＜数理計画モデル＞

- 線形計画問題
- ネットワーク計画問題
- 非線形計画問題
- 組合せ計画問題

＜輸送問題＞

A社は2つの工場A_1,A_2で製品を生産し、３つの取引先B_1,B_2,B_3に納入している。

注文量

B_1	70
B_2	40
B_3	60

生産量

A_1	90
A_2	80

輸送コスト

	B_1	B_2	B_3
A_1	4	7	12
A_2	11	6	3

＜変数＞

x_{ij} : 工場A_iから取引先B_jへの輸送量

(i= 1,2 ;　j= 1,2,3)

＜制約条件＞

$x_{11} + x_{12} + x_{13} = 90$

$x_{21} + x_{22} + x_{23} = 80$

工場A_1,A_2での生産量

$x_{11} + x_{21} = 70$

$x_{12} + x_{22} = 40$

$x_{13} + x_{23} = 60$

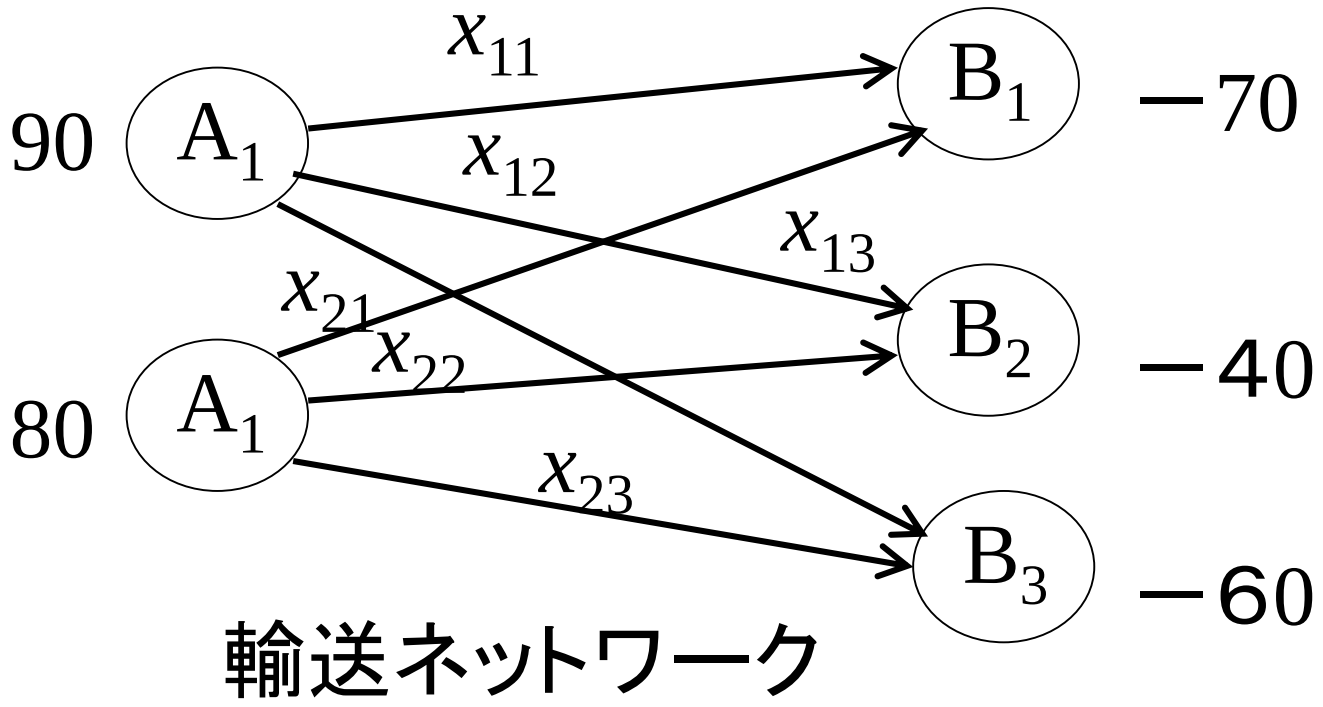
取引先B_1,B_2,B_3の注文量

$x_{ij} \geqq 0$　(i= 1,2 ;　j= 1,2,3)　輸送量の非負条件

＜目的関数＞

$$4x_{11} + 7\,x_{12} + 12\,x_{13} + 11x_{21} + 6x_{22} + 3x_{23}$$

輸送コストの総和が最小となるようにする



輸送ネットワーク

ネットワークモデル

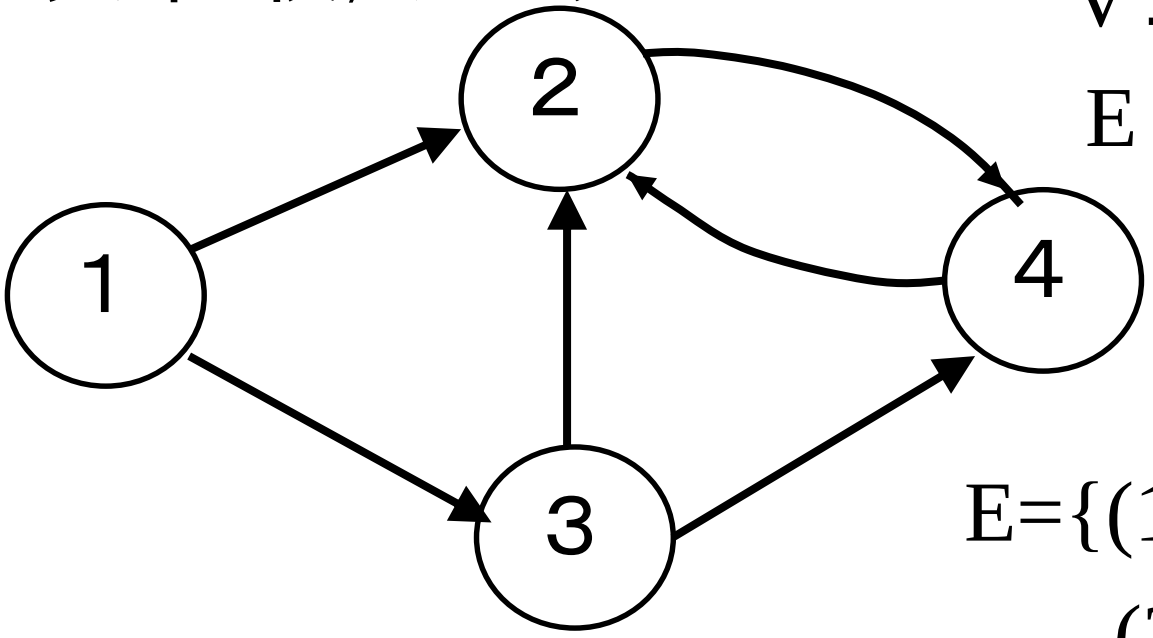
<グラフとネットワーク＞

点：節点，ノード　　枝(i,j):節点iから節点jへの枝

矢印：枝，アーク

V：節点全体の集合

E：枝全体の集合



有向グラフ

V={1,2,3,4}

E={(1,2), (1,3), (2,4),
(3,2),(3,4),(4,2)}

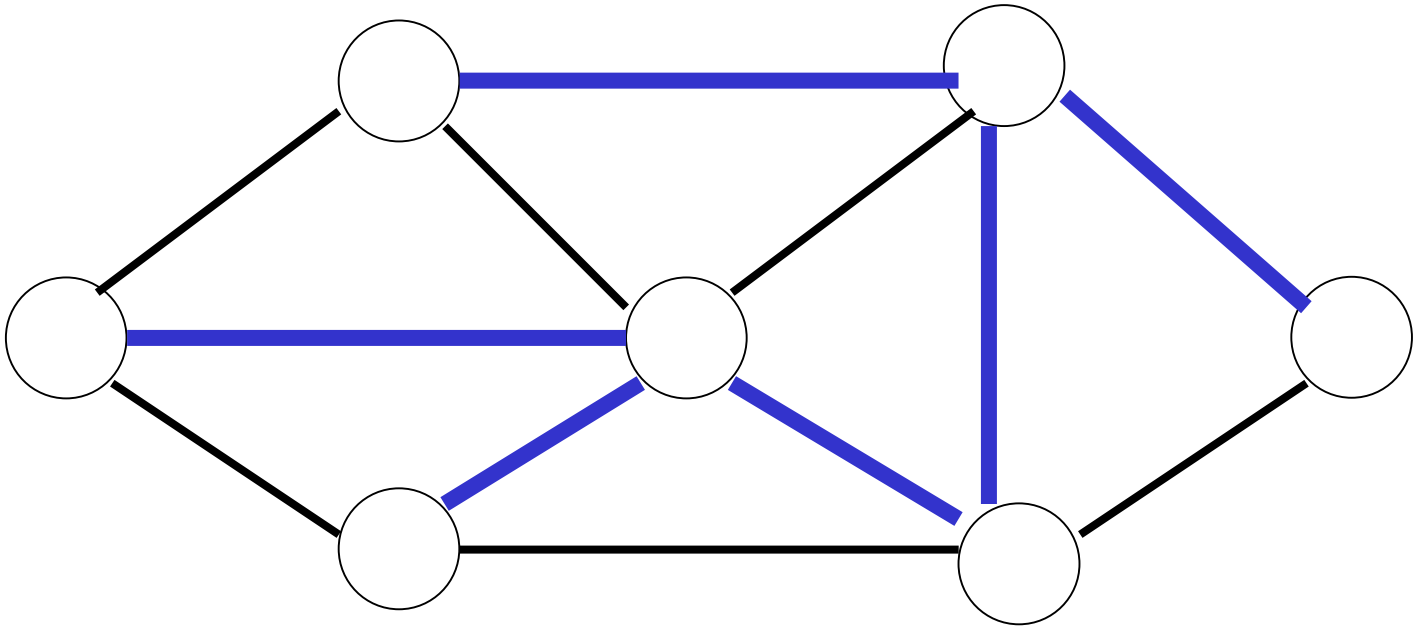
ネットワーク計画

線形計画問題の特別な場合

- 最小木問題
- 最短路問題
- 最長路問題（PERT：日程計画）
- 最大流問題
- 最小費用流問題
- マッチング問題（割当問題）

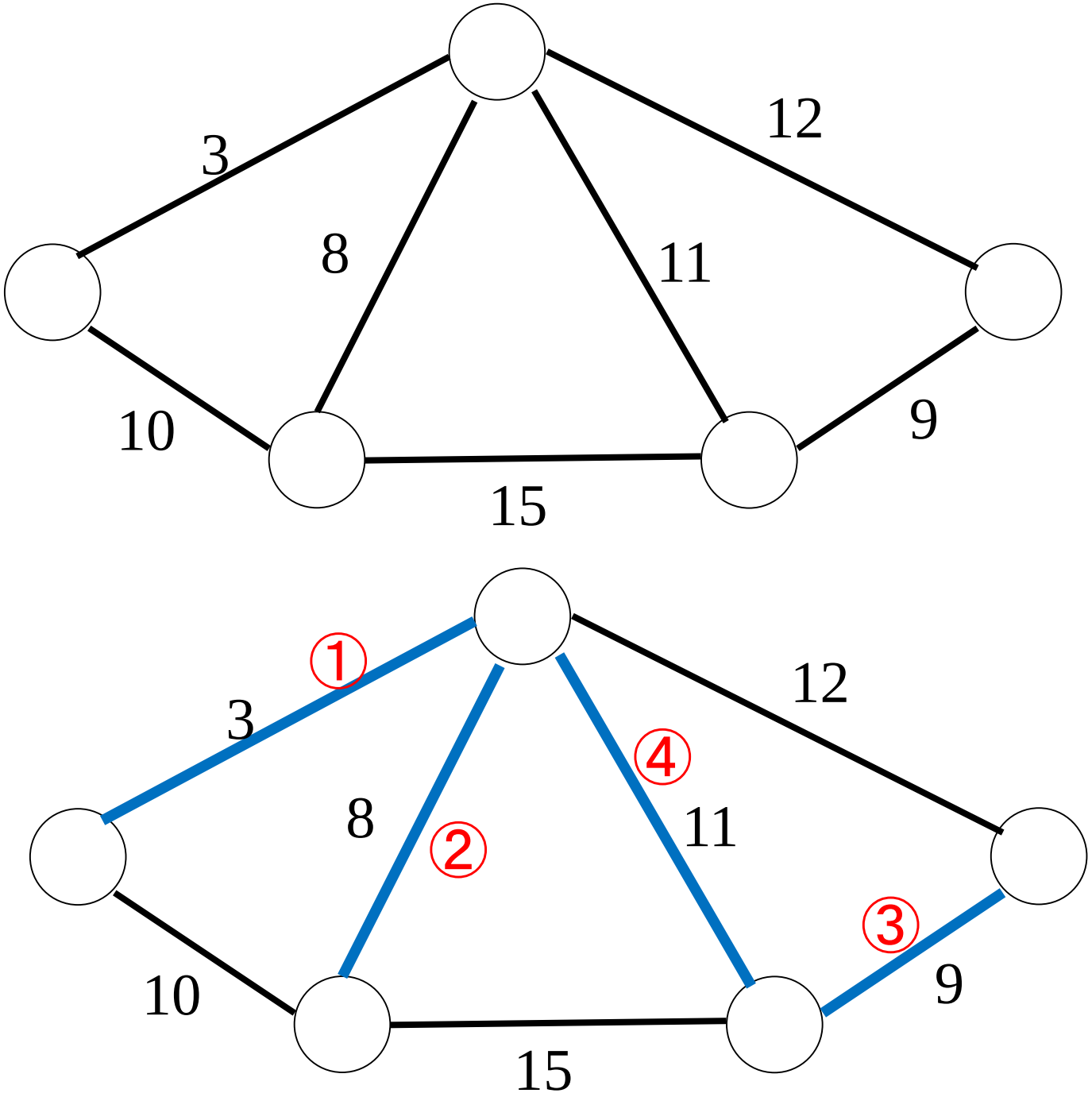
最小木問題

木：閉路を含まない連結グラフ



最小木：長さ最小の全域木

a_i：枝e_iの長さ


3
12
8
11
10
9
15
①
3
12
④
8
②
11
③
10
9
15

＜クラスカル法＞　J.B.Kruskal (1956)

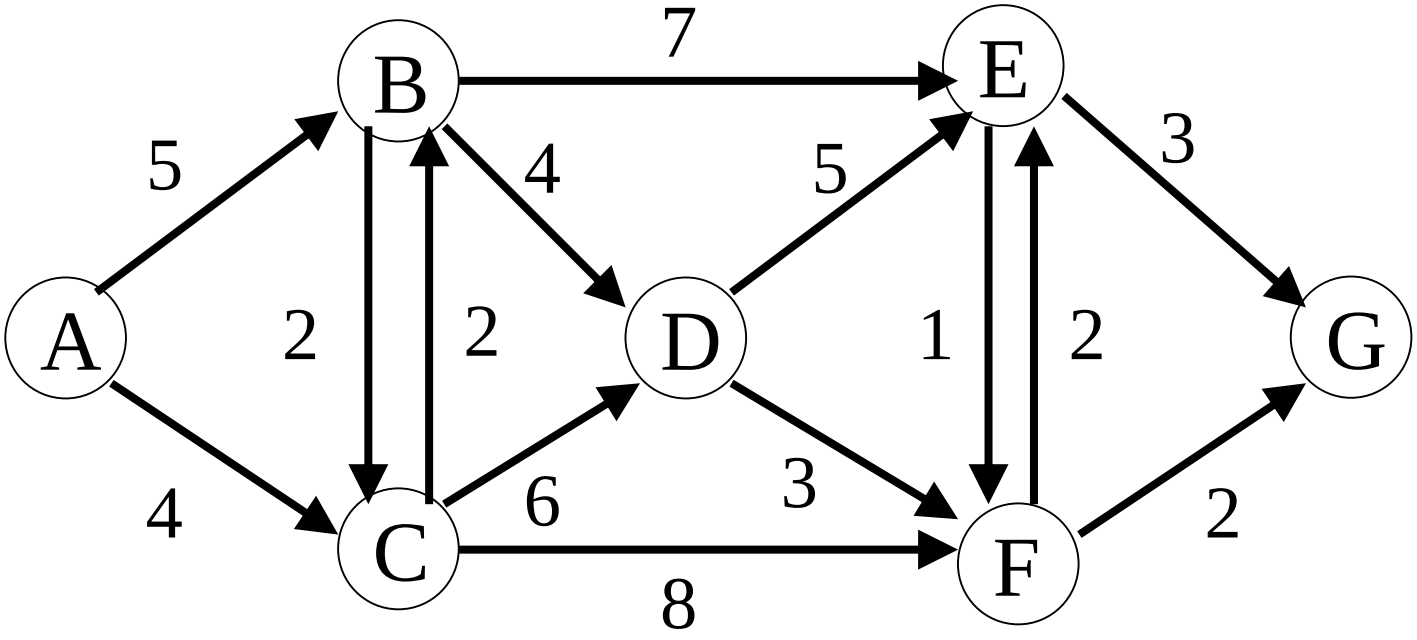
(0) グラフGの枝を短い順に並べ、$a_1 \leqq a_2 \leqq \cdots \leqq a_m$を満たすように枝$e_i$の番号（添字）を付けかえる．　T:＝$\{e_1\}$, k :＝2 とおく.

(1) 枝集合T$\cup \{e_k\}$が閉路を含まないならば、T:＝T$\cup \{e_k\}$ とする.

(2) TがGの全域木になっていれば計算終了.
さもなければ k :＝k+1 としてステップ(1)へ.

欲張り法（Greedy Algorithm）

最短路問題



路の例：A→C→D→E→F→G

＜最適性の原理＞

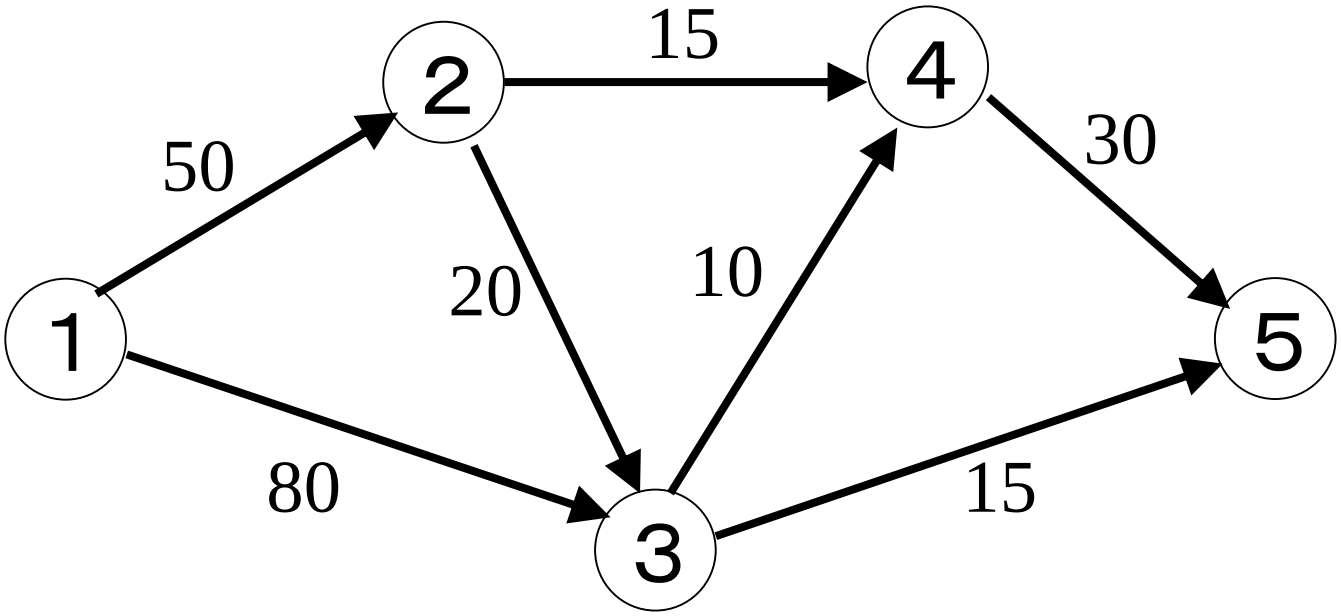
P : 節点sからtへの最短路

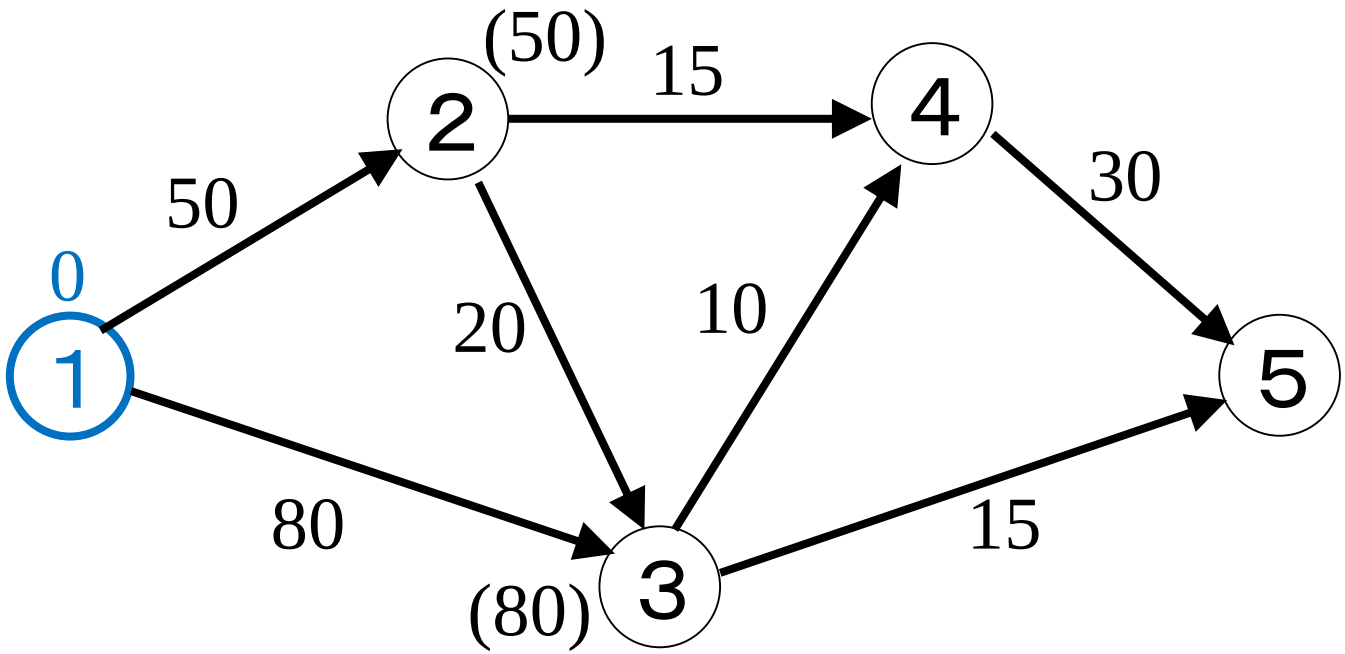
最短路Pのどの一部分を取り出しても，その両端の節点間の最短路になっている.

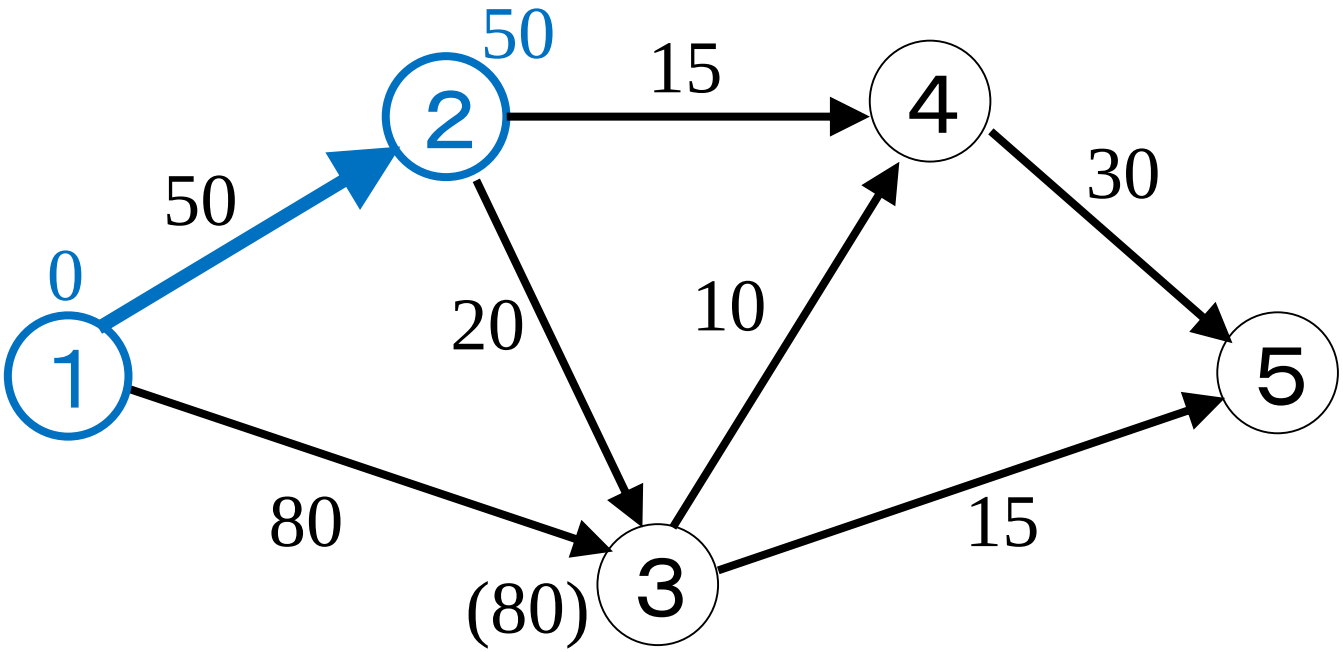
a_{ij} : 枝(i, j)の長さ

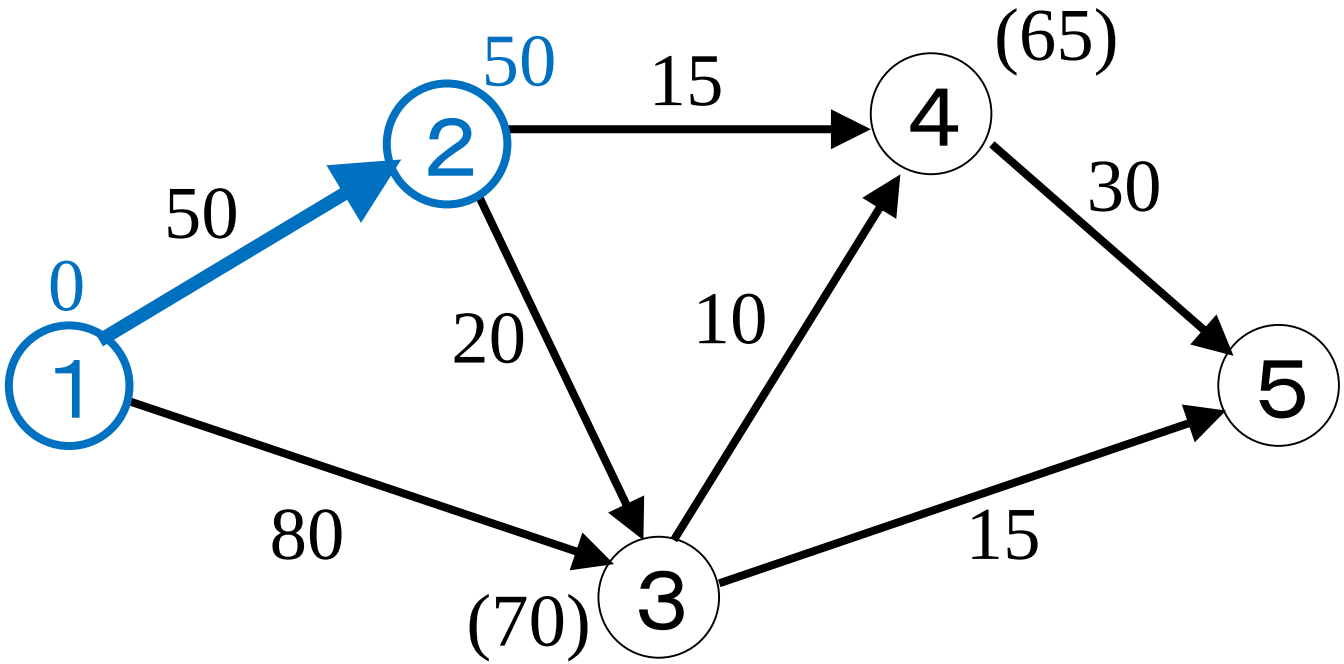
d(i) : 節点sから節点iへの最短路の長さの上限値

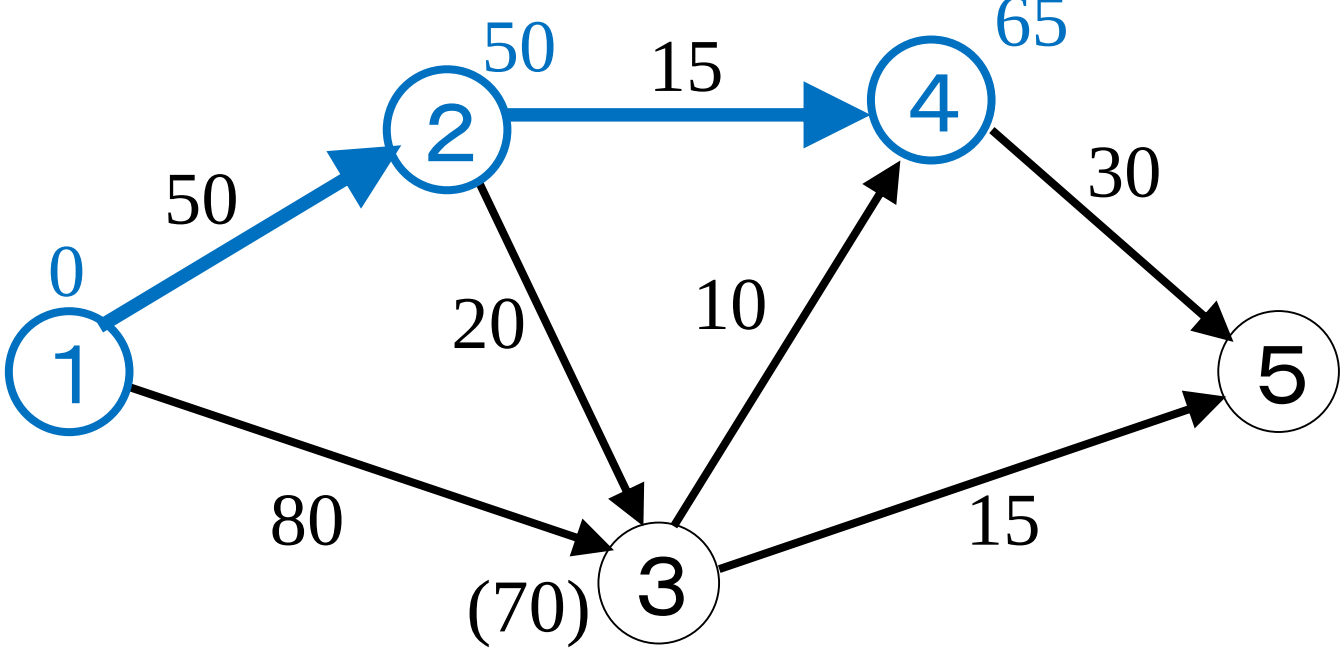
p(i) : 節点sから節点iへの最短路において節点iの直前に位置する節点

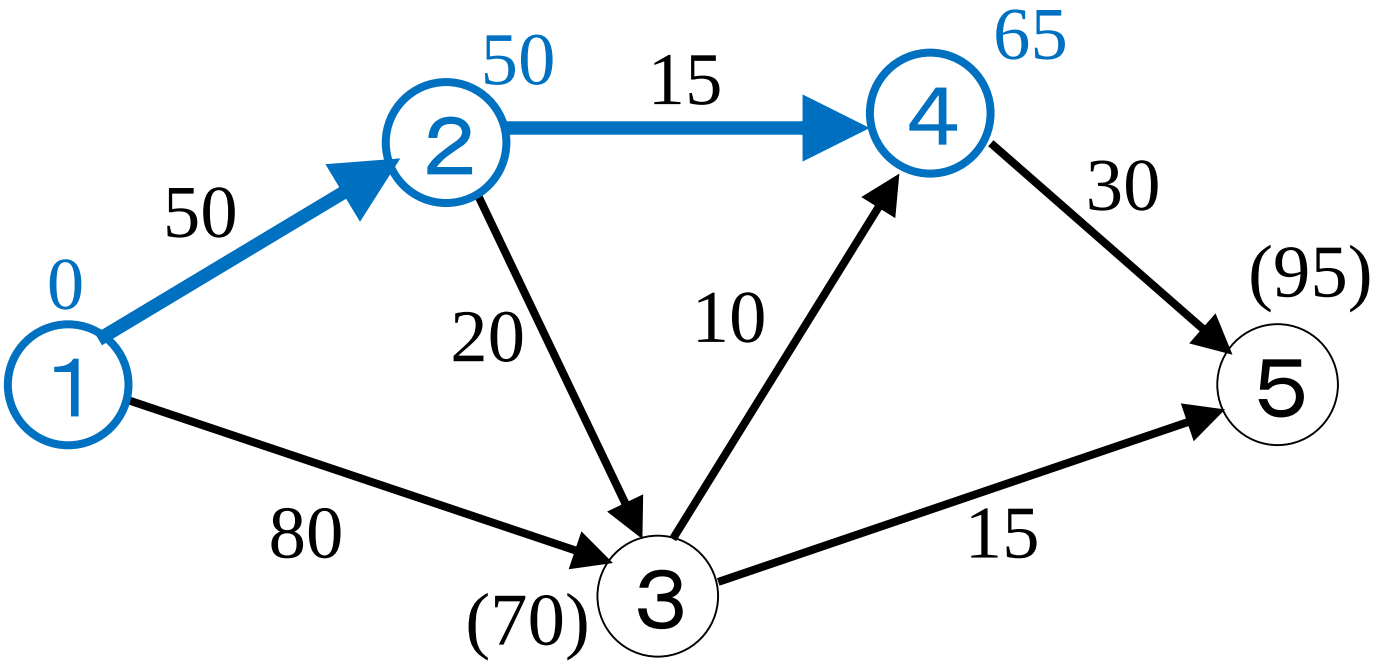

1
2
3
4
5
50
15
30
20
10
80
15

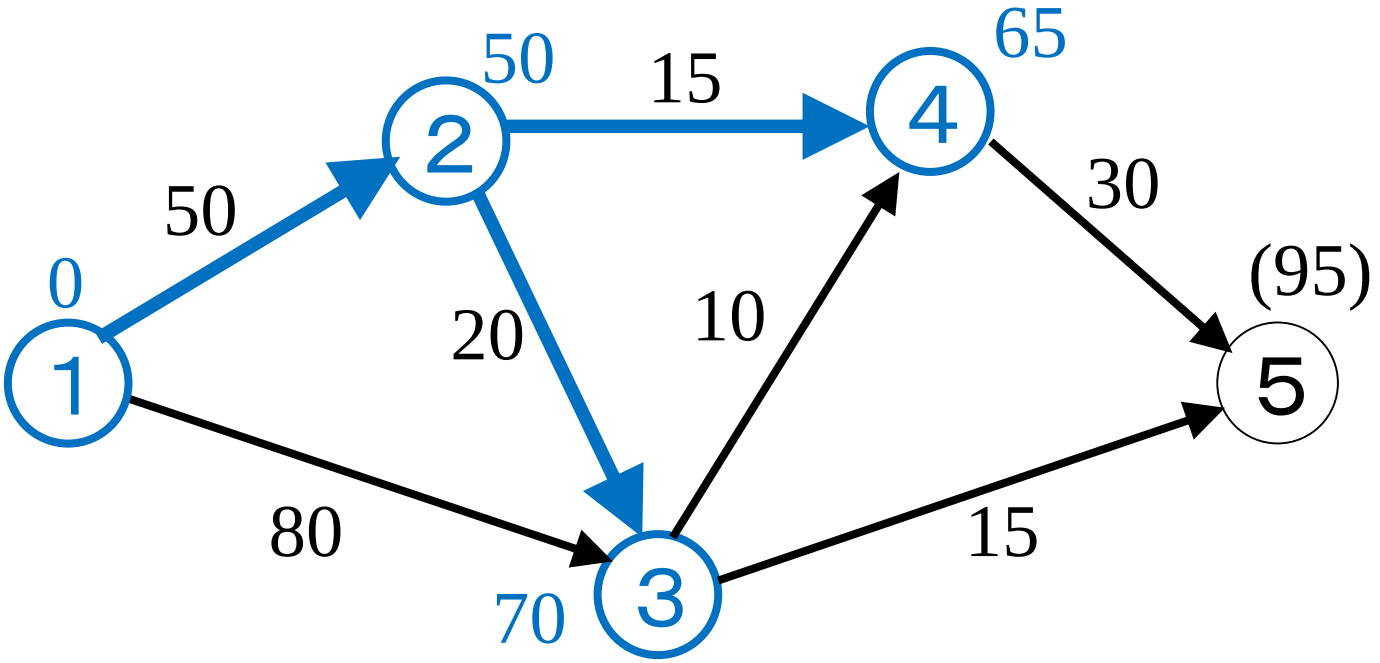

0
1
(50)
2
3
(80)
4
5
50
15
30
20
10
80
15

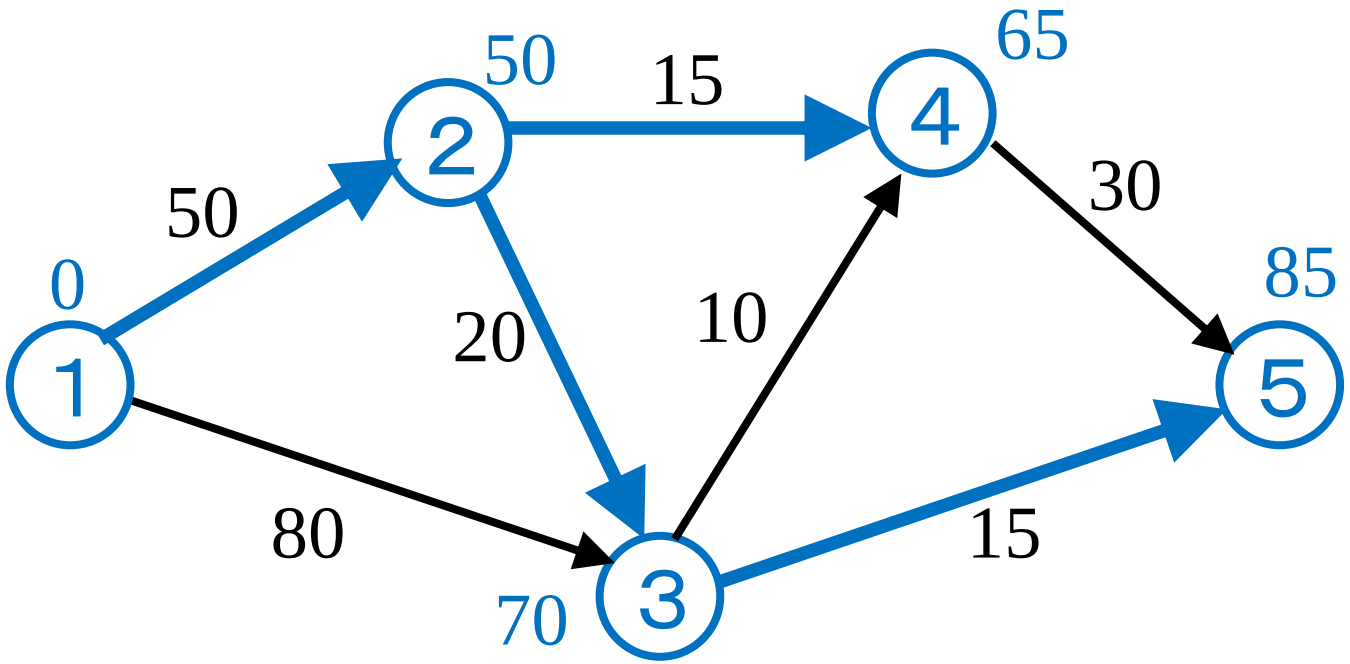

50
15
50
30
0
20
10
1
2
4
5
80
15
(80)
3


(65)
50
15
50
30
0
20
10
1
2
4
5
80
15
(70)
3


0
1
50
2
50
15
4
65
30
5
20
10
80
(70)
3
15


0
1
50
2
50
15
4
65
30
(95)
5
20
10
80
(70)
3
15


0
50
15
65
50
30
(95)
20
10
80
15
70


0
50
50
15
65
30
85
20
10
80
15
70

＜ダイクストラ法＞　E.Dijkstra (1959)

(0) $S := \varphi$,　$\overline{S} := V$,　$d(s) := 0$,
　　$d(i) := \infty$ $(i \in V - \{s\})$とおく.

(1) $S = V$なら計算終了. そうでないなら,

$d(v) = \min\{d(i) \mid i \in \overline{S}\}$ である節点$v \in \overline{S}$を選ぶ.

(2) $S := S \cup \{v\}$,　$\overline{S} := \overline{S} - \{v\}$)とする.

　　$d(j) > d(v) + a_{vj}$ ならば $d(j) := d(v) + a_{vj}$
　　　　$((v,j) \in E,\ j \in \overline{S})$

　　$p(j) := v$　とし, ステップ(1)へ.

＜計算の複雑さ＞

計算量：アルゴリズムが停止するまでに実行される演算の総数

大きさNの問題を$f(N)$回の演算で解くことができる　⟶　計算量$O(f(N))$

$f(N)$がNのべき乗で表される

⟶　多項式時間アルゴリズム

$f(N)$が2^NやN！

⟶　指数時間アルゴリズム

＜多項式時間アルゴリズム＞

大規模な問題に対しても効率的である

＜指数時間アルゴリズム＞

計算量が爆発的に増加

多項式時間アルゴリズムが存在する問題

⟶ クラスPに属する　　多項式時間（polynomial time）

NP困難な問題：多項式時間アルゴリズムの存在が知られていない

非決定計算による多項式時間（nondeterministic polynomial time ）

＜ダイクストラ法の計算量＞

アルゴリズムの反復回数：節点数 n

ステップ(1)：$O(n^2)$

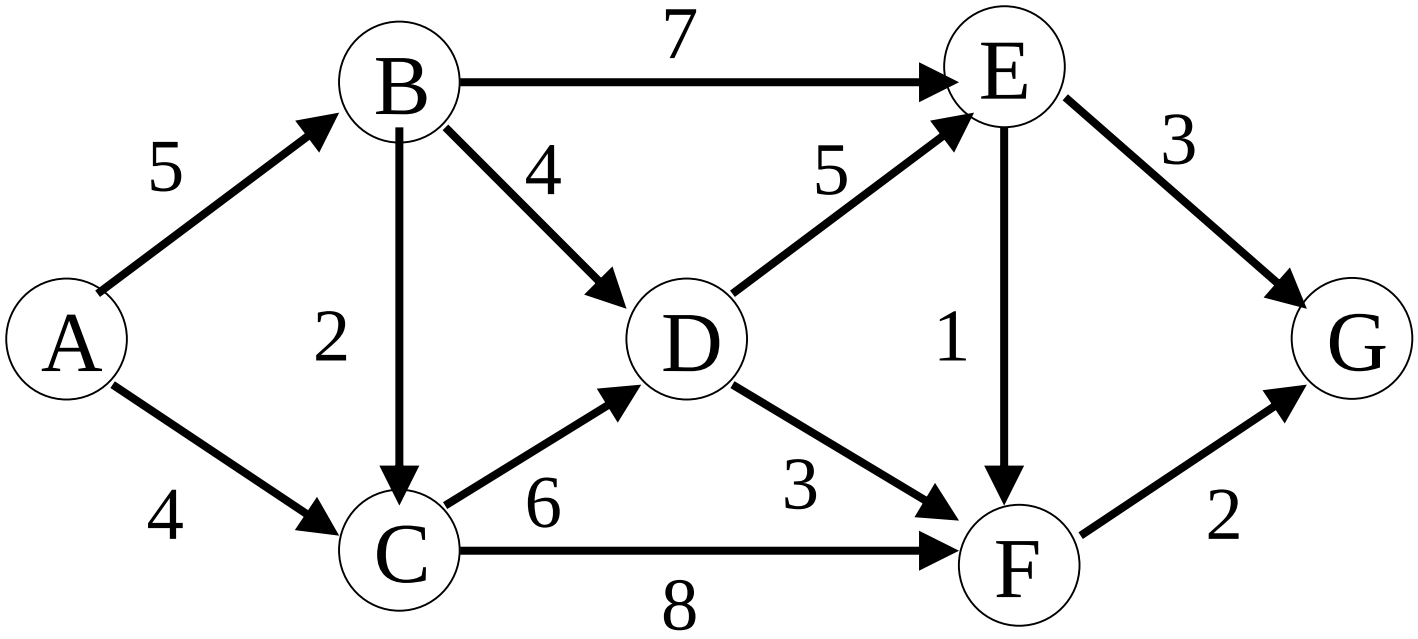
ステップ(2)：$O(m)$　　(m：枝数)

アルゴリズム全体の計算量：$O(n^2 + m) = O(n^2)$

$m \leqq n^2$

最長路問題

アサイクリックグラフ：閉路を含まない有向グラフ



計算量：O(｜V｜＋｜E｜)

＜PERT：日程計画＞

PERT(program evaluation and review technique)

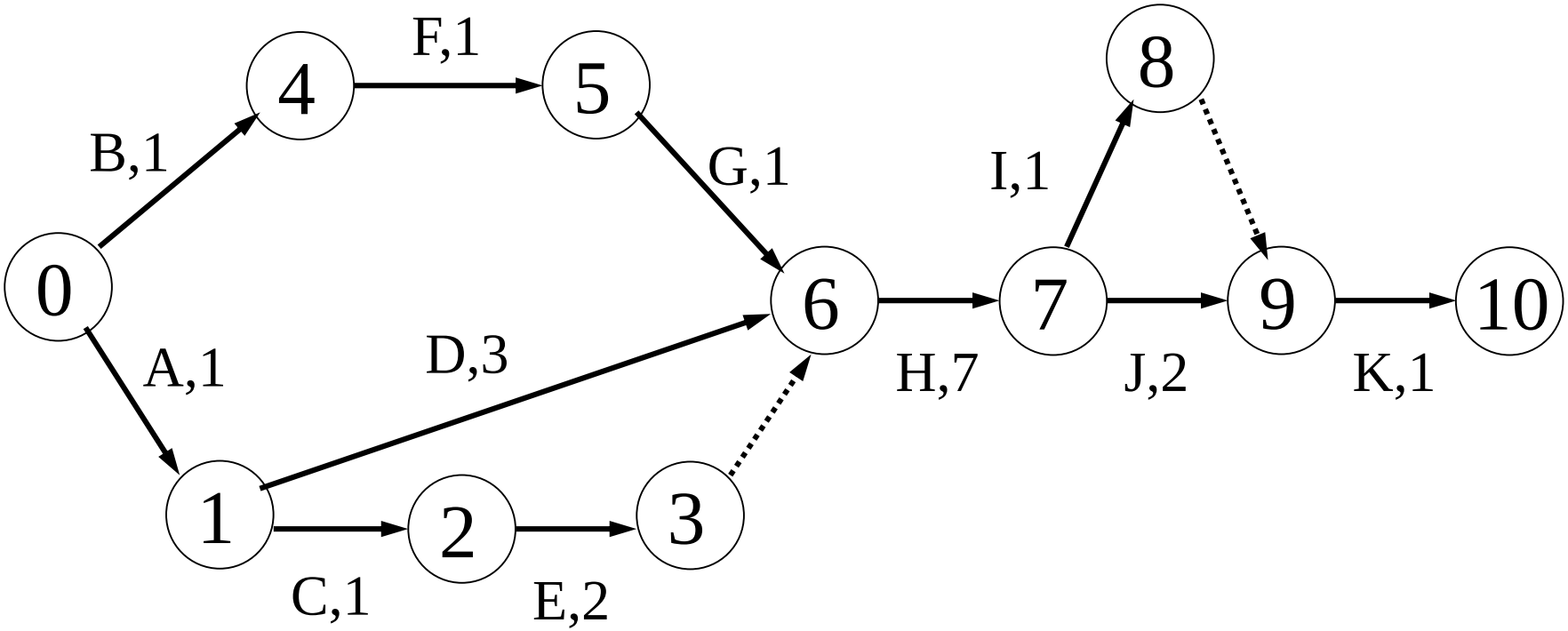
プロジェクトなどの工程管理

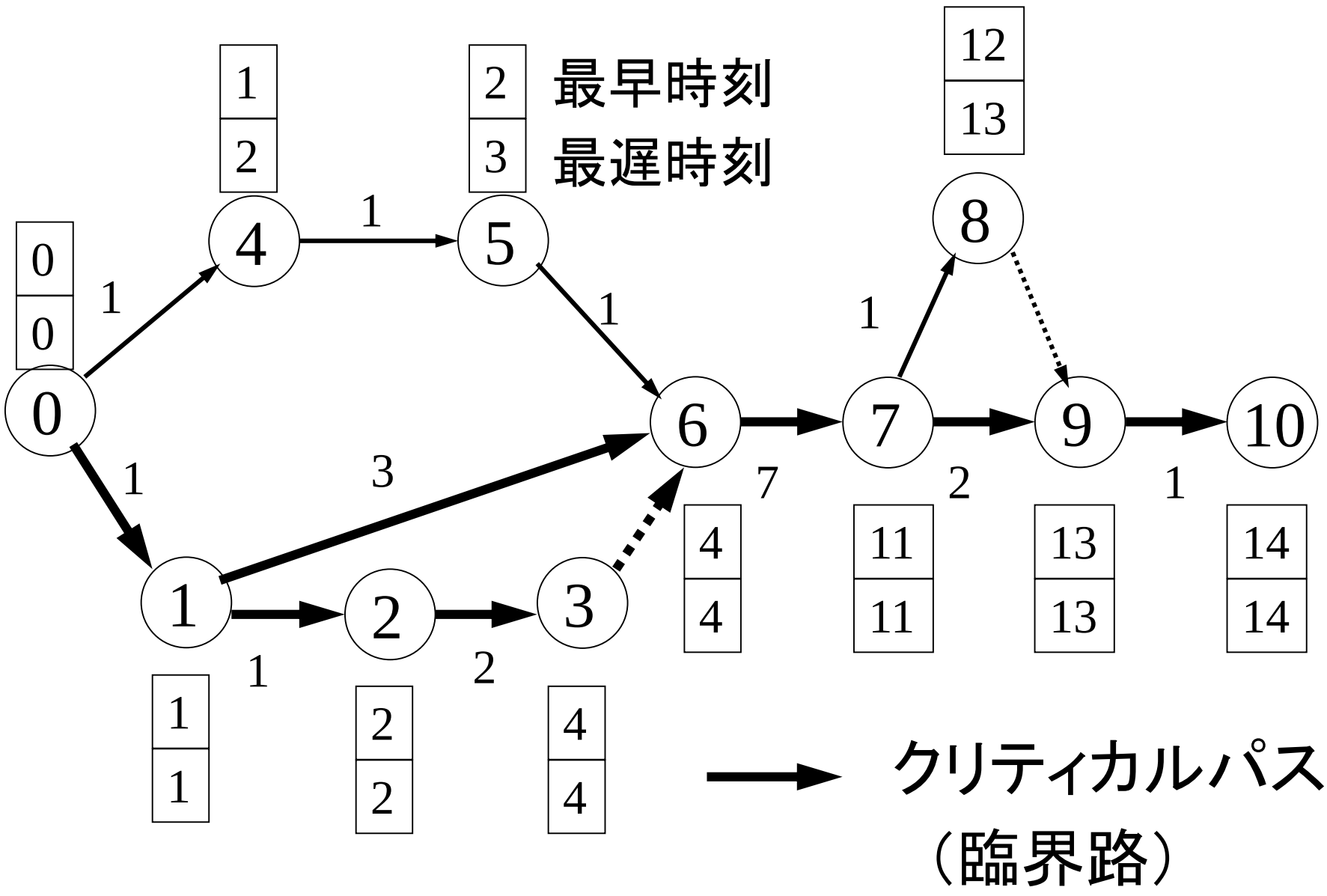
作業名	先行作業	所要日数
A	―	1
B	―	1
C	A	1
D	A	3
E	C	2
F	B	1
G	F	1
H	D, E ,G	7
I	H	1
J	H	2
K	I, J	1

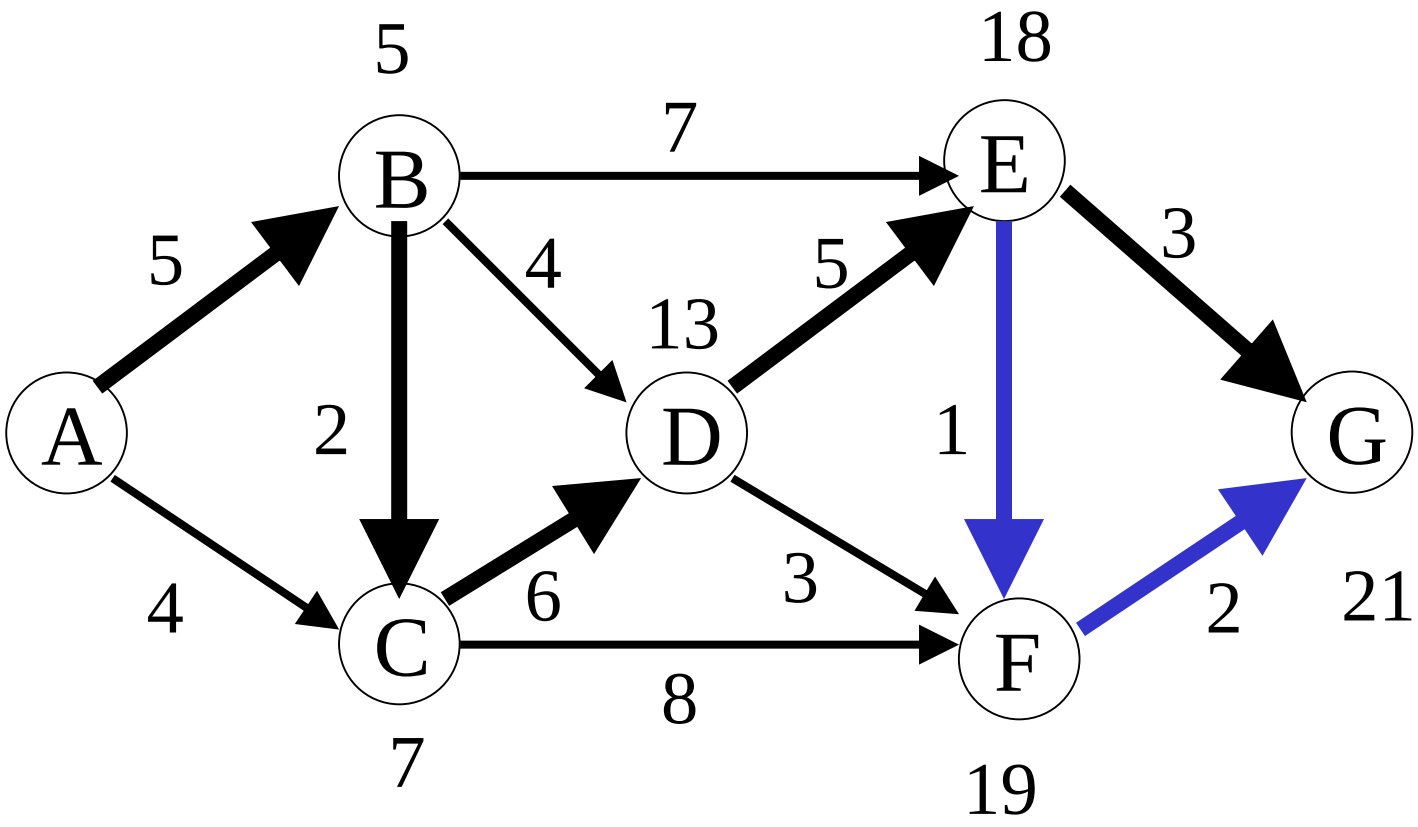
アメリカ海軍のポラリス潜水艦開発計画

D.G.Malcolm ら
(1950年代後半)

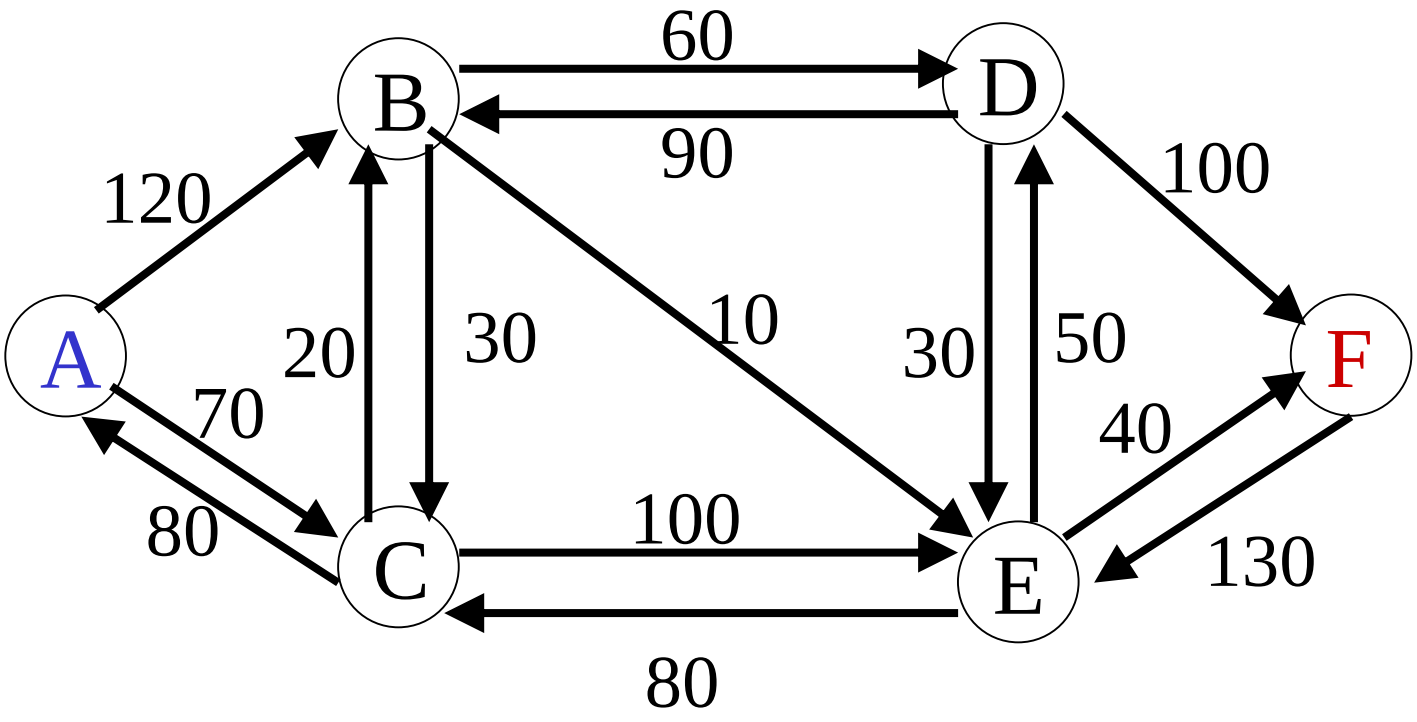
＜プロジェクト・ネットワーク＞




最早時刻＝点0からの最長路の長さ
最早時刻
最遅時刻
0
1
2
3
4
5
6
7
8
9
10
0
0
1
2
1
1
1
1
3
7
1
2
4
4
11
11
13
13
14
14
12
13
1
1
2
2
4
4
クリティカルパス
（臨界路）


5
18
7
B
E
5
4
5
3
13
A
2
D
1
G
4
6
3
2
21
C
F
8
7
19

＜最大流問題と最小費用流問題＞



A:ソース（フローを送り出す節点）

F:シンク（フローが流入する節点）

最大流問題

目的関数: $f \longrightarrow$ 最大

s:ソース

t:シンク

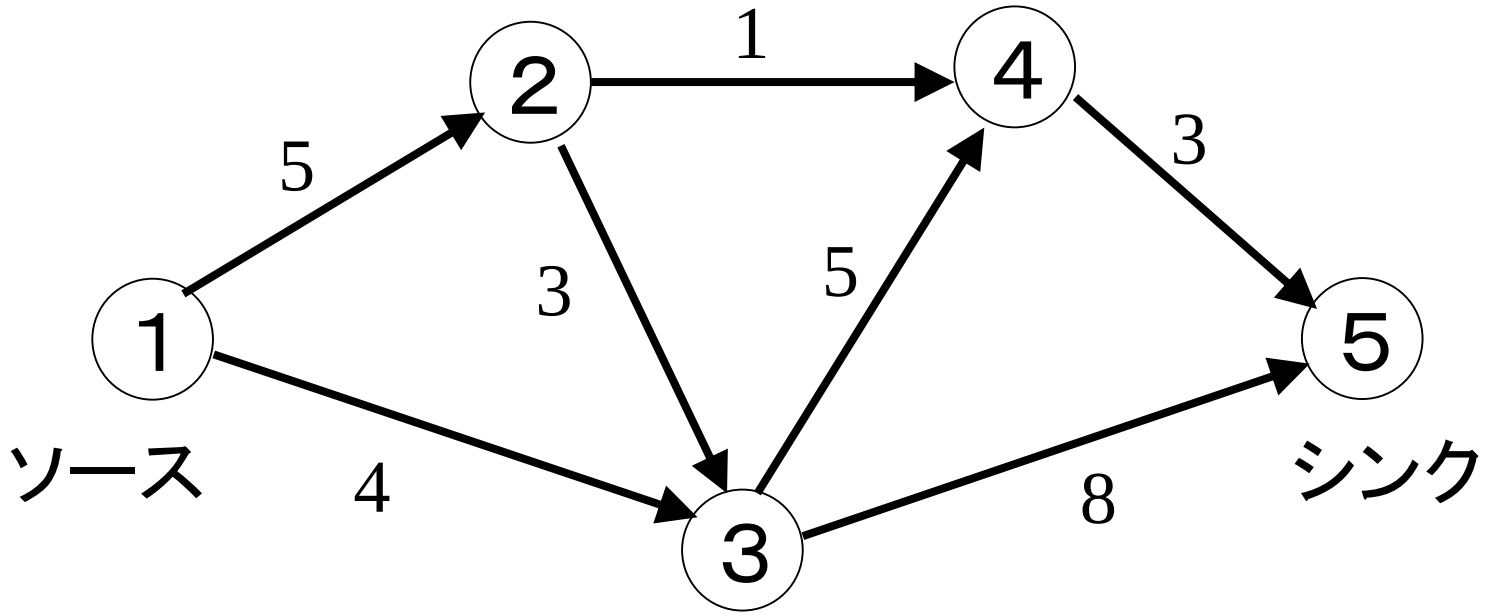
制約条件:

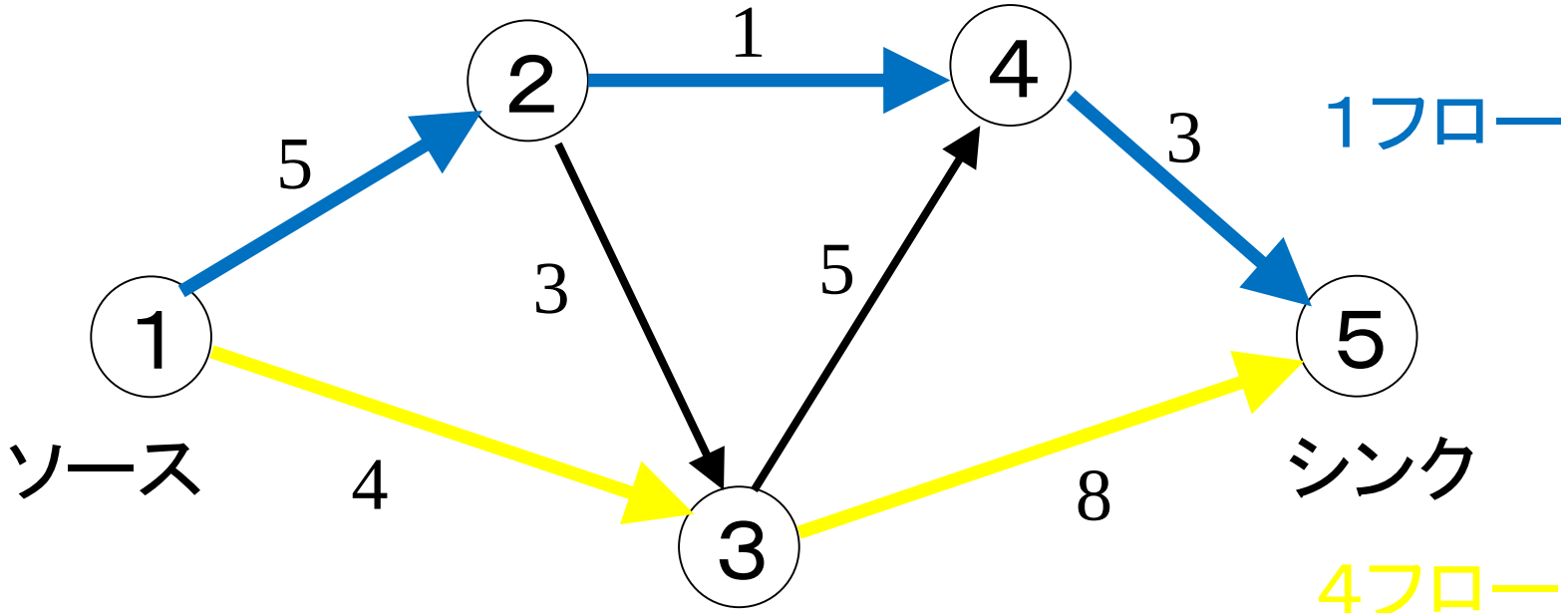
$$\sum_{\{j|(s,j)\in E\}} x_{sj} - \sum_{\{j|(j,s)\in E\}} x_{js} = f$$ 流出量

$$\sum_{\{j|(i,j)\in E\}} x_{ij} - \sum_{\{j|(j,i)\in E\}} x_{ji} = 0$$ $(i\in V-\{s,t\})$ 流れ保存則

$$\sum_{\{j|(t,j)\in E\}} x_{tj} - \sum_{\{j|(j,t)\in E\}} x_{jt} = -f$$ 流入量

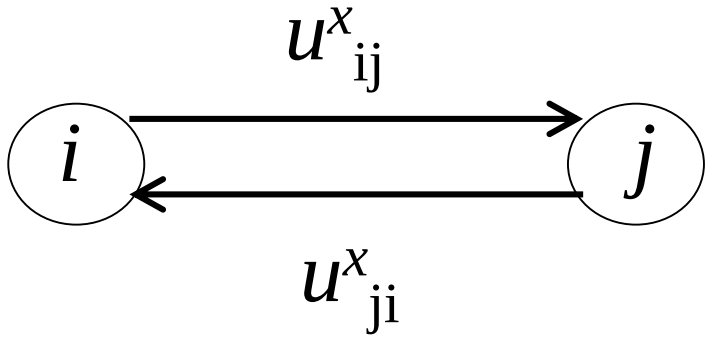
$$0 \leqq x_{ij} \leqq u_{ij} \quad ((i,j)\in E)$$ 容量制約条件


1
2
3
4
5
5
3
1
5
3
4
8
ソース
シンク


1
2
3
4
5
5
3
1
5
3
4
8
ソース
シンク
1フロー
4フロー

$\boldsymbol{x}=(x_{ij})$: フロー　　f: フロー$\boldsymbol{x}$の流量

あるフロー$\boldsymbol{x}$が与えられているとする



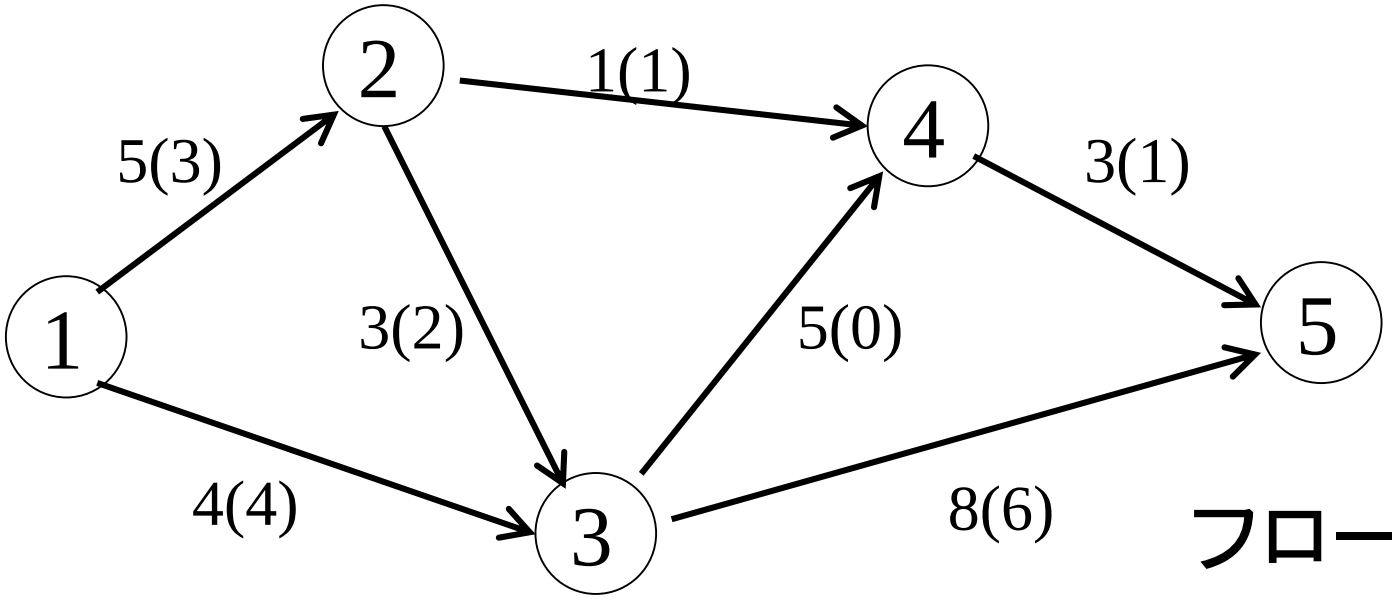
$u^x_{ij} = u_{ij} - x_{ij}$

$u^x_{ji} = x_{ij}$

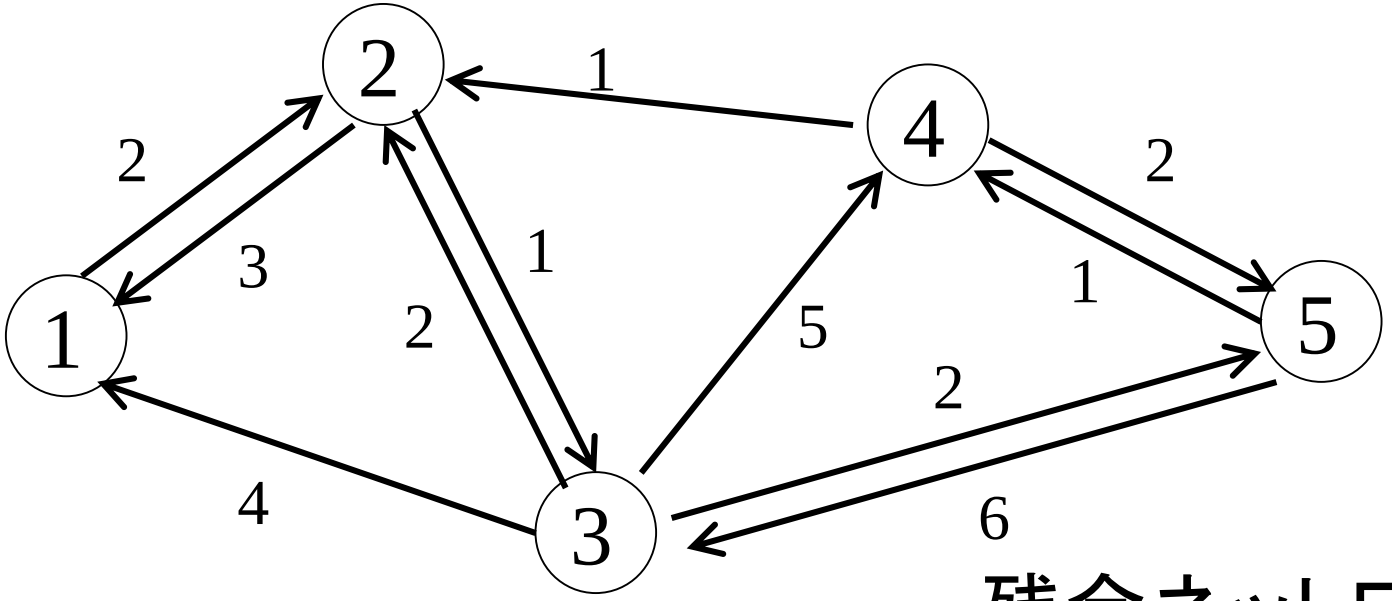
u^x_{ij} , u^x_{ji} : フロー$\boldsymbol{x}$に対する枝(i,j)の残余容量

$G^x=(V,E^x)$: フロー$\boldsymbol{x}$に対する残余ネットワーク

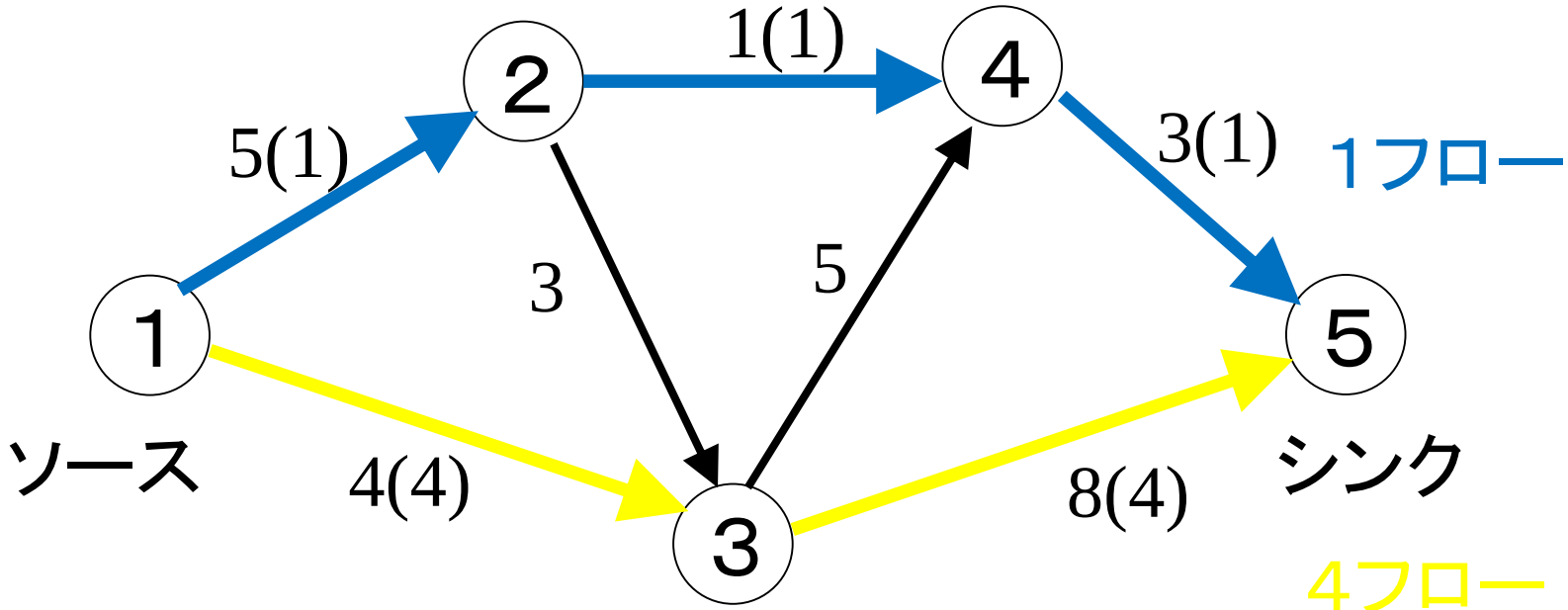
フロー増加路 : 残余ネットワークにおける
ソースからシンクへの路

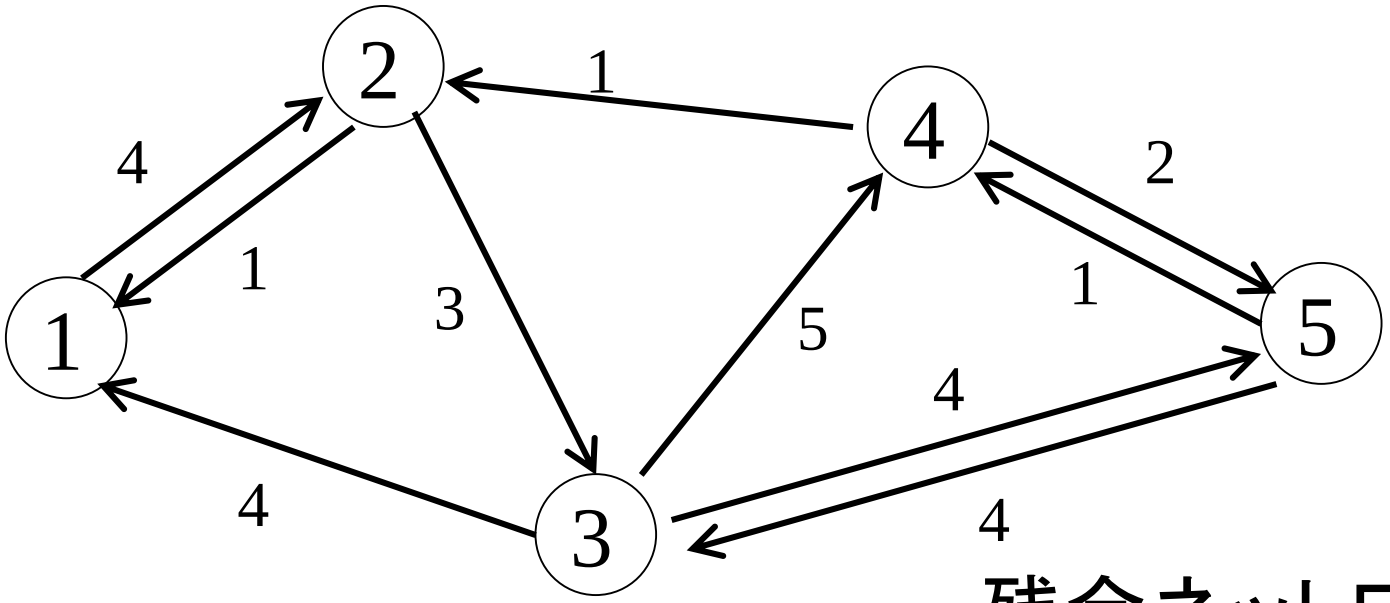


フローの例

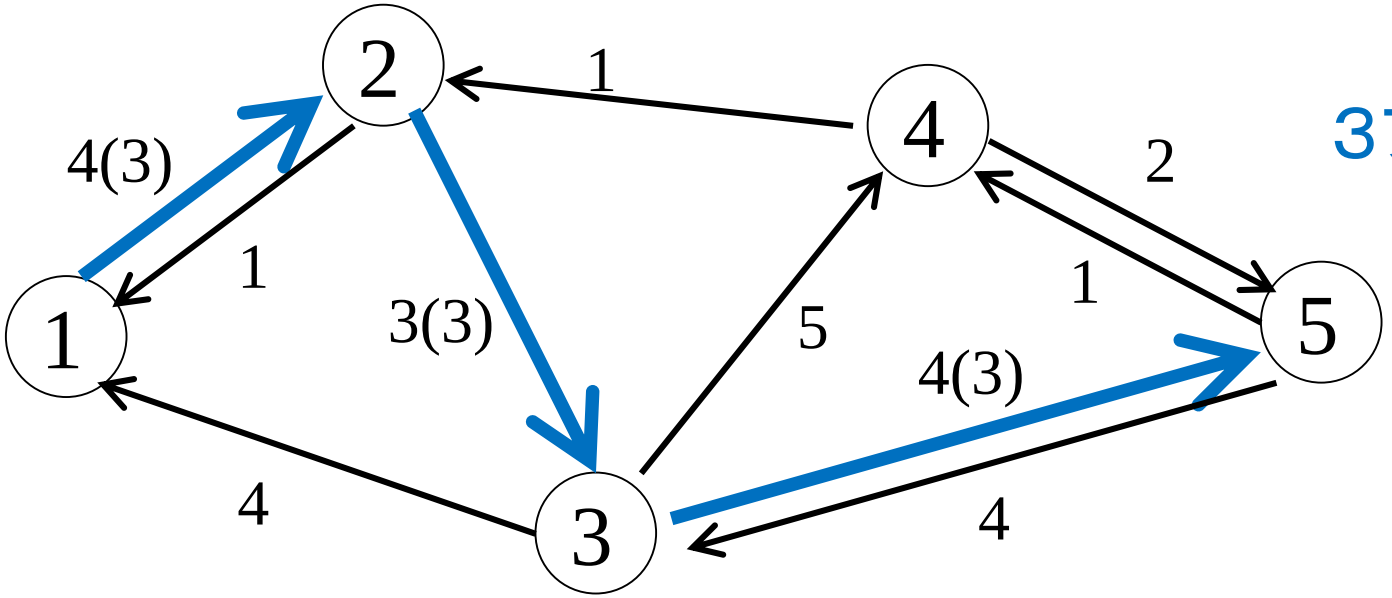


残余ネットワーク

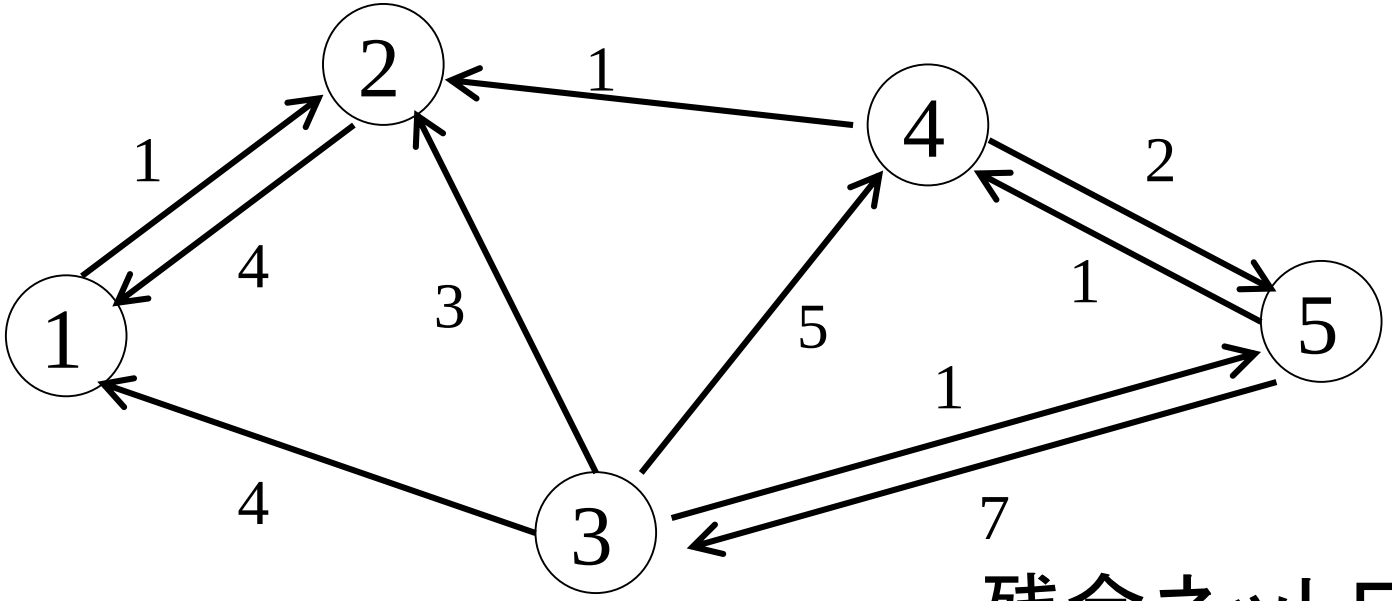




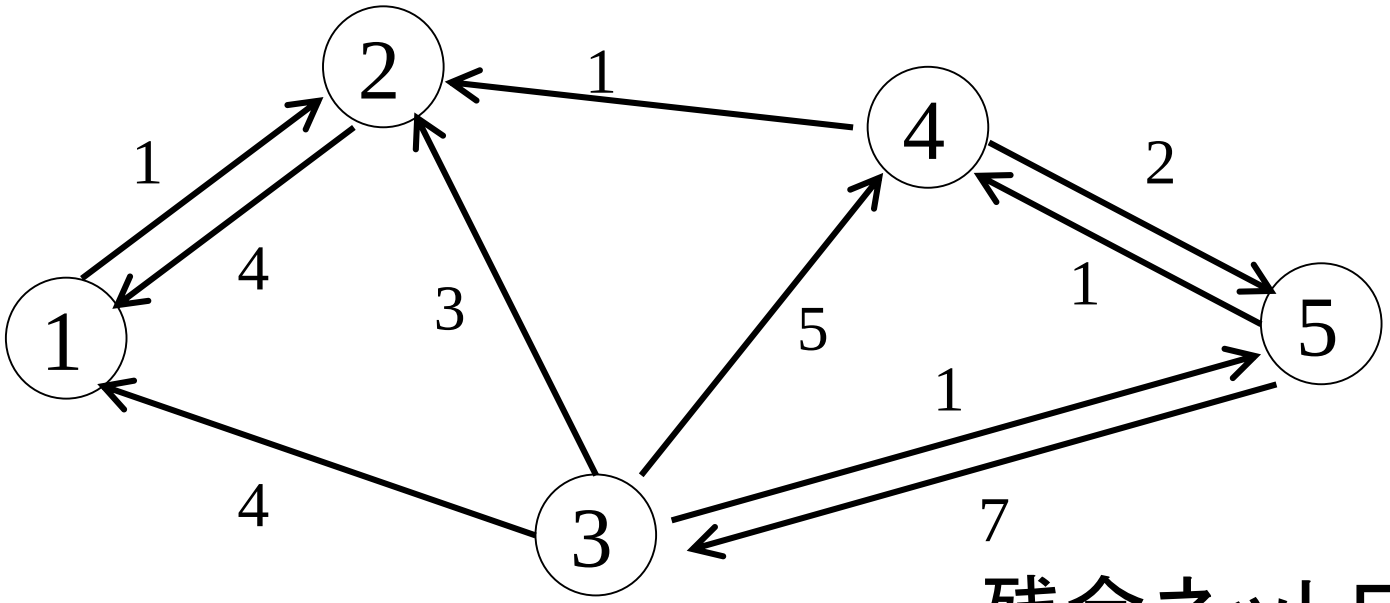
残余ネットワーク



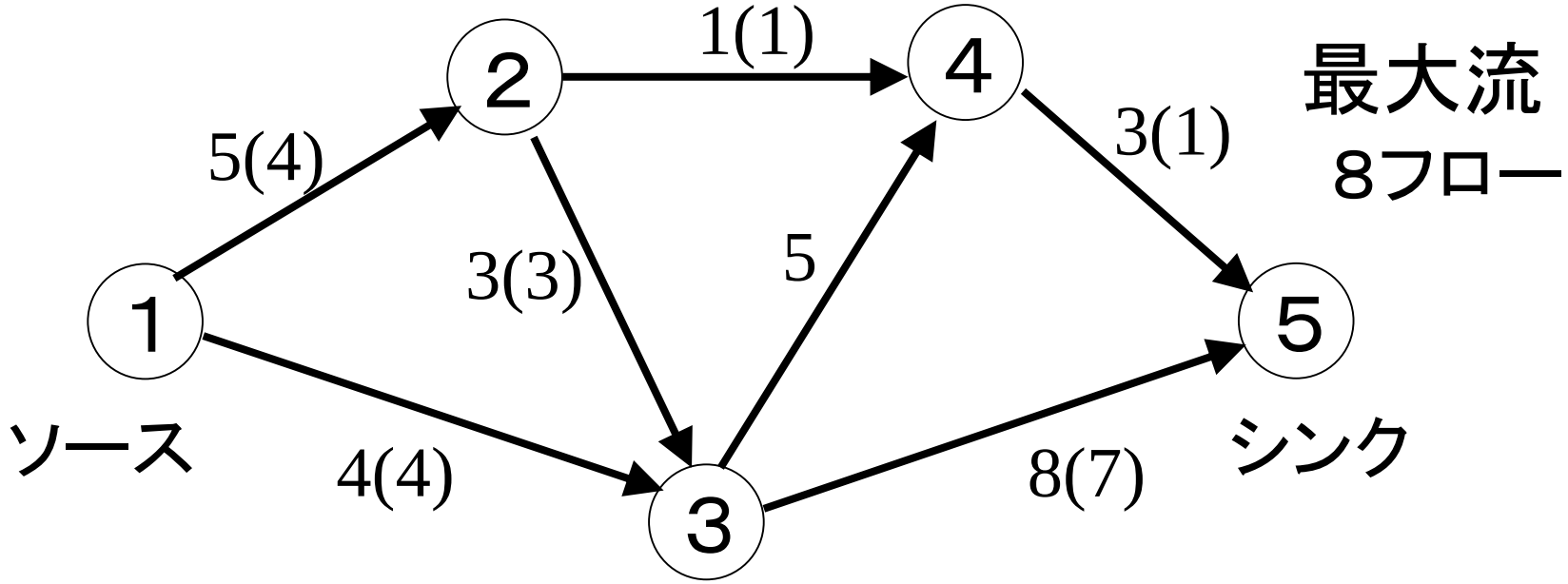
3フロー



残余ネットワーク



残余ネットワーク



最大流
8フロー

＜フロー増加法＞

(0) 適当な初期フロー$\boldsymbol{x}$を定める (例えば $x_{ij}=0$)

(1)フロー増加路を見つける.
　存在しなければ計算終了.

(2)フロー増加路に沿って可能な限りフローを追加し，新しいフロー$\boldsymbol{x}$を得る. ステップ(1)に戻る.

最大流最小カット定理

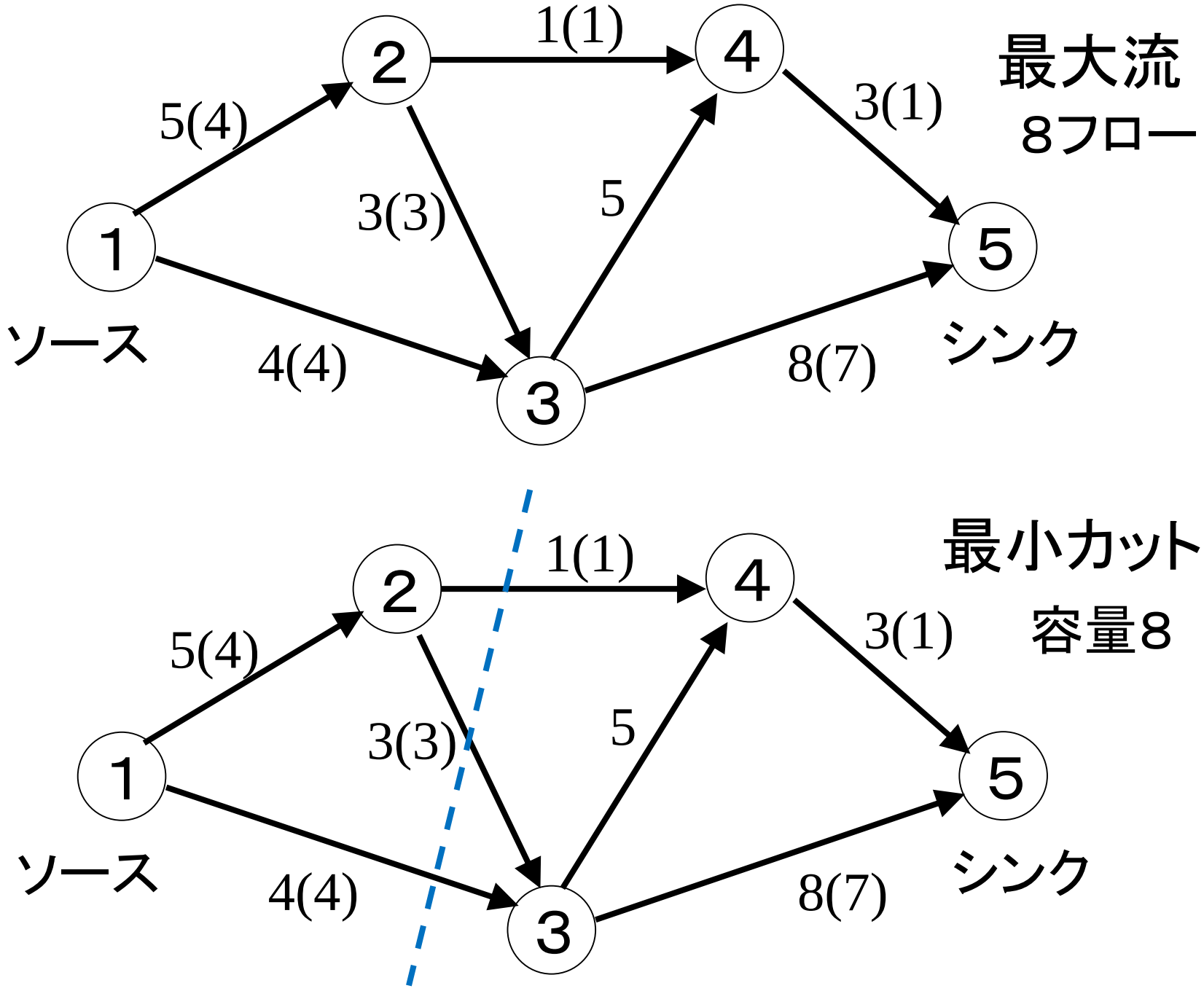
カット(S,T)：節点集合Vをソースsを含む集合Sと
シンクtを含む集合Tに分割したもの

C(S,T)：カット(S,T)の容量

$$C(S,T) = \sum_{(i,j)\in(S,T)} u_{ij}$$

$$f = \sum_{(i,j)\in(S,T)} x_{ij} - \sum_{(j,i)\in(T,S)} x_{ji} \leqq C(S,T)$$

$f^* = C(S^*,T^*)$　フロー増加法の終了時点


1(1)
2
4
最大流
8フロー
5(4)
3(1)
3(3)
5
1
5
ソース
4(4)
3
8(7)
シンク
最小カット
容量8

最大流問題の計算量

フロー増加法(Ford, Fulkerson(1956)):

$O(mf_{max})=O(m^2U)$, $U=\max\{u_{ij}|(i,j)\in E\}$

多項式時間アルゴリズムではない

Edmonds, Karp (1969): $O(m^2n)$

Dinic (1970): $O(mn^2)$

プリフロープッシュ法(Goldberg, Tarjan): $O(mn^2)$

改良版: $O(n^2m^{1/2})$

最小費用流問題

目的関数: $c_{ij}x_{ij} \longrightarrow$ 最小

制約条件:

$$\sum_{\{j|(i,j)\in E\}} x_{ij} - \sum_{\{j|(j,i)\in E\}} x_{ji} = b_i \qquad (i\in V)$$

需要・供給量

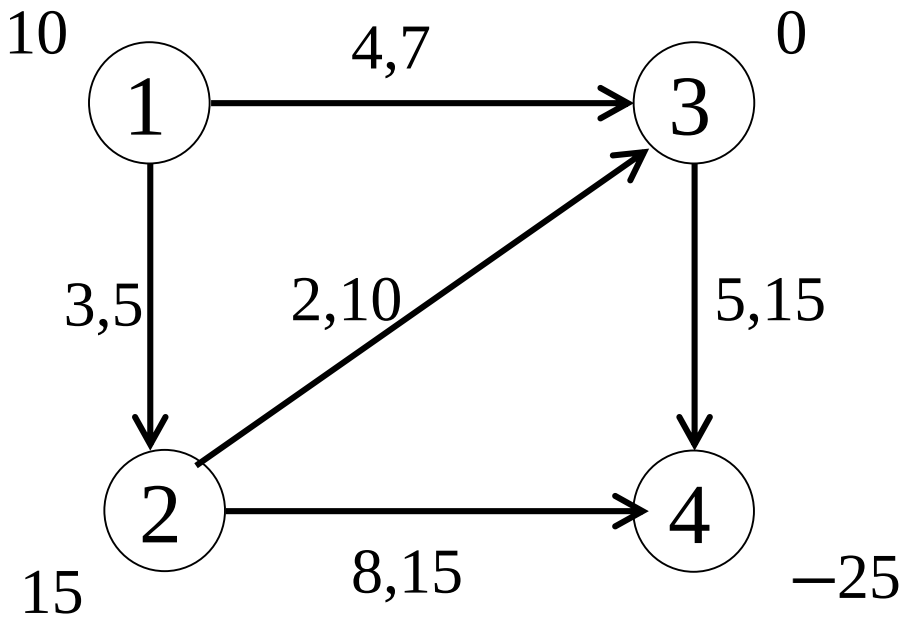
$$0 \leqq x_{ij} \leqq u_{ij} \qquad ((i,j)\in E)$$

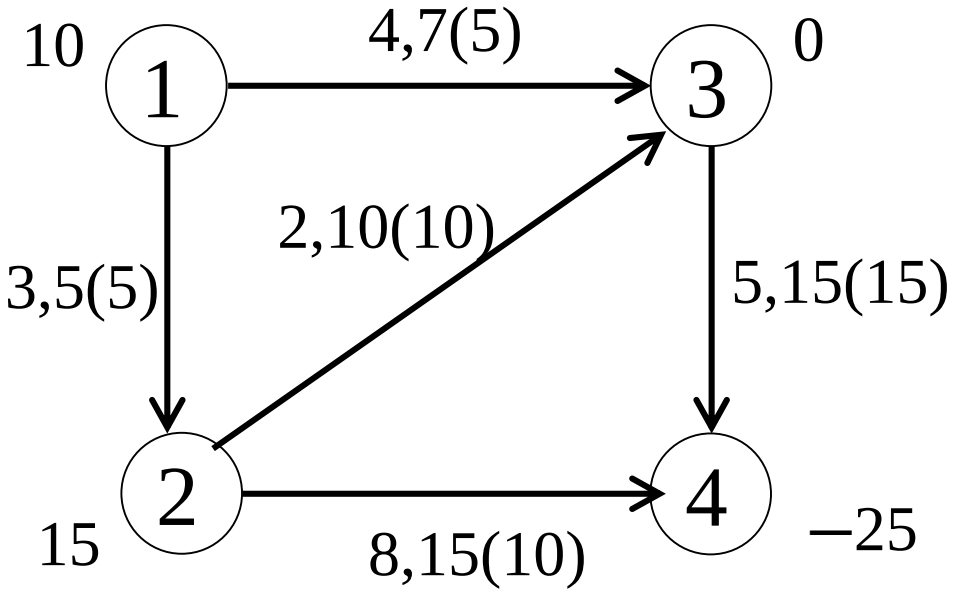
容量制約条件

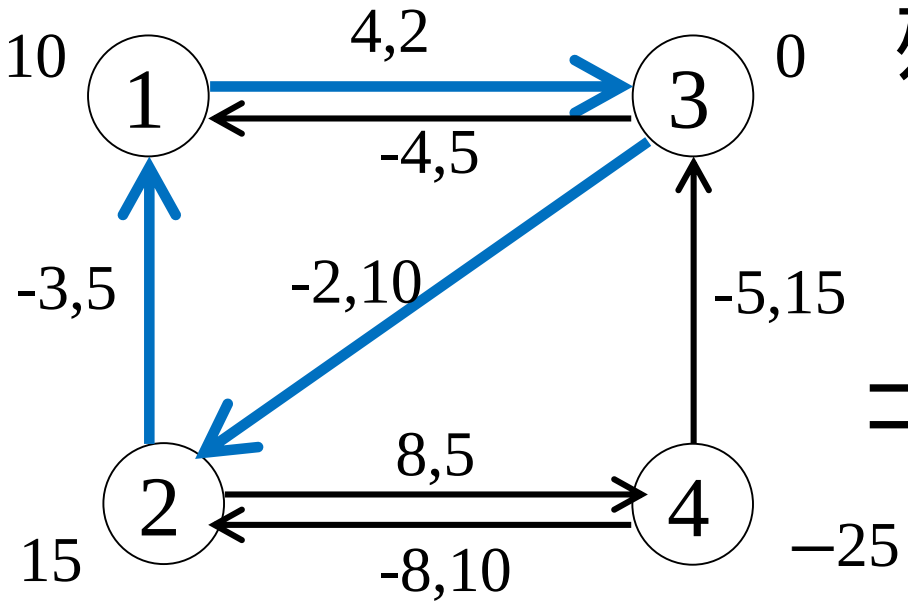
c_{ij} : 枝(i,j)の１単位の流れに対するコスト

$$\sum_{i\in V} b_i = 0$$

＜最小費用流問題＞







残余ネットワーク

負閉路

コスト：4−2−3= −1

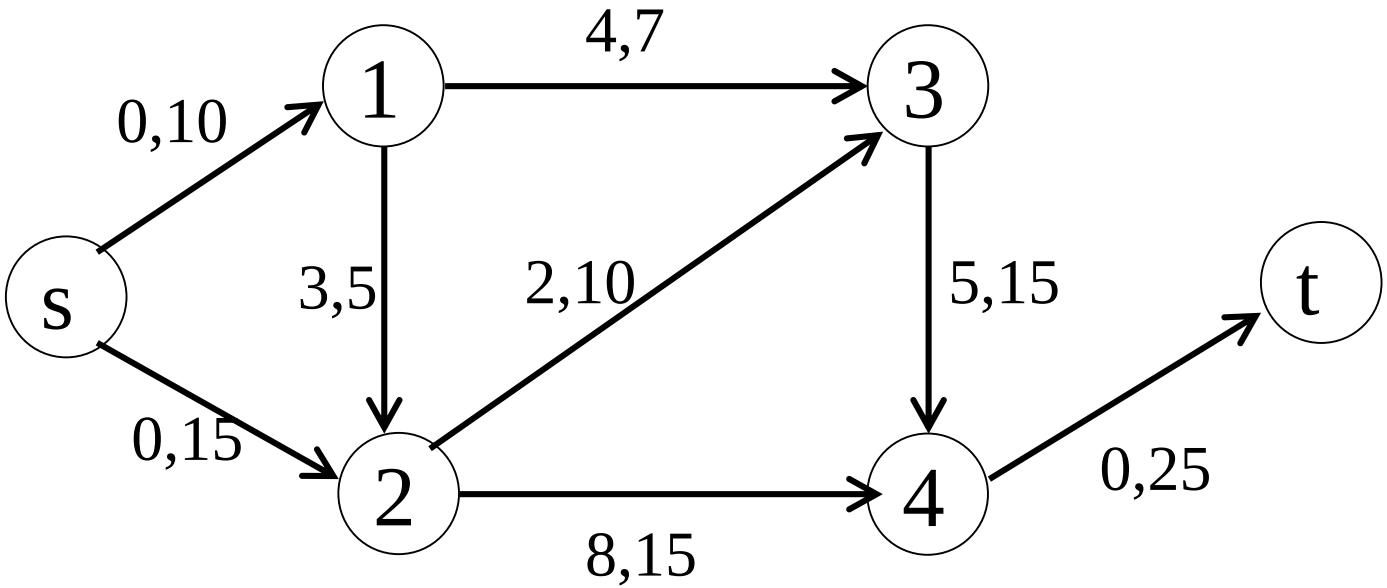
2フロー

＜負閉路除去法＞

(0) 適当な初期フローxを定める

(1)残余ネットワーク$G^x=(V,E^x)$において負閉路を見つける．存在しなければ計算終了．

(2)負閉路に沿って可能な限りフローを追加し，新しいフローxを得る．ステップ(1)に戻る．



$$f = 25 = \sum_{b_i>0\ (i \in V)} b_i$$

$$u_{si} = b_i$$

$$u_{it} = -b_i$$

$$c_{si} = c_{it} = 0$$

最小費用流問題

目的関数: $c_{ij}x_{ij} \longrightarrow$ 最小

s:ソース

t:シンク

制約条件:

$$\sum_{\{j|(s,j)\in E\}} x_{sj} - \sum_{\{j|(j,s)\in E\}} x_{js} = f \qquad \text{流出量}$$

$$\sum_{\{j|(i,j)\in E\}} x_{ij} - \sum_{\{j|(j,i)\in E\}} x_{ji} = 0 \qquad (i\in V-\{s,t\})$$

流れ保存則

$$\sum_{\{j|(t,j)\in E\}} x_{tj} - \sum_{\{j|(j,t)\in E\}} x_{jt} = -f \qquad \text{流入量}$$

$$0 \leqq x_{ij} \leqq u_{ij} \qquad ((i,j)\in E) \qquad \text{容量制約条件}$$

＜フロー増加法＞

(0) 適当な初期フローxを定める (例えば $x_{ij}=0$)

(1)残余ネットワーク$G^x=(V,E^x)$においてソースからシンクへの最短路を求める .
存在しなければ計算終了.

(2)最短路に沿って可能な限りフローを追加し, 新しいフローxを得る. 流量＝ f になれば計算終了. そうでなければステップ(1)に戻る.

最小費用流問題の計算量

負閉路除去法：反復回数=O(mCU)

$C=\max\{c_{ij}|(i,j)\in E\}$, $U=\max\{u_{ij}|(i,j)\in E\}$

多項式時間アルゴリズムではない

改良版：$O(n^2m^3\log n)$

フロー増加法：$O(n^2f)$

Goldberg, Tarjan(1988)：$O(nm^2\log^2 n)$

最小費用循環流問題

目的関数: $c_{ij}x_{ij} \longrightarrow$ 最小

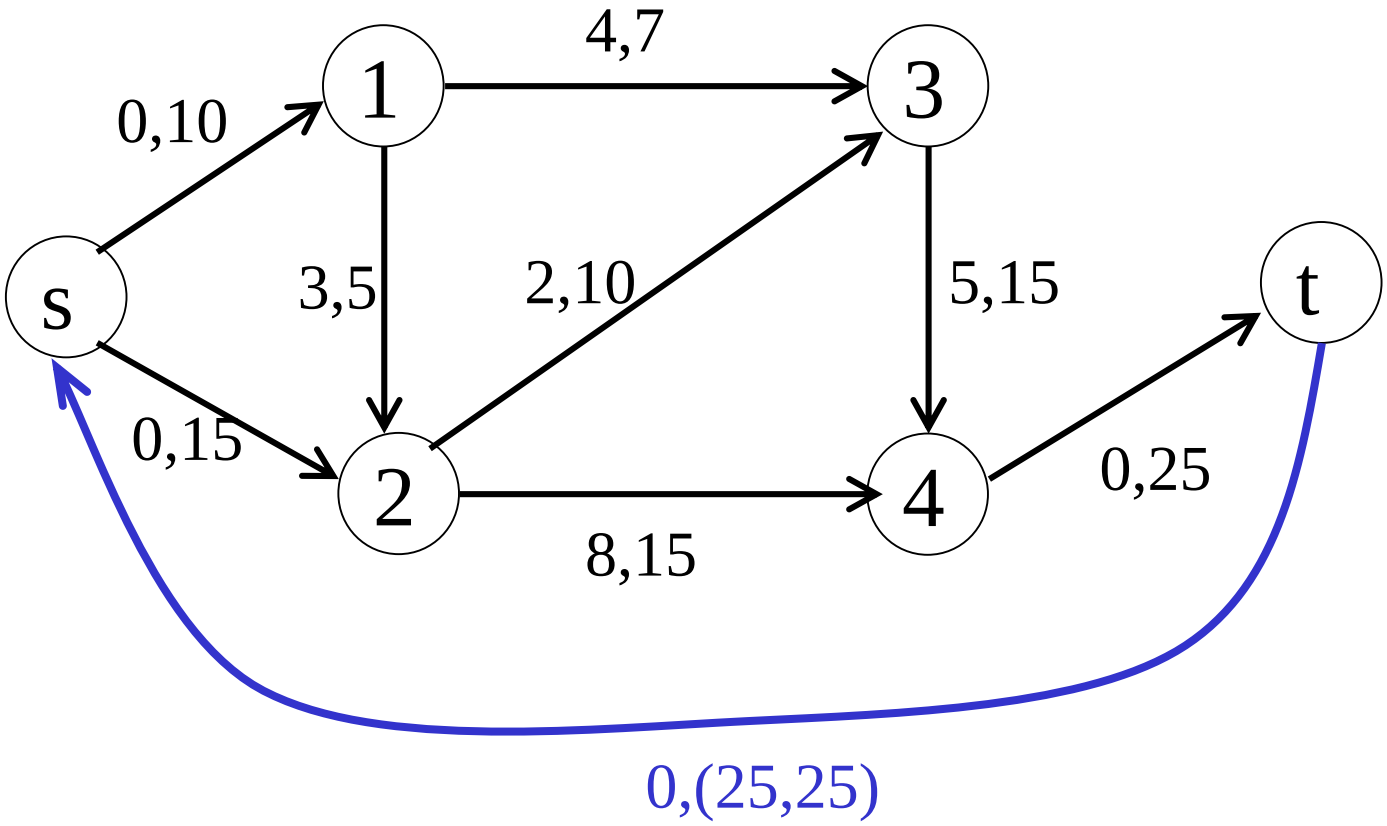
制約条件:

$$\sum_{\{j|(i,j)\in E\}} x_{ij} - \sum_{\{j|(j,i)\in E\}} x_{ji} = 0 \qquad (i\in V)$$

流れ保存則

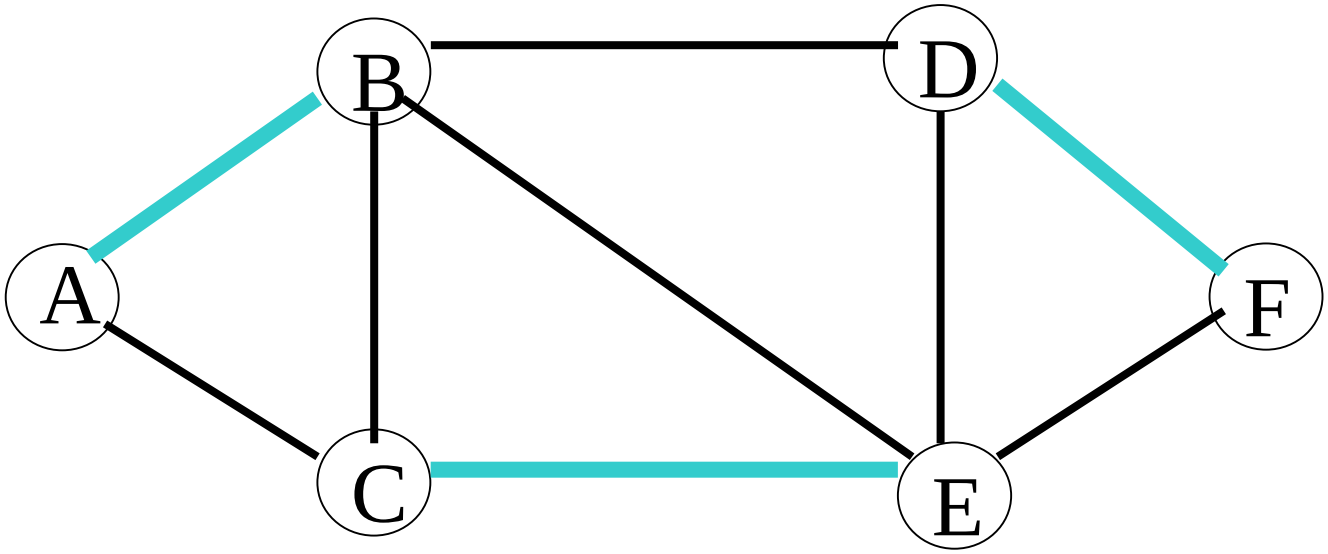
$$l_{ij} \leqq x_{ij} \leqq u_{ij} \qquad ((i,j)\in E)$$

容量制約条件


4,7
1
3
0,10
s
3,5
2,10
5,15
t
0,15
2
4
0,25
8,15
0,(25,25)

マッチング問題

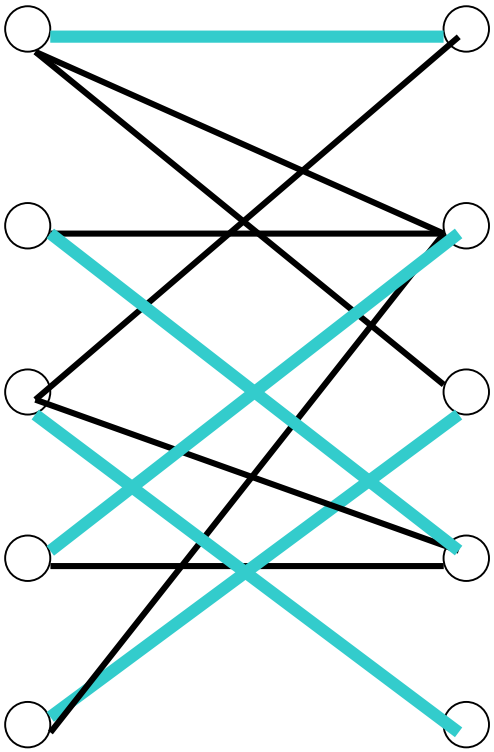
マッチングM（M⊂E）：どの相異なる2つの枝
k,l∈ Mも端点を共有しない



最大マッチング問題の計算量：$O(n^3)$
最適 k-マッチング問題の計算量：$O(kn^2)$

割当問題

２部グラフ



２部グラフの最大マッチング

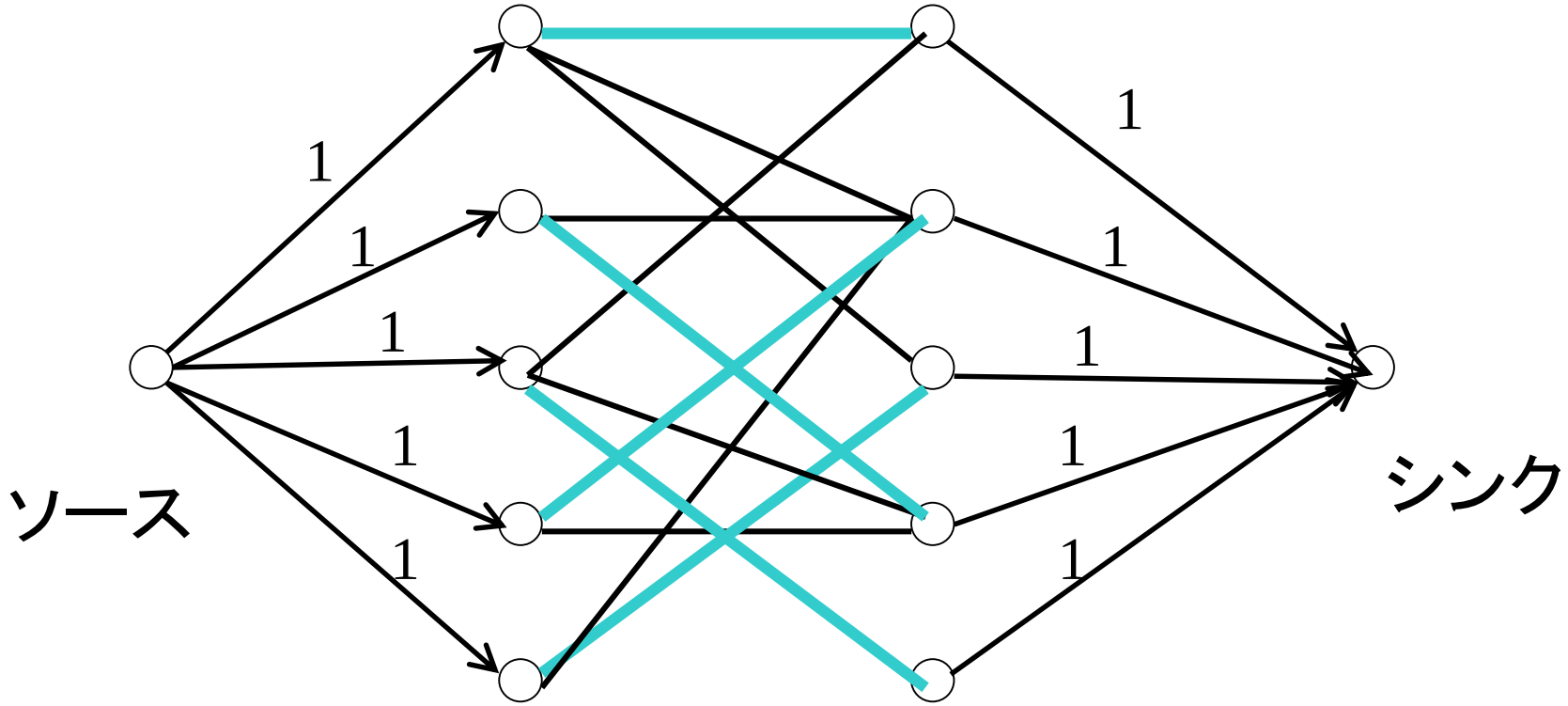
計算量：$O(n^{2.5})$

⟶ 最大流問題

２部グラフの最適k-割当

計算量：$O(kn^2)$

⟶ 最小費用流問題


1
1
1
1
1
1
1
1
1
1
ソース
シンク