算法数理工学
第11回

定兼　邦彦

成長法による最適全域木の構成

- 「繰り返しの各ステップの直前で，A はある最小全域木の部分集合」というループ不変式を維持
- 安全な辺=それを加えてもループ不変式を満たす

$A := \emptyset$

while A は最小全域木ではない

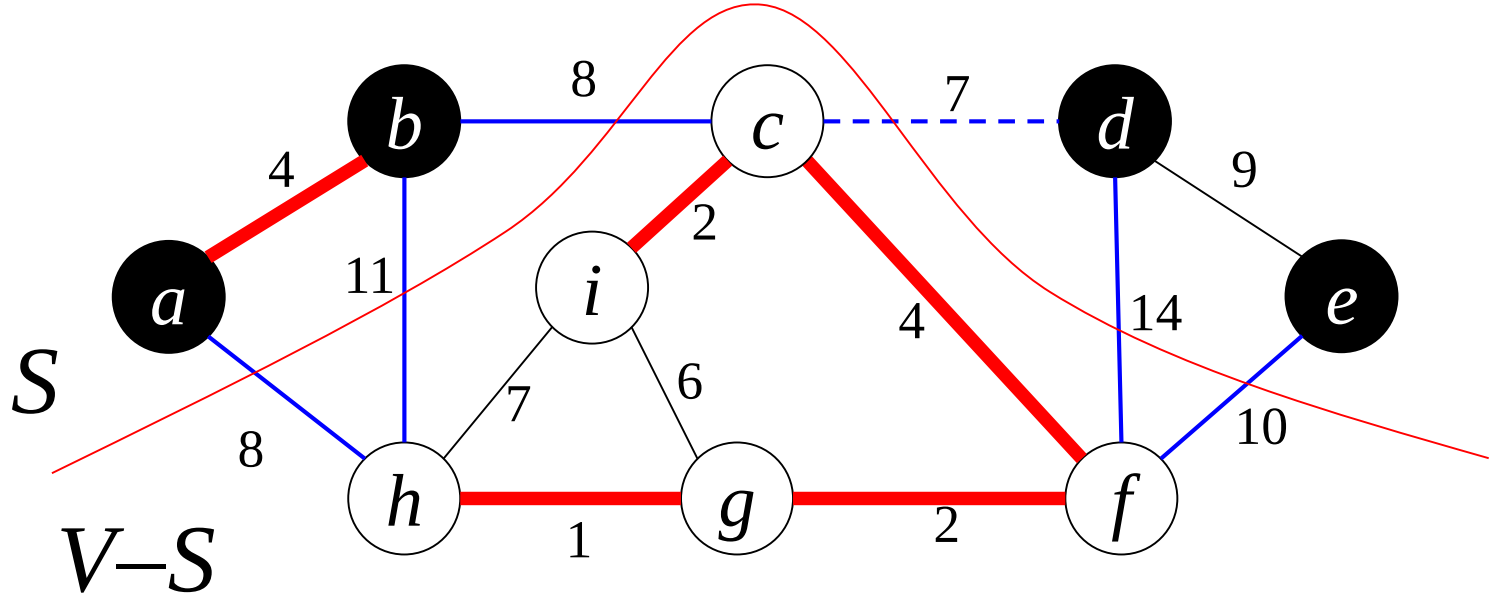
　　A に対して安全な辺 (u,v) を求める

　　$A := A \cup \{(u,v)\}$

return A

定義

- 無向グラフ $G = (V, E)$ のカット $(S, V–S)$ とは，V の分割
- 辺 $(u,v) \in E$ の一方の端点が S にあり，他方が $V–S$ にあるとき，(u,v) はカット $(S, V–S)$ と交差するという
- 辺集合 A に属するどの辺もカットと交差しないとき，このカットは A を侵害しないという
- カットと交差する辺の中で重み最小の辺を軽い辺 (light edge) という



カットと交差する辺

軽い枝

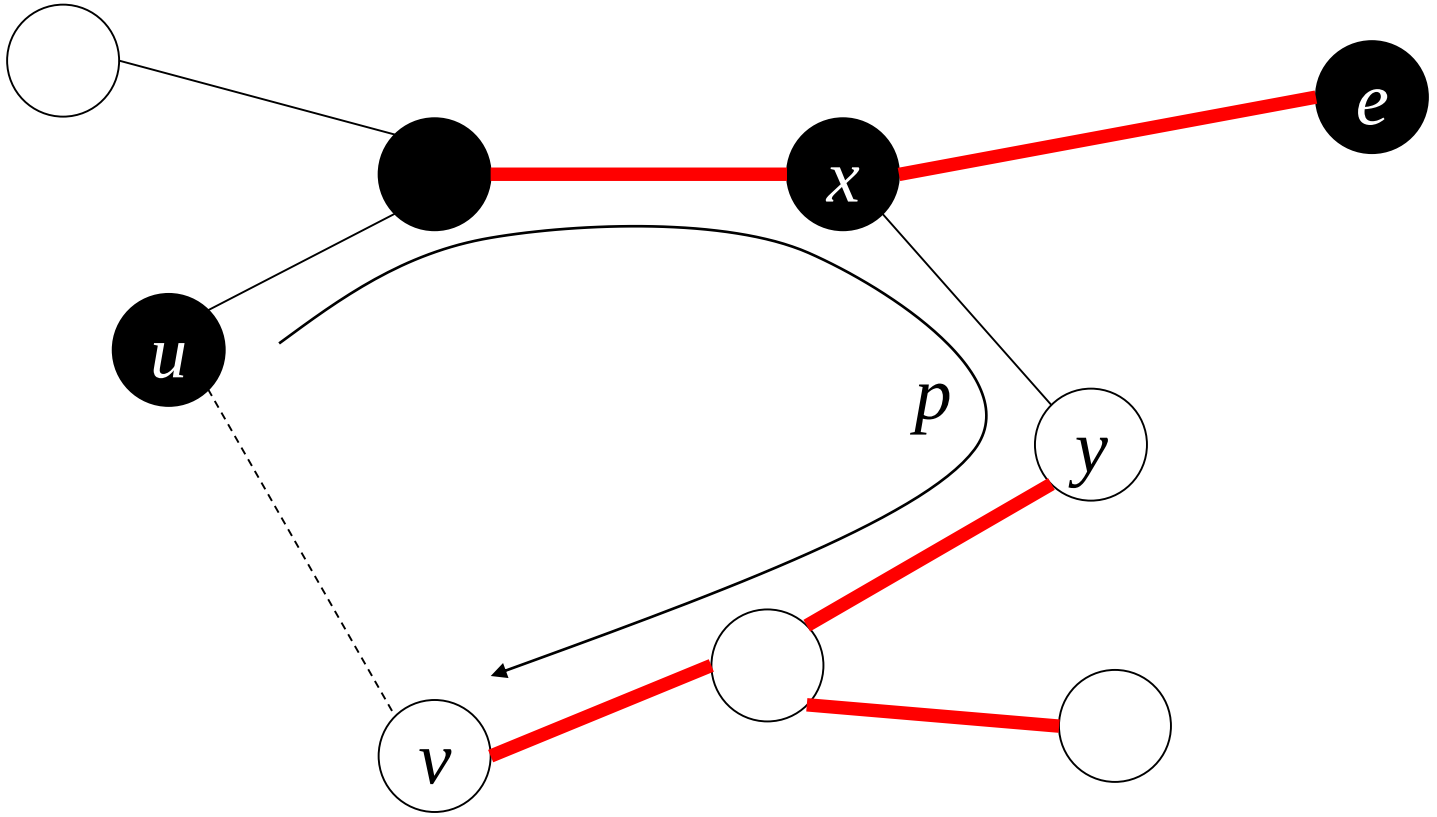
A: 最小全域木の枝の部分集合

定理 1: グラフ $G = (V, E)$ のある最小全域木に含まれる E の任意の部分集合を A, A を侵害しない G の任意のカットを $(S, V–S)$, そのカットと交差する任意の軽い辺を (u,v) とする. このとき, 辺 (u,v) は A に対して安全である.

証明: A を含む任意の最小全域木 T に対し,

- T が (u,v) を含む→OK
- T が (u,v) を含まない→$A \cup \{(u,v)\}$ を含む別の最小全域木 T' が存在することを示す.

T の u から v への経路 p 上の辺に (u,v) を加えると閉路ができる. u と v はカット $(S, V–S)$ の反対側にあるから, 経路 p 上にこのカットと交差する T の辺が少なくとも1つ存在する.
このような辺の1つを (x,y) とする.
カット $(S, V–S)$ は A を侵害しないから, A は辺 (x,y) を含まない. (x,y) は T における u から v への唯一の経路上にあるから, (x,y) を削除すると T は2つの成分に分かれる. これに (u,v) を加えると新たな全域木 $T' = T–\{(x,y)\}\cup\{(u,v)\}$ ができる.



(u,v) はカットと交差する軽い辺
(x,y) は T における u から v への唯一の経路 p の辺
T から辺 (x,y) を削除し，辺 (u,v) を加えることにより
(u,v) を含む最小全域木 T' が形成できる.

T' が最小全域木であることを示す.

(u,v) はカットと交差する軽い辺であり, (x,y) もこのカットと交差するから, $w(u,v) \leq w(x,y)$ である.

$$w(T') = w(T) - w(x,y) + w(u,v) \leq w(T)$$

となるが, T は最小全域木なので $w(T) \leq w(T')$.

従って, T' も最小全域木である.

(u,v) が A に対する安全な辺であることを示す.

$A \subseteq T$ かつ $(x,y) \notin A$ より, $A \subseteq T'$ である.

従って, $A \cup \{(u,v)\} \subseteq T'$ である.

T' は最小全域木なので, (u,v) は安全である.

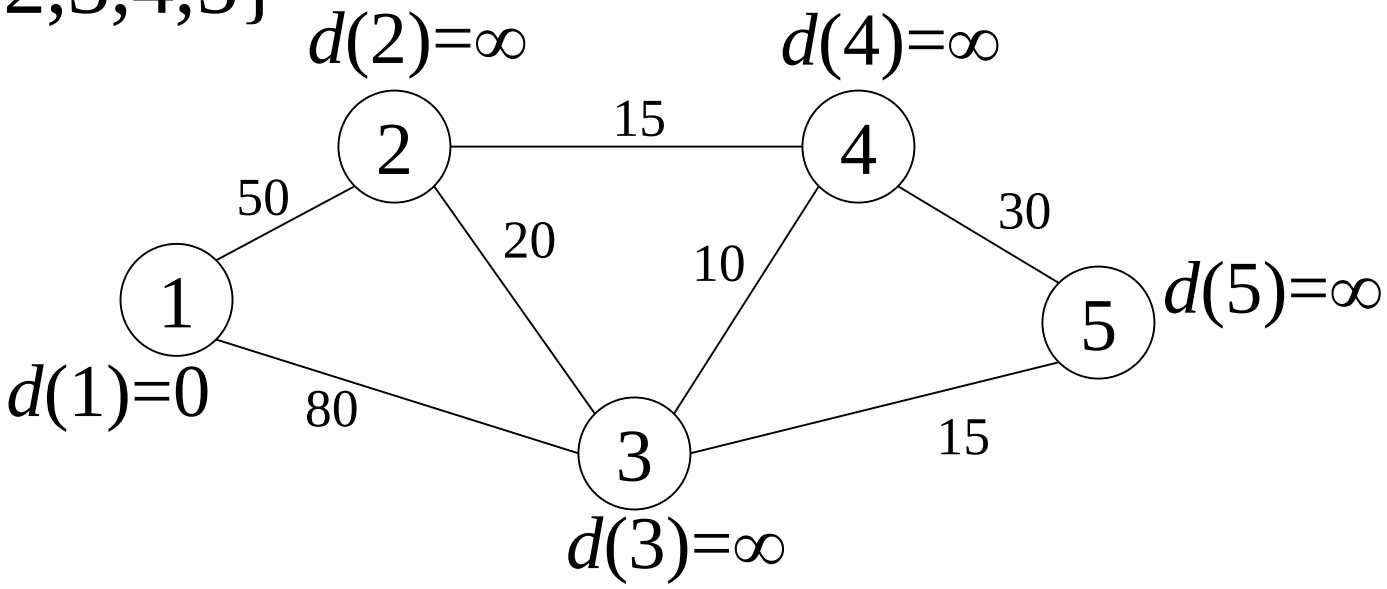
プリム (Prim) のアルゴリズム

- 最小全域木の部分集合 A を常に連結にしながら更新していくアルゴリズム
- A に含まれない点の集合を Q とする
- 初期状態は $Q = V - \{r\}$ (r はある任意の点)
- カットは $(Q, V–Q)$ になっている
- ダイクストラ法に似ている

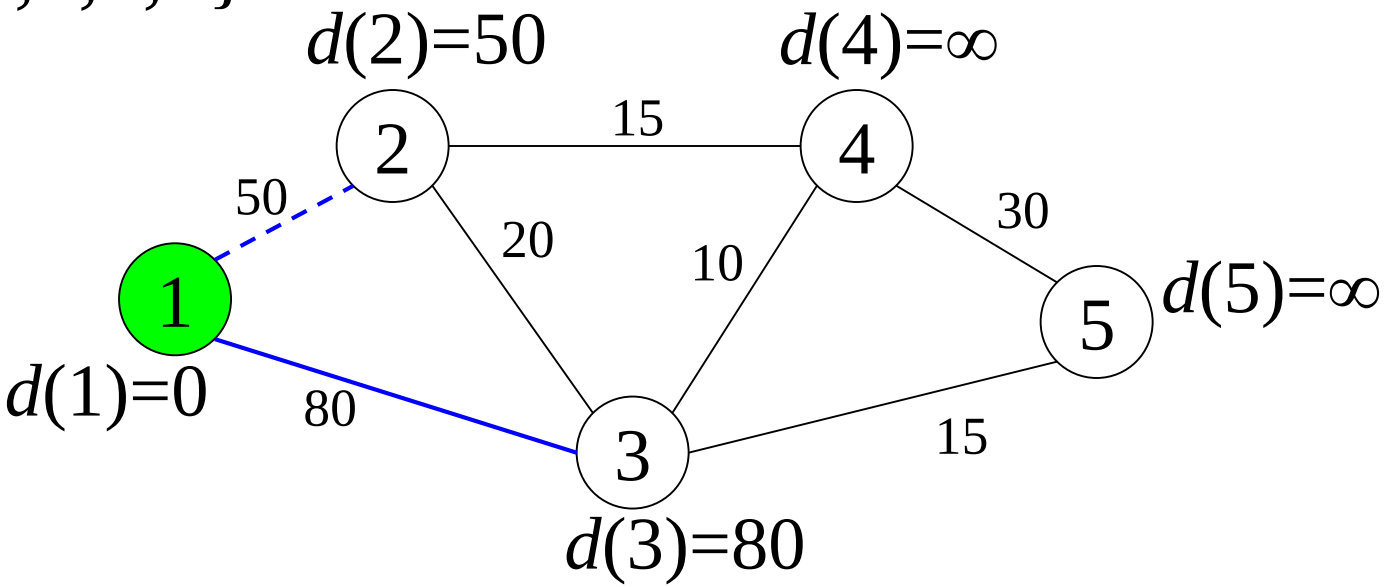
mst_prim(G, w, r)

1. for each $u \in V(G)$ $d[u] := \infty$
2. $d[r] := 0$; $\pi[r] :=$ NIL
3. $Q := V(G)$;
4. while $Q \neq \emptyset$
5. $u :=$ extract_min(Q)
6. for each $v \in Adj[u]$
7. if $v \in Q$ かつ $w(u,v) < d[v]$ then
8. $\pi[v] := u$
9. $d[v] := w(u,v)$

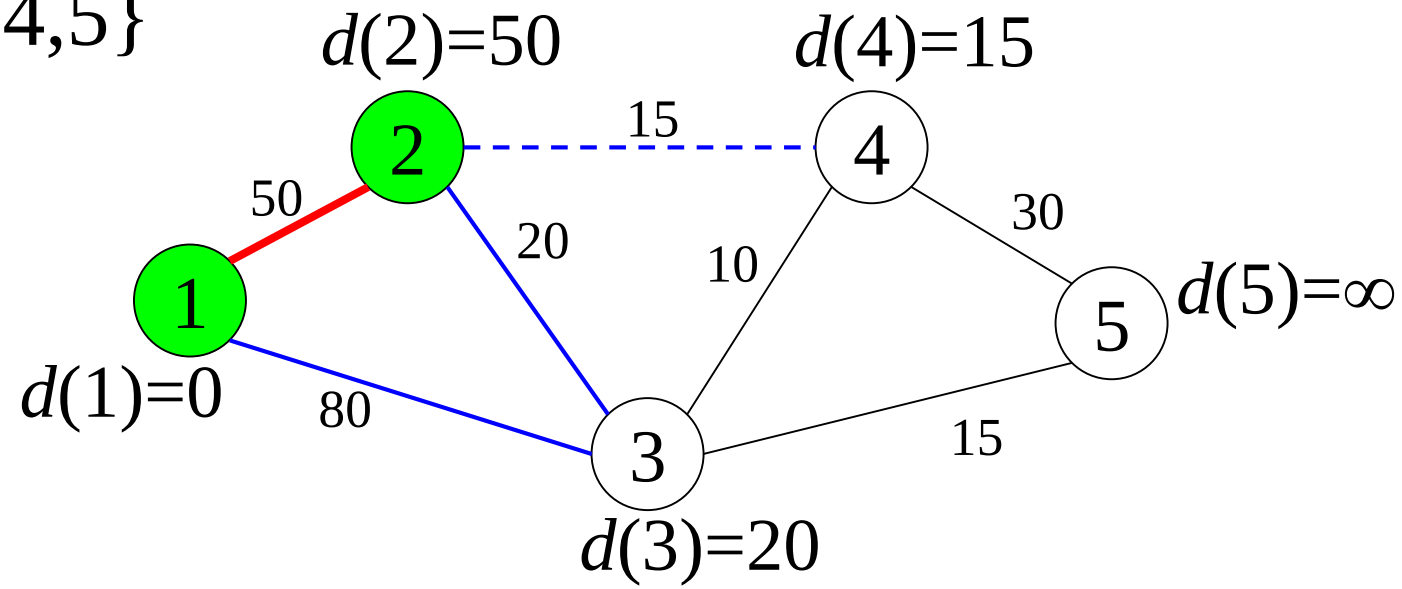
$Q = \{1,2,3,4,5\}$



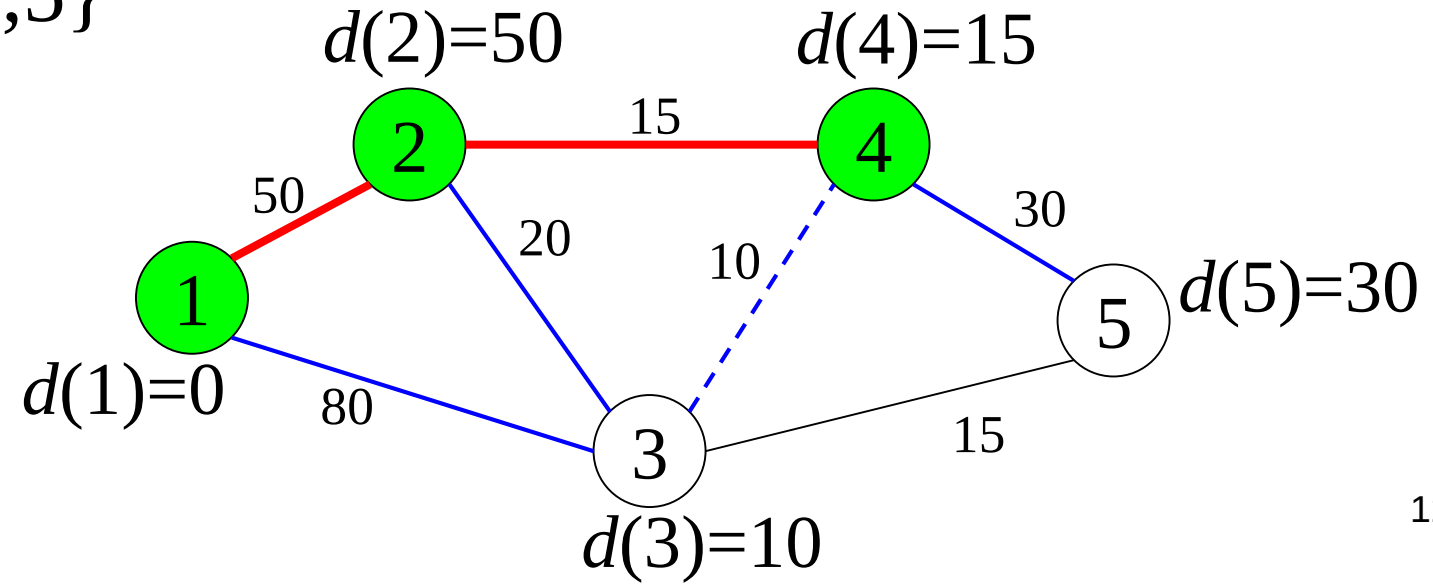
$Q = \{2,3,4,5\}$



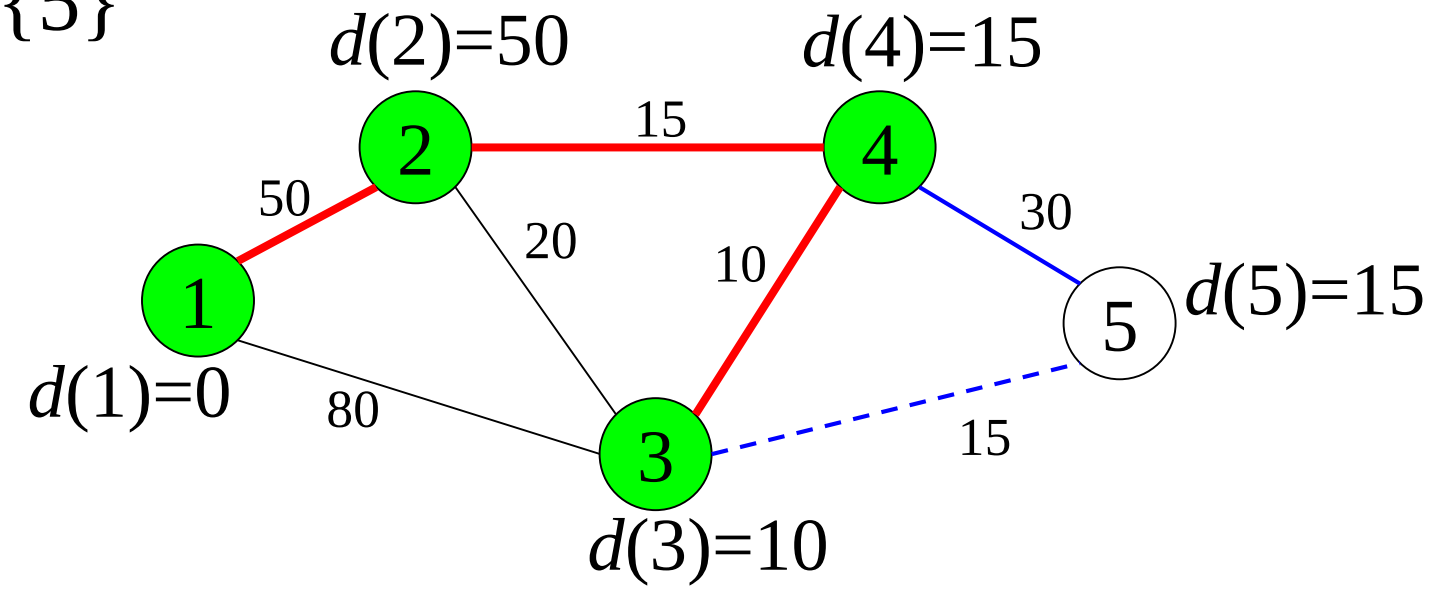
$Q = \{3,4,5\}$


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

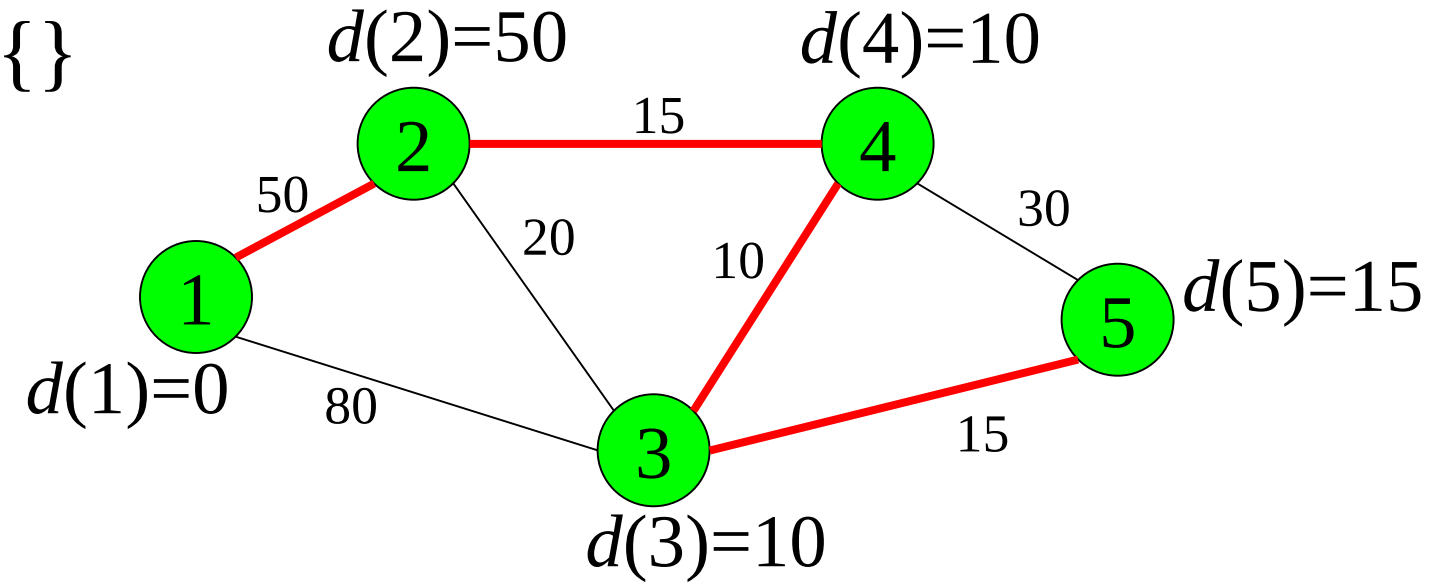
$Q = \{3,5\}$


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

$Q = \{5\}$


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

$Q = \{\}$


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

- Whileループの各繰り返しの直前では
  - $A = \{(v, \pi[v] \mid v \in V - \{r\} - Q\}$
  - 最小全域木の中にすでに置かれた点は $V - Q$ の点
  - $v \in Q$ に対し，$\pi[v] \neq \mathrm{NIL}$ ならば，
    $d[v] < \infty$
    $d[v]$ は v と最小全域木にすでに置かれた点を結ぶ軽い辺 $(v, \pi[v])$ の長さ
- 注：集合 A は明示的には持たない
- アルゴリズムでは $d[u]$ が最小である $u \in Q$ を Q から削除する $\Rightarrow$ $(u, \pi[u])$ を A に加える

データ構造の詳細

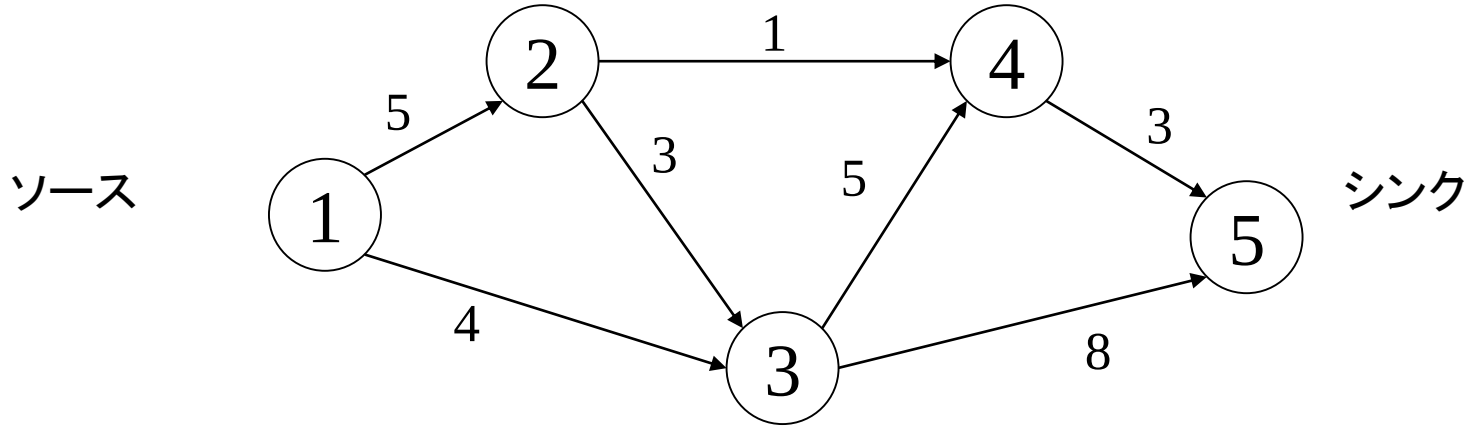
- グラフは隣接リストで表現
- Q はヒープで表現．キーは $d[\cdot]$
- $d[v] := w(u,v)$ を行うとき，ヒープも更新する
  - v をヒープからいったん削除し，$d[v]$ を更新してから挿入し直す
  - 削除を $O(\log n)$ 時間で行うために，v がヒープのどこに格納されているかをサイズ n の配列で管理
- if $v \in Q$ の判定は，上記のヒープの拡張を用いれば定数時間で可能

プリム法の計算量

- extract_min: $O(n)$ 回
- 各点の隣接点を列挙する処理は，各点について1回のみ $\Rightarrow$ 処理回数は $2m$ 以下
- d の更新 (heapの挿入削除): $O(m)$ 回
- 総計算量: $O(n \log n + m \log n) = O(m \log n)$
  - クラスカル法と同じ
- d の更新を挿入と削除ではなく，フィボナッチヒープのdecrease_keyで行うと，総計算量は $O(m + n \log n)$

最大流問題とフロー増加法

- 各枝に容量が与えられたネットワーク $G = (V,E)$ において，ソース $s \in V$ からシンク $t \in V$ に向かってできるだけ多くのものを送ることを考える
- 枝 (i,j) 上の流れの大きさを x_{ij}，枝 (i,j) の容量，つまり流れ x_{ij} の上限値を u_{ij}，ソースからシンクへの総流量を f とすると，問題は次のような線形計画問題として定式化できる



目的関数: $f \rightarrow$ 最大

制約条件:

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

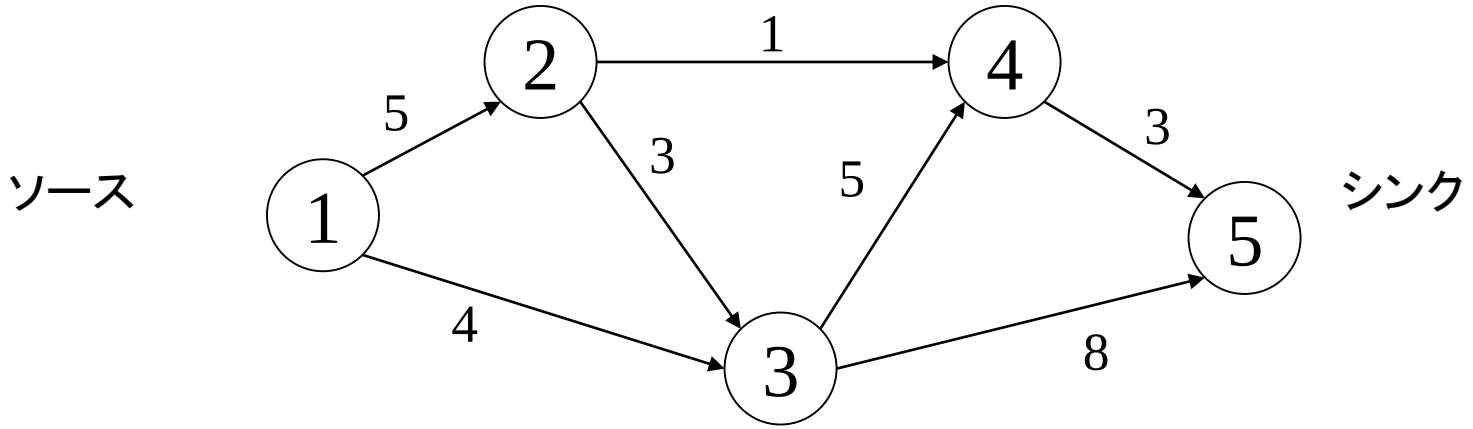
$$\sum_{\{j|(i,j)\in E\}} x_{ij} - \sum_{\{j|(j,i)\in E\}} x_{ji} = 0 \qquad (i \in V - \{s,t\})$$

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

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

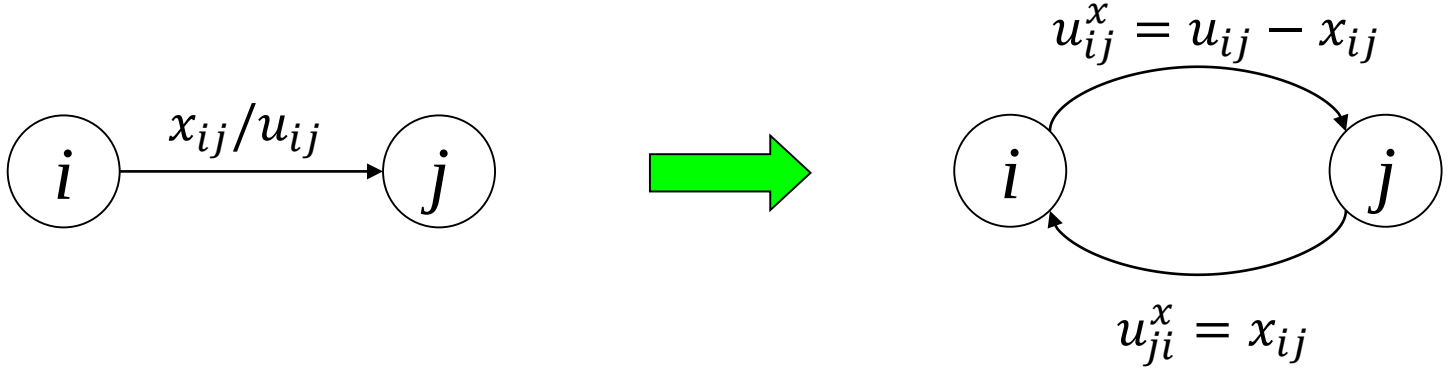
- 流れ保存則
  - 制約条件1: s からの正味の流出量が f
  - 制約条件3: t への正味の流入量が f
  - 制約条件2: s と t 以外の節点では流入量 = 流出量
- 容量制約条件
  - 制約条件4: 各枝の流量が非負かつ枝の容量以下

- 流れ保存則と容量制約条件を満たす $x = (x_{ij})$ をフローと呼びそれに対する f をフローの流量と呼ぶ
- 最大流問題は，流量が最大になるフローを求める問題
- 最大流問題は線形計画問題なので単体法などで解くことができるが，ここではネットワーク問題の特性を利用したアルゴリズムを考える



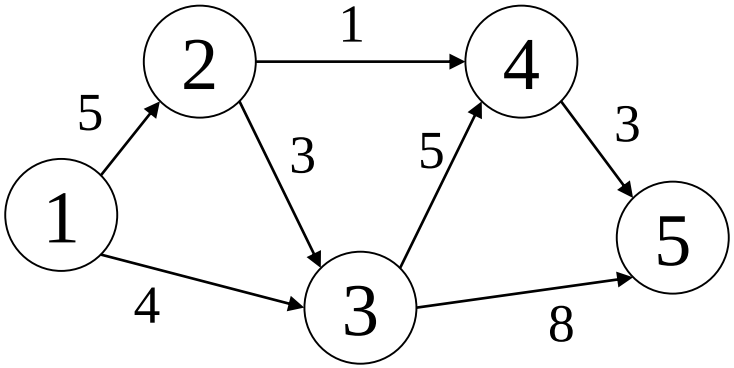
残余容量と残余ネットワーク

- ネットワークにおいて，任意の2節点 $i, j \in V$ の間に枝(i,j)と(j,i)の両方が存在することはないと仮定
- あるフロー $x = (x_{ij})$ が与えられているとする．ネットワーク $G = (V,E)$ の各枝 $(i,j) \in E$ を容量 $u_{ij}^x = u_{ij} - x_{ij}$ を持つ枝 (i,j) と，容量 $u_{ji}^x = x_{ij}$ を持つ枝 (j,i) で置き換える
- $u_{ij}^x = 0$ なら枝 (j,i) のみ，$u_{ji}^x = 0$ なら枝 (i,j) のみ

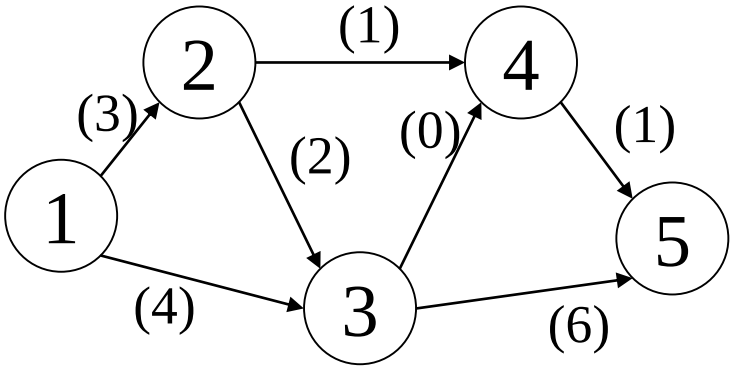


- u_{ij}^x, u_{ji}^x をフロー x に対する枝 (i,j) の残余容量という
- ネットワーク G の各枝 (i,j) を上のように置き換えたネットワークを，フロー x に対する残余ネットワークと言い，$G^x = (V,E^x)$ と書く

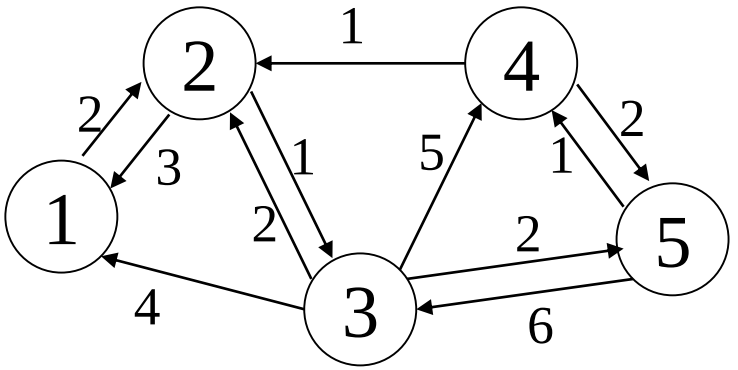
元のネットワーク G



フロー x

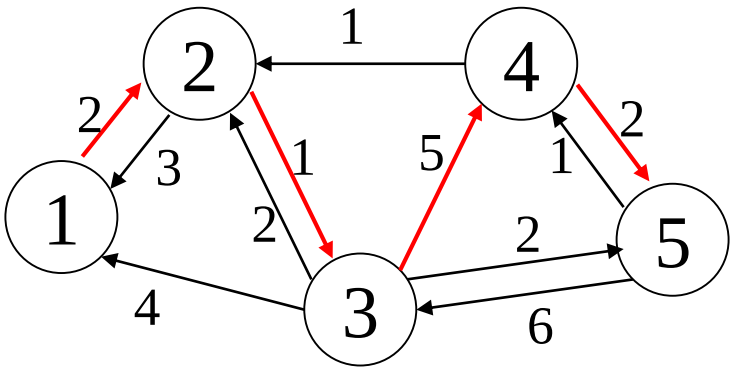


残余ネットワーク G^x

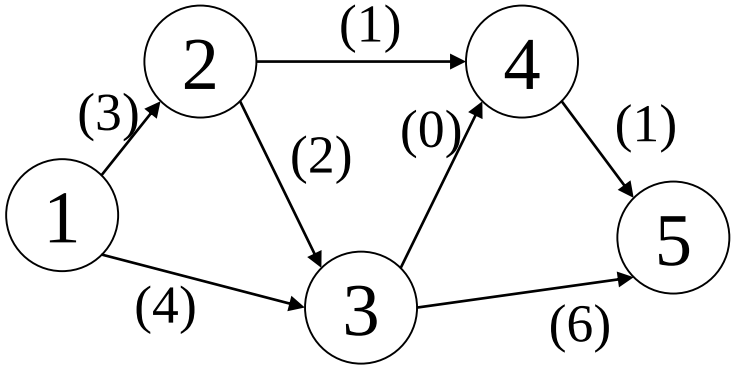


- 残余ネットワークにおいて，ソースからシンクへの路が存在すれば，その路に沿ってフローを追加することによって，元のネットワーク上での流量 f を増やせることがわかる
- そのような路を，(現在のフロー x に対する) フロー増加路と呼ぶ
- フロー増加路に沿って追加できるフローの量は，その路に含まれる枝の残余容量の最小値に等しい

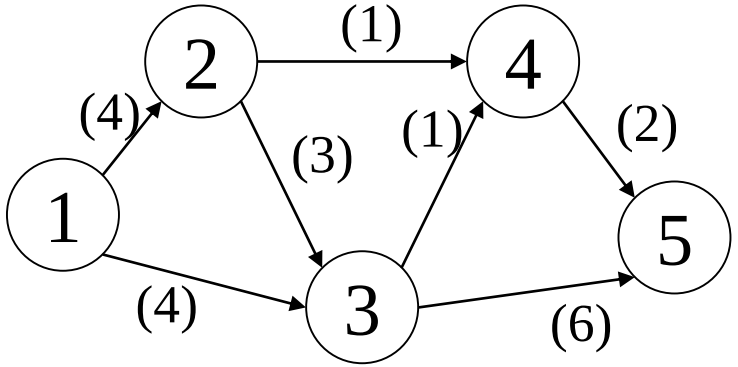
残余ネットワーク G^x



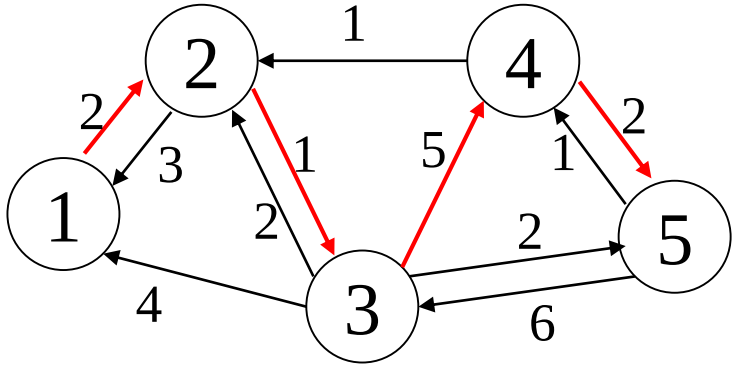
フロー x



新しいフロー x



残余ネットワーク G^x
フロー増加路 (流量1)



フロー増加法 (Ford-Fulkerson法)

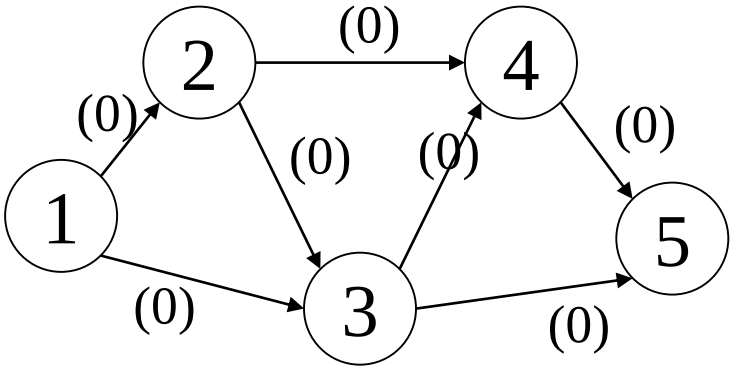
- 最大流を求めるアルゴリズム
  (正当性の証明は後)

(0) 適当な初期フロー x を求める
  例えば，全ての枝 (i,j) に対し $x_{ij} = 0$ とする

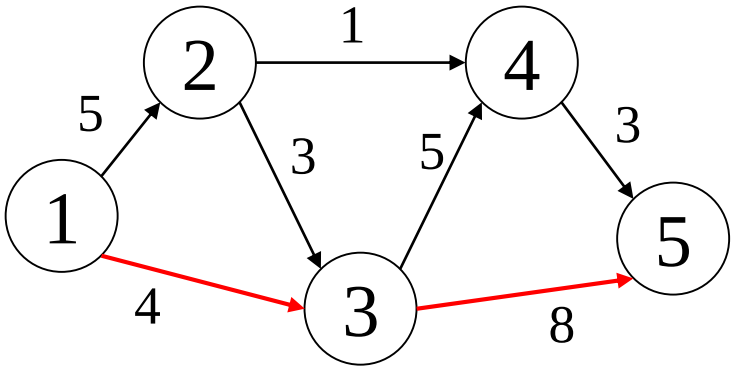
(1) 残余ネットワーク G^x においてソースからシンクへの路 (フロー増加路) を見つける
  存在しなければ計算終了

(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 x を得る．ステップ(1)に戻る

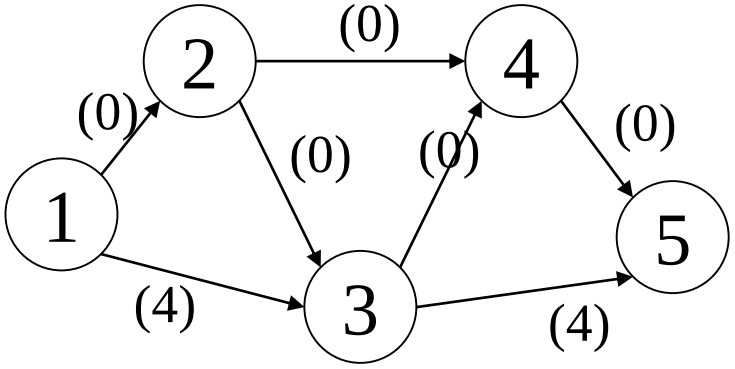
初期フロー



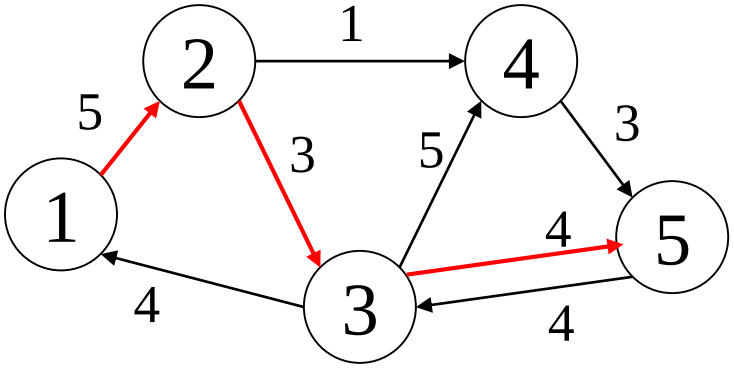
残余ネットワーク



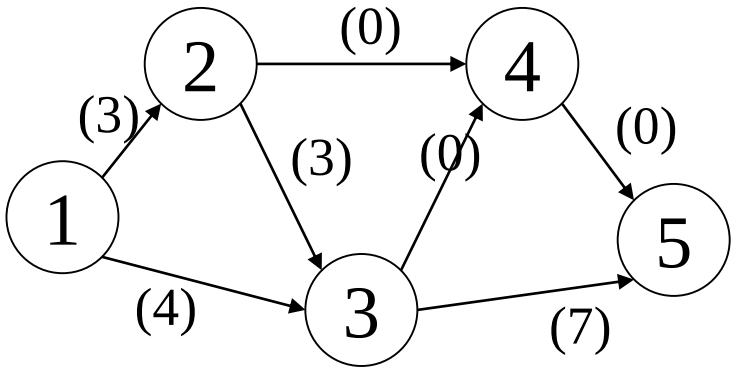
フロー



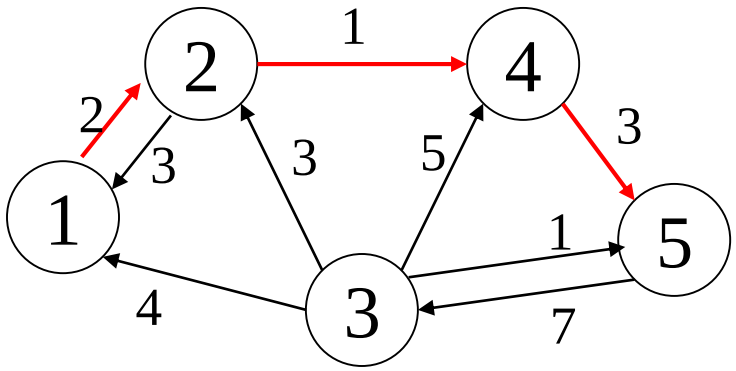
残余ネットワーク



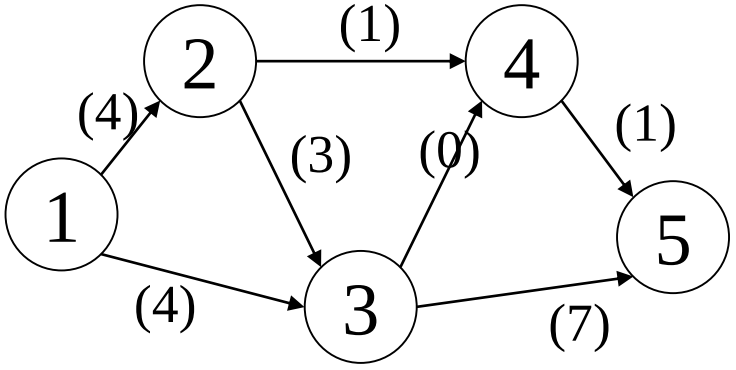
フロー



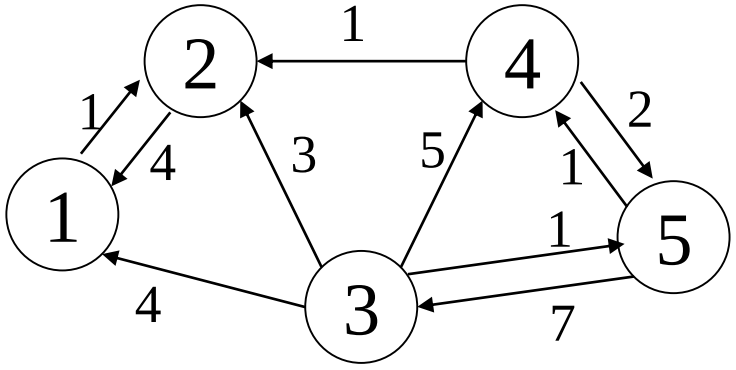
残余ネットワーク



フロー



残余ネットワーク



フロー増加路が存在しないので終了
フローの流量は 8

フロー増加法の正当性と最大流最小カット定理

- 枝の容量が全て整数なら，初期フローを整数値としてフロー増加法を用いると，ソースからシンクへの流量は1回の反復で少なくとも1は増える
- よって有限回の反復の後にアルゴリズムは終了
- 終了した時点でのフローが最大流になっていることを示す

- 頂点集合Vをソースsを含む集合Sとシンクtを含む集合Tに分割したものをs-tカットと言い，(S,T)と表す
- 任意のカット(S,T)に対して，Sの節点を始点とし，Tの節点を終点とする枝を$(i,j) \in (S,T)$，逆に，Tの節点を始点とし，Sの節点を終点とする枝を$(j,i) \in (T,S)$と書く
- 全ての枝$(i,j) \in (S,T)$の容量u_{ij}の合計をカット(S,T)の容量と呼び，$C(S,T)$で表す．つまり

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

- x を任意のフロー，f をその流量，(S,T) を任意の s-t カットとする
- $f = \sum_{(i,j)\in(S,T)} x_{ij} - \sum_{(j,i)\in(T,S)} x_{ji}$ が成立する

- 全ての枝 $(i,j) \in E$ に対して $0 \le x_{ij} \le u_{ij}$ であるから，カット容量の定義より，$f \le C(S,T)$ を得る

$$
\begin{aligned}
f &= \sum_{(i,j)\in(S,T)} x_{ij} - \sum_{(j,i)\in(T,S)} x_{ji} \\
&\le \sum_{(i,j)\in(S,T)} x_{ij} \le \sum_{(i,j)\in(S,T)} u_{ij} \\
&= C(S,T)
\end{aligned}
$$

- フロー増加法の計算が終了した時点で得られているフローを x^*, その流量を f^* とする
- 残余ネットワークで s から到達できる点の集合を S^*, その補集合を T^* とする
- $s \in S^*$ かつ $t \in T^*$ より, (S^*, T^*) はカットとなる
- 残余ネットワークの定義より

$$(i,j) \in (S^*, T^*) \Rightarrow x_{ij}^* = u_{ij}$$
$$(j,i) \in (T^*, S^*) \Rightarrow x_{ji}^* = 0$$

- 従って，

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

- 任意のフローに対して $f \leq C(S,T)$が成り立つので，上の式は x^* が実行可能な全てのフローの中で最大流量を与えるものであり，同時に，(S^*,T^*) が全てのカットの中で最小の容量をもつものであることを示している (最大流最小カットの定理)
- 以上より，フロー増加法が終了したときのフローは最大流になっている

フロー増加法の計算量とその改良

- 容量が整数値のとき，フロー増加法では1回の反復で少なくとも1単位の流量が追加される

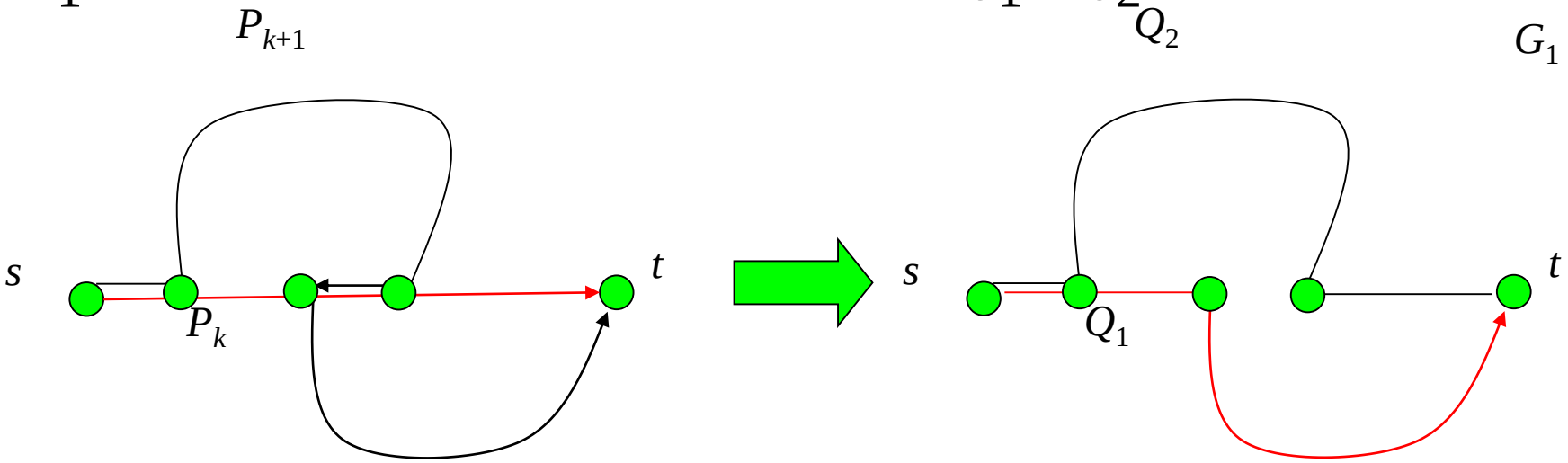
  $\Rightarrow$ 全体の反復回数は最大流量の値を越えない
- 枝の容量の最大値を $U = \max\{u_{ij} | (i,j) \in E\}$ とすると，最大流量の上限は mU
- 1つのフロー増加路を見つける計算量は $O(m)$
- 全体の計算量は $O(m^2U)$
- 計算量に U が現れるので，理論的には多項式時間アルゴリズムとは言えない

Edmonds-Karpアルゴリズム

- フロー増加法において，フロー増加路を求める際に辺数最小のものを求める
  - 幅優先探索で路を求めればいい
- 反復回数が $mn/2$ 以下になる (後述)
- 全体の時間計算量は $O(m^2n)$

- i 回目の反復後のフローを f_i とする
- i+1 回目の反復では，残余ネットワーク G^{f_i} で辺数最小のフロー増加路 P_i を求め，フロー f_{i+1}を流す
- 補題：フローの系列 $f_1, f_2,...$ に対し
  (a) 全ての k で $|E(P_k)| \leq |E(P_{k+1})|$
  (b) $P_k \cup P_l$ が逆辺対を持つような全ての $k < l$ に対して，$|E(P_k)| + 2 \leq |E(P_l)|$

- 証明:(a) $P_k \cup P_{k+1}$ から逆辺対を全て除去したグラフを G_1 とする
- P_k と P_{k+1} の両方に現れる辺は多重辺とする
- G_1 には2つの辺素な s-t パス Q_1, Q_2 が存在する



- $E(G^{f_{k+1}}) \setminus E(G^{f_k})$ の任意の辺は P_k の辺の逆辺となる

$E(G^{f_{k+1}}) \setminus E(G^{f_k})$ の辺は P_k にそってフローを流したときにできたので，それは流量が 0 のところにフロー P_k を流したあとの残余ネットワークの辺であるから

- G_1 の枝は $P_k \cup P_{k+1}$ に含まれるが，後者の枝の中で $E(G^{f_{k+1}}) \setminus E(G^{f_k})$の枝は P_k の逆辺なので削除されている．よって $E(G_1) \subseteq E(G^{f_k})$
- Q_1, Q_2 の枝は G_1 に含まれるので，Q_1, Q_2 は G^{f_k}での増加路である
- P_k は辺数最小の増加路なので，$|E(P_k)| \leq |E(Q_1)|$ かつ $|E(P_k)| \leq |E(Q_2)|$

$$
\begin{aligned}
2|E(P_k)| &\leq |E(Q_1)| + |E(Q_2)| \\
&\leq |E(G_1)| && (Q_1 \text{と} Q_2 \text{は} G_1 \text{の辺素パス}) \\
&\leq |E(P_k)| + |E(P_{k+1})| && (G_1 \text{は逆辺を削除})
\end{aligned}
$$

以上より $|E(P_k)| \leq |E(P_{k+1})|$

- (b) の証明: $k < i < l$ である全ての i に対し$P_i \cup P_l$ が逆辺を持たないようなk, l に対して証明できれば, (a) より全ての $k < l$ に対しても成り立つ
- $P_k \cup P_l$ から逆辺対を全て除去したグラフを G_1 とする
- $E(P_k) \subseteq E(G^{f_k}), E(P_l) \subseteq E(G^{f_l})$ であり, $E(G^{f_l}) \setminus E(G^{f_k})$ の任意の辺は P_k, P_{k+1},…,P_{l-1} のいずれかに含まれる辺の逆辺である
- よって, P_l の辺で P_k の逆辺でないものは, $E(G^{f_k})$ の辺か P_{k+1},…,P_{l-1} の辺の逆辺である
- k と l の定め方より, P_l は P_{k+1},…,P_{l-1} の辺の逆辺を持たない.

- よって，P_l の辺で P_k の逆辺でないものは $E(G^{f_k})$ の辺となり，$E(G_1) \subseteq E(G^{f_k})$ が得られる
- G_1 には2つの辺素な s-t パス Q_1, Q_2 が存在し，それらは G^{f_k} での増加路である
- P_k は辺数最小の増加路なので，$|E(P_k)| \leq |E(Q_1)|$ かつ $|E(P_k)| \leq |E(Q_2)|$
- $2|E(P_k)| \leq |E(Q_1)| + |E(Q_2)| \leq |E(P_k)| + |E(P_l)| - 2$

  (少なくとも2辺を除去しているので)

  より，命題が成り立つ

- 定理: 辺数 m, 点数 n のネットワークに対して, Edmonds-Karpアルゴリズムは辺の容量に関わらず, 高々 $mn/2$ 回の反復で終了する
- 証明:アルゴリズムの実行中に選ばれた増加路を $P_1, P_2,...$ とする
- 増加路で可能な限りフローを追加するので, 各増加路には残余ネットワークでの容量いっぱいまで流す枝が存在する. それをボトルネック辺と呼ぶ
- 任意の辺 e に対して, e をボトルネック辺にもつ増加パスを $P_{i_1}, P_{i_2},\ldots$とする
- P_{i_j} と $P_{i_{j+1}}$の間に, e の逆辺を含む増加路 P_k が存在する $(i_j < k < i_{j+1})$

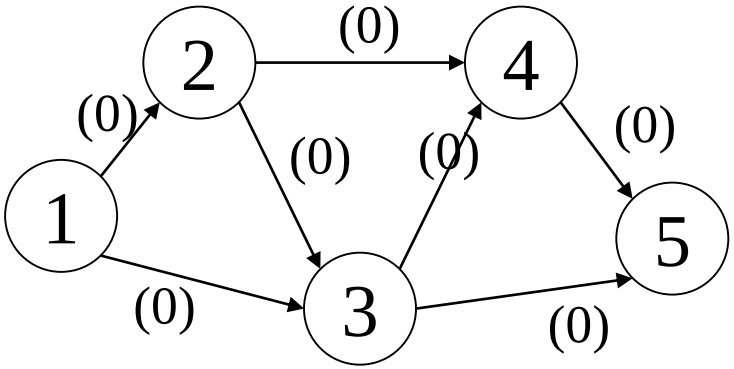
- 補題 (b) より，全ての j に対し
$$\left|E(P_{i_j})\right| + 4 \leq |E(P_k)| + 2 \leq \left|E(P_{i_{j+1}})\right|$$
- e が端点として s と t を含まなければ，全ての j で $3 \leq \left|E(P_{i_j})\right| \leq n-1$ となり，e をボトルネック辺としてもつ増加路は高々 $n/4$ 個
- e が端点として s と t のいずれかを含むときは，e またはその逆辺をボトルネック辺として持つ増加路は高々1つ
- 任意の増加路は G の辺かその逆辺を少なくとも1本含むので，高々 $2m \cdot n/4 = mn/2$ 個の増加路しか選ばれない

Dinicアルゴリズム

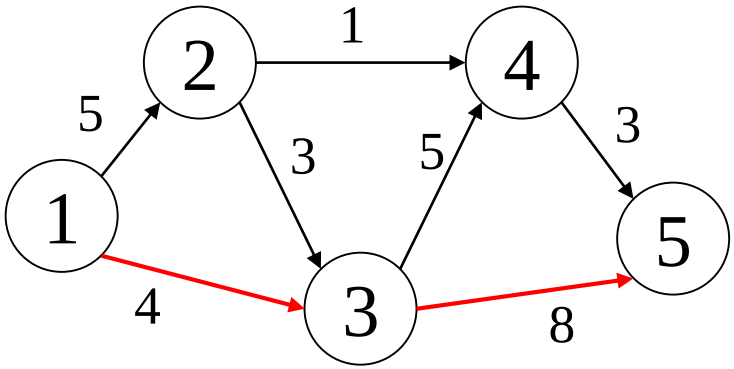
- フロー増加法の各反復において，1本のフロー増加路だけを用いてフローを追加するのではなく，複数のフロー増加路を求めてそれらに同時にフローを追加するアルゴリズム
- Edmonds-Karpアルゴリズムでは各反復で最短の増加路を求めているが，補題 (a) より路の長さは単調増加
- 路の長さが等しい増加路の系列をアルゴリズムのフェイズと呼ぶ
- 1つのフェイズの全ての増加路を同時に求めて加える

- 命題：あるフェイズの始まりのフローを f とする．そのフェイズの増加路は全て G^f の増加路になる．
- 証明：補題 (b) より，ある増加路 P_k と P_l に対して，$P_k \cup P_l$ が逆辺対を持つならば $|E(P_k)| + 2 \leq |E(P_l)|$ なので，同じフェイズに属する増加路の共通部分は逆辺対を持たない
- $E(G^{f_l}) \setminus E(G^{f_k})$ の任意の辺は $P_k, P_{k+1},\ldots,P_{l-1}$ のいずれかに含まれる辺の逆辺であるが，今は逆辺は存在しないので，$E(G^{f_l}) \subseteq E(G^{f_k})$ となる
- つまり，あるフェイズの増加路は全て G^f の増加路

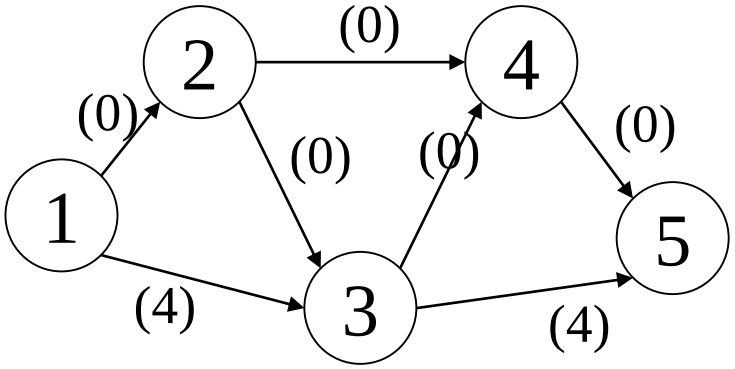
初期フロー



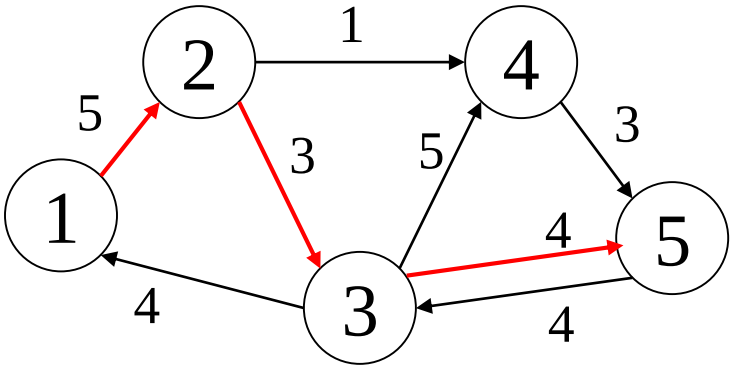
残余ネットワーク



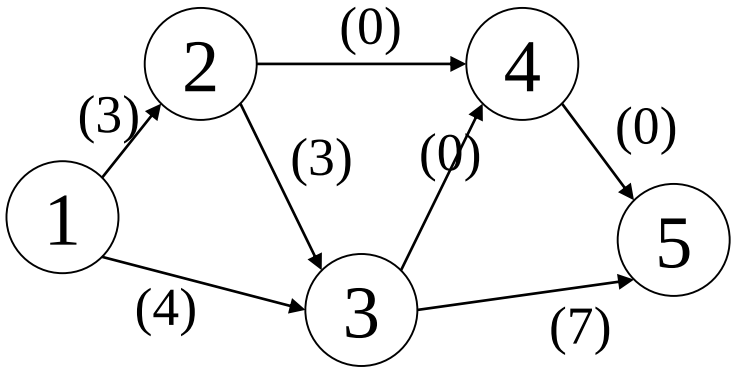
フロー



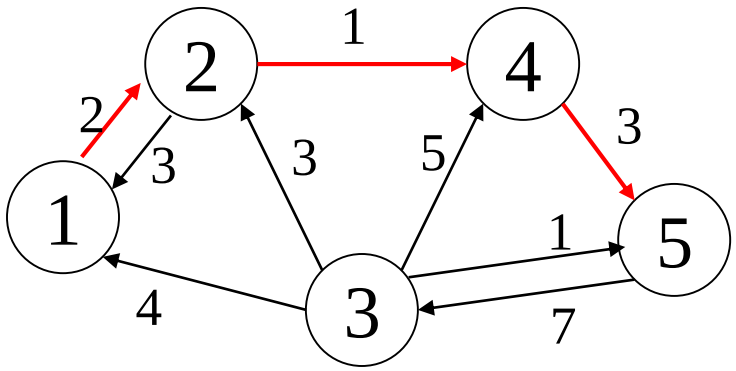
残余ネットワーク



フロー



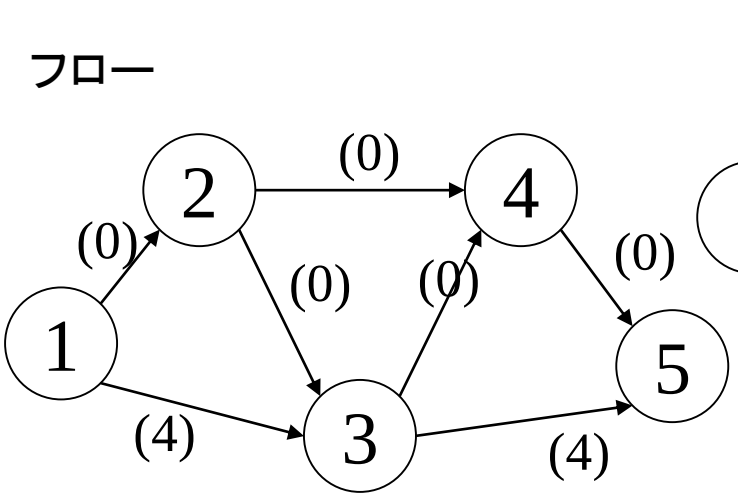
残余ネットワーク

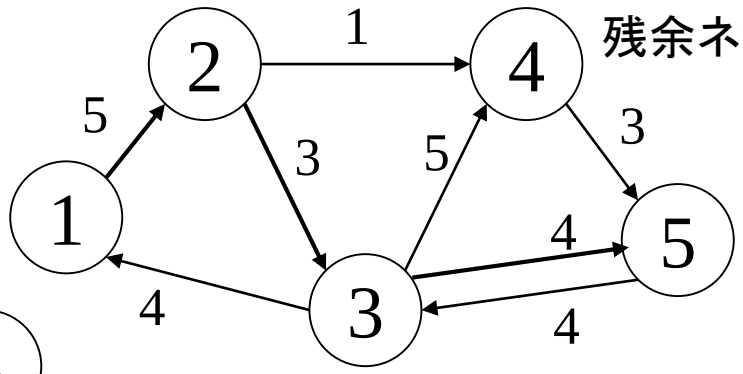


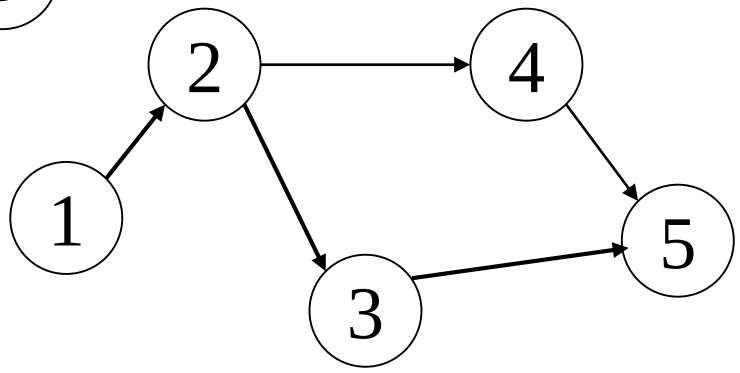
この増加路は1つ前の
残余ネットワークにも存在
(長さ3の増加路全てで成立)

- 残余ネットワークから，路長最小の路の集合を求める必要がある．
- 定義：ネットワークの s-t フロー f に対して，G^f のレベルグラフ G_L^f は

$$\left(V(G), \left\{e = (x, y) \in E(G^f) | \mathrm{dist}_{G_f}(s, x) + 1 = \mathrm{dist}_{G_f}(s, y)\right\}\right)$$

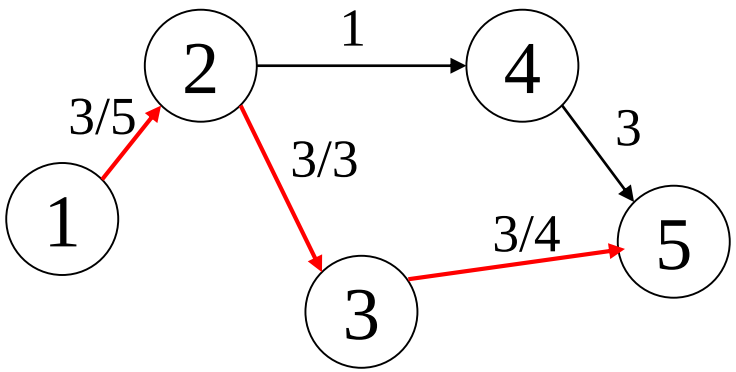




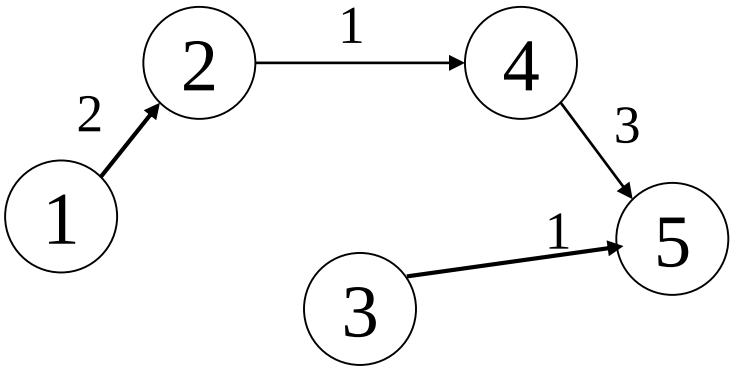


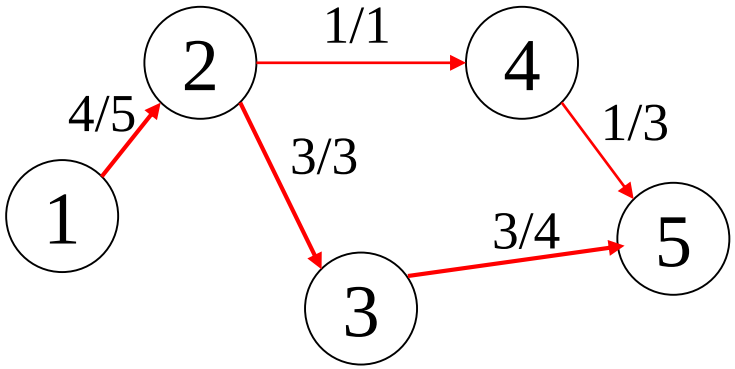
- 定義：ネットワークの s-t フロー f は，$\left(V(G), \{e \in E(G_f) | f(e) < u(e)\}\right)$ が s-t パスを含まないとき，ブロックフロー (blocking flow) と呼ぶ

残余ネットワーク
のレベルグラフ

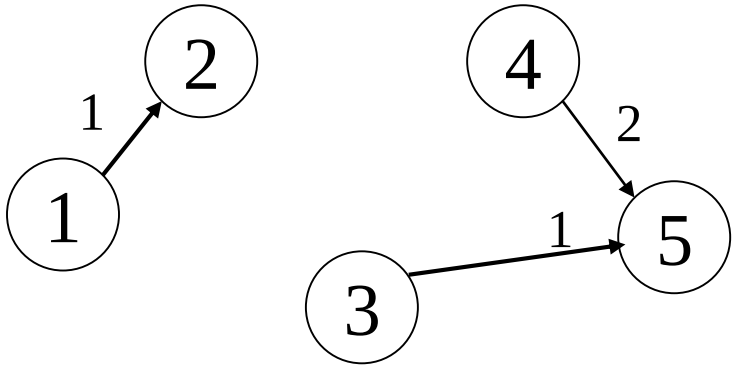


s-t フローが存在 ⇒
左はブロックフローではない





s-t フローが存在しない ⇒
左はブロックフロー

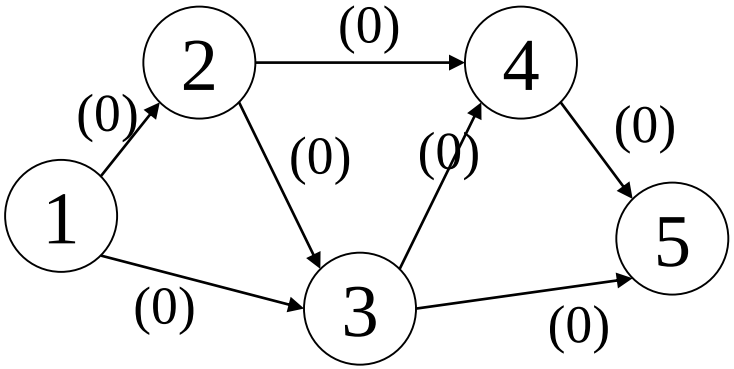


Dinicのアルゴリズム

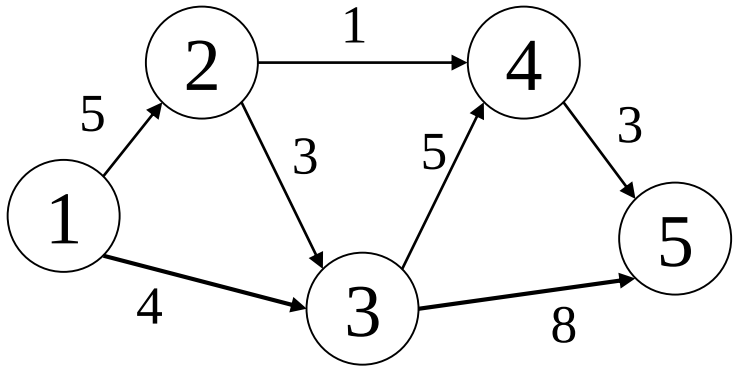
1. 全ての $e \in E(G)$ で $f(e) = 0$ とする
2. G^f のレベルグラフを作る
3. レベルグラフでのブロックフロー f' を求める.
   $f' = 0$ ならば終了
4. f を f' だけ増加させる. 2. へ行く

注: f を増加させるときは, 逆辺に対しては
$f(e) := f(e) - f'(e)$ とする

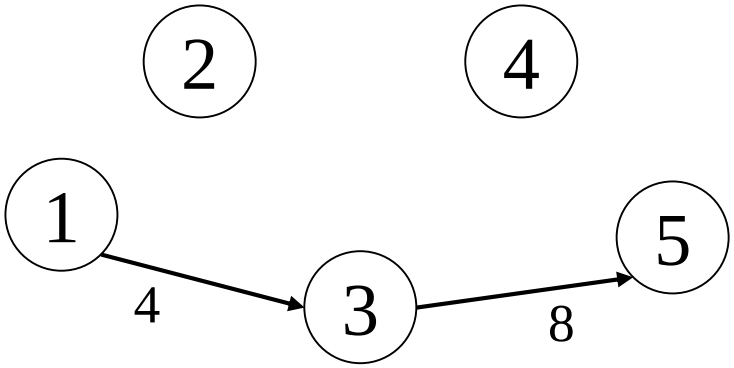
初期フロー



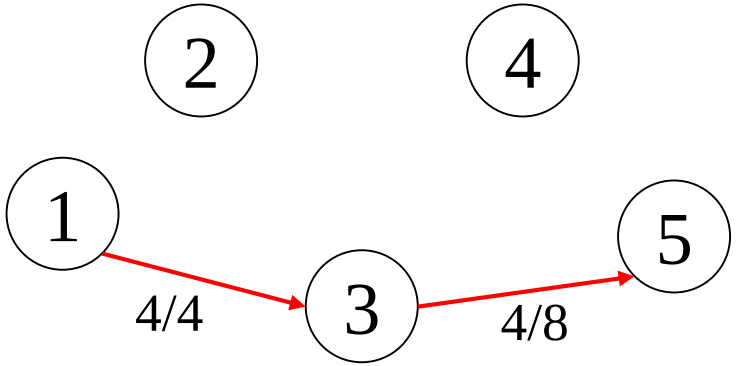
残余ネットワーク



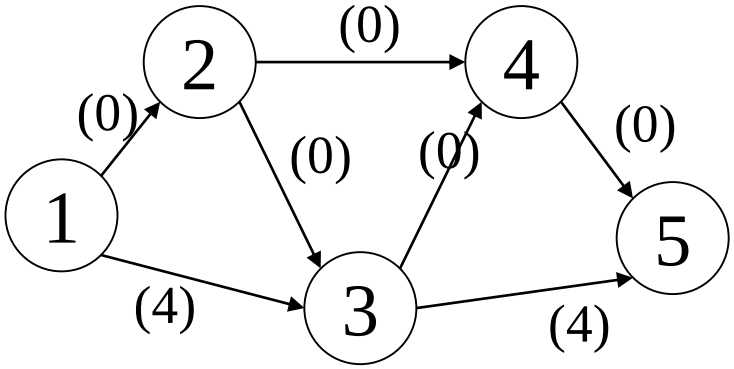
レベルグラフ



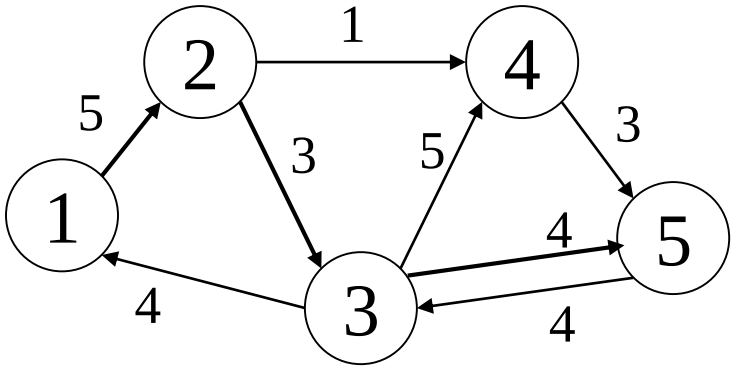
ブロックフロー



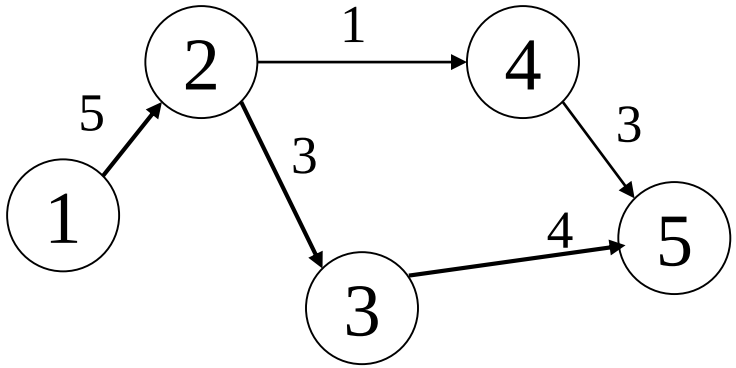
フロー



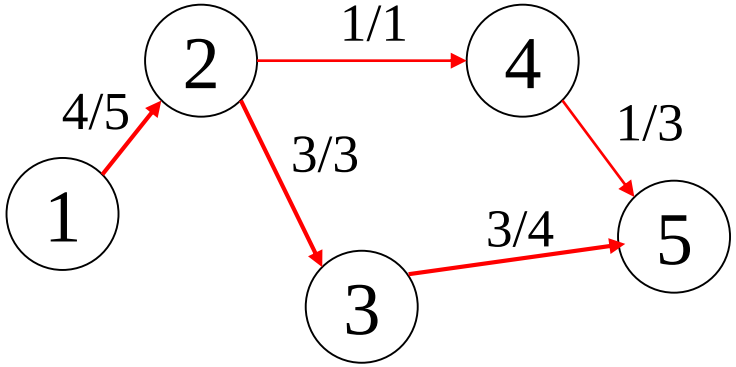
残余ネットワーク



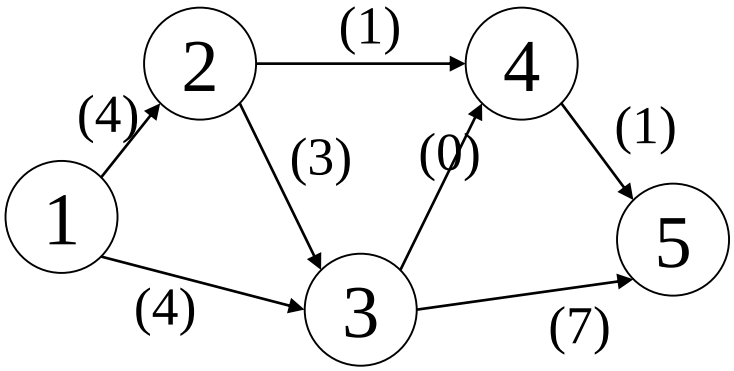
レベルグラフ



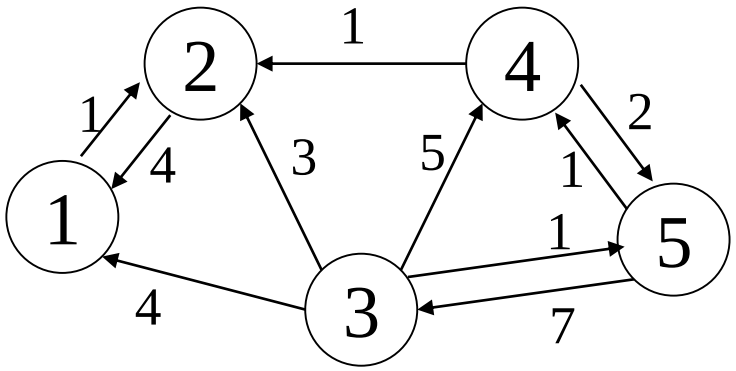
ブロックフロー



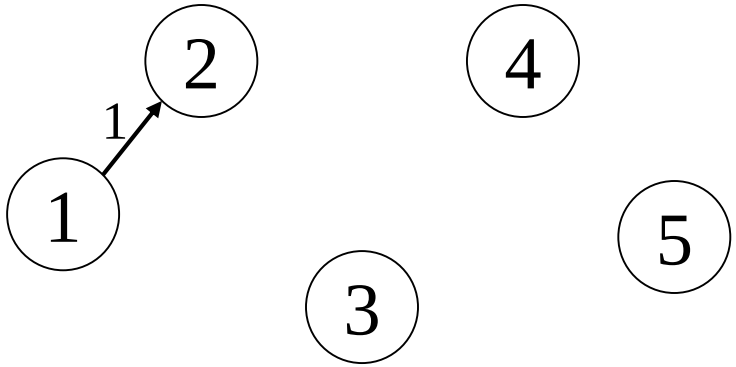
フロー



残余ネットワーク



レベルグラフ



レベルグラフに *s*-*t* フローが存在しないので終了
フローの流量は 8

Dinicアルゴリズムの計算量

- 辺数最小の増加路の辺数はフェイズごとに厳密に増加する. よってDinicアルゴリズムは高々 $n-1$ 回のフェイズ後終了する.
- 各フェイズではブロックフローは $O(mn)$ 時間で求まる
  - レベルグラフは幅優先探索で $O(m)$ 時間
  - レベルグラフ上で1つのs-tパスは $O(n)$ 時間
  - 1つのs-tパス上には少なくとも1つの飽和辺が存在するそれを削除してさらにs-tパスを探す. 高々 m 回で終了
- 全体の計算量は $O(mn^2)$
- 動的木というデータ構造を用いるとブロックフローは $O(m \log n)$ 時間. 全体で $O(mn \log n)$ 時間.