Didžiausios keliamosios galios uždavinys

Žilvinas Verseckas

Tikslas

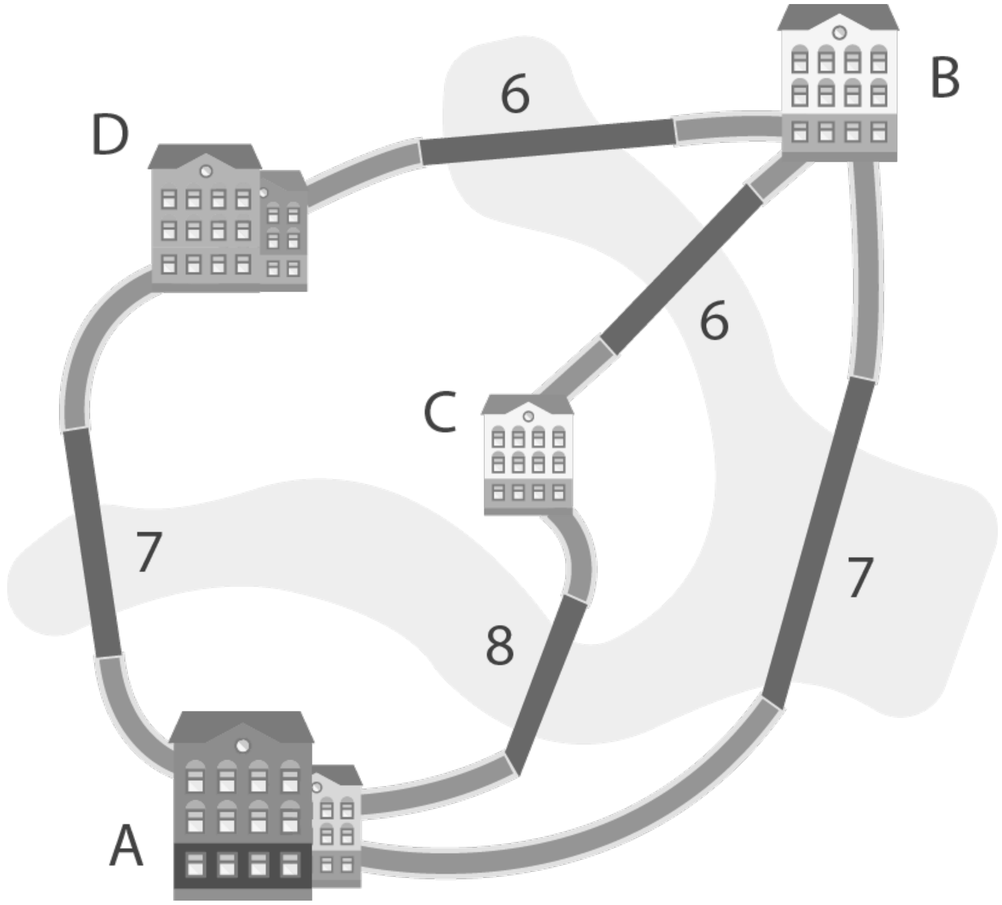
Suformuluoti didžiausios keliamosios galios apskaičiavimo **uždavinį**, pristatyti jo **sprendimo algoritmą**, pateikti algoritmo veikimo pavyzdį.

Uždavinio formuluotė

Duotas svorinis grafas $G(V, U, C)$, kur C yra grafo briaunų svorių masyvas.

Tarkime, kad briauna yra kelias, tarp gyvenvietes atitinkančių viršūnių.

Tarkime, tame kelyje yra tiltas per upę, ir briaunos svoris reiškia to tilto keliamąją galią. Apskaičiuoti, kokius didžiausio svorio krovinius galima nuvežti iš viršūnės v_0 iki visų kitų viršūnių v_i.



Algoritmas

1: **Duota:**
2: $G(V, U, C)$ // Viršūnių aibė V, briaunų aibė U, briaunų svorių aibė C
3: s // Pradžios viršūnė, $s \in [1..n], n = |V|$
4: **Rezultatas:**
5: d // Didžiausių svorių, galimų nuvežti į kiekvieną viršūnę, sąrašas
6: $prec$ // Kelių atsekamumo sąrašas

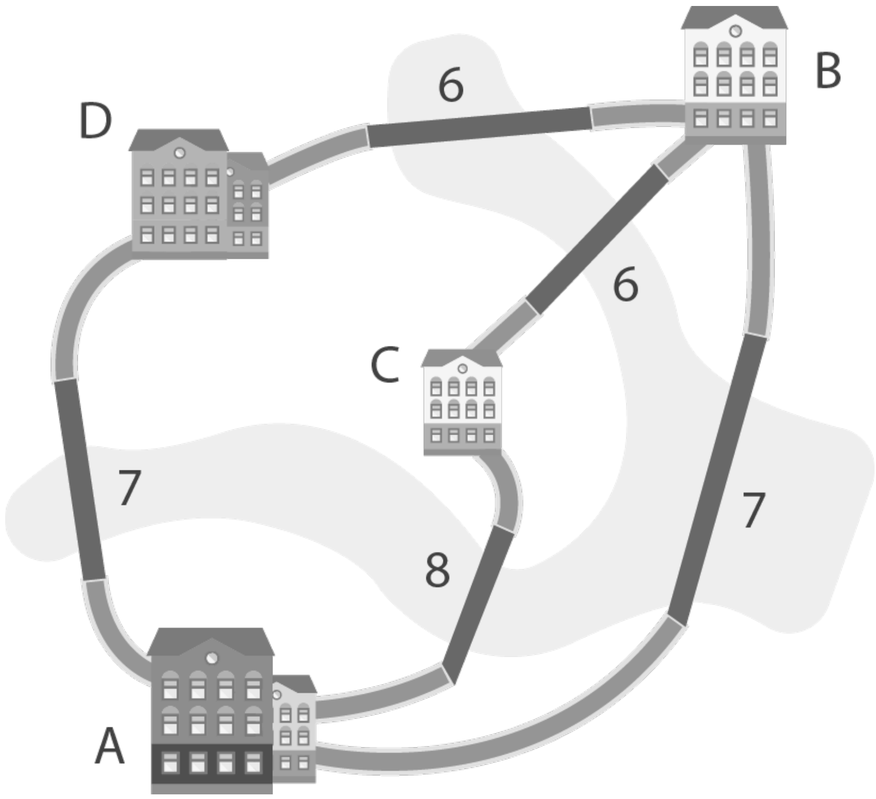
Algoritmas (2)

7: **Inicializacija:**
8: Visos grafo viršūnės neaplankytos
9: $d[i] \leftarrow 0; prec[i] \leftarrow 0$ // $i \in [1..n]$
10: $d[s] \leftarrow \sum_{i=1}^{m} c_i + 1$ // Begalybė, $m = |U|$, $c_i \in C$
11: $prec[s] \leftarrow s$

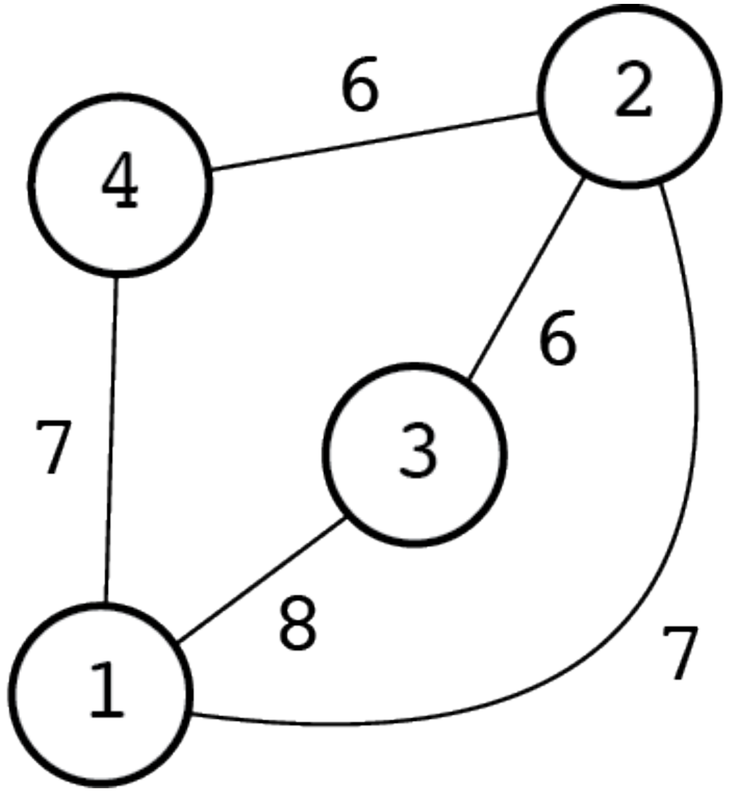
Algoritmas (3)

12: **Skaičiavimai:**
13: **while** neaplankytos != 0 **do**
14: $maxSvoris \leftarrow maxNeaplankyta(d)$
15: $k \leftarrow indexOf(maxSvoris, d)$
16: **if** maxSvoris = 0 **then**
17: stop // Grafas nejungus
18: aplankyti(k)
19: **for** u $\in$ kaimynai(k) **do**
20: **if** neaplankyta(k) and $d[u] < min(d[k], c(k, u))$ **then**
21: $d[u] \leftarrow min(d[k], c(k, u))$
22: $prec[u] \leftarrow k$

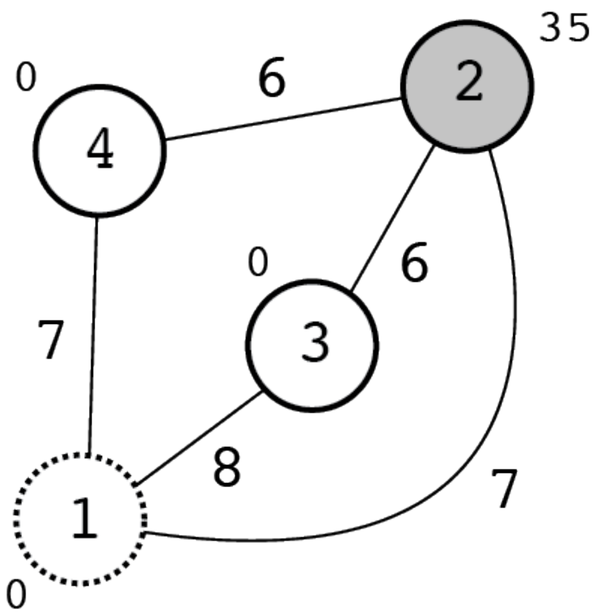
Algoritmo veikimo pavyzdys



~



Algoritmo veikimo pavyzdys (2)



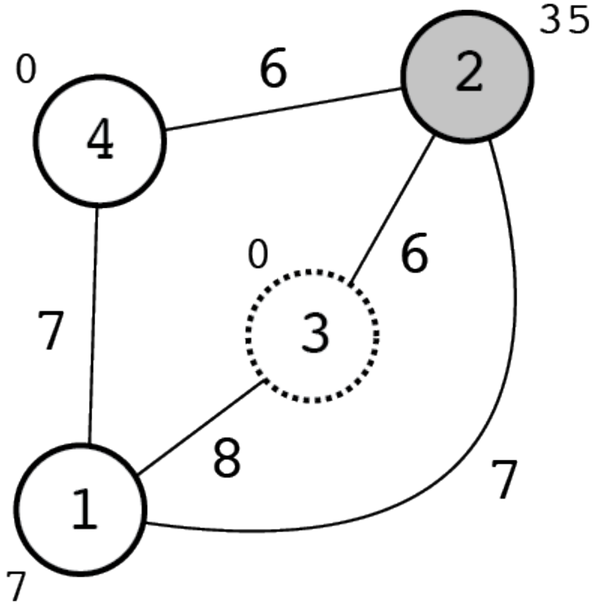
d:

	1	2	3	4
0:	0	**35**	0	0

prec:

	1	2	3	4
0:	0	2	0	0

Algoritmo veikimo pavyzdys (3)



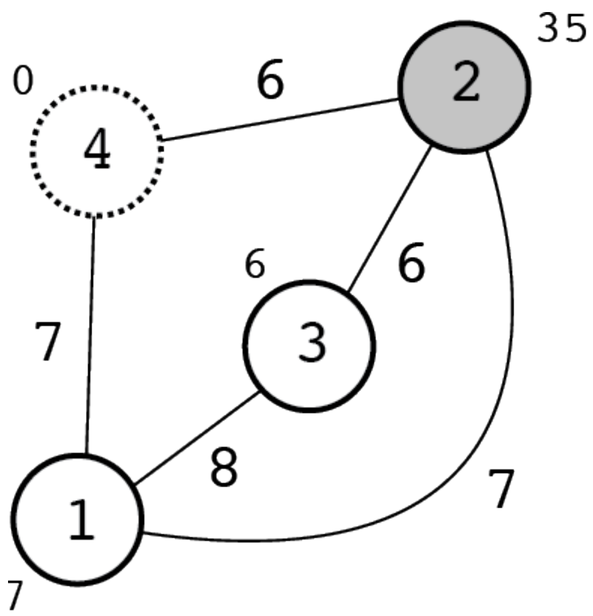
d:

	1	2	3	4
0:	0	35	0	0
1:	7	35	0	0

prec:

	1	2	3	4
0:	0	2	0	0
1:	2	2	0	0

Algoritmo veikimo pavyzdys (4)



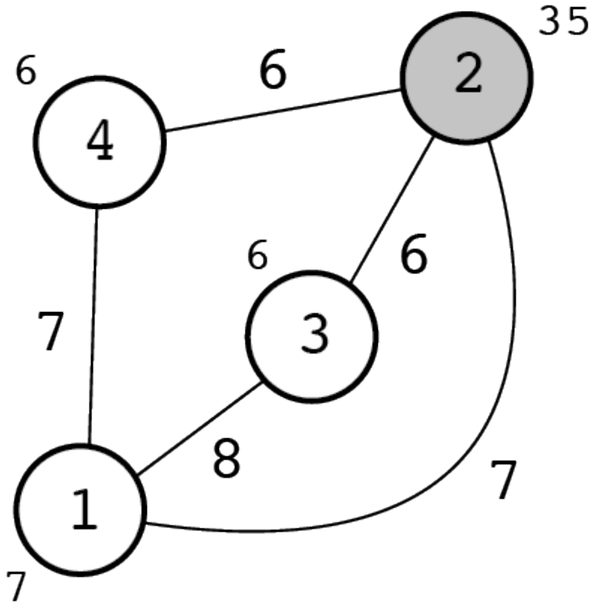
d:

	1	2	3	4
0:	0	35	0	0
1:	7	35	0	0
2:	**7**	**35**	**6**	**0**

prec:

	1	2	3	4
0:	0	2	0	0
1:	2	2	0	0
2:	**2**	**2**	**2**	**0**

Algoritmo veikimo pavyzdys (5)



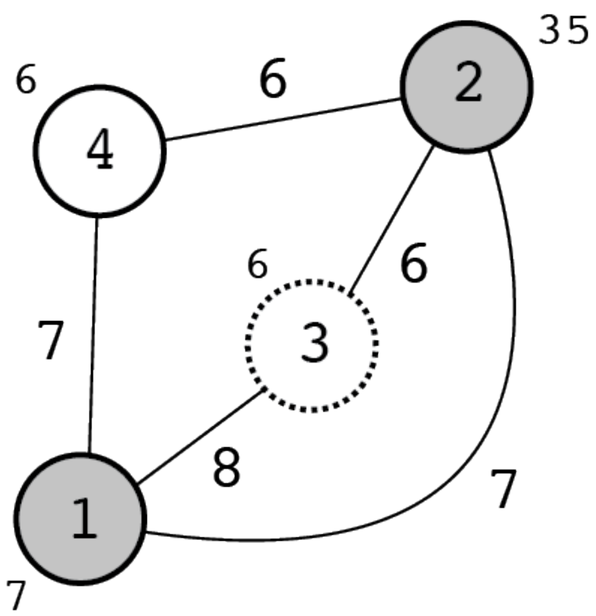
d:

	1	2	3	4
0:	0	35	0	0
1:	7	35	0	0
2:	7	35	6	0
3:	**7**	**35**	**6**	**6**

prec:

	1	2	3	4
0:	0	2	0	0
1:	2	2	0	0
2:	2	2	2	0
3:	**2**	**2**	**2**	**2**

Algoritmo veikimo pavyzdys (6)



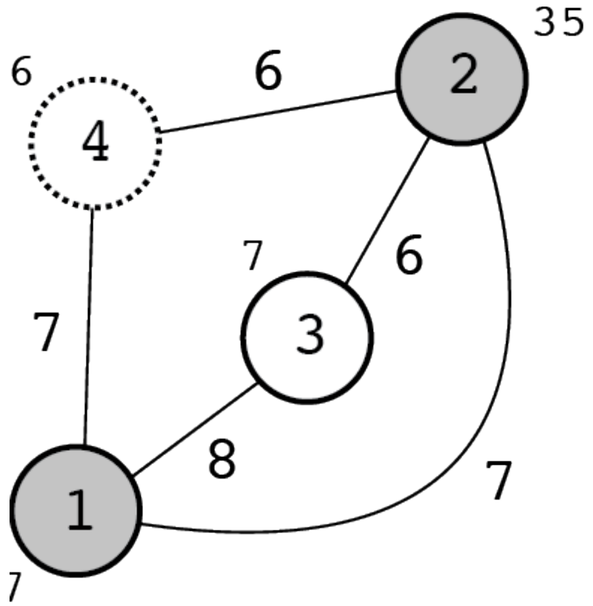
d:

	1	2	3	4
0:	0	35	0	0
1:	7	35	0	0
2:	7	35	6	0
3:	**7**	35	6	6

prec:

	1	2	3	4
0:	0	2	0	0
1:	2	2	0	0
2:	2	2	2	0
3:	2	2	2	2

Algoritmo veikimo pavyzdys (7)



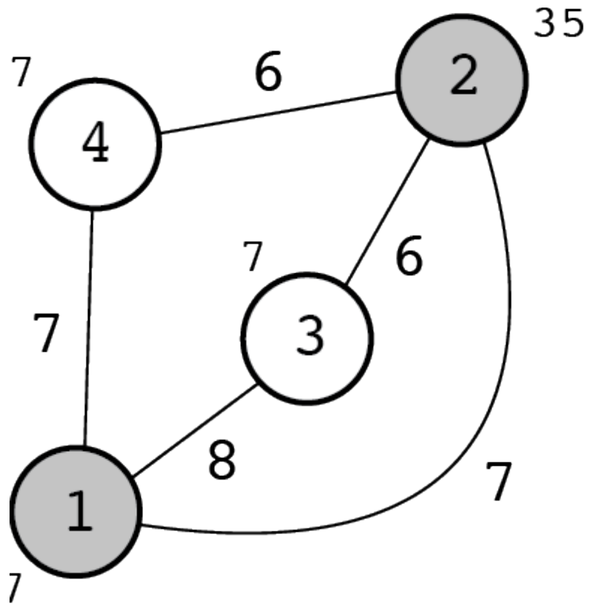
d:

	1	2	3	4
0:	0	35	0	0
1:	7	35	0	0
2:	7	35	6	0
3:	7	35	6	6
4:	**7**	**35**	**7**	**6**

prec:

	1	2	3	4
0:	0	2	0	0
1:	2	2	0	0
2:	2	2	2	0
3:	2	2	2	2
4:	**2**	**2**	**1**	**2**

Algoritmo veikimo pavyzdys (8)



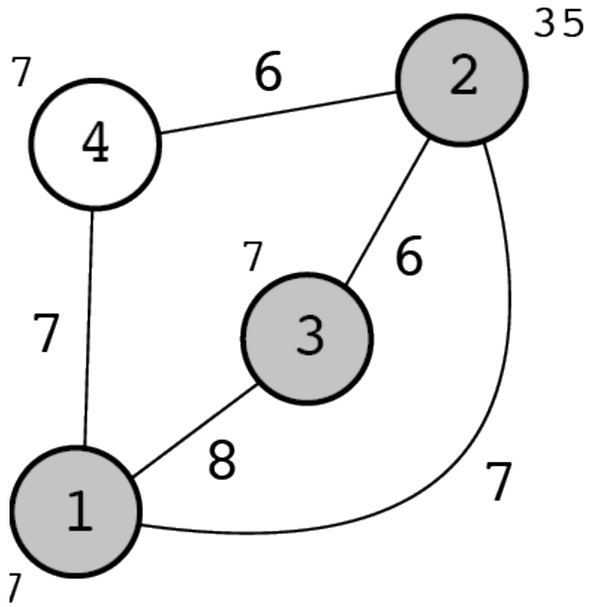
d:

	1	2	3	4
0:	0	35	0	0
1:	7	35	0	0
2:	7	35	6	0
3:	7	35	6	6
4:	7	35	7	6
5:	**7**	**35**	**7**	**7**

prec:

	1	2	3	4
0:	0	2	0	0
1:	2	2	0	0
2:	2	2	2	0
3:	2	2	2	2
4:	2	2	1	2
5:	**2**	**2**	**1**	**1**

Algoritmo veikimo pavyzdys (9)



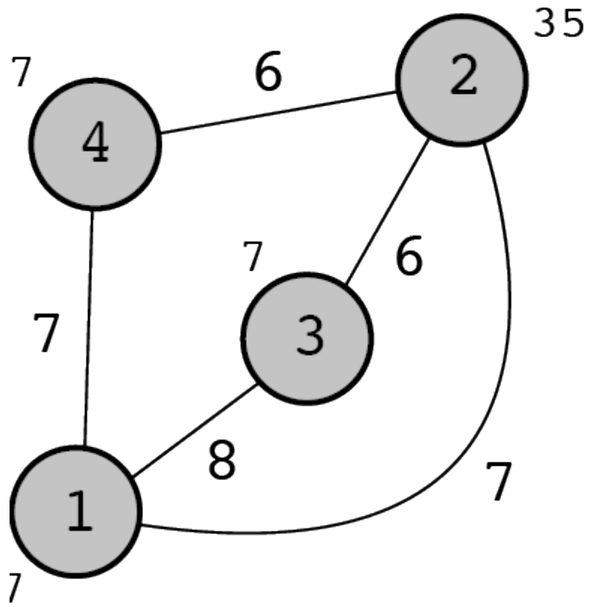
d:

	1	2	3	4
0:	0	35	0	0
1:	7	35	0	0
2:	7	35	6	0
3:	7	35	6	6
4:	7	35	7	6
5:	**7**	**35**	**7**	**7**

prec:

	1	2	3	4
0:	0	2	0	0
1:	2	2	0	0
2:	2	2	2	0
3:	2	2	2	2
4:	2	2	1	2
5:	**2**	**2**	**1**	**1**

Algoritmo veikimo pavyzdys (10)



d:

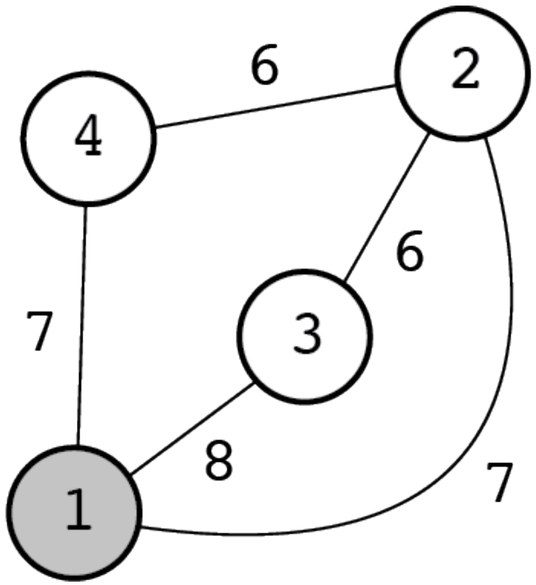
	1	2	3	4
0:	0	35	0	0
1:	7	35	0	0
2:	7	35	6	0
3:	7	35	6	6
4:	7	35	7	6
5:	**7**	**35**	**7**	**7**

prec:

	1	2	3	4
0:	0	2	0	0
1:	2	2	0	0
2:	2	2	2	0
3:	2	2	2	2
4:	2	2	1	2
5:	**2**	**2**	**1**	**1**

```
   d = [7,35,7,7]
prec = [2,2,1,1]
```

Pratimas



d:

	1	2	3	4
0:	**35**	0	0	0

prec:

	1	2	3	4
0:	1	0	0	0