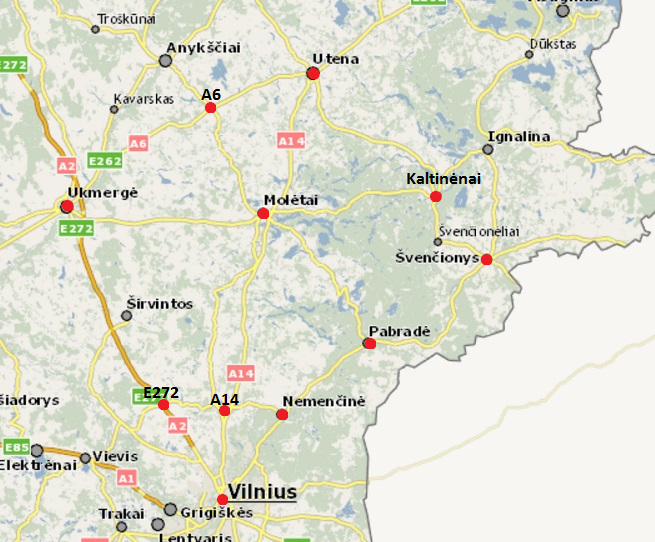
Viskas matuojama kalorijomis ir atitinkama kelio atkarpos kalorijų skaičius priklauso nuo šių parametrų:

* Lietus (l)
* Sniegas (sn)
* Vėjas (v)
* Aukštas tempas (at)
* Bekelė (b)
* Kalnuotumas (k)
* Didelis eismas (de)

1 kelio metras įvertinamas 1 kalorija. Priklausomai nuo parametrų 1 metras gali kainuoti daugiau kalorijų, gali mažiau. Kiek kalorijų reikia sunaudoti tam, kad įveikti atitinkamą atkarpą bus skaičiuojama pagal tokią formulę:

1lentelė. Papildomų sąlygų parametrai

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Parametras | l | sn | v | at | b | k | de |
| Koeficiento reikšmė | 0.1 | 0.2 | 0.1 | 0.1 | 0.2 | 0.1 | 0.2 |

Visas žemėlapis (žr. 1pav) susideda iš 11 sužymėtų tašku, pradinis taškas yra Vilnius, o galutinis Utena. D1 visada startuoja nuo Vilniaus, o D2 pradinį tašką reikia pasirinkti. D2 pradinė vieta negali būti Vilnius ir Utena.

2pav. Žemėlapis su taškais

Iš viso žemėlapyje yra 23 atkarpos (žr. 2lentelė).

2lentelė. Galimos atkarpos

|  |  |
| --- | --- |
| Atkarpa | Atstumas (km) |
| Vilnius – E272 | 24 |
| Vilnius – A14 | 19 |
| Vilnius – Nemenčinė | 22 |
| E272 – A14 (A14 – E272) | 14 |
| E272 – Ukmergė | 45 |
| A14 – Molėtai | 45 |
| A14 – Nemenčinė ( Nemenčinė – A14) | 14 |
| Nemenčinė – Pabradė | 24 |
| Ukmergė – Molėtai (Molėtai – Ukmergė) | 41 |
| Ukmergė – A6 | 42 |
| Molėtai – Utena | 35 |
| Molėtai – Kaltinėnai ( Kaltinėnai – Molėtai) | 40 |
| Molėtai – A6 | 29 |
| Pabradė – Molėtai | 43 |
| Pabradė – Švenčionys | 37 |
| Švenčionys – Kaltinėnai | 21 |
| A6 - Utena | 24 |
| Kaltinėnai – Utena | 43 |

# Rezultatai

Programa buvo realizuota panaudojant JAVA programavimo kalbą. Grafinė vartotojo sąsaja buvo sukurta panaudojant libGDX JAVA žaidimų programavimo karkasą. Taip pat buvo naudotas Maven projektų valdymo ir Git kodo versijavmo įrankiai.

# Išvados

Realizuotas djikstra algoritmas, padaryta grafinė vartotojo sąsaja, kurioje vartotojas gali nustatyti D2 pradinę vietą, taip pat gali kiekvienai kelio atkarpai nurodyti papildomas parametrus, kurie apsunkina tą atkarpą. Apskaičiuotas kelias yra pateikiamas vartotojui ekrane, o taip pat atvaizduojama papildoma informacija apie įveiktą kelią.

Vartotojui negalima keisti žemėlapio, tačiau programa padaryta taip, kad joje labai papratai galima pakeisti, kad būtų naudojamas kitas žemėlapis ar kitos kelio atkarpos.

# Literatūros sąrašas

1. *Svoriniai grafai, Trumpiausio kelio paieška – Dijkstros algoritmas,* [žiūrėta 2016-11-28]. Prieiga per internetą: < <http://inf.knyga.nmakademija.lt/10_dijkstra.html>>
2. *Dijkstros algoritmas,* [žiūrėta 2016-11-29]. Prieiga per internetą: < <https://lt.wikipedia.org/wiki/Dijkstros_algoritmas>>
3. *LibGDX Documentation,* [žiūrėta 2016-11-30]. Prieiga per internetą: < <https://libgdx.badlogicgames.com/documentation.html>>