DAUGAVPILS UNIVERSITĀTE
Dabaszinātņu un matemātikas fakultāte
Matemātikas katedra
Maģistra studiju programma "Matemātika"

Studiju kurss

Diskrētā matemātika

13.lekcija

Docētājs: Dr. P. Daugulis

2012./2013.studiju gads

Saturs

Lekcijas mērķis:

- apgūt svarīgākos grafu algoritmus.

Lekcijas kopsavilkums:

- var definēt divus algoritmus grafa virsotņu apiešanai.
- dažādu grafu problēmu risināšanai var izmantot rijīgos vai pārlases tipa algoritmu.

Svarīgākie jēdzieni: plašummeklēšanas koks, dziļummeklēšanas koks, minimāla svara pārklājošā koka problēma, rijīgais algoritms, minimālā svara pārklājošā koka problēma, īsākā ceļa problēma.

Svarīgākie fakti un metodes: plašummeklēšanas algoritms, dziļummeklēšanas algoritms, Kraskala algoritms, Prima algoritms, Dijkstras algoritms.

1. Grafu algoritmi

1.1. Vispārīgie apsvērumi

Grafu algoritmisko problēmu klasifikācija:

- invariantu aprēķināšana vai novērtēšana, ja dots grafs,
- invariantu aprēķināšana visiem grafiem dotā grafu kopā,
- grafu konstruēšana ar dotām invariantu vērtībām vai īpašībām,
- grafu pārveidojumu veikšana.

Grafu algoritmisko problēmu izcelsme:

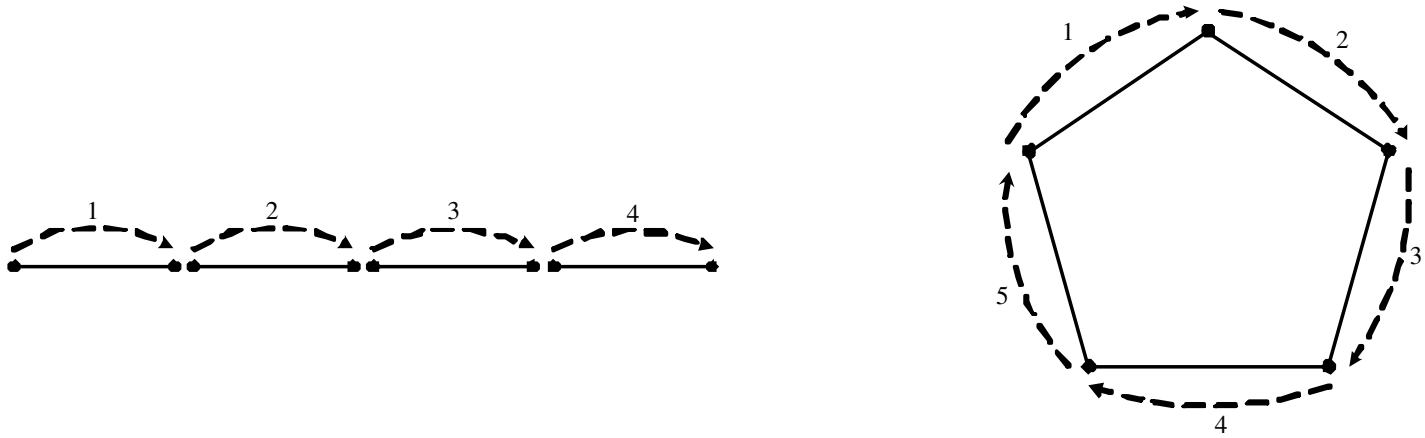
- noteikta veida invariantu, īpašību un algoritmu svarīgums dažādos modeļos,
- konkrētiem modeļiem specifiski invarianti/algoritmi,
- teorētisko pētījumu atbalsts - hipotēžu pārbaude.

1.2. Meklēšana/pārlase

1.2.1. Vispārīgie apsvērumi

Bieži risinot teorētiskus vai praktiskus uzdevumus, ir nepieciešams apiet (pārmeklēt, pārlasīt, iezīmēt) grafa virsotnes noteiktā, algoritmiskā kārtībā, izmantojot grafa šķautnes.

Šāda algoritma efektivitātes dēļ ir lietderīgi minimizēt katras virsotnes iezīmēšanas reižu skaitu, piemēram, pieprasīt, ka katra virsotne tiek iezīmēta ne vairāk kā vienu reizi. Kāda varētu būt grafu virsotņu apiešanas dabiskā kārtība? Šo uzdevumu var viegli atrisināt, ja grafs ir, piemēram, ķēde vai cikls (skatīt 3.65.attēlā).



3.65. attēls. Ķēdes un cikla apiešanas veidi

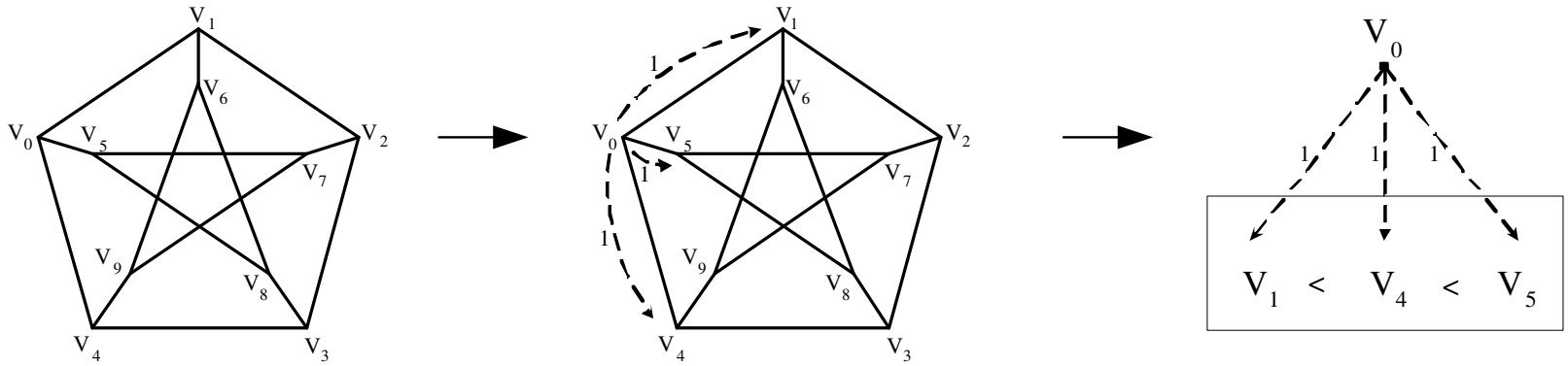
Ko darīt patvaļīga grafa gadījumā?

1.2.2. Plašummeklēšana

Plašummeklēšanas (PM) algoritma apraksts

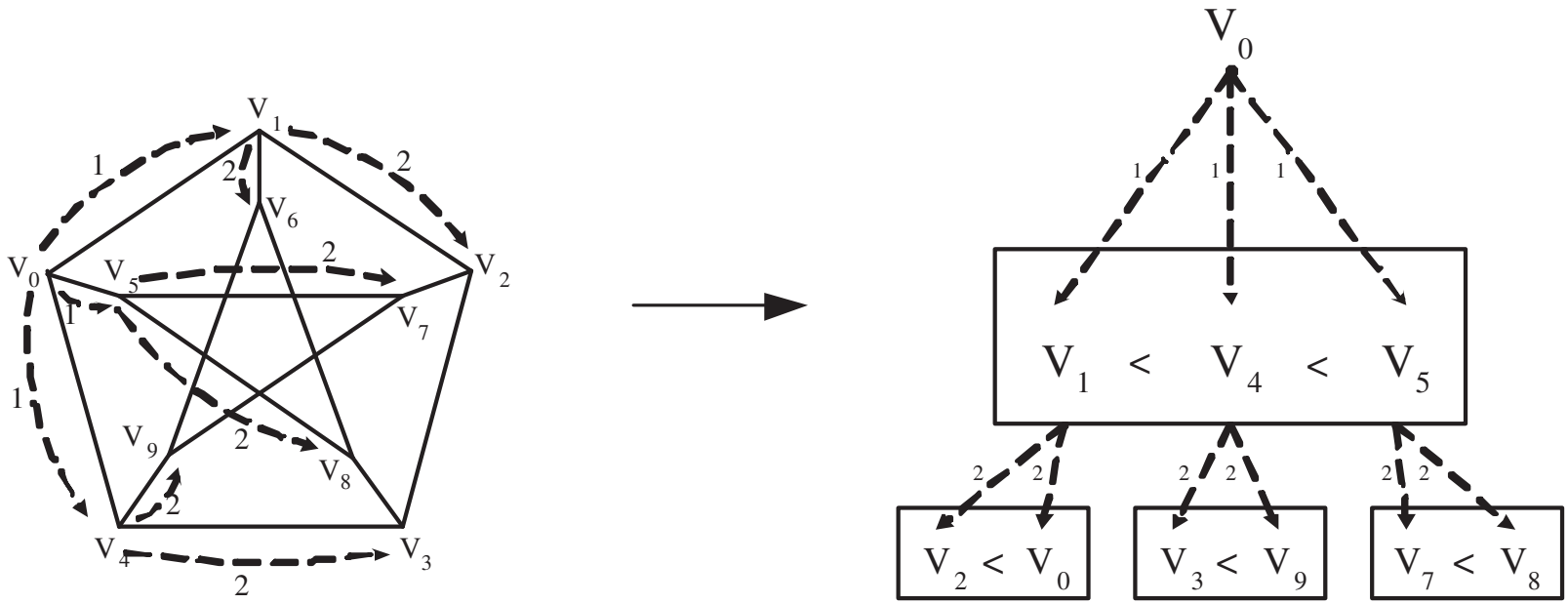
- Sākam ar fiksētu virsotni v_0,
- atzīmējam visas ar to savienotās virsotnes $\{v_{01}^1, v_{02}^1, ..., v_{0n_1}^1\}$, pilnīgi sakārtojam tās veidā $v_{01}^1 < v_{02}^1 < ... < v_{0n_1}^1$, konstruējam

sakārtotu koku B_1.



3.66. attēls. Pirmais solis PM algoritmā Petersena grafā

- Nākošajā solī konstruējam sakārtotu koku B_2 šādā veidā: apejam pilnīgi sakārtoto virkni $\{v_{01}^1, v_{02}^1, ..., v_{0n_1}^1\}$ tās elementu pieaugošajā kārtībā un katrai virsotnei v_{0i}^1 atzīmējam un piekārtojam kā dēlu kopu visas virsotnes $\{v_{i1}^2, v_{i2}^2, ..., v_{in_1}^2\}$, kas ir savienotas ar v_{0i}^1 un vēl nav atzīmētas (skatīt 3.67.attēlā).
- Turpinām šo procesu, konstruējot sakārtotu koku virkni B_1 ,B_2 ,..., kuras pēdējais loceklis tiek saukts par *plašummeklēšanas koku*.
- PM algoritms beidz darbu, kad ir atzīmētas visas virsotnes.



3.67. attēls. Pirmie divi soļi PM algoritmā Petersena grafā

Piezīmēsim, ka šis koks nav noteikts viennozīmīgi, tas ir atkarīgs no pirmās virsotnes v_0 un no katras iekšējās virsotnes dēlu sakārtojuma.

Operāciju skaita novērtēšana

Definēsim $g(x) = O(f(x)) \iff \exists\, c_1, c_2 > 0 :$

$$c_1 f(x) \leq g(x) \leq c_2 f(x),\ \forall\ x > N.$$

$\Gamma = (V, E)$ ir uzdots ar blakusattiecības sarakstu.

- PM procesā $\forall$ virsotne tiek ievietota kokā vienu reizi, tātad summārais operāciju skaits ir $O(|V|)$.
- $\forall$ virsotnes blakusattiecības saraksts tiek apskatīts vienu reizi, tātad summārais operāciju skaits šajā algoritma daļā ir $O(|E|)$.
- Papildus operāciju skaits grafa inicializācijai ir $O(|V|)$. Var redzēt, ka kopējais algoritma darba laiks ir $O(|V| + |E|)$.

Pielietojumi

Izmantojot PM, var atrisināt šādus uzdevumus:

- noteikt, vai grafs ir sakarīgs, un atrast grafa komponenšu skaitu - to var izdarīt, veicot PM no jebkuras virsotnes, grafs ir sakarīgs tad un tikai tad, ja tiek apietas visas grafa virsotnes, komponenšu skaits ir vienāds ar PM skaitu, kas ir nepieciešamas, lai apietu visas virsotnes;
- noteikt attālumu starp divām virsotnēm - to var izdarīt, veicot PM no vienas virsotnes, attālums starp virsotnēm ir vienāds ar attālumu no sākotnējās virsotnes līdz otrai virsotnei PM kokā.

1.2.3. Dziļummeklēšana

Otrs svarīgs grafa virsotņu pārlases veids ir *dziļummeklēšana (DzM, pārlase dziļumā, depth-first search)*.

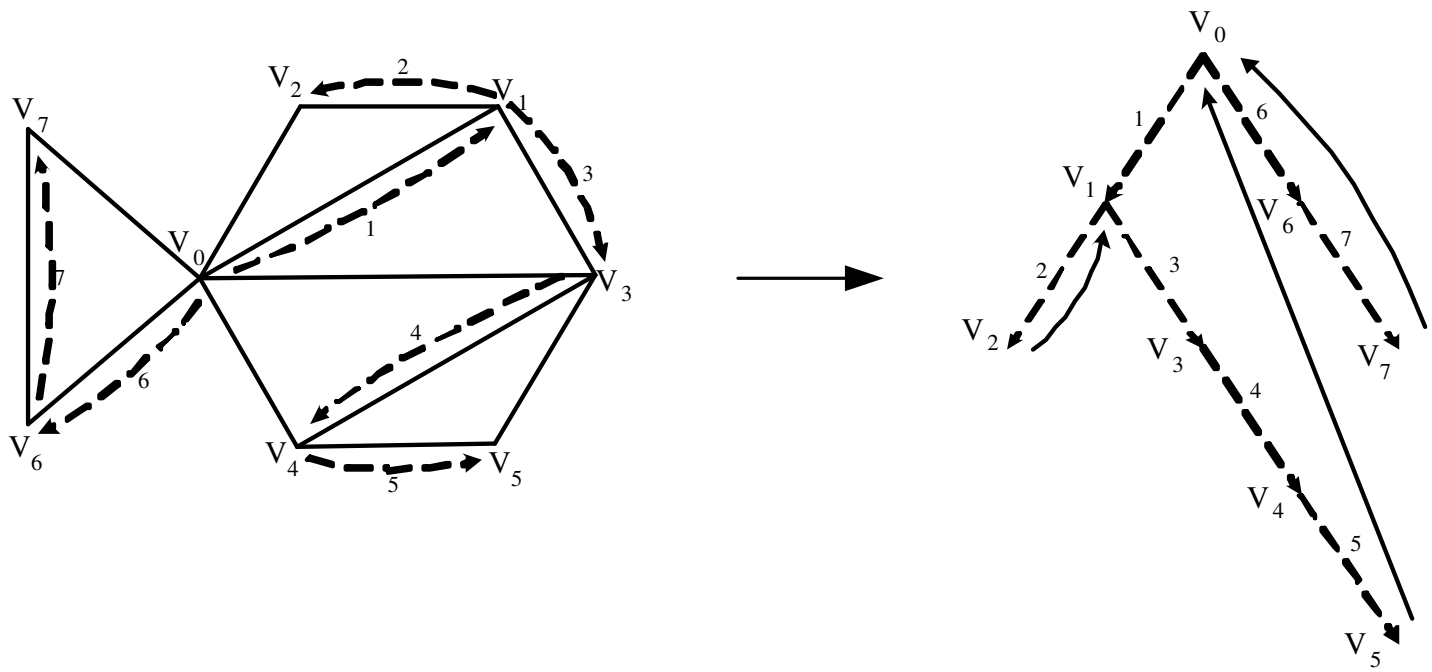
Algoritma apraksts

- Sākam ar fiksētu virsotni v_0 kā topošā dziļummeklēšanas koka sakni,
- atzīmējam jebkuru vēl neatzīmētu virsotni v_1, kas ir savienota ar v_0, DzM kokā zīmējam šķautni $v_0 \to v_1$ un pārvietojamies uz virsotni v_1.
- Vispārīgā gadījumā rīkojamies šādi: ja mēs atrodamies virsotnē v un eksistē vēl neatzīmēta virsotne w, kas ir savienota ar v, tad
  1) atzīmējam virsotni w,
  2) dziļummeklēšanas kokā zīmējam šķautni $v \to w$,
  3) pārvietojamies uz virsotni w.

  Pretējā gadījumā (ja visas virsotnes, kas ir savienotas ar v, jau ir atzīmētas) pārejam (atkāpjamies) uz virsotnes v tēvu DzM kokā.
- Turpinām šo algoritmu tik ilgi, kamēr atgriežamies atpakaļ virsotnē v_0.

- Rezultātā iegūsim *DzM koku*. Bieži vien ir lietderīgi papildus atzīmēt arī "atkāpšanās" šķautnes.

1.1. piemērs.



3.68. attēls. DzM algoritma realizācijas piemērs

Operāciju skaita novērtēšana

Var pierādīt, ka DzM algoritma darba laiks ir $O(|V| + |E|)$.

Pielietojumi

DzM tāpat kā plašummeklēšanu pielieto

- grafa komponenšu un stingri sakarīgo komponenšu noteikšanā,
- metrisko invariantu (diametra, ekscentritātes, centra) noteikšanā.

DzM pielieto arī algoritmos, kuros tiek risināti uzdevumi par augstāku kārtu sakarīgumu:

- šarnīru un bloku noteikšana.

1.3. Minimālā svara pārklājoša koka meklēšana

1.3.1. Problēma

$\Gamma = (V, E, w)$ nosvērts (neorientēts) sakarīgs grafs, $\forall$ šķautnei e ir piekārtots pozitīvs skaitlis $w(e)$.

Apskatīsim šādu uzdevumu: konstruēt Γ pārklājošu koku, kura šķautņu svaru summa ir minimāla.

1.1. piezīme. Inženiertehniska (ceļu tīkla), elektrotehniska vai datortehniska tīkla sakaru līniju kopējā garuma minimizēšana.

1.3.2. Rijīgie algoritmi

Minimāla svara pārklājoša koka meklēšanai tiek izmantoti "rijīgie" algoritmi. Algoritmu sauksim par *rijīgu (alkatīgu, greedy)*, ja katrā solī tiek izdarīta lokāli optimāla izvēle (šajā laika momentā vai situācijā).

Nebūt nav acīmredzami, ka lokāli optimāla izvēle ir arī globāli

optimāla, bet minimāla svara pārklājoša koka uzdevumā tas tā ir.

1.3.3. Algoritmi

Ir divi svarīgi rijīgie algoritmi - *Kraskala algoritms* un *Prima algoritms*, kuros tiek pakāpeniski būvēts mežs vai koks, pievienojot šķautnes ar minimāli iespējamu svaru.

Kraskala algoritms

$\Gamma = (V, E)$ - sakarīgs, $|V| = n$:

1) konstruējam grafu $T_1 = O_n + e_1$, pievienojam bezšķautņu grafam ar virsotņu kopu V šķautni e_1, kuras svars ir minimāls;

2) ja grafs T_i jau ir konstruēts un $i < n-1$, tad konstruējam grafu $T_{i+1} = T_i + e_{i+1}$, kur e_{i+1} ir grafa Γ šķautne, kurai ir minimāls svars starp visām tām šķautnēm, kas neieiet grafā T_i un neveido ciklus ar grafa T_i šķautnēm, citiem vārdiem sakot, T_{i+1} ir mežs.

1.1. teorēma. Ja $i < n-1$, tad grafu T_{i+1} ir iespējams konstruēt saskaņā ar Kraskala algoritma aprakstu. Grafs T_{n-1} ir minimāla svara pārklājošs koks.

PIERĀDĪJUMS Grafā T_i ir i šķautnes $\Longrightarrow$ T_i ir nesakarīgs, ja $i < n-1$.

Γ ir sakarīgs $\Longrightarrow$ tajā ir vēl vismaz viena šķautne, kas neveido ciklus ar grafa T_i šķautnēm $\Longrightarrow$ meklējamā šķautne e_{i+1} $\exists$ un ir iespējams konstruēt grafu T_{i+1}.

T_{n-1} ir grafs ar n virsotnēm un $n-1$ šķautnēm bez cikliem $\Longrightarrow$ T_{n-1} ir koks.

Jāpierāda, ka koka T_{n-1} svars ir minimāls. Pierādīsim to izmantojot indukciju. Pierādīsim, ka $\forall\ i : 0 \leq i \leq n-1$ mežs T_i ir kāda minimāla svara pārklājoša koka apakšgrafs.

Indukcija bāze : Ja $i = 0$, tad apgalvojums ir acīmredzams, jo jebkurš pārklājošs grafs satur bezšķautņu grafu.

Indukcijas solis : Pieņemsim, ka apgalvojums ir patiess visiem T_i, ja $i < n-1$ un pierādīsim, ka pievienojot vēl vienu šķautni saskaņā ar Kraskala algoritmu apgalvojums arī ir patiess.

Pieņemsim, ka ir konstruēts mežs T_i. Pēc pieņēmuma $T_i \leq T$, kur T - minimāla svara pārklājošs koks.

Pieņemsim, ka nākamā izvēlētā šķautne ir e. Ja e ir kokā T, tad apgalvojums ir pierādīts, jo $T_i + e \leq T$. Ja e nav kokā T, tad grafam $T + e$ ir cikls C. Ciklā C ir šķautne f, kas nav mežā T_i. Pretējā gadījumā mēs nevarētu pievienot e, jo grafā $T_i + e$ rastos cikls.

$T+e-f$ tiek iegūts, izmetot vienu šķautni no grafa $T+e$ ar vienu ciklu $\Longrightarrow T + e - f$ ir koks.

$$w(T + e - f) = w(T) + w(e) - w(f) \geq w(T) \Longrightarrow w(e) \geq w(f).$$

Nevar būt, ka $w(e) > w(f)$, jo tad būtu izvēlēta šķautne f, nevis e - cikli nevar veidoties, jo tad pievienojot šķautni e, grafam $T + e$ būtu 2 cikli (grūti, ja pievienojot f veidotos cikls grafā $T_i + f$, tas būtu arī cikla grafā $T + f$, tātad grafā $T + f + e$ būtu 2 cikli). Seko, ka $w(f) = w(e) \Longrightarrow w(T + e - f) = w(T)$.

Seko, ka $T + e - f$ ir minimāla svara pārklājošs koks, kas satur $T_i + e = T_{i+1}$ un indukcijas solis ir pierādīts. ■

Kraskala algoritma rezultātā tiek iegūts *Kraskala koks*, kaut arī algoritma gaitā tiek būvēts *Kraskala mežs*.

Prima algoritms

Prima algoritms atšķiras no Kraskala algoritma ar to, ka katrā solī tiek konstruēts koks (*Prima koks*), kam pievieno minimāla svara šķautnes, kas neveido ciklus ar jau esošo koku, koka pirmā virsotne var tikt izvēlēta patvaļīgi.

Var pierādīt, ka Prima koks ir minimāla svara pārklājošs koks.

Operāciju skaits

Var pierādīt, ka Kraskala algoritma patērēto operāciju skaits ir $O(|E| \log |E|)$ un Prima algoritma operāciju skaits ir $O(|E| \log |V|)$.

1.4. Īsākā ceļa meklēšana

1.4.1. Problēma

$\Gamma = (V, E, w)$ - nosvērts (neorientēts vai orientēts) sakarīgs grafs ar pozitīviem svariem w, apzīmēsim škautnes svaru ar $w(e)$, bet (v, t)-ķēdes summāro svaru ar $dist(v, t)$.

Apskatīsim šādu uzdevumu: atrast minimāla svara ķēdi, kas savieno divas dotās virsotnes (*īsāko ceļu starp divām dotajām virsotnēm*).

Par dotā grafa *v-īsāko ceļu koku* sauksim pārklājošu orientētu koku, kura sakne ir v un kurā (v, w)-maršruts līdz virsotnei w realizē īsāko (v, w)-ķēdi.

1.2. teorēma.

1. ja virsotņu virkne $(v, v_1, ..., v_{n-1}, t)$ ir īsākais ceļš no v līdz t, tad jebkurām divām virsotnēm v_i un v_j, kur $i < j, v_0 = v, v_n = t$, virkne $v_i, v_{i+1}, ..., v_j$ ir īsākais ceļš no v_i līdz v_j.

2. Funkcija *dist* ir metrika kopā V (apmierina simetriju, trijstūra nevienādību, nedeģenerētību).

PIERĀDĪJUMS 1. Pieņemot pretējo iegūsim pretrunu.

2. Līdzīgi parastajai attāluma funkcijai (kas ir speciālgadījums). ■

1.2. piemērs. Tipiski īsākā ceļa pielietošanas piemēri ir saistīti ar optimāla maršruta noteikšanu transporta sistēmās (Google Maps).

1.4.2. Algoritmi

Plašummeklēšana

Ja visu šķautņu svari ir vienādi, tad šis uzdevums var tikt atrisināts ar plašummeklēšanas palīdzību.

Dijkstras algoritms

Vispārīgā gadījumā var izmantot *Dijkstras algoritmu*, kas ir līdzvērtīgs Prima algoritmam ar nelielu modifikāciju. Dijkstras algoritms balstās uz šādu teorēmu.

1.3. teorēma. Ja ir dotas minimāla svara ķēdes no virsotnes u līdz katrai virsotnei no kopas $W = \{u, v_1, ..., v_k\}$, tad $\exists\ v \in V \backslash W$, kur īsākā ķēde no u līdz v ir formā $(u, ..., t, v)$, kur ķēde $(u, ..., t)$ ir viena no dotajām ķēdēm vai kāda no to nepārtrauktajām apakšķēdēm, kas sākas ar u.

PIERĀDĪJUMS Ja ķēde $(u, ..., t)$ ir īsākā ķēde, tad jebkura tās nepārtraukta apakšķēde, kas sākas ar u, ir īsākā ķēde līdz tās galapunktam.

Apskatīsim virsotni v:

v minimizē lielumu $dist(u, t') + w(t', v)$, kur $t' \in \{u, v_1, ..., v_k\}$.

Ķēde $(u, ..., t', v)$, kas minimizē $dist(u, t') + w(t', v)$ ir minimāla

garuma (u, v)-ķēde. To pierāda pieņemot pretējo: ja $\exists$ īsāka (u, v)-ķēde, tad tai izejot no $\{u, v_1, ..., v_k\}$ iegūsim pretrunu.

$\Longrightarrow$ virsotne v ir meklējamā virsotne. ■

Tātad, lai atrastu īsāko ceļu starp divām virsotnēm u un v nosvērtā grafā, mums ir jākonstruē Prima koks ar papildus nosacījumiem:

- ir jāsāk ar vienu no divām dotajām virsotnēm, $\begin{cases} V := \{u\} \\ E := \varnothing \end{cases}$,
- katrā solī ir jāpievieno šķautne (u_i, t), kas neveido ciklus un minimizē lielumu $dist(u, v_i)+w(v_i, t)$, $v_i \in V$, $\begin{cases} V := V \cup t \\ E := E \cup \{u_i, t\} \end{cases}$;
- minimizēšana tiek veikta, pārskatot visas jau uzbūvētā koka virsotnes v_i un visas pārējās virsotnes t grafā.

Pēc galīga skaita soļu tiks konstruēts koks, kas satur abas no dotajām virsotnēm, attālums starp tām ir vienāds ar attālumu konstruētajā kokā.

Operāciju skaits Dijkstras algoritmā - $O(|V|^2 + |E|)$.

Attālumu matricas atrašana

Izmantojot īsākā ceļa atrašanas uzdevumu, var aizpildīt grafa virsotņu *attālumu matricu*, kuras rūtiņās tiek ierakstīti grafu teorētiskie attālumi starp rindai un kolonnai atbilstošajām virsotnēm. Attālumu matricu izmanto grafa metrisko invariantu aprēķināšanai.

2. 13.mājasdarbs

2.1. Obligātie uzdevumi

13.1 $m \times n$ taisnstūrī paralēli malām ir jāievieto maksimāls skaits "stūrīšu" (ar laukumu 3) tā, lai nevienu citu "stūrīti" vairs nevar ievietot. Noformulēt šo problēmu grafu terminos un izstrādāt algoritmu tās atrisināšanai.

13.2 Ir dota cilvēku grupa. Šajā grupā ir jāorganizē pāru sarunas, viens pāris runā 1 stundu atsevišķā telpā, telpu skaits ir/nav ierobežots, pāru saraksts ir dots. Uzdevums ir minimizēt kopējo sarunu laiku. Noformulēt šo problēmu grafu terminos un izstrādāt algoritmu tās atrisināšanai.

13.3 Ģenerēt nosvērtu neorientētu grafu ar 20 virsotnēm un realizēt Kraskala un Prima algoritmu.

13.4 Kādā valstī ir iesūtīti vairāki kaimiņu valsts spiegi. Starp jebkuriem diviem spiegiem i un j ir iespējams nodibināt divpusējus sakarus, kuru atklāšanas varbūtība ir p_{ij}. Ir jānodibina

divpusēju sakaru sistēma tā, lai jebkurš spiegs varētu nodot informāciju jebkuram citam spiegam un kopējā atklāšanas varbūtība ir mininmāla. Noformulēt šo problēmu grafu terminos un izstrādāt algoritmu tās atrisināšanai.

13.5 Ģenerēt nosvērtu neorientētu grafu ar 30 virsotnēm un realizēt Dijkstras algoritmu ar izvēlētu sākuma virsotni.

2.2. Paaugstinātas grūtības un pētnieciska raksturа uzdevumi

13.6 Izstrādāt grafu modeli spēles "Burbuļi" (no www.draugiem.lv) optimālai spēlēšanai.

13.7 Izstrādāt grafu modeli pilsētas ielu tīkla modelēšanai un uzlabošanai. Definējiet invariantus, kas varētu raksturot

(a) konkrētu ielu vai krustojumu īpašības,
(b) ielas un krustojumus ar ekstremālām īpašībām,
(c) sagaidāmo braukšanas laiku starp diviem dotiem punktiem.

Izstrādājiet algoritmu, ar kura palīdzību varētu optimālā veidā uzlabot ielu grafu (būvēt jaunas ielas, mainīt vai likvidēt esošās, noteikt kustības veidu ielās un krustojumos), lai

(a) samazinātu kopējo braukšanas laiku starp diviem punktiem,
(b) samazinātu korķus pārslogotajās vietās,
(c) vienmērīgāk sadalītu ielu un krustojumu parametrus,
(d) pārkārtotu transporta plūsmu vēlamajā veidā (kokveida struktūras guļamrajonos, caurbraucošās struktūras u.c.),
(e) vienmērīgi sadalītu ielu seguma noslodzi u.c.

13.8 Izstrādāt grafu modeli cilvēku populācijas savstarpējo attiecību un/vai informācijas plūsmas tīkla modelēšanai un uzlabošanai. Definējiet invariantus, kas varētu raksturot

(a) konkrētu indivīdu īpašības,
(b) indivīdus un attiecības ar ekstremālām īpašībām,
(c) informācijas plūsmu un tās īpašības definētajā grafā.

Izstrādājiet algoritmu, ar kura palīdzību varētu optimālā veidā uzlabot definētos grafu (optimizēt individuālos un sabiedriskos procesus), lai

(a) optimizētu informācijas izplatīšanos.
(b) optimāli sadalītu grafa parametrus pa virsotnēm un šķautnēm u.c.