Algoritmai ir duomenų struktūros

10 paskaita

2024-04-16

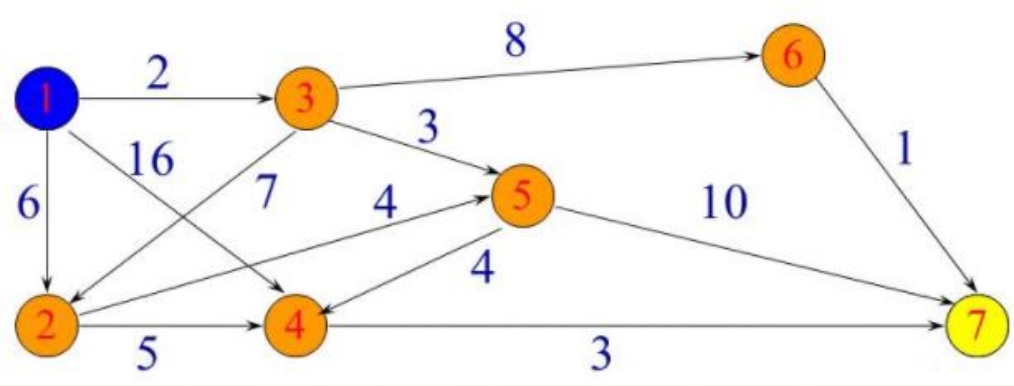
10 paskaitos tikslas

Susipažinti su trumpiausių takų paieškos algoritmais grafuose ir digrafuose:

- Trumpiausi takai tarp pradinės viršūnės ir kitų likusių:
  - Dijkstros algoritmas,
  - Belmano–Fordo algoritmas.
- Trumpiausi takai tarp visų viršūnių porų:
  - Lėtas trumpiausių takų paieškos algoritmas,
  - Greitas trumpiausių takų paieškos algoritmas,
  - Floido–Varšalo algoritmas.

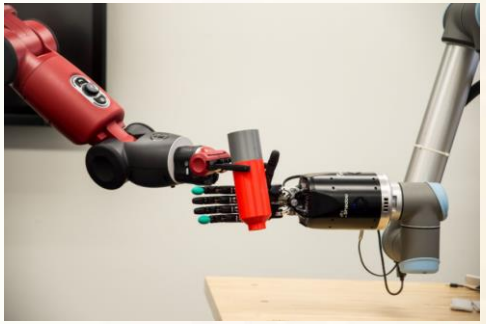
Tako ilgis svoriniame (di)grafe

- Tako ilgiu svoriniame grafe arba digrafe vadinama takui priklausančių briaunų svorių suma.
- Jei grafas nėra svorinis, tako ilgiu arba atstumu tarp viršūnių laikomas briaunų, priklausančių takui, skaičius.

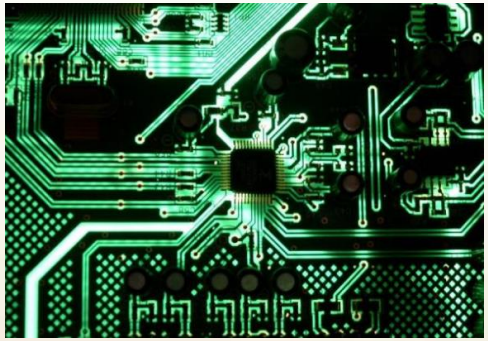


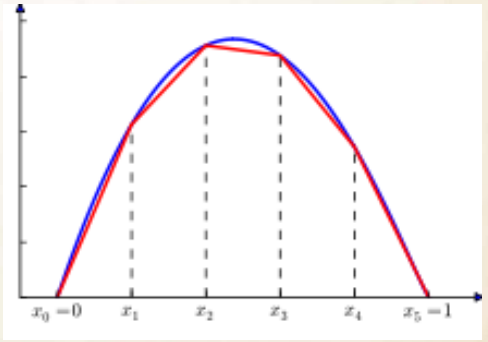
Kiek iš viso skirtingų takų veda iš viršūnės 1 į viršūnę 7?

Trumpiausių takų algoritmai taikomi



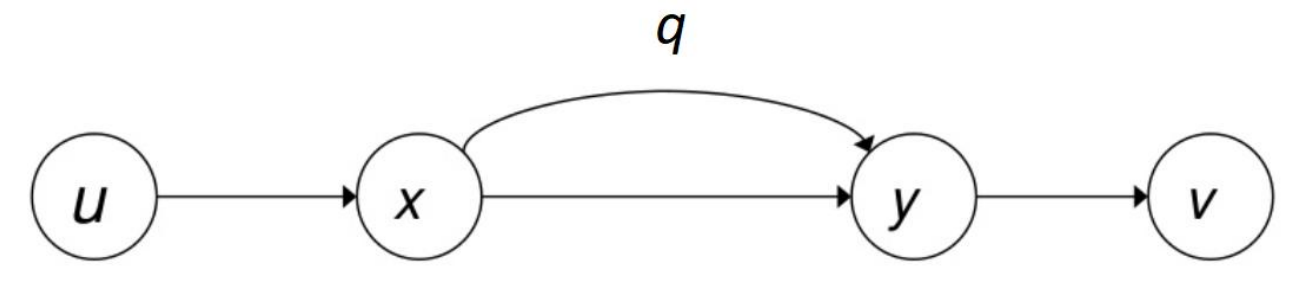
- Robotų navigacijai.
- Kelionės maršrutams sudaryti.
- Mikroschemų architektūroje.
- Gamybos optimizavimo uždaviniuose.
- Transporto uždaviniams spręsti.
- Tiesinių funkcijų rinkinio (laužčių) aproksimavimui.
- Skaitmeninių modelių apdorojimui (pvz.: *texture mapping*).





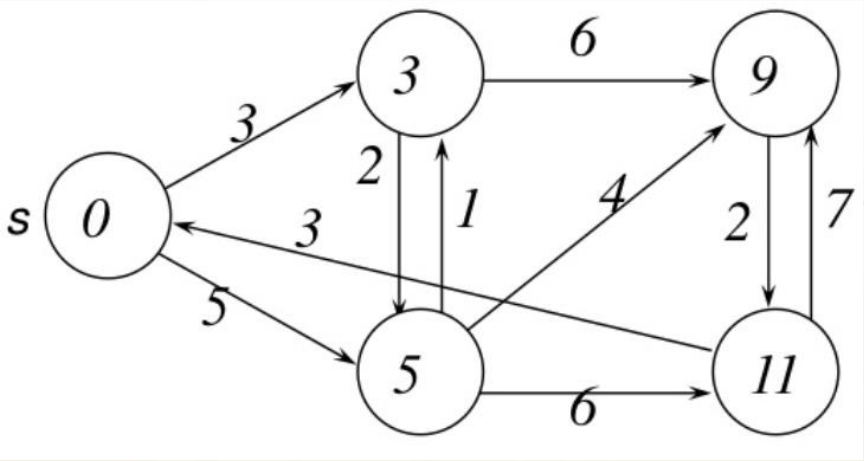
Trumpiausio tako savybė

- **Teorema.** Jei p yra trumpiausias takas nuo viršūnės u iki viršūnės v, tai kiekvienas kitas potakis priklausantis takui p irgi yra trumpiausias pradžios ir pabaigos viršūnių atžvilgiu.
- **Įrodymas.** Tegu q yra takui p priklausantis potakis. Jei egzistuotų trumpesnis potakis už q, tai takas q irgi būtų trumpesnis.



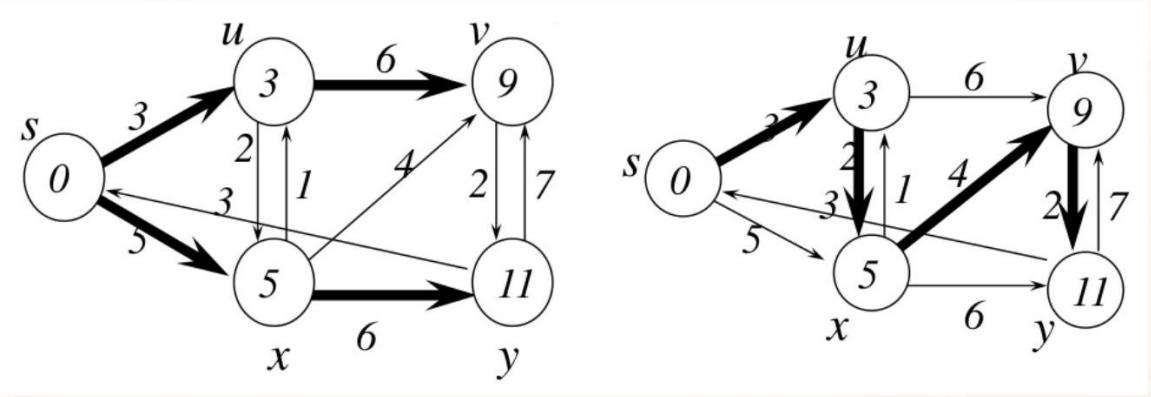
Trumpiausi takai nuo šaltinio viršūnės

- **Užduotis.** Svoriniame grafe ar digrafe reikia rasti trumpiausius takus nuo šaltinio viršūnės s iki likusių viršūnių.
- **Pastaba.** Trumpiausias takas tarp viršūnių u ir v negali būti rastas asimptotiškai greičiau nei trumpiausias takas tarp u ir likusių (di)grafo viršūnių.



Trumpiausių takų paieškos idėja (1)

- **Užduotis.** Rasti trumpiausią taką (ne tik jo ilgį).
- **Sprendimas.** Sukonstruoti jungiantį medį (di)grafe, kuriame egzistuotų vienintelis takas nuo šaltinio iki tikslo viršūnės.



- **Klausimas diskusijai.** Ar šis medis yra minimaliai jungus?

Trumpiausių takų paieškos idėja (2)

- Tegu $\delta(u,v)$ – trumpiausio tako ilgis tarp viršūnių u ir v.
- Visi trumpiausių takų (nuo šaltinio viršūnės) algoritmai turi kiekvienai viršūnei u turi atributą d[u], kuris apytiksliai lygus $\delta(s,u)$.
- Algoritmo vykdymo metu d[u] kaskart atnaujinamas, kol galiausiai $d[u] = \delta(s,u)$.
- d[u] kaskart iš naujo perskaičiuojamas, jei teisinga nelygybė

$$d[u] \geq \delta(s,u).$$

- Tuo tikslu naudojamas viršūnės u tėvystės atributas π[u], kuris parodo trumpiausią atstumą nuo s iki u. π[u] iš naujo perskaičiuojamas tada, kai atnaujinamas d[u].

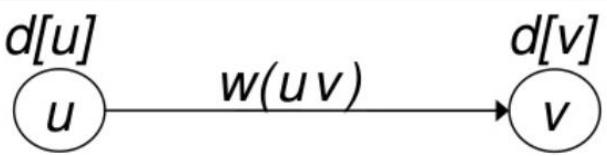
Inicializacija ir briaunos relaksacija

INIT(G,s):

1. **for each** $v \in V$
2. **do** $d[v] \leftarrow \infty$
3. $\pi[v] \leftarrow NIL$
4. $d[s] \leftarrow 0$

RELAX (uv, w) :

1. **If** $d[v] > d[u] + w(uv)$
2. **then** $d[v] \leftarrow d[u] + w(uv)$
3. $\pi[v] \leftarrow u$



- Atributai d[v] ir π[v] – kintantys.
- Po briaunos uv relaksacijos bus teisinga nelygybė

$$d[v] \leq d[u] + w(uv).$$

Dijkstros algoritmas

DIJKSTRA (G, w, s) :

1. INIT(G,s)
2. $S \leftarrow \emptyset$
3. $Q \leftarrow V$
4. **while** $Q \neq \emptyset$
5. **do** $u \leftarrow$ "išskirk $\min(Q)$" (ir u išmetama iš Q)
6. $S \leftarrow S \cup \{u\}$
7. **for each** $v \in Adj[u]$
8. **do** RELAX(uv,w)
9. END

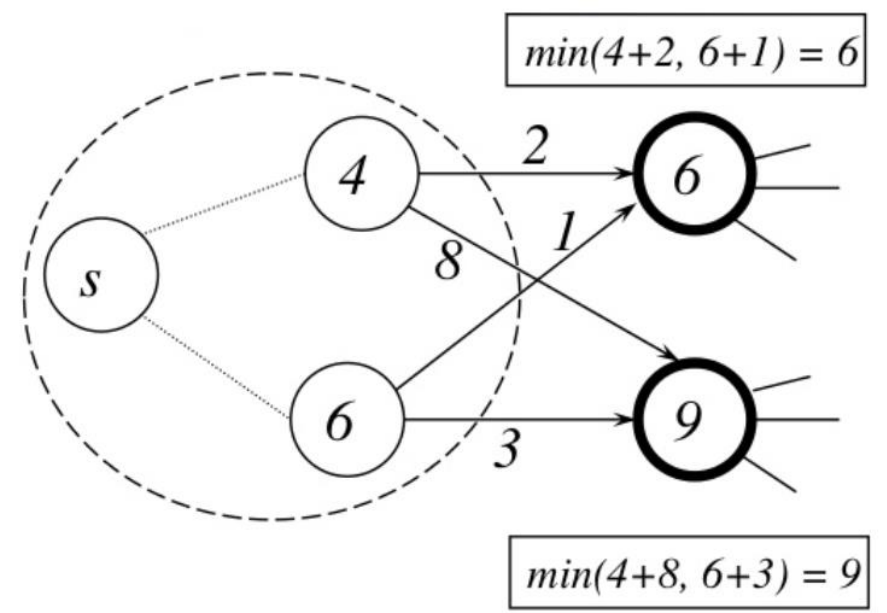
- Algoritmas pasiūlytas 1956 metais ir publikuotas 1959 metais olando W. Dijkstra.
- Algoritmo vykdymo eigoje formuojamas viršūnių aibės poaibis S.
- Iš likusių viršūnių sudaroma prioritetinė eilė Q pagal d[u] didėjimo tvarką.

Dijkstros algoritmo sudėtingumas

- Iš vienmačio masyvo Q = V – S minimalus elementas išrenkamas per O(|V|) žingsnių.
- Šį išrinkimą galimą pagreitinti masyvą Q suvedus į dvejetainį paieškos medį, tada mažiausio elemento paieška užtruktų O(log |V|) žingsnių.
- Išrinkimų skaičius lygus O(|V|) .
- Gretimumo sąrašas nuskaitomas O(|E|) kartų.
- Išvada: Dijkstros algoritmo sudėtingumas asimptotiškai lygus

$$O((|V| + |E|)\log |V|).$$

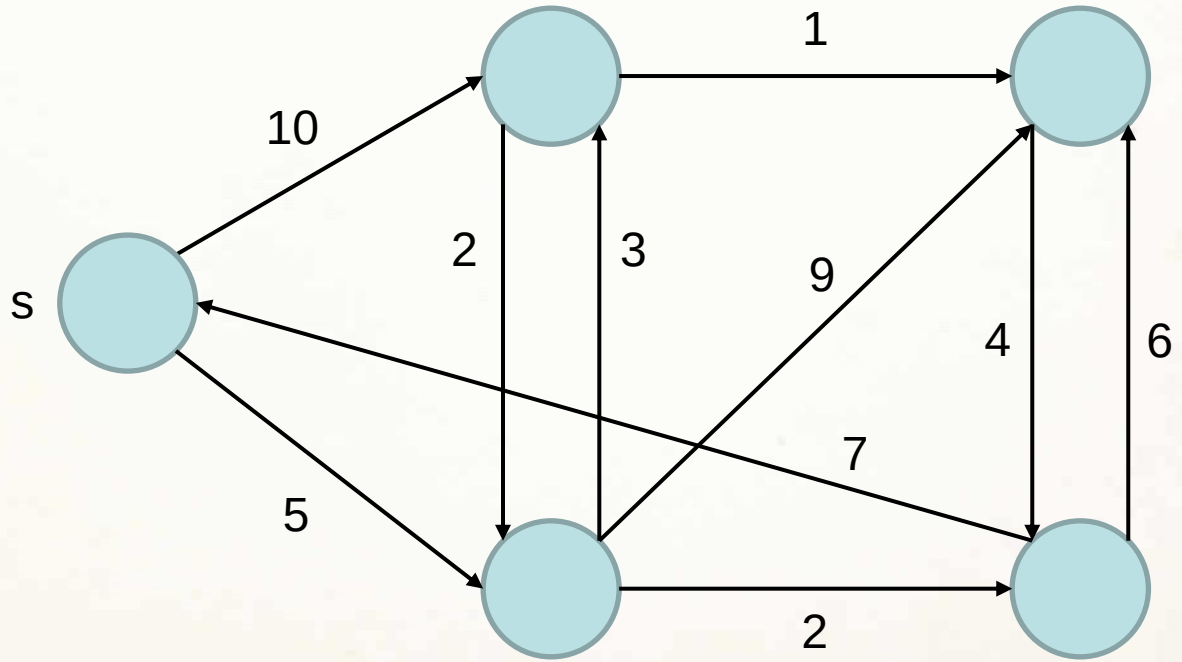
Dijsktros algoritmo savybė (teorema)



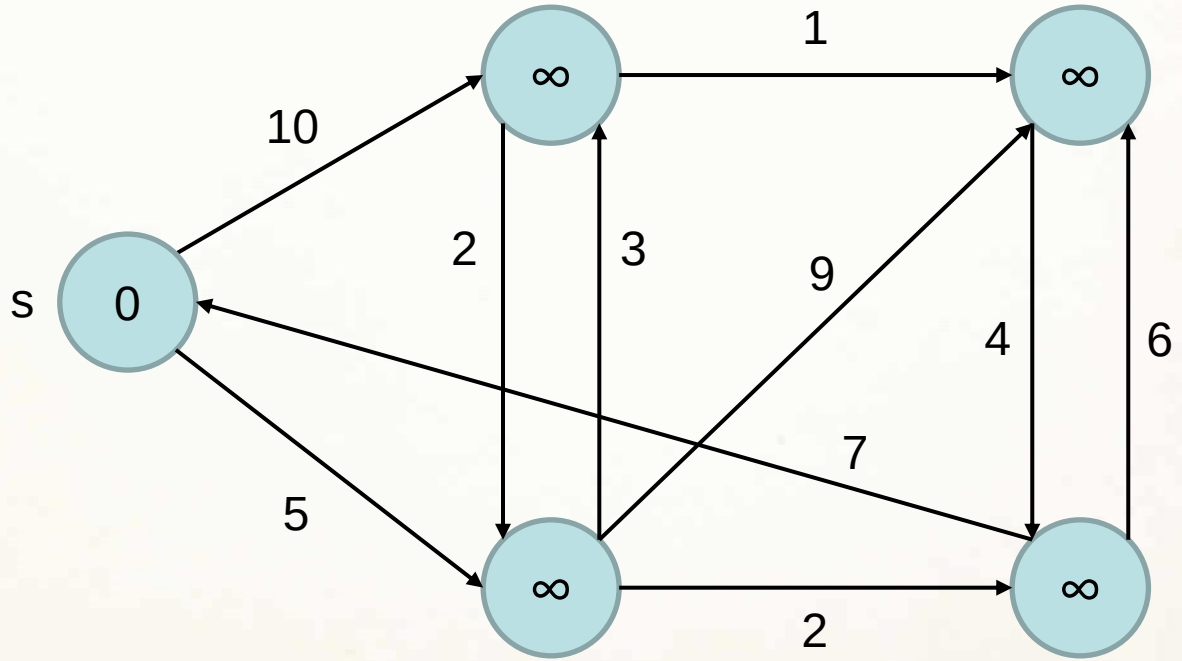
- Tegu kontūru pažymėtoje srityje yra žinomi trumpiausi atstumai nuo šaltinio viršūnės s.
- Atnaujinus gretimų (bet ne kontūru pažymėtoje srityje priklausančių) viršūnių atributus d[u], bus teisinga lygybė

$$d[u] = \delta(s,u).$$

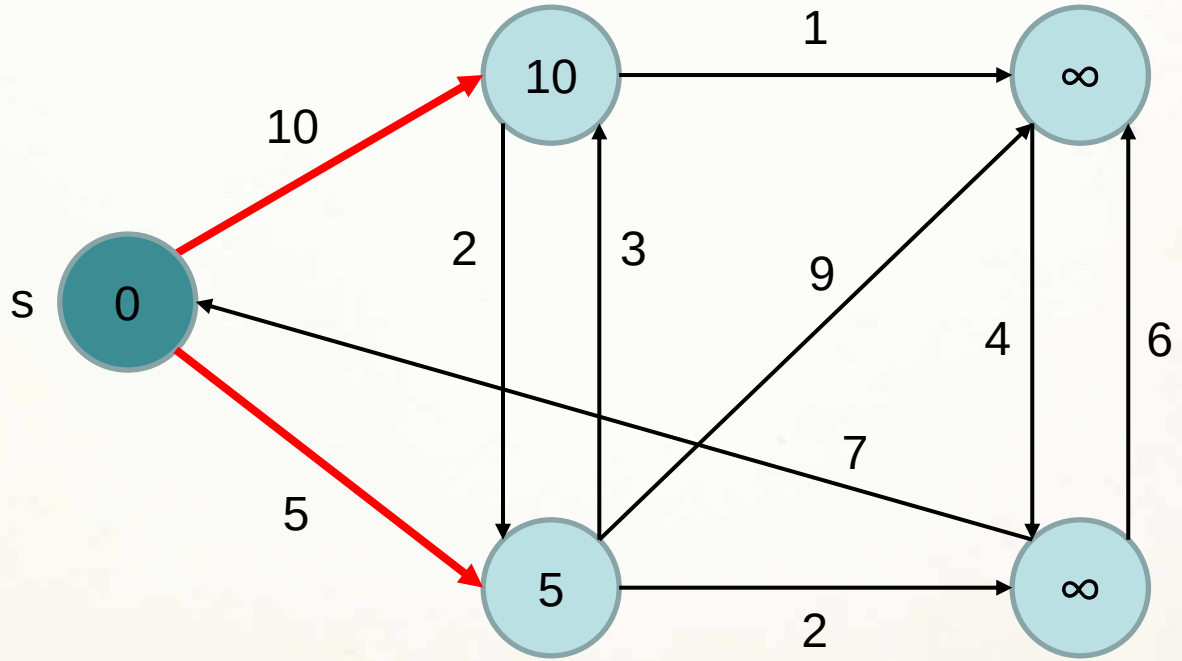
Dijkstros algoritmo pavyzdys



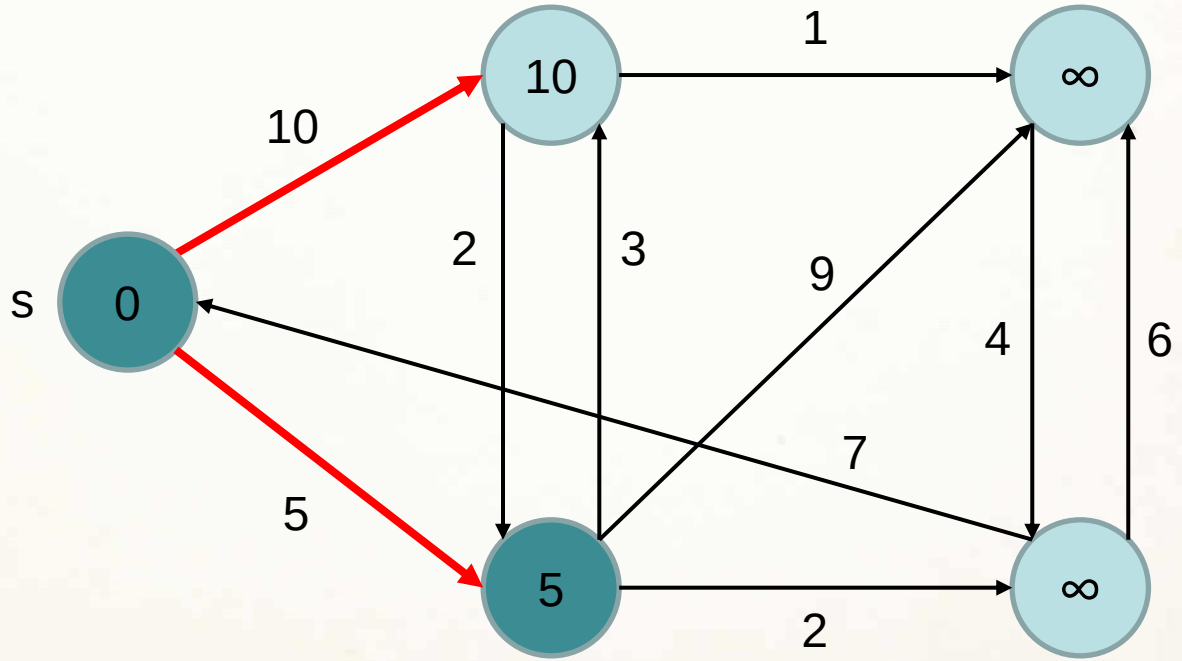
Dijkstros algoritmo pavyzdys



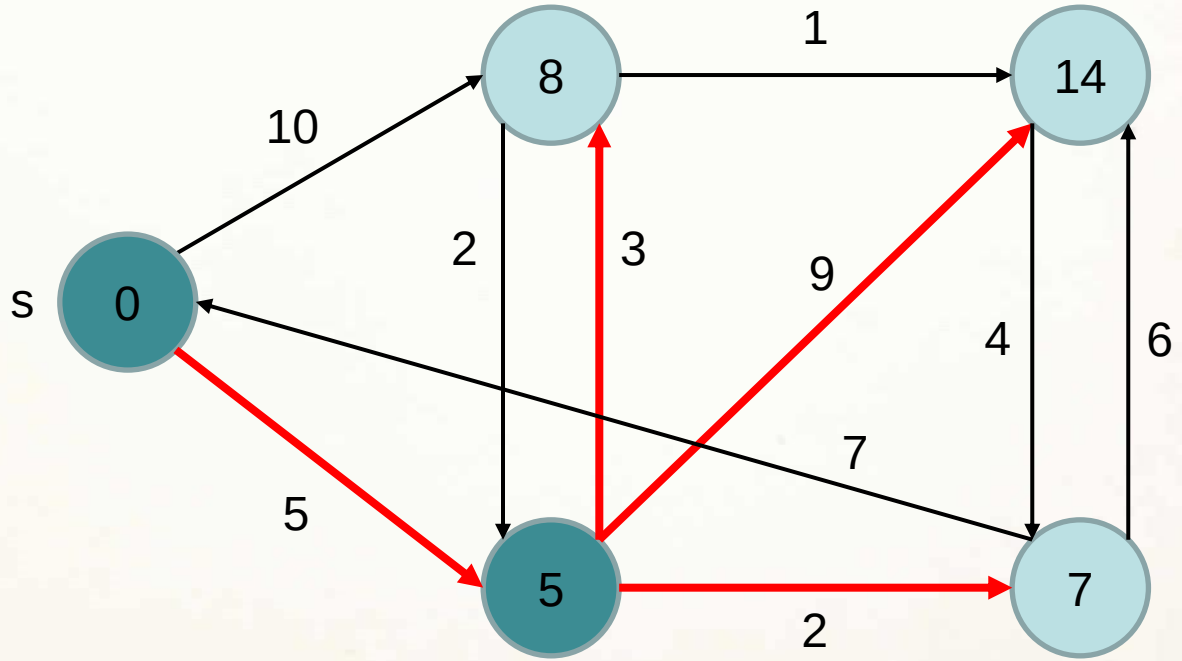
Dijkstros algoritmo pavyzdys



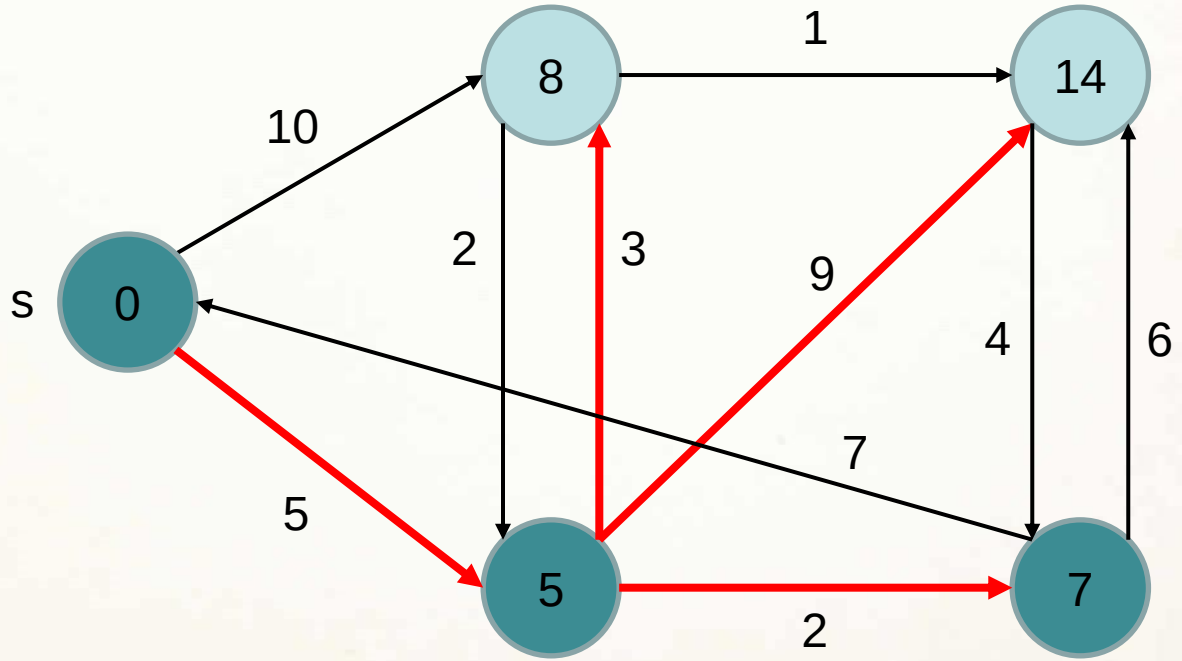
Dijkstros algoritmo pavyzdys



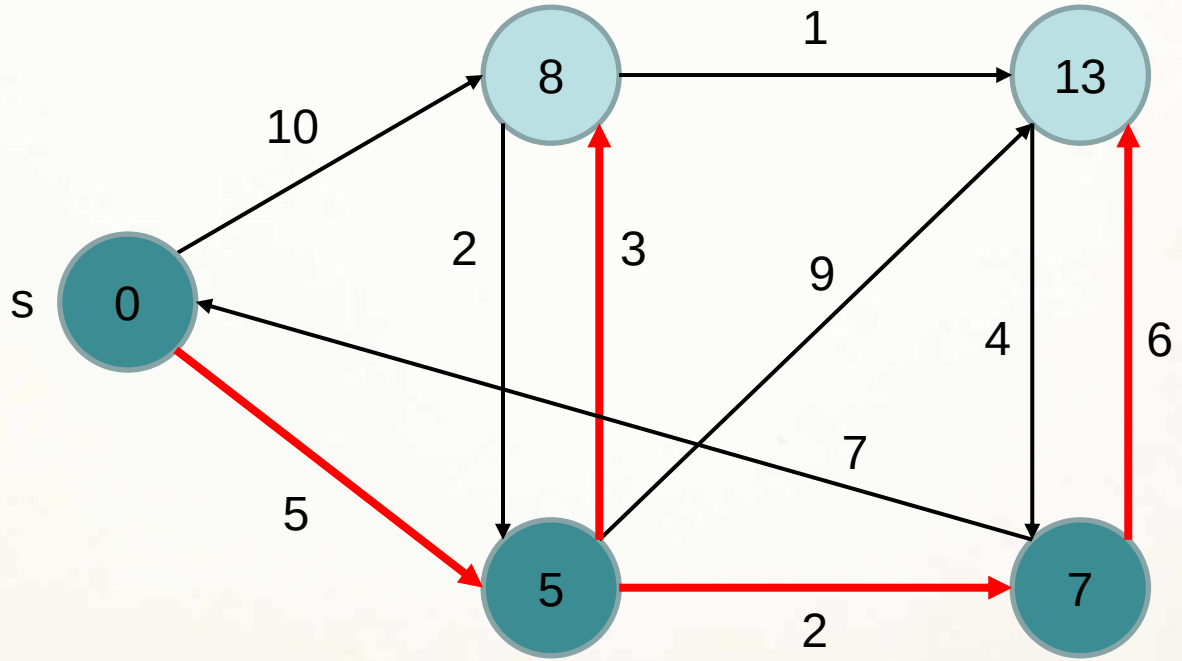
Dijkstros algoritmo pavyzdys



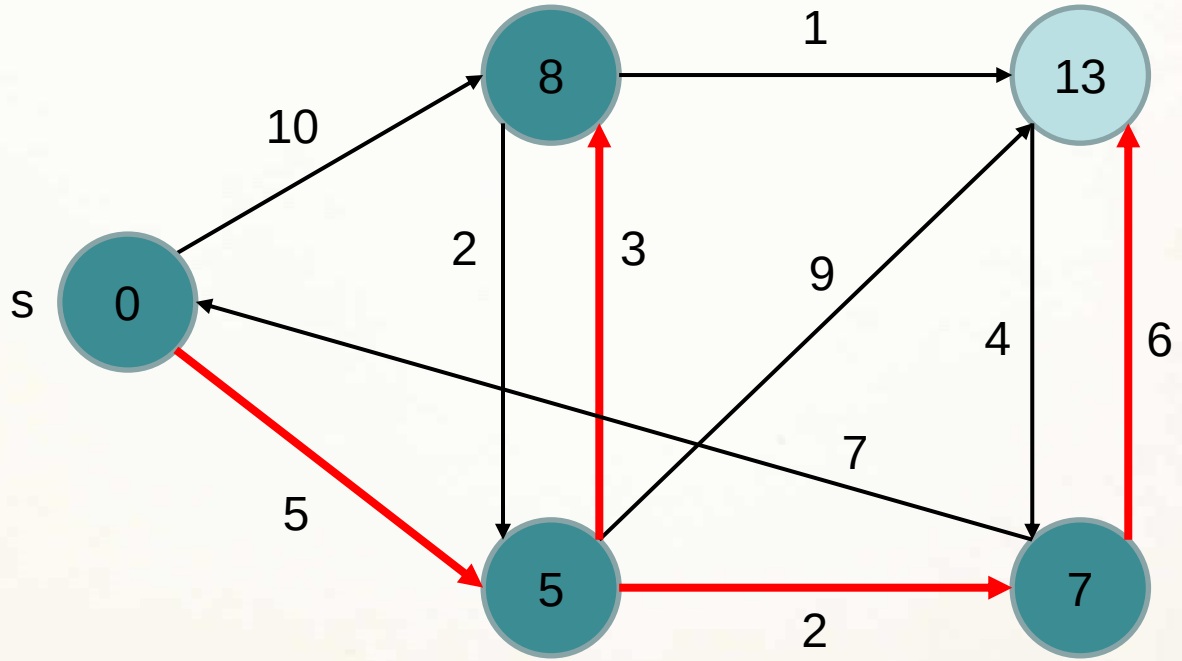
Dijkstros algoritmo pavyzdys



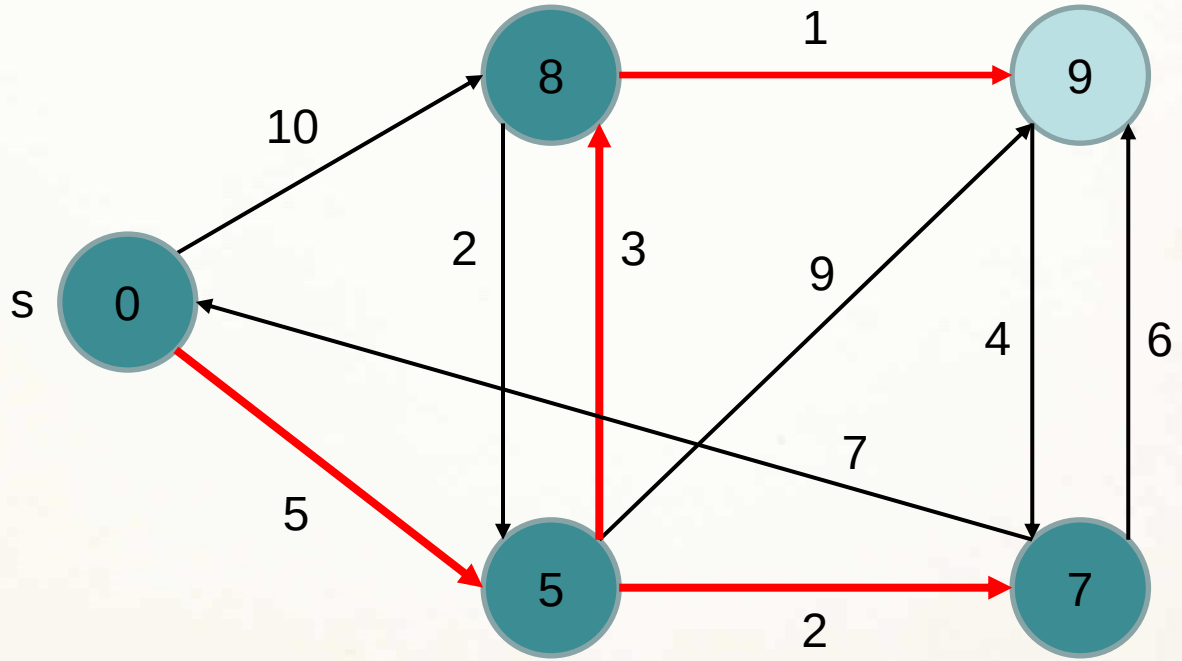
Dijkstros algoritmo pavyzdys



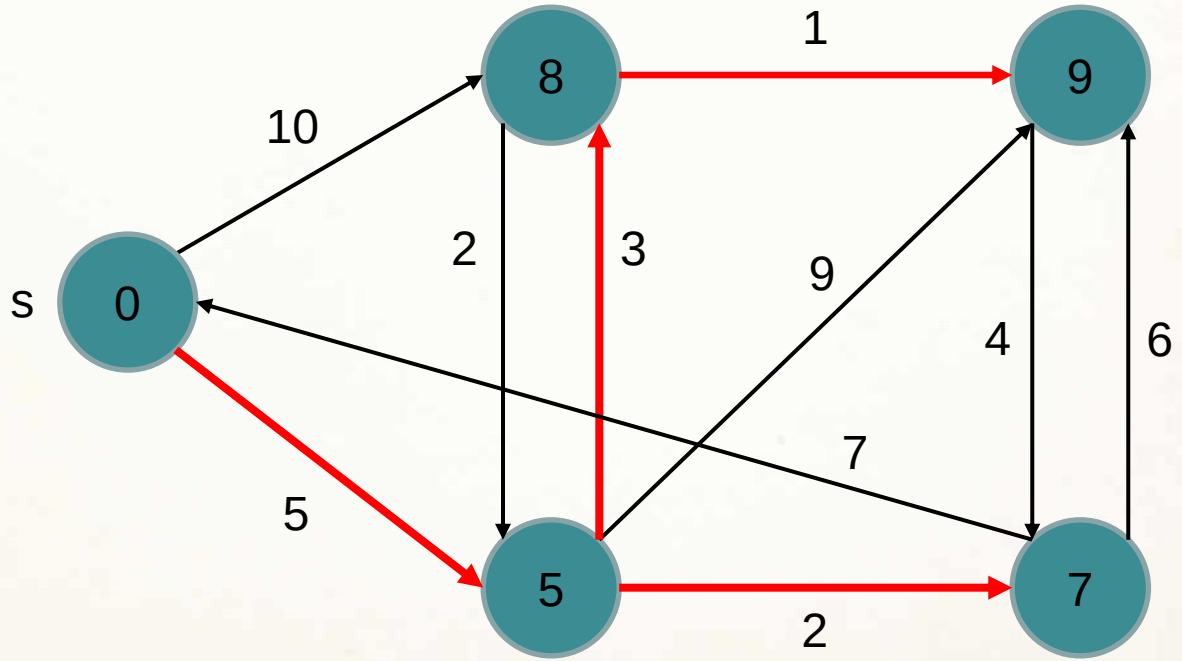
Dijkstros algoritmo pavyzdys



Dijkstros algoritmo pavyzdys

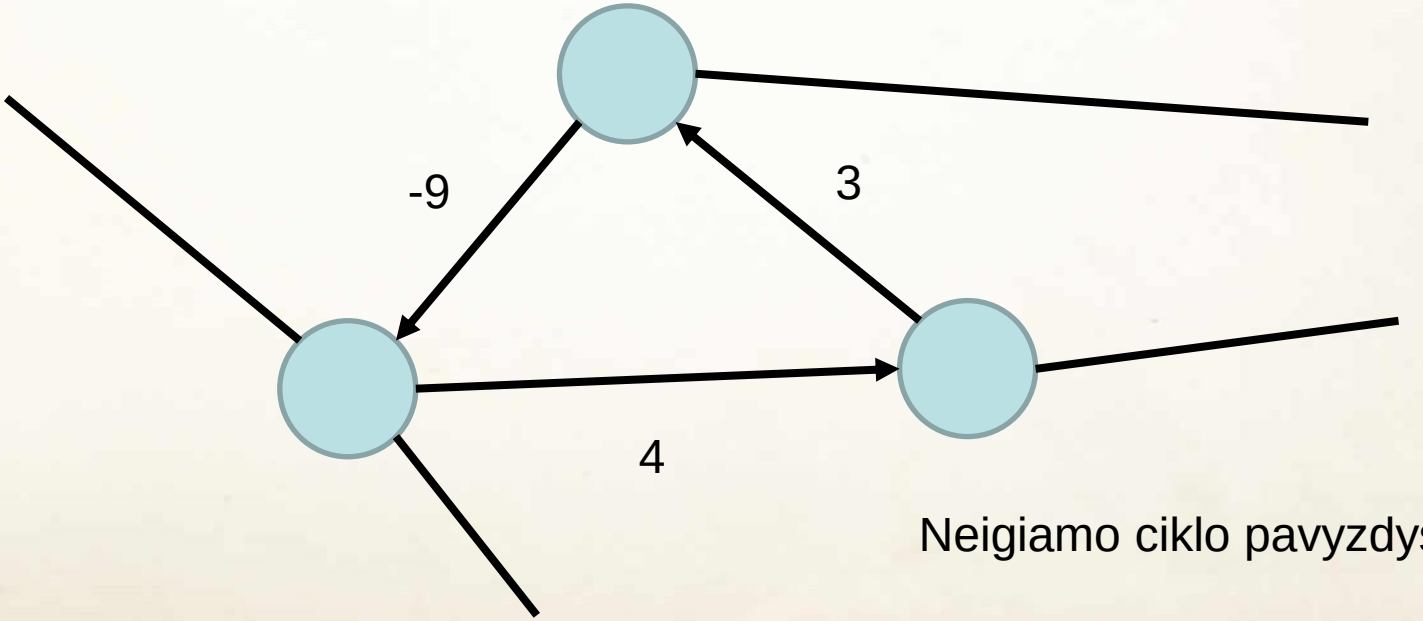


Dijkstros algoritmo pavyzdys



Belmano–Fordo algoritmas

- Tinkamas svoriniam (di)grafui su neigiamais briaunų svoriais (skirtingai nei Dijkstros algoritmas).
- Aptinka neigiamus ciklus.
- Lėtesnis už Dijkstros algoritmą.



Neigiamo ciklo pavyzdys.

Belmano–Fordo algoritmo idėja (1)

- Kaskart atnaujinti atributą d visoms viršūnių poroms, kurias jungia briauna.
- **Teorema.** Jei viršūnes u ir v jungia briauna (digrafo atveju kryptis iš u į v), $s \Rightarrow u \rightarrow v$ yra trumpiausias takas ir $d[u] = \delta(s,u)$, tada $d[u] + w(uv)$, yra trumpiausias atstumas nuo s iki v.
- **Įrodymas.** Kadangi $s \Rightarrow u \rightarrow v$ yra trumpiausias takas, jo ilgis lygus $\delta(s,u) + w(uv) = d[u] + w(uv)$.

Belmano–Fordo algoritmo idėja (2)

- Iš pradžių apskaičiuojami atributai δ(s,u) visoms viršūnėms, kurios gretimos šaltinio viršūnei s.
- Po to apskaičiuojami d[u] = δ(s,u) atributai viršūnėms, kurios nutolusios 2 briaunų atstumu nuo šaltinio viršūnės s ir t. t.
- Kadangi joks trumpiausias takas negali turėti daugiau nei |V|-1 briauną, po baigtinio žingsnių skaičiaus d atributai bus teisingi.
- Pastaba. Nepasiekiamų viršūnių iš s atributas d lygus begalybei.

Belmano–Fordo algoritmas

```
BELLMAN-FORD(G, w, s)
1  for each  v ∈V  do
2      d[v] ← ∞
3      π[v] ← NIL
4  d[s] ← 0
5  for i ← 1 to |V|-1 do
6      for each  uv ∈ E  do
7          RELAX(uv, w)
8  for each  uv ∈ E  do
9      if d[v] > d[u] + w(uv) then
10         return FALSE
11 return TRUE
```

Algoritmo sudėtingumas O(VE).

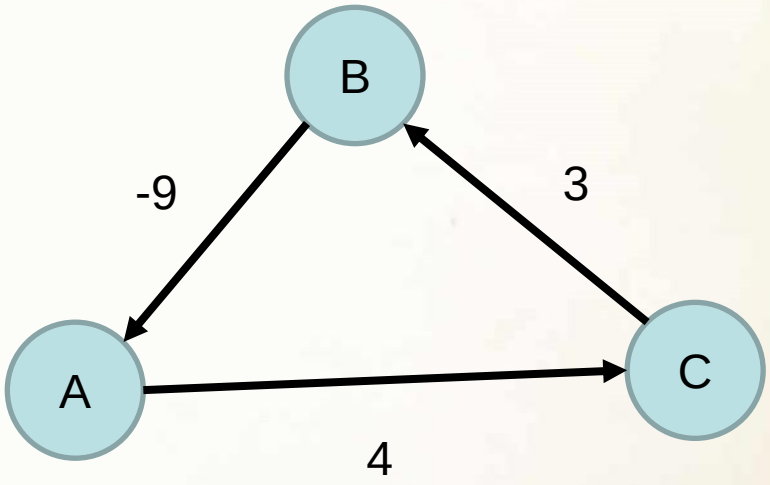
Neigiamų ciklų aptikimas

Tarkime lygybės teisingos:

$d[A] \le d[B] - 9$,

$d[B] \le d[C] + 3$,

$d[C] \le d[A] + 4$.



Jas sudėjus gaunama prieštara:

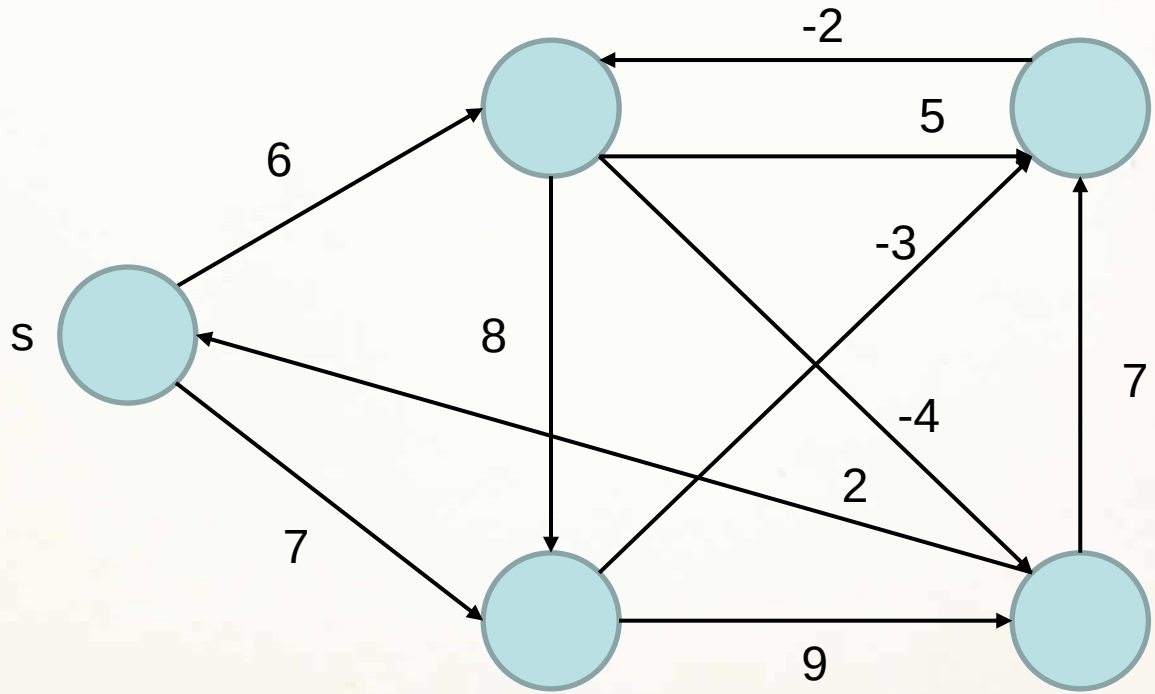
$0 \le -2$.

Vadinasi, bent vienai briaunai uv galios teisinga nelygybė

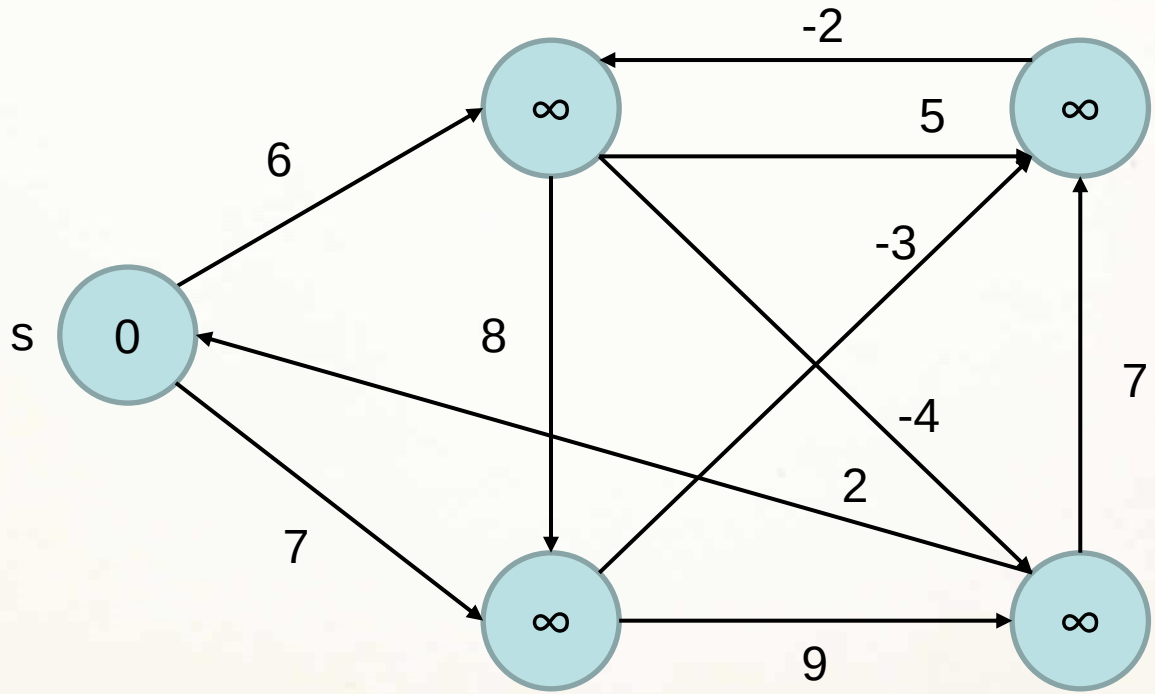
$$d[v] > d[u] + w(uv).$$

8–10 Belmano–Fordo algoritmas eilutėse tai yra tikrinama.

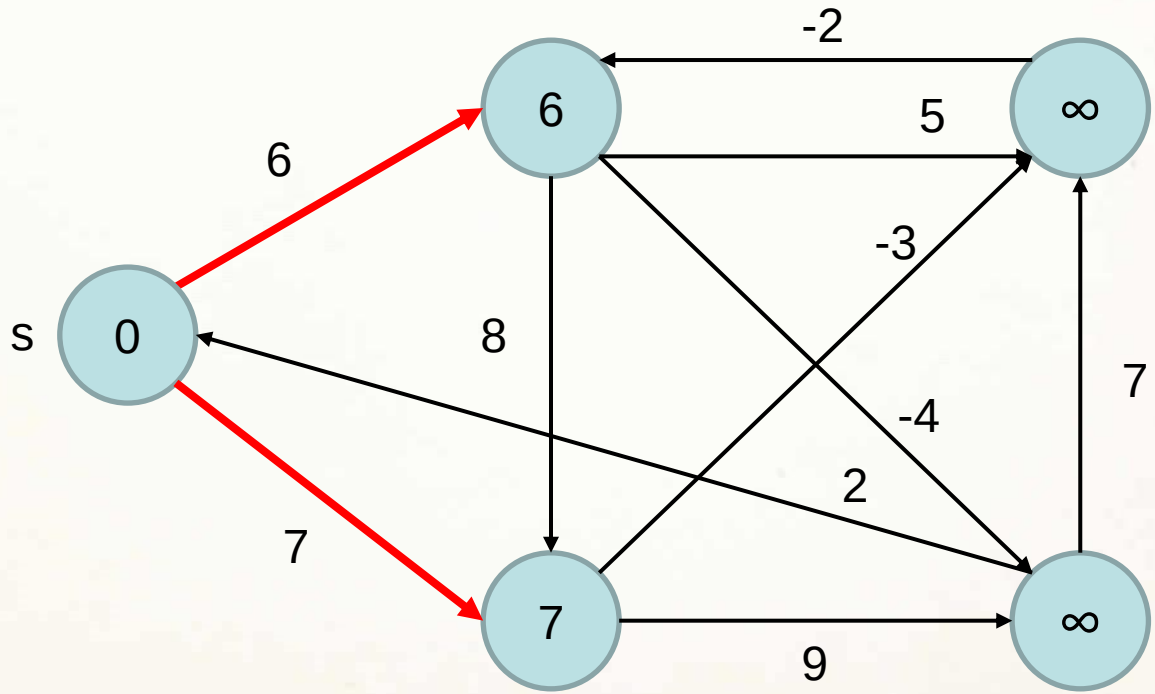
Belmano–Fordo algoritmo pavyzdys



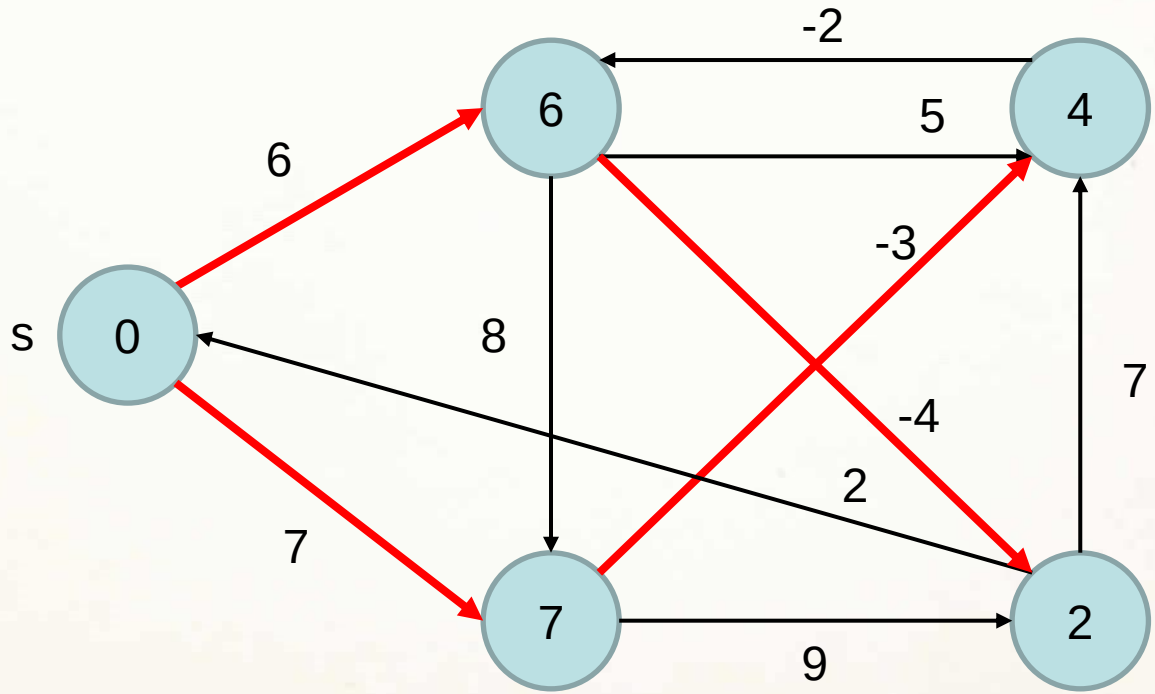
Belmano–Fordo algoritmo pavyzdys



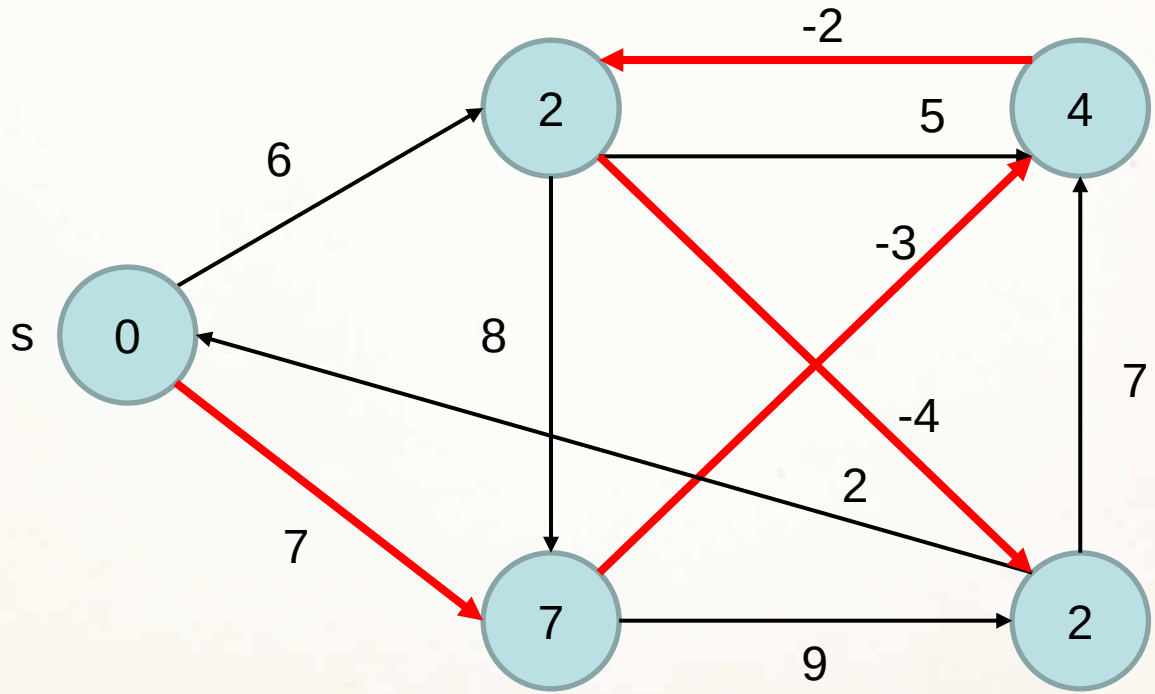
Belmano–Fordo algoritmo pavyzdys



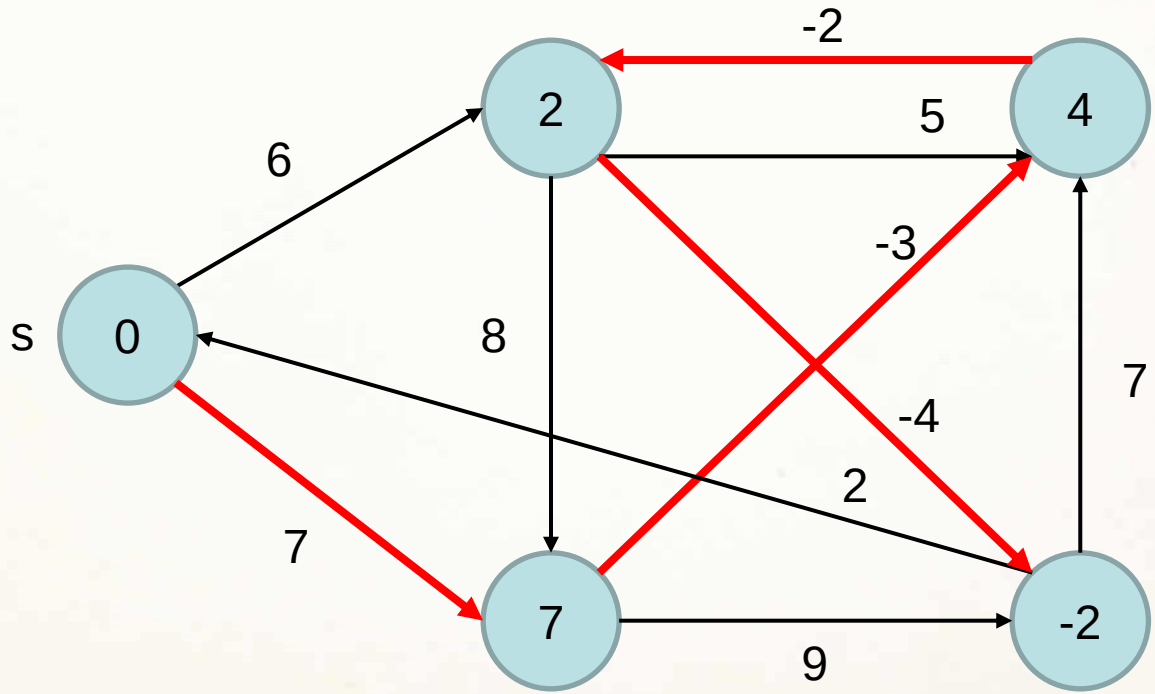
Belmano–Fordo algoritmo pavyzdys



Belmano–Fordo algoritmo pavyzdys



Belmano–Fordo algoritmo pavyzdys



Trumpiausi takai tarp visų viršūnių

- **Užduotis.** Svoriniame grafe ar digrafe reikia rasti trumpiausius takus tarp visų viršūnių porų.

- **Pastaba.** Bendruoju atveju (kai leidžiami neigiami svoriai), tai galima padaryti taikant Belmano–Fordo algoritmą |V| kartų, todėl algoritmo sudėtingumas tampa $O(V^2E)$. Jei grafas G tankus (turintis daug briaunų), trumpiausių takų paieškos algoritmo sudėtingumas artėja į $O(V^4)$.

Trumpiausi takai tarp visų viršūnių

- Pažymėkime mažiausią svorį d_{ij}^m tako (iš viršūnės *i* į viršūnę *j*), kuriame yra ne daugiau *m* briaunų:

$$d_{ij}^0 = \begin{cases} 0, \text{ jei } i = j, \\ \infty, \text{ jei } i \neq j. \end{cases}$$

- Apskaičiavus d_{ij}^{m-1}, teisingos lygybės:

$$\begin{aligned} d_{ij}^m &= \min\left\{d_{ij}^{m-1}, \min_{1\leq k\leq n}(d_{ik}^{m-1} + w_{kj})\right\} \\ &= \min_{1\leq k\leq n}(d_{ik}^{m-1} + w_{kj})\Big\}. \end{aligned}$$

- Tikslas – apskaičiuoti iteracijų seką (matricas D):

$$W = D^{(1)} \to D^{(2)} \to \cdots \to D^{(n-1)}.$$

Trumpiausi takai tarp visų viršūnių

- Pagrindinė algoritmo procedūra – iš dviejų žinomų matricų *D* ir *W* apskaičiavimas trečios matricos D' (algoritmas labai panašus matricų *D* ir *W* daugybai):

Tr-T (D, W):

1. $n \leftarrow$ eilutės[D]
2. Tegu $D' = (d'_{ij})$ – $n \times n$ matrica
3. **for** $i \leftarrow 1$ **to** n
4. **do for** $j \leftarrow 1$ **to** n
5. **do** $d'_{ij} \leftarrow \infty$
6. **for** $k \leftarrow 1$ **to** n
7. **do** $d'_{ij} \leftarrow \min\{d'_{ij}, d_{ik} + w_{kj}\}$
8. **return** D'

- Rezultatai gaunami tokia tvarka:

$$\textbf{Tr-T}(D_0, W) = D^{(1)},\ \textbf{Tr-T}(D^{(1)}, W) = D^{(2)},\ \textbf{Tr-T}(D^{(k)}, W) = D^{(k+1)}, \ldots$$

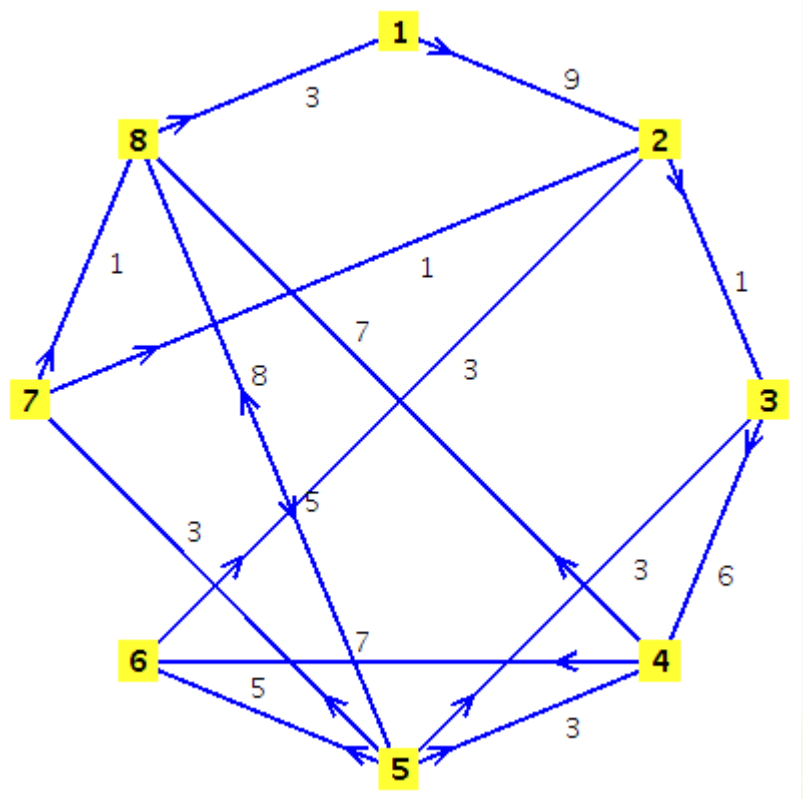
Lėtas trumpiausių takų paieškos algoritmas

Lėtas Tr-T(W):

1. $n \leftarrow$ eilutės[W]
2. $D^{(1)} \leftarrow W$
3. **for** $m \leftarrow 2$ **to** $n-1$
4. **do** $D^{(m)} \leftarrow \mathbf{Tr-T}(D^{(m-1)}, W)$
5. **return** $D^{(n-1)}$

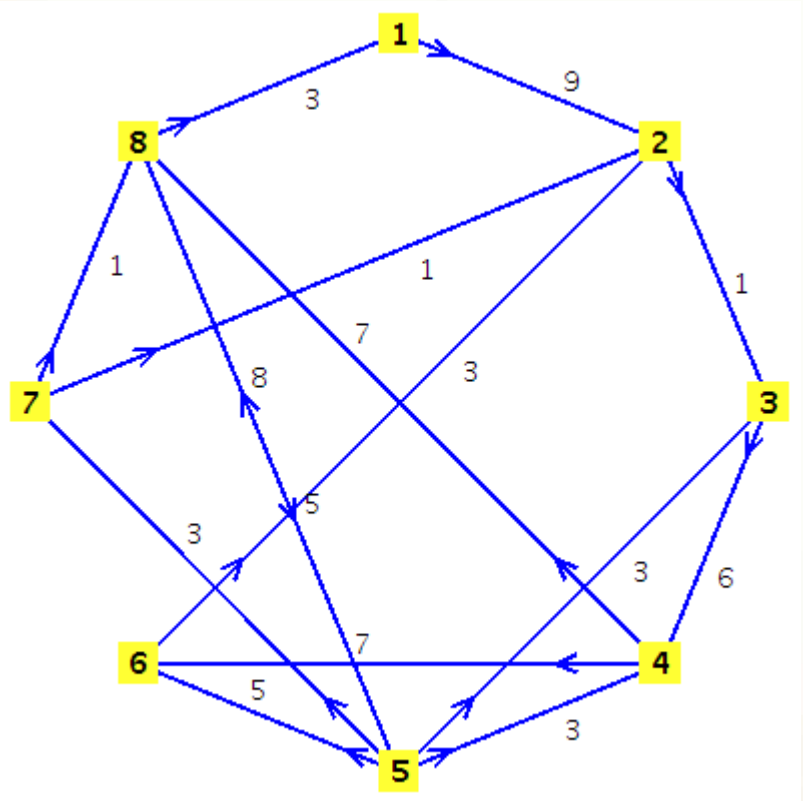
- Algoritmo sudėtingumas $O(n^4)$. Kodėl?

Lėtas trumpiausių takų paieškos algoritmas



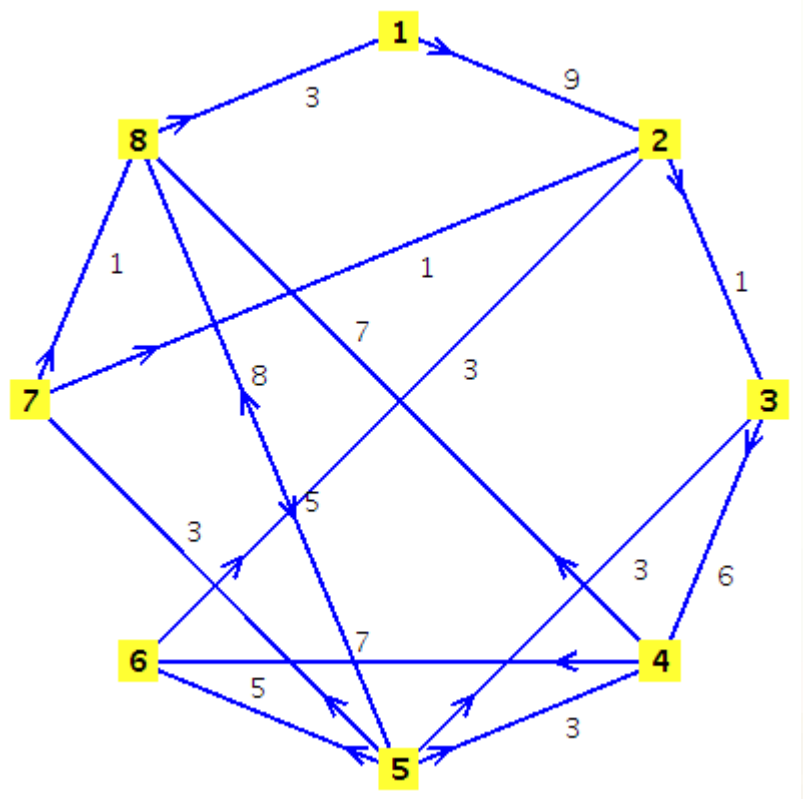
$$W := \begin{bmatrix} 0 & 9 & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 1 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 6 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & 7 & \infty & 7 \\ \infty & \infty & 3 & 3 & 0 & 5 & 3 & 8 \\ \infty & 3 & \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & 1 & \infty & \infty & \infty & \infty & 0 & 1 \\ 3 & \infty & \infty & \infty & 5 & \infty & \infty & 0 \end{bmatrix}$$

Lėtas trumpiausių takų paieškos algoritmas



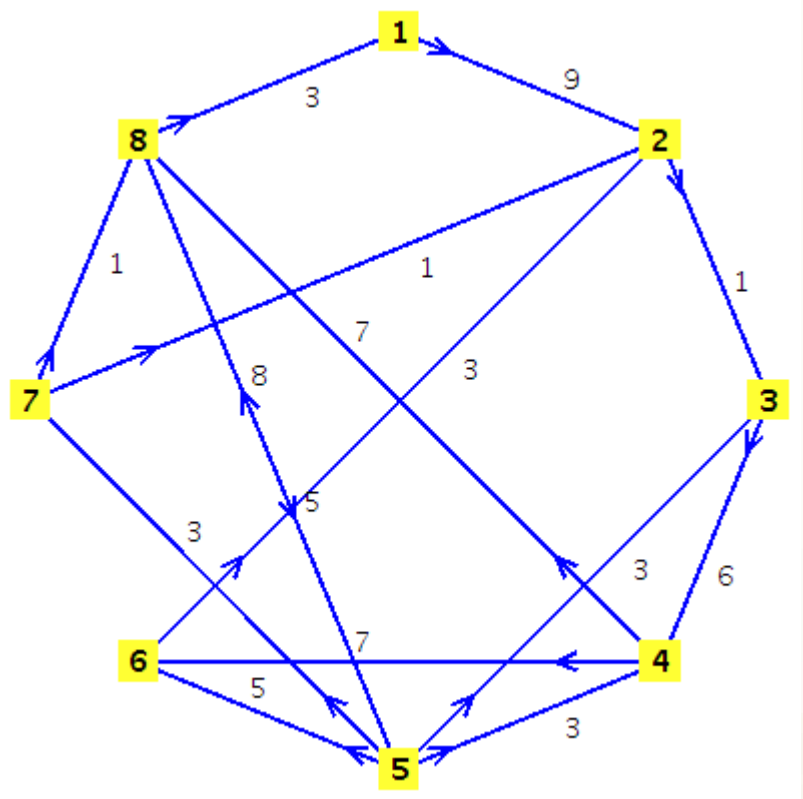
$$D_1 := \begin{bmatrix} 0 & 9 & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 1 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 6 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & 7 & \infty & 7 \\ \infty & \infty & 3 & 3 & 0 & 5 & 3 & 8 \\ \infty & 3 & \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & 1 & \infty & \infty & \infty & \infty & 0 & 1 \\ 3 & \infty & \infty & \infty & 5 & \infty & \infty & 0 \end{bmatrix}$$

Lėtas trumpiausių takų paieškos algoritmas



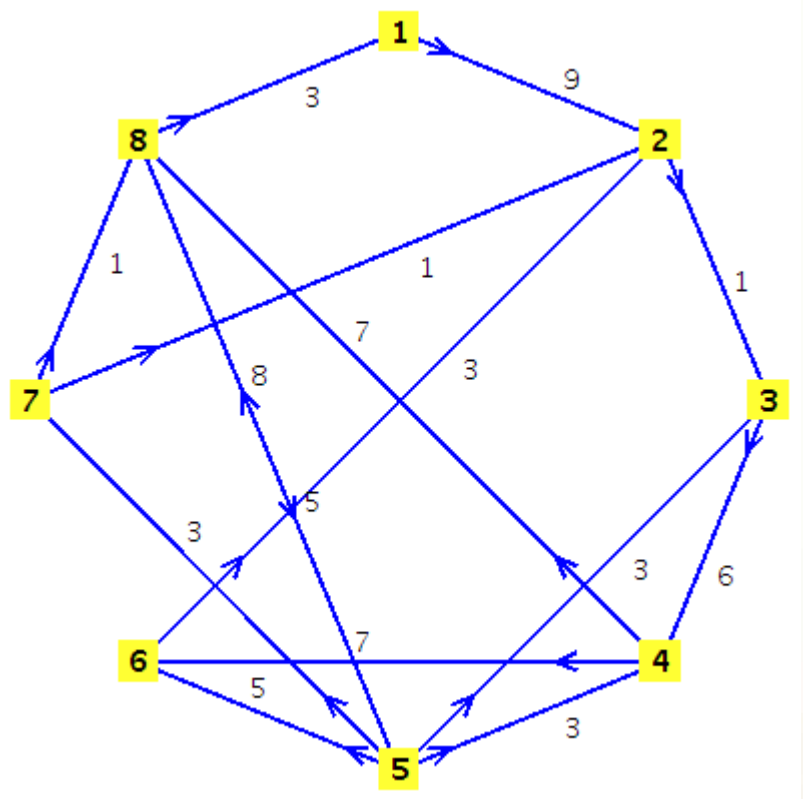
$$D_2 := \begin{bmatrix} 0 & 9 & 10 & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 1 & 7 & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 6 & \infty & 13 & \infty & 13 \\ 10 & 10 & \infty & 0 & 12 & 7 & \infty & 7 \\ 11 & 4 & 3 & 3 & 0 & 5 & 3 & 4 \\ \infty & 3 & 4 & \infty & \infty & 0 & \infty & \infty \\ 4 & 1 & 2 & \infty & 6 & \infty & 0 & 1 \\ 3 & 12 & 8 & 8 & 5 & 10 & 8 & 0 \end{bmatrix}$$

Lėtas trumpiausių takų paieškos algoritmas



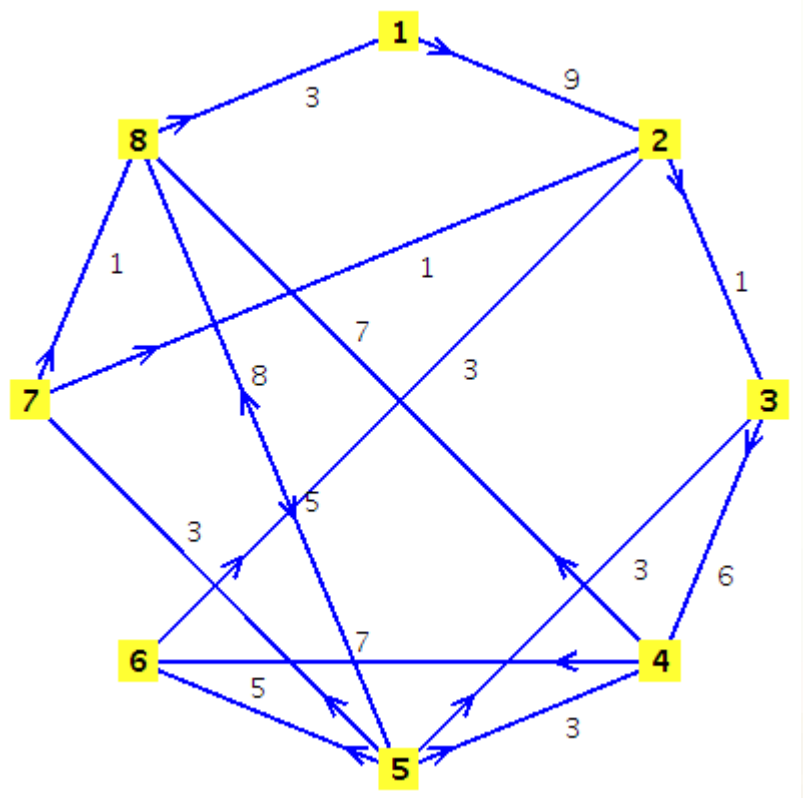
$$D_3 := \begin{bmatrix} 0 & 9 & 10 & 16 & \infty & \infty & \infty & \infty \\ \infty & 0 & 1 & 7 & \infty & 14 & \infty & 14 \\ 16 & 16 & 0 & 6 & 18 & 13 & \infty & 13 \\ 10 & 10 & 11 & 0 & 12 & 7 & 15 & 7 \\ 7 & 4 & 3 & 3 & 0 & 5 & 3 & 4 \\ \infty & 3 & 4 & 10 & \infty & 0 & \infty & \infty \\ 4 & 1 & 2 & 8 & 6 & 11 & 0 & 1 \\ 3 & 9 & 8 & 8 & 5 & 10 & 8 & 0 \end{bmatrix}$$

Lėtas trumpiausių takų paieškos algoritmas



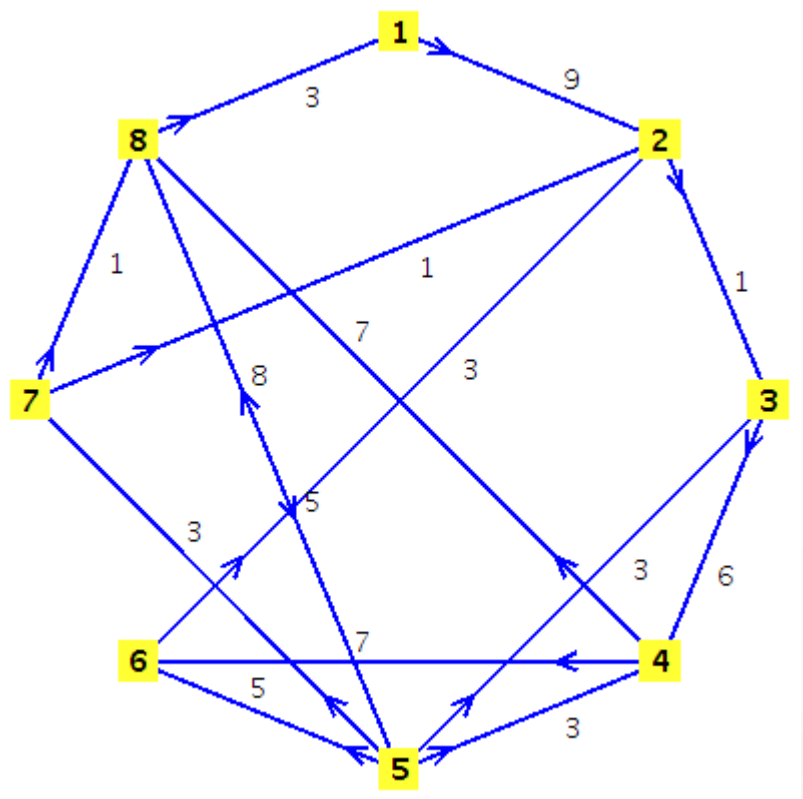
$$D_4 := \begin{bmatrix} 0 & 9 & 10 & 16 & \infty & 23 & \infty & 23 \\ 17 & 0 & 1 & 7 & 19 & 14 & \infty & 14 \\ 16 & 16 & 0 & 6 & 18 & 13 & 21 & 13 \\ 10 & 10 & 11 & 0 & 12 & 7 & 15 & 7 \\ 7 & 4 & 3 & 3 & 0 & 5 & 3 & 4 \\ \infty & 3 & 4 & 10 & \infty & 0 & \infty & 17 \\ 4 & 1 & 2 & 8 & 6 & 11 & 0 & 1 \\ 3 & 9 & 8 & 8 & 5 & 10 & 8 & 0 \end{bmatrix}$$

Lėtas trumpiausių takų paieškos algoritmas



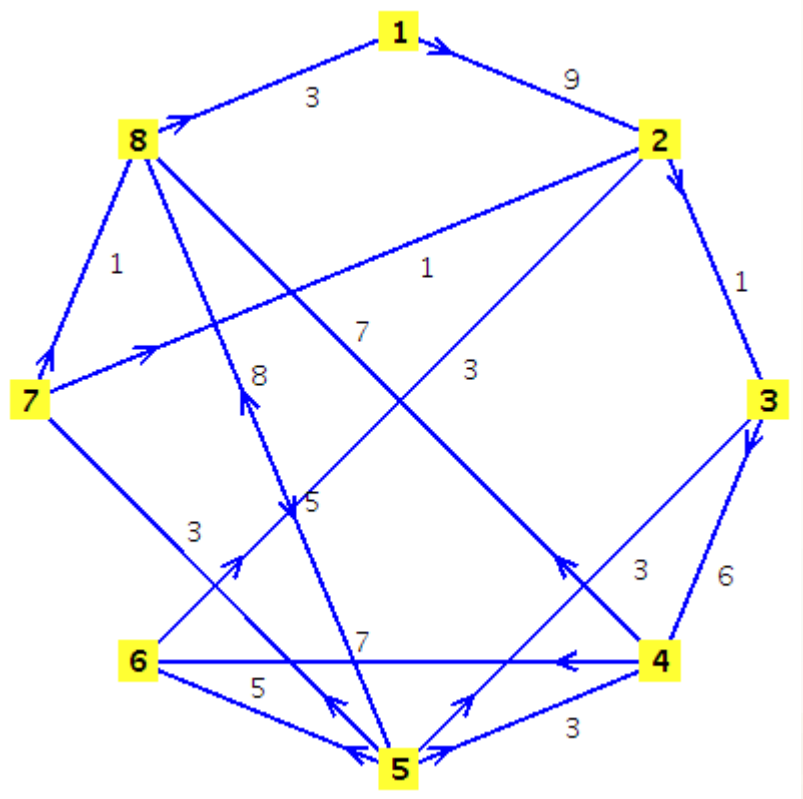
$$D_5 := \begin{bmatrix} 0 & 9 & 10 & 16 & 28 & 23 & \infty & 23 \\ 17 & 0 & 1 & 7 & 19 & 14 & 22 & 14 \\ 16 & 16 & 0 & 6 & 18 & 13 & 21 & 13 \\ 10 & 10 & 11 & 0 & 12 & 7 & 15 & 7 \\ 7 & 4 & 3 & 3 & 0 & 5 & 3 & 4 \\ 20 & 3 & 4 & 10 & 22 & 0 & \infty & 17 \\ 4 & 1 & 2 & 8 & 6 & 11 & 0 & 1 \\ 3 & 9 & 8 & 8 & 5 & 10 & 8 & 0 \end{bmatrix}$$

Lėtas trumpiausių takų paieškos algoritmas



$$D_6 := \begin{bmatrix} 0 & 9 & 10 & 16 & 28 & 23 & 31 & 23 \\ 17 & 0 & 1 & 7 & 19 & 14 & 22 & 14 \\ 16 & 16 & 0 & 6 & 18 & 13 & 21 & 13 \\ 10 & 10 & 11 & 0 & 12 & 7 & 15 & 7 \\ 7 & 4 & 3 & 3 & 0 & 5 & 3 & 4 \\ 20 & 3 & 4 & 10 & 22 & 0 & 25 & 17 \\ 4 & 1 & 2 & 8 & 6 & 11 & 0 & 1 \\ 3 & 9 & 8 & 8 & 5 & 10 & 8 & 0 \end{bmatrix}$$

Lėtas trumpiausių takų paieškos algoritmas



$$D_7 := \begin{bmatrix} 0 & 9 & 10 & 16 & 28 & 23 & 31 & 23 \\ 17 & 0 & 1 & 7 & 19 & 14 & 22 & 14 \\ 16 & 16 & 0 & 6 & 18 & 13 & 21 & 13 \\ 10 & 10 & 11 & 0 & 12 & 7 & 15 & 7 \\ 7 & 4 & 3 & 3 & 0 & 5 & 3 & 4 \\ 20 & 3 & 4 & 10 & 22 & 0 & 25 & 17 \\ 4 & 1 & 2 & 8 & 6 & 11 & 0 & 1 \\ 3 & 9 & 8 & 8 & 5 & 10 & 8 & 0 \end{bmatrix}$$

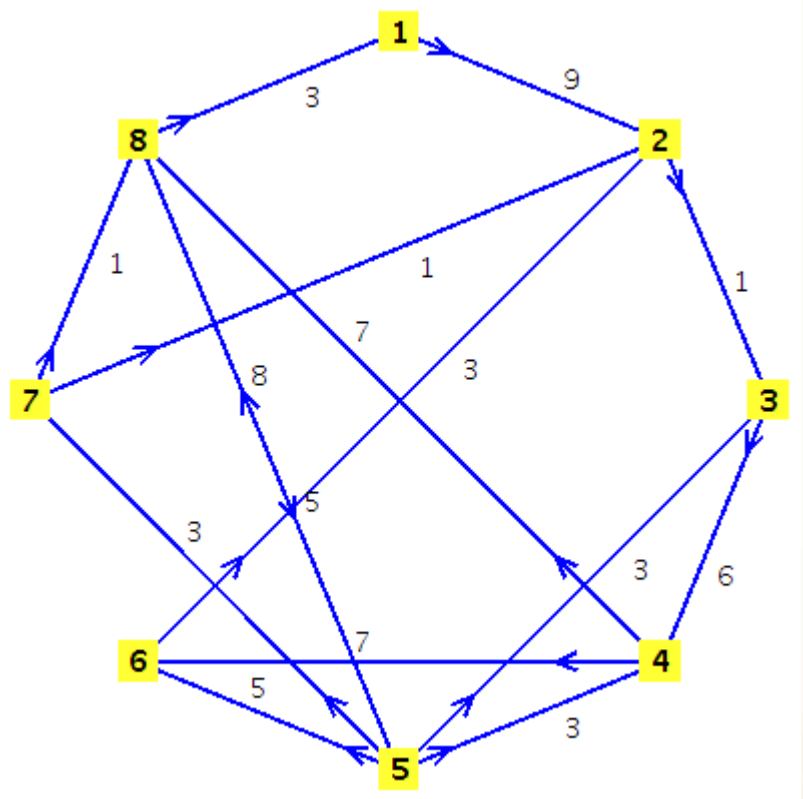
Greitas trumpiausių takų paieškos algoritmas

Greitas Tr-T(W):

1. $n \leftarrow$ eilutės[W]
2. $D^{(1)} \leftarrow W$
3. $m \leftarrow 1$
4. **while** $n - 1 > m$
5. **do** $D^{(2m)} \leftarrow \mathbf{Tr} - \mathbf{T}(D^{(m)}, D^{(m)})$
6. $m \leftarrow 2m$
7. **return** $D^{(m)}$

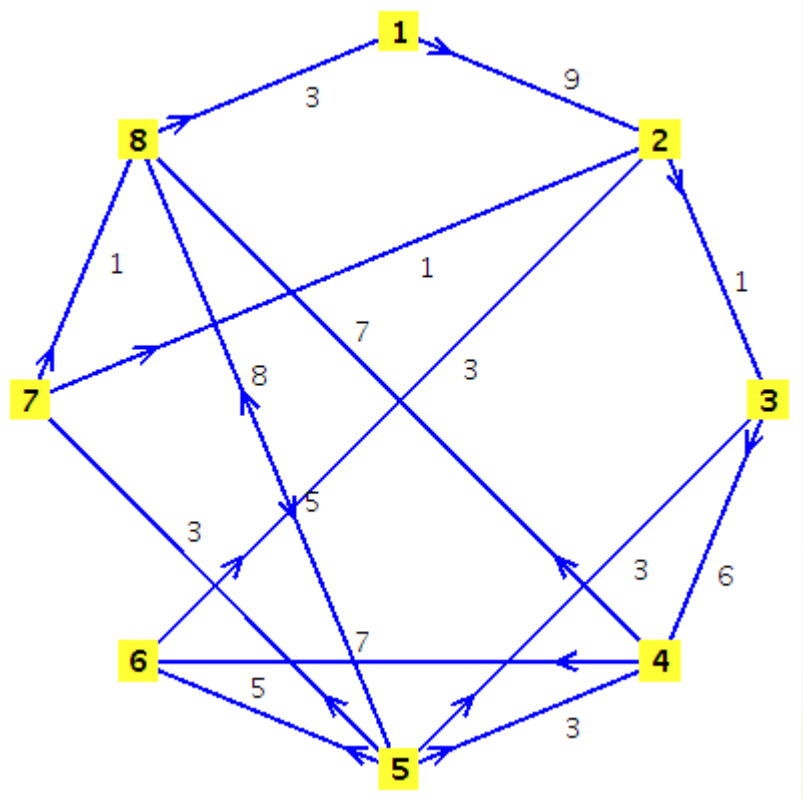
- Algoritmo sudėtingumas O(n^3 log n). Kodėl?

Greitas trumpiausių takų paieškos algoritmas



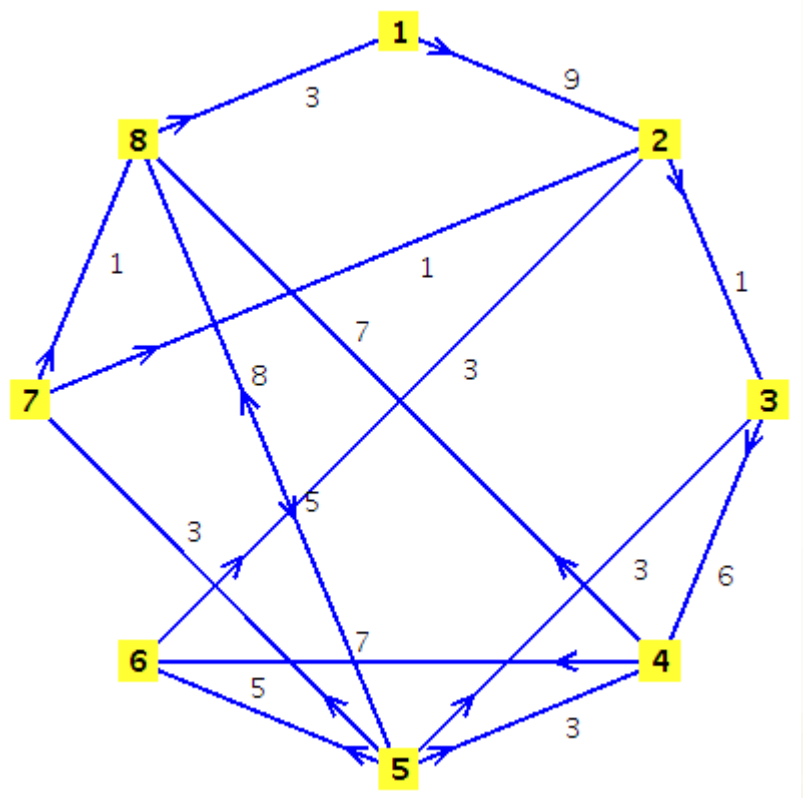
$$W := \begin{bmatrix} 0 & 9 & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 1 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 6 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & 7 & \infty & 7 \\ \infty & \infty & 3 & 3 & 0 & 5 & 3 & 8 \\ \infty & 3 & \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & 1 & \infty & \infty & \infty & \infty & 0 & 1 \\ 3 & \infty & \infty & \infty & 5 & \infty & \infty & 0 \end{bmatrix}$$

Greitas trumpiausių takų paieškos algoritmas



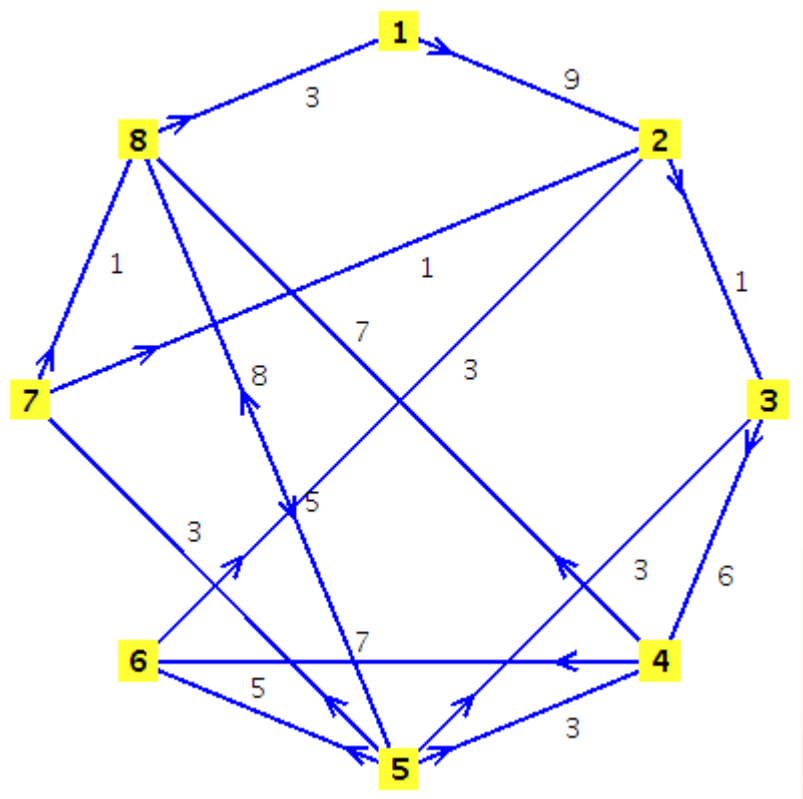
$$D_1 := \begin{bmatrix} 0 & 9 & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 1 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 6 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & 7 & \infty & 7 \\ \infty & \infty & 3 & 3 & 0 & 5 & 3 & 8 \\ \infty & 3 & \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & 1 & \infty & \infty & \infty & \infty & 0 & 1 \\ 3 & \infty & \infty & \infty & 5 & \infty & \infty & 0 \end{bmatrix}$$

Greitas trumpiausių takų paieškos algoritmas



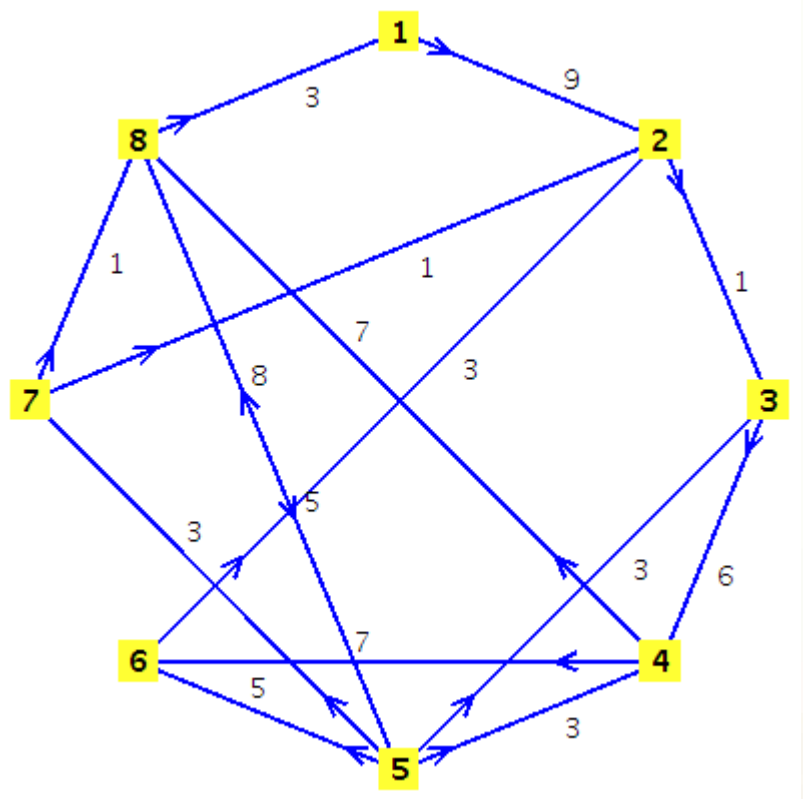
$$D_2 := \begin{bmatrix} 0 & 9 & 10 & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 1 & 7 & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 6 & \infty & 13 & \infty & 13 \\ 10 & 10 & \infty & 0 & 12 & 7 & \infty & 7 \\ 11 & 4 & 3 & 3 & 0 & 5 & 3 & 4 \\ \infty & 3 & 4 & \infty & \infty & 0 & \infty & \infty \\ 4 & 1 & 2 & \infty & 6 & \infty & 0 & 1 \\ 3 & 12 & 8 & 8 & 5 & 10 & 8 & 0 \end{bmatrix}$$

Greitas trumpiausių takų paieškos algoritmas



$$D_4 := \begin{bmatrix} 0 & 9 & 10 & 16 & \infty & 23 & \infty & 23 \\ 17 & 0 & 1 & 7 & 19 & 14 & \infty & 14 \\ 16 & 16 & 0 & 6 & 18 & 13 & 21 & 13 \\ 10 & 10 & 11 & 0 & 12 & 7 & 15 & 7 \\ 7 & 4 & 3 & 3 & 0 & 5 & 3 & 4 \\ \infty & 3 & 4 & 10 & \infty & 0 & \infty & 17 \\ 4 & 1 & 2 & 8 & 6 & 11 & 0 & 1 \\ 3 & 9 & 8 & 8 & 5 & 10 & 8 & 0 \end{bmatrix}$$

Greitas trumpiausių takų paieškos algoritmas



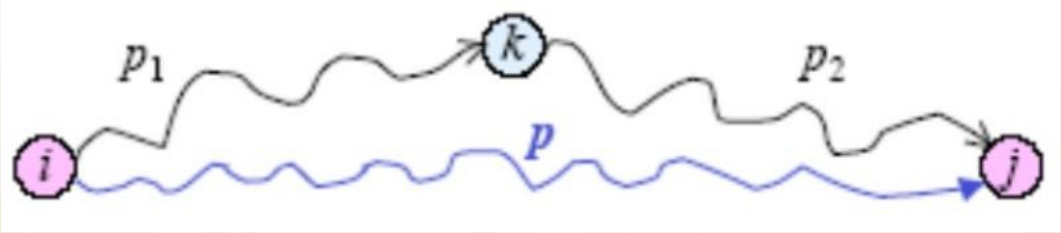
$$D_8 := \begin{bmatrix} 0 & 9 & 10 & 16 & 28 & 23 & 31 & 23 \\ 17 & 0 & 1 & 7 & 19 & 14 & 22 & 14 \\ 16 & 16 & 0 & 6 & 18 & 13 & 21 & 13 \\ 10 & 10 & 11 & 0 & 12 & 7 & 15 & 7 \\ 7 & 4 & 3 & 3 & 0 & 5 & 3 & 4 \\ 20 & 3 & 4 & 10 & 22 & 0 & 25 & 17 \\ 4 & 1 & 2 & 8 & 6 & 11 & 0 & 1 \\ 3 & 9 & 8 & 8 & 5 & 10 & 8 & 0 \end{bmatrix}$$

Floido–Varšalo algoritmo idėja

- Floido–Varšalo algoritmas išnaudoja kitokią tako struktūrą nei kiti trumpiausių takų paieškos algoritmai.
- Algoritmo vykdymo metu klasifikuojami takai pagal tai, kokios yra tarpinės viršūnės $v_2 \dots v_{k-1}$ take $p = iv_2 \dots v_{k-1}j$.
- Tegu p_{k-1} – takas, kurio tarpinės viršūnės priklauso aibei $\{1, \dots, k-1\}$ ir p_k – takas, kurio tarpinės viršūnės priklauso aibei $\{1, \dots, k-1, k\}$.
- Jei k nebuvo tarpine viršūne take p_k, tai $p_k = p_{k-1}$. Kitu atveju

$$p_k : i \Rightarrow k \Rightarrow j,$$

čia esantys daliniai takai priklauso takams p_{k-1}.



Floido–Varšalo algoritmas

- Pažymėkime mažiausią svorį d_{ij}^k tako (iš viršūnės *i* į viršūnę *j*), kuriame yra k tarpinių viršūnių {1,..., k}.
- Jei tarpinių viršūnių nėra (k=0), tai

$$d_{ij}^0 = w_{ij}.$$

- Apibrėžiant rekursyviai:

$$d_{ij}^k = \begin{cases} w_{ij}, & \text{jei } k = 0, \\ \min\left\{d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}\right\}, & \text{jei } k \geq 1. \end{cases}$$

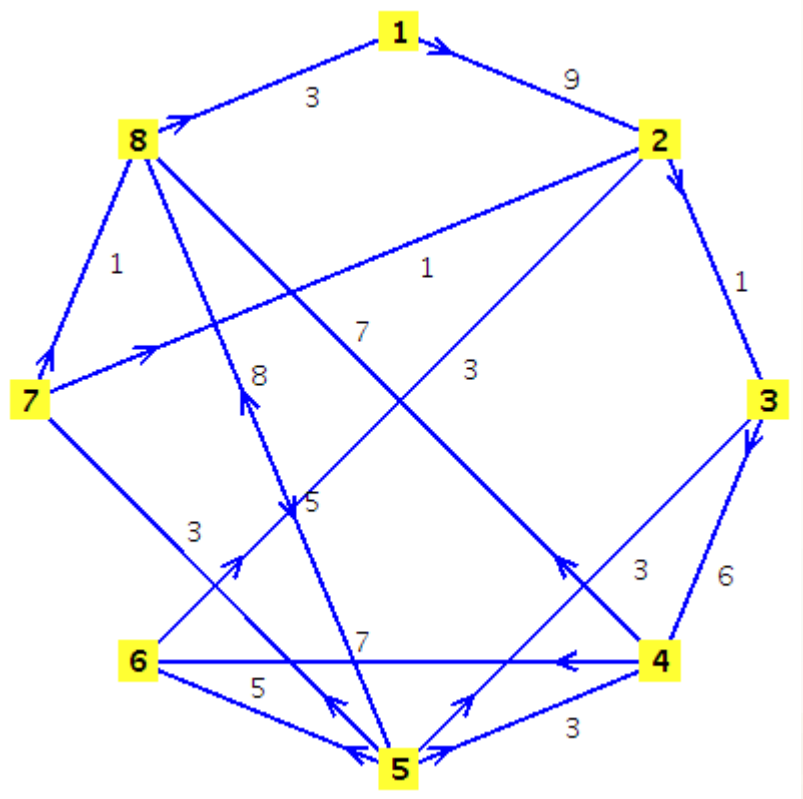
Floido–Varšalo algoritmas

Fl-W (W):

1. $n \leftarrow$ eilutės[W]
2. $D^{(0)} \leftarrow W$
3. **for** $k \leftarrow 1$ **to** n
4. **do for** $i \leftarrow 1$ **to** n
5. **do for** $j \leftarrow 1$ **to** n
6. $d_{ij}^{k} \leftarrow \min\left\{d_{ij}^{k-1}, d_{ik}^{k-1} + d_{kj}^{k-1}\right\}$
7. **return** $D^{(n)}$

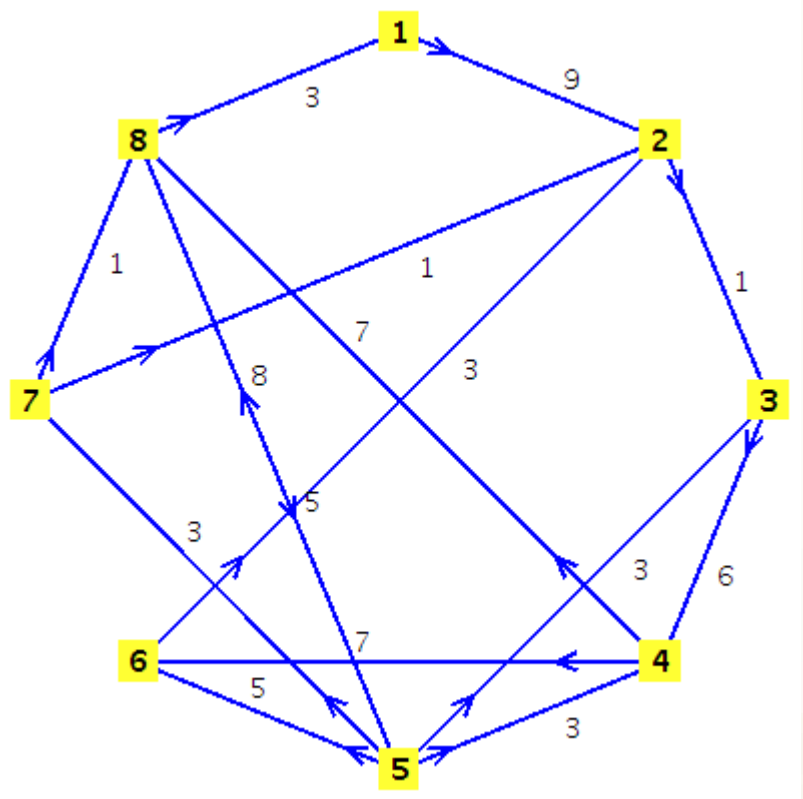
- Algoritmo sudėtingumas $O(n^3)$. Kodėl?

Floido–Varšalo algoritmas



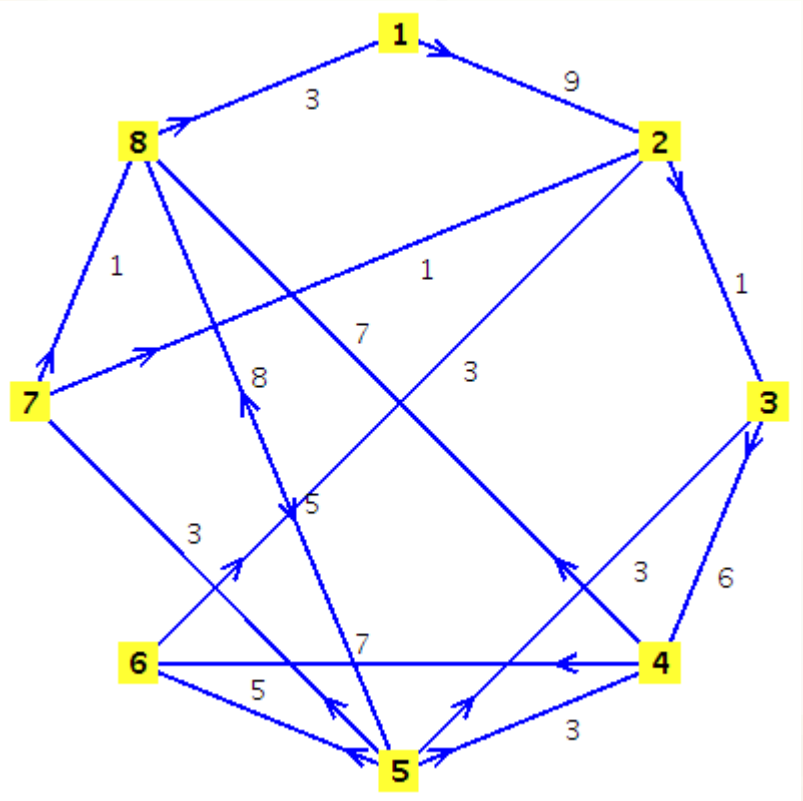
$$W := \begin{bmatrix} 0 & 9 & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 1 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 6 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & 7 & \infty & 7 \\ \infty & \infty & 3 & 3 & 0 & 5 & 3 & 8 \\ \infty & 3 & \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & 1 & \infty & \infty & \infty & \infty & 0 & 1 \\ 3 & \infty & \infty & \infty & 5 & \infty & \infty & 0 \end{bmatrix}$$

Floido–Varšalo algoritmas



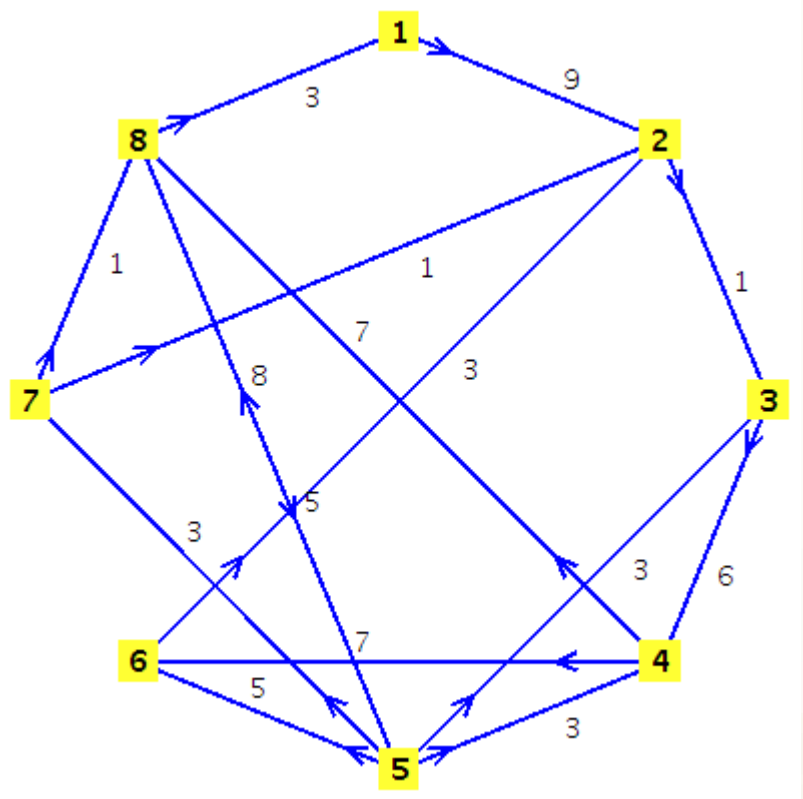
$$D_0 = \begin{bmatrix} 0 & 9 & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 1 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 6 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & 7 & \infty & 7 \\ \infty & \infty & 3 & 3 & 0 & 5 & 3 & 8 \\ \infty & 3 & \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & 1 & \infty & \infty & \infty & \infty & 0 & 1 \\ 3 & \infty & \infty & \infty & 5 & \infty & \infty & 0 \end{bmatrix}$$

Floido–Varšalo algoritmas



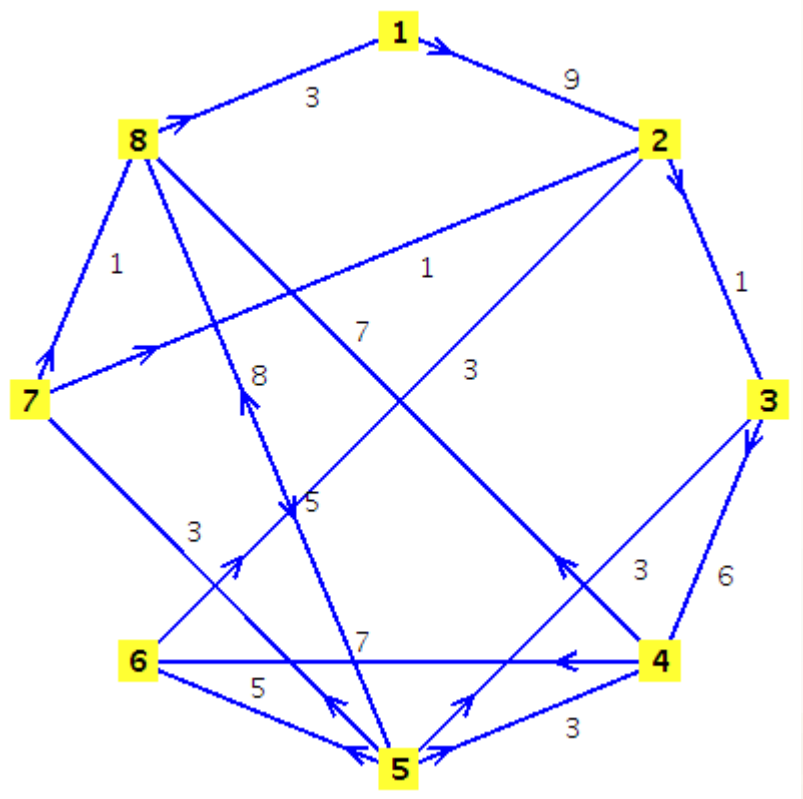
$$D_1 = \begin{bmatrix} 0 & 9 & \infty & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 1 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 6 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & 7 & \infty & 7 \\ \infty & \infty & 3 & 3 & 0 & 5 & 3 & 8 \\ \infty & 3 & \infty & \infty & \infty & 0 & \infty & \infty \\ \infty & 1 & \infty & \infty & \infty & \infty & 0 & 1 \\ 3 & 12 & \infty & \infty & 5 & \infty & \infty & 0 \end{bmatrix}$$

Floido–Varšalo algoritmas



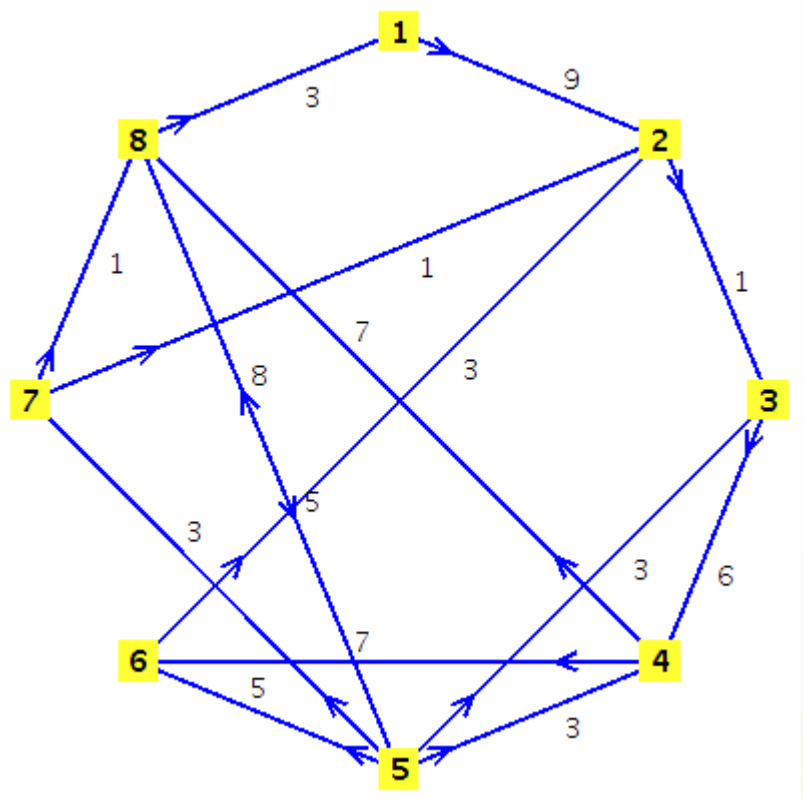
$$D_2 = \begin{bmatrix} 0 & 9 & 10 & \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 1 & \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 6 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & 7 & \infty & 7 \\ \infty & \infty & 3 & 3 & 0 & 5 & 3 & 8 \\ \infty & 3 & 4 & \infty & \infty & 0 & \infty & \infty \\ \infty & 1 & 2 & \infty & \infty & \infty & 0 & 1 \\ 3 & 12 & 13 & \infty & 5 & \infty & \infty & 0 \end{bmatrix}$$

Floido–Varšalo algoritmas



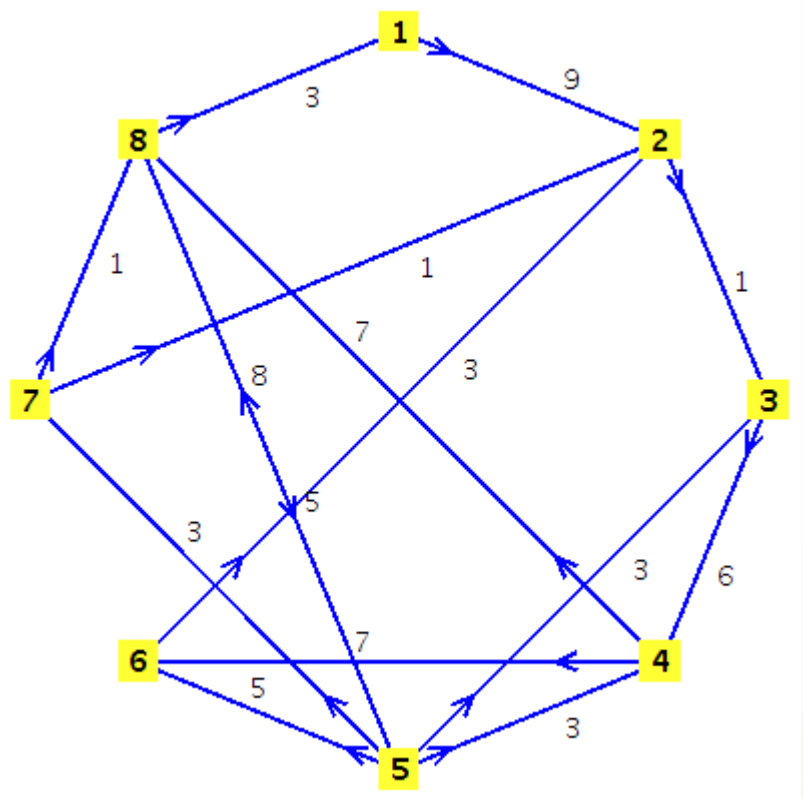
$$D_3 = \begin{bmatrix} 0 & 9 & 10 & 16 & \infty & \infty & \infty & \infty \\ \infty & 0 & 1 & 7 & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & 6 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & 0 & \infty & 7 & \infty & 7 \\ \infty & \infty & 3 & 3 & 0 & 5 & 3 & 8 \\ \infty & 3 & 4 & 10 & \infty & 0 & \infty & \infty \\ \infty & 1 & 2 & 8 & \infty & \infty & 0 & 1 \\ 3 & 12 & 13 & 19 & 5 & \infty & \infty & 0 \end{bmatrix}$$

Floido–Varšalo algoritmas



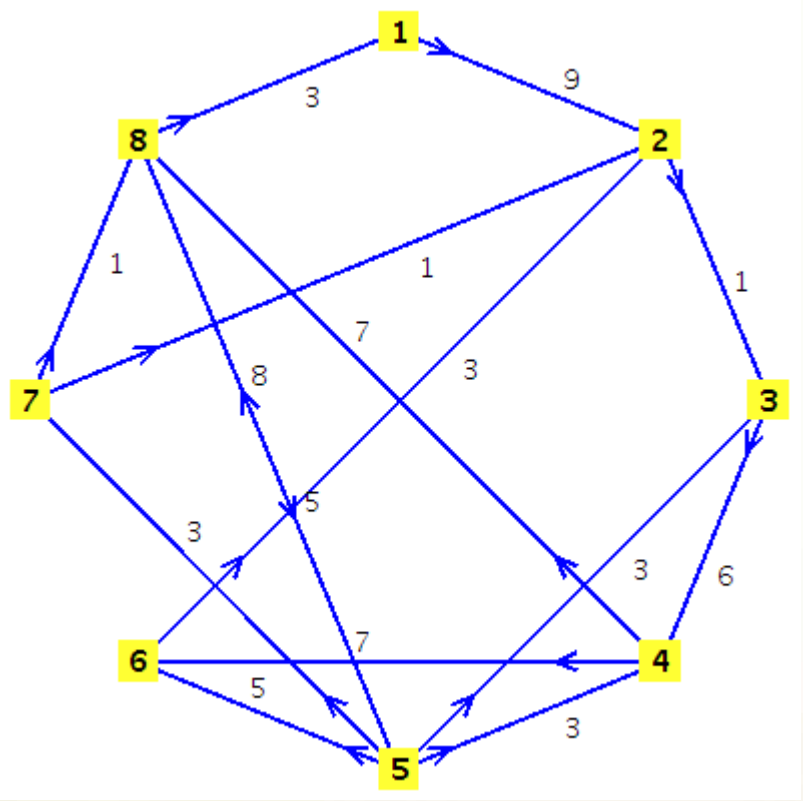
$$D_4 = \begin{bmatrix} 0 & 9 & 10 & 16 & \infty & 23 & \infty & 23 \\ \infty & 0 & 1 & 7 & \infty & 14 & \infty & 14 \\ \infty & \infty & 0 & 6 & \infty & 13 & \infty & 13 \\ \infty & \infty & \infty & 0 & \infty & 7 & \infty & 7 \\ \infty & \infty & 3 & 3 & 0 & 5 & 3 & 8 \\ \infty & 3 & 4 & 10 & \infty & 0 & \infty & 17 \\ \infty & 1 & 2 & 8 & \infty & 15 & 0 & 1 \\ 3 & 12 & 13 & 19 & 5 & 26 & \infty & 0 \end{bmatrix}$$

Floido–Varšalo algoritmas



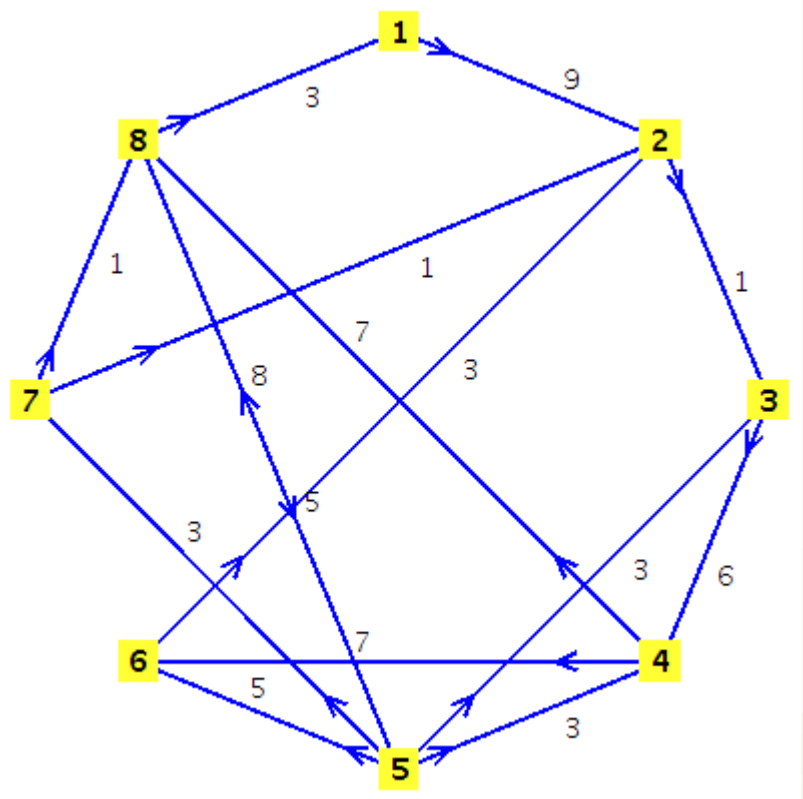
$$D_5 = \begin{bmatrix} 0 & 9 & 10 & 16 & \infty & 23 & \infty & 23 \\ \infty & 0 & 1 & 7 & \infty & 14 & \infty & 14 \\ \infty & \infty & 0 & 6 & \infty & 13 & \infty & 13 \\ \infty & \infty & \infty & 0 & \infty & 7 & \infty & 7 \\ \infty & \infty & 3 & 3 & 0 & 5 & 3 & 8 \\ \infty & 3 & 4 & 10 & \infty & 0 & \infty & 17 \\ \infty & 1 & 2 & 8 & \infty & 15 & 0 & 1 \\ 3 & 12 & 8 & 8 & 5 & 10 & 8 & 0 \end{bmatrix}$$

Floido–Varšalo algoritmas



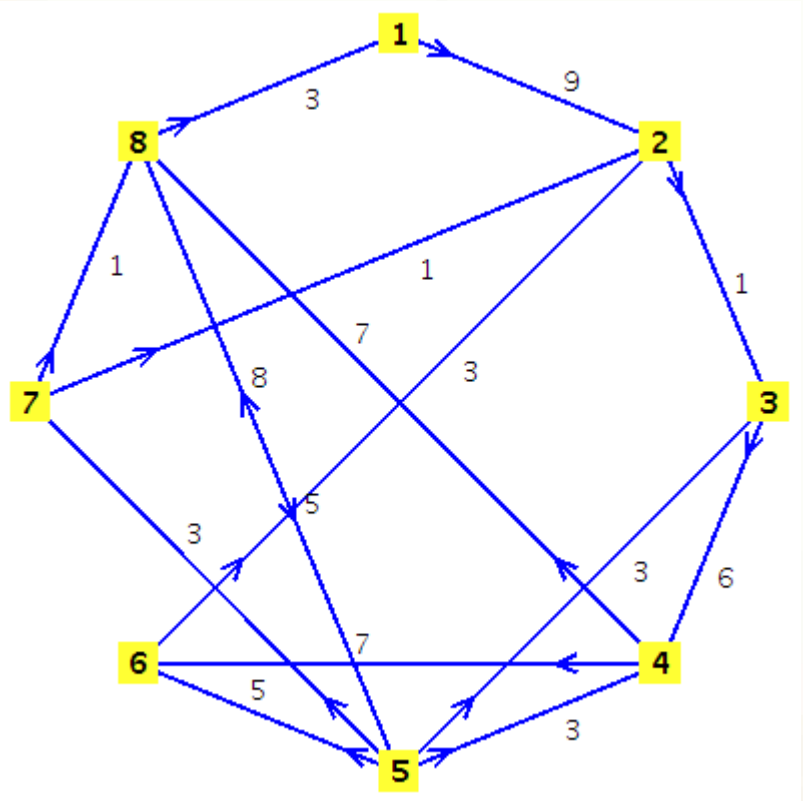
$$D_6 = \begin{bmatrix} 0 & 9 & 10 & 16 & \infty & 23 & \infty & 23 \\ \infty & 0 & 1 & 7 & \infty & 14 & \infty & 14 \\ \infty & 16 & 0 & 6 & \infty & 13 & \infty & 13 \\ \infty & 10 & 11 & 0 & \infty & 7 & \infty & 7 \\ \infty & 8 & 3 & 3 & 0 & 5 & 3 & 8 \\ \infty & 3 & 4 & 10 & \infty & 0 & \infty & 17 \\ \infty & 1 & 2 & 8 & \infty & 15 & 0 & 1 \\ 3 & 12 & 8 & 8 & 5 & 10 & 8 & 0 \end{bmatrix}$$

Floido–Varšalo algoritmas



$$D_7 = \begin{bmatrix} 0 & 9 & 10 & 16 & \infty & 23 & \infty & 23 \\ \infty & 0 & 1 & 7 & \infty & 14 & \infty & 14 \\ \infty & 16 & 0 & 6 & \infty & 13 & \infty & 13 \\ \infty & 10 & 11 & 0 & \infty & 7 & \infty & 7 \\ \infty & 4 & 3 & 3 & 0 & 5 & 3 & 4 \\ \infty & 3 & 4 & 10 & \infty & 0 & \infty & 17 \\ \infty & 1 & 2 & 8 & \infty & 15 & 0 & 1 \\ 3 & 9 & 8 & 8 & 5 & 10 & 8 & 0 \end{bmatrix}$$

Floido–Varšalo algoritmas



$$D_8 = \begin{bmatrix} 0 & 9 & 10 & 16 & 28 & 23 & 31 & 23 \\ 17 & 0 & 1 & 7 & 19 & 14 & 22 & 14 \\ 16 & 16 & 0 & 6 & 18 & 13 & 21 & 13 \\ 10 & 10 & 11 & 0 & 12 & 7 & 15 & 7 \\ 7 & 4 & 3 & 3 & 0 & 5 & 3 & 4 \\ 20 & 3 & 4 & 10 & 22 & 0 & 25 & 17 \\ 4 & 1 & 2 & 8 & 6 & 11 & 0 & 1 \\ 3 & 9 & 8 & 8 & 5 & 10 & 8 & 0 \end{bmatrix}$$

Ačiū už dėmesį.

Klausimai?