経営・経済のための最適化理論
第8回
最大流問題

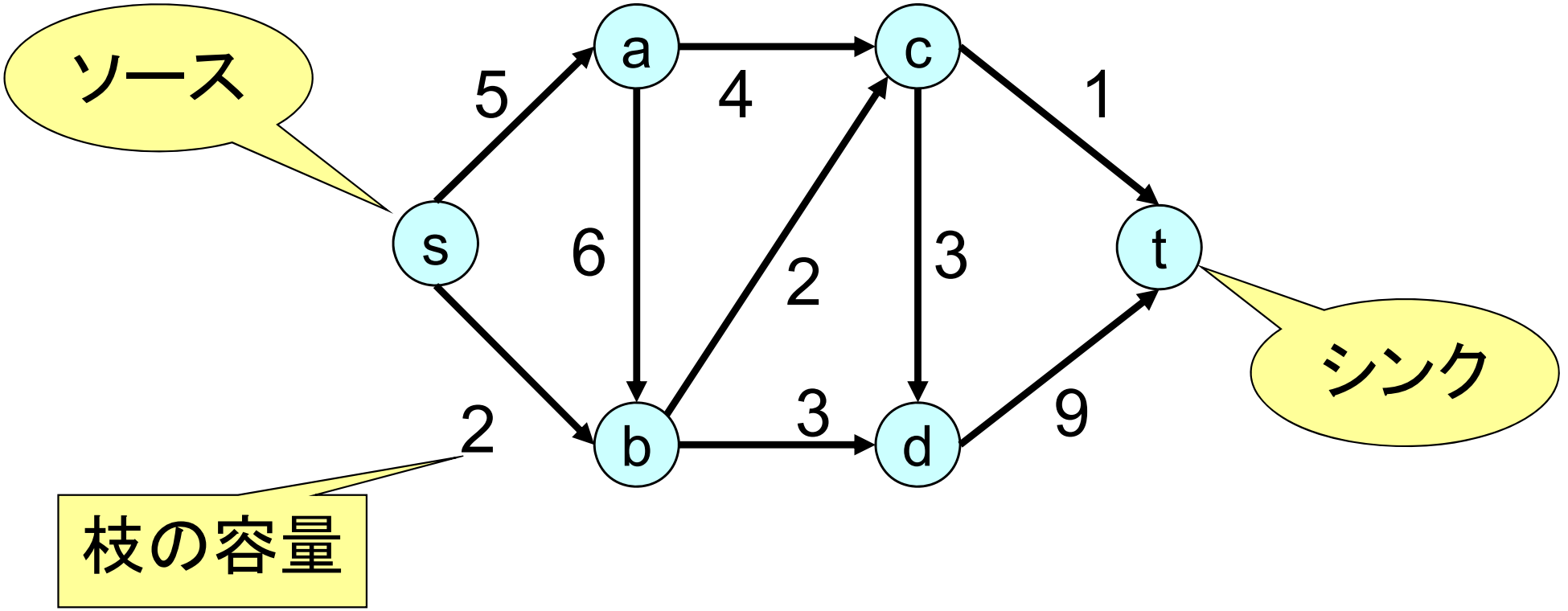
塩浦昭義
東京工業大学 経営工学系
shioura.a.aa@m.titech.ac.jp

最大流問題の定義(その1)

入力: 有向グラフ $G = (V, E)$

ソース(供給点) $s \in V$, シンク(需要点) $t \in V$

各枝 $(i, j) \in V$ の容量 $u_{ij} \geqq 0$



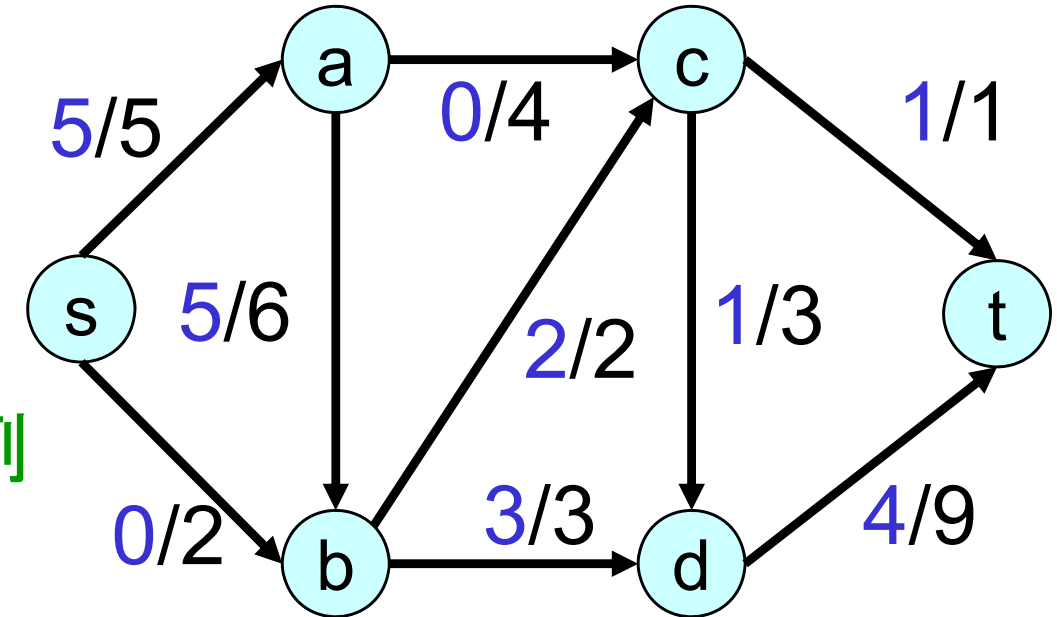
最大流問題の定義(その2)

目的:ソースからシンクに向けて，枝と頂点を経由して
「もの」を出来るだけたくさん流す

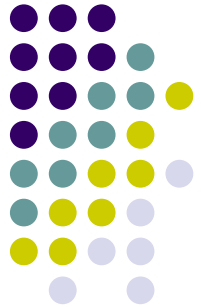
条件1(容量条件):
0 ≦ 各枝を流れる「もの」の量 ≦ 枝の容量

条件2(流量保存条件):
頂点から流れ出す「もの」の量＝　流れ込む「もの」の量

実行可能解の例

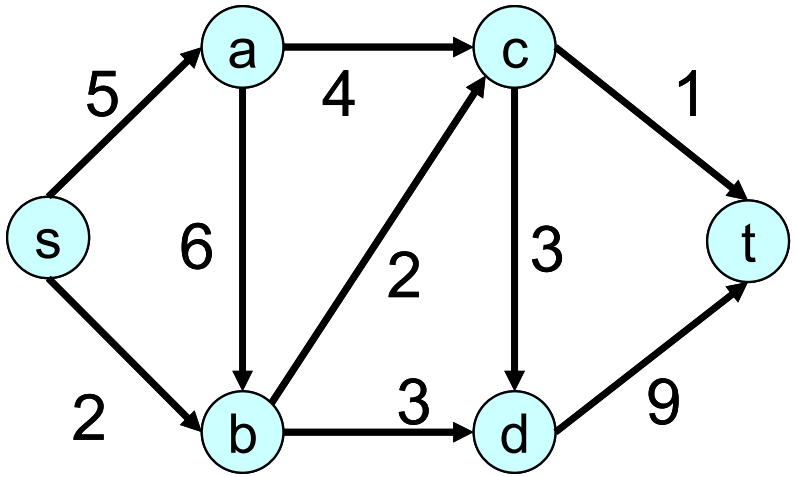


最大流問題の定式化：変数，目的関数と容量条件



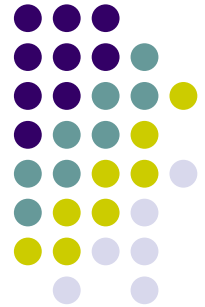
変数 x_{ij}：フロー＝枝 (i, j) を流れる「もの」の量
変数 f：総流量　＝ シンクに流れ込む「もの」の総量
　　＝ ソースから流れ出す「もの」の総量

目的：ソースからシンクに「もの」を出来るだけ多く流したい
　　⇒ 最大化　f
容量条件：0 ≦ 各枝を流れる「もの」の量 ≦ 枝の容量
　　⇒　$0 \leqq x_{ij} \leqq u_{ij}$　　((i,j) ∈ E)



最大化　f
容量条件：
$0 \leqq x_{sa} \leqq 5,\ 0 \leqq x_{sb} \leqq 2,\ 0 \leqq x_{ab} \leqq 6,$
$0 \leqq x_{ac} \leqq 4,\ 0 \leqq x_{bc} \leqq 2,$
…

最大流問題の定式化：流量保存条件



流量保存条件：

(頂点から流れ出す「もの」の量) − (流れ込む「もの」の量) ＝ 0

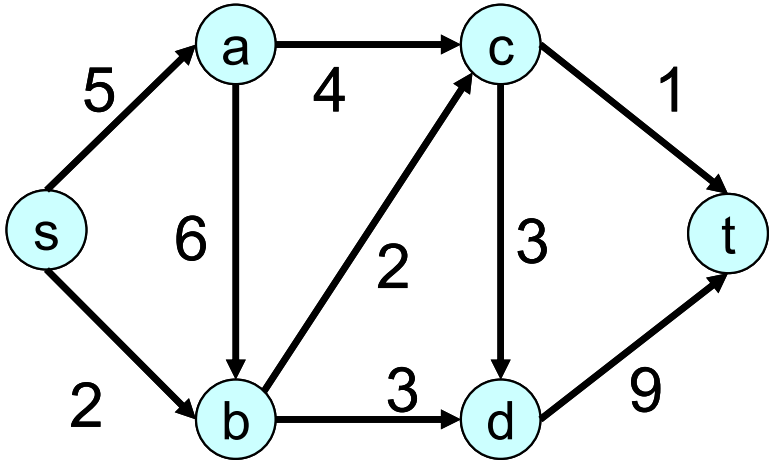
⇒ $\sum\{x_{kj} \mid$ 枝 (k,j) は 頂点 k から出る$\}$

$\quad - \sum\{x_{ik} \mid$ 枝 (i,k) は 頂点 k に入る$\} \;=\; 0 \quad (k \in V - \{s, t\})$

ソースとシンクに関する条件：

$\sum\{x_{sj} \mid (s,j)$ は s から出る$\} - \sum\{x_{is} \mid (i,s)$ は s に入る$\} = f$

$\sum\{x_{tj} \mid (t,j)$ は t から出る$\} \; - \; \sum\{x_{it} \mid (i,t)$ は t に入る$\} = - f$



流量保存条件の例：

$x_{ac} + x_{ab} - x_{sa} = 0$

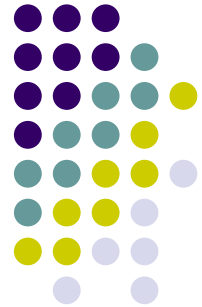
$x_{bc} + x_{bd} - x_{ab} - x_{sb} = 0$

$x_{ct} + x_{cd} - x_{ac} - x_{cb} = 0$

$x_{dt} - x_{cd} - x_{bd} = 0$

$x_{sa} + x_{sb} = f, \qquad - x_{ct} - x_{dt} = - f$

最大流問題の定式化：まとめ



最大化　f

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

$\sum\{x_{kj} \mid (k,j)$ は k から出る$\} - \sum\{x_{ik} \mid (i,k)$ は k に入る$\} = 0$

(k：s, t 以外の頂点)

$\sum\{x_{sj} \mid (s,j)$ は s から出る$\} - \sum\{x_{is} \mid (i,s)$ は s に入る$\} = f$

$\sum\{x_{tj} \mid (t,j)$ は t から出る$\} - \sum\{x_{it} \mid (i,t)$ は t に入る$\} = -f$

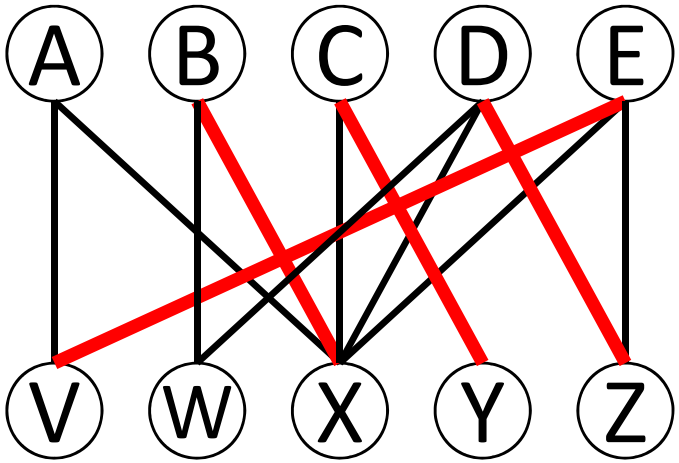
この問題の実行可能解 x_{ij} --- 実行可能フロー
総流量が最大の実行可能フロー --- 最大フロー

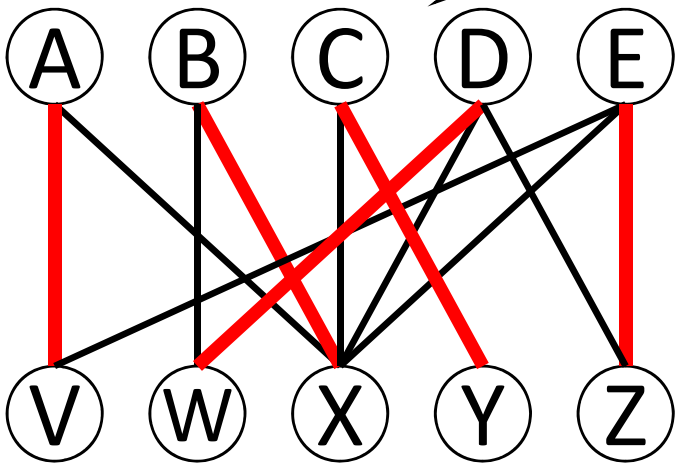
※ フロー x_{ij} の値は実数値でもよい

最大マッチング問題

- できるだけ多くの労働者に仕事を割り当てたい
  - 各労働者には，高々1つの仕事が割り当て可能
  - 各仕事は，高々一人の労働者に割り当て可能

  ➔ 仕事の割当を枝集合で表現可能





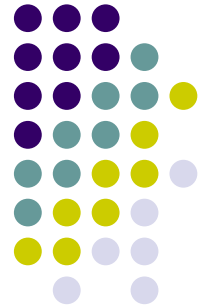
最大
マッチング

- グラフのマッチング：

  グラフの枝集合であって，各頂点に接続する枝の数（次数）が1以下
- 最大（サイズ）マッチング問題：

  与えられたグラフのマッチングの中で，枝数最大のものを求める

二部グラフの最大マッチングとの関係



二部グラフの最大マッチング問題は最大流問題に帰着可能

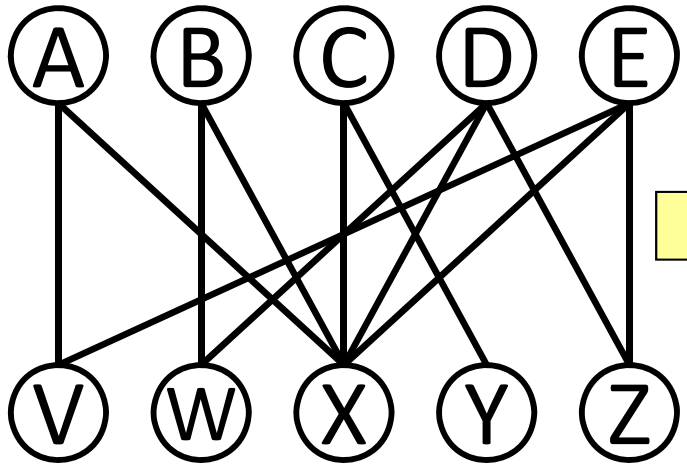
(1) 新しい頂点 s, t をおく

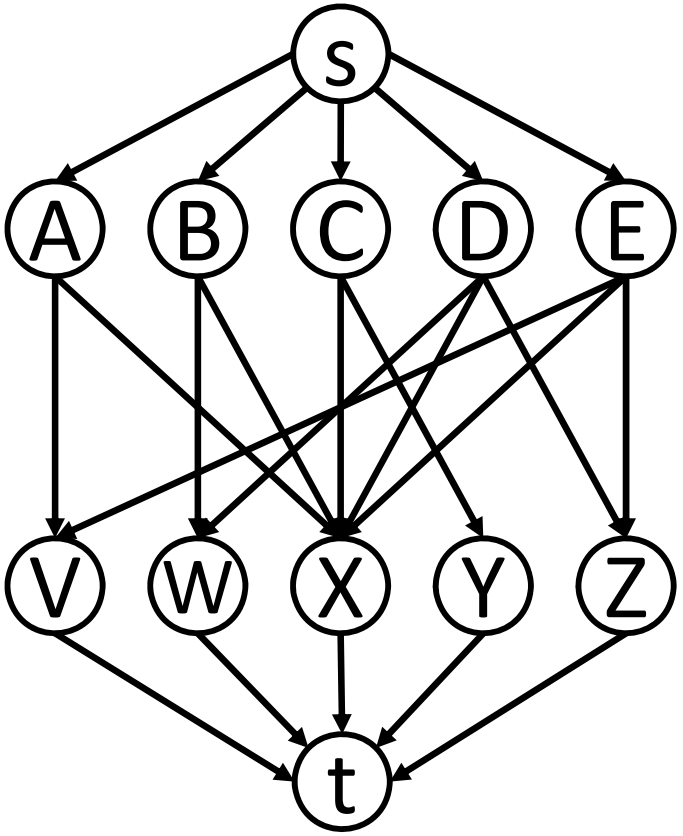
(2) s から上側の各頂点に向かう枝をおく．容量=1

(3) 下側の各頂点から t に向かう枝をおく．容量=1

(4) 元々の枝を上から下に向き付け．容量=+∞

容量=1 としても
よいが，+∞の方が
後々都合が良い



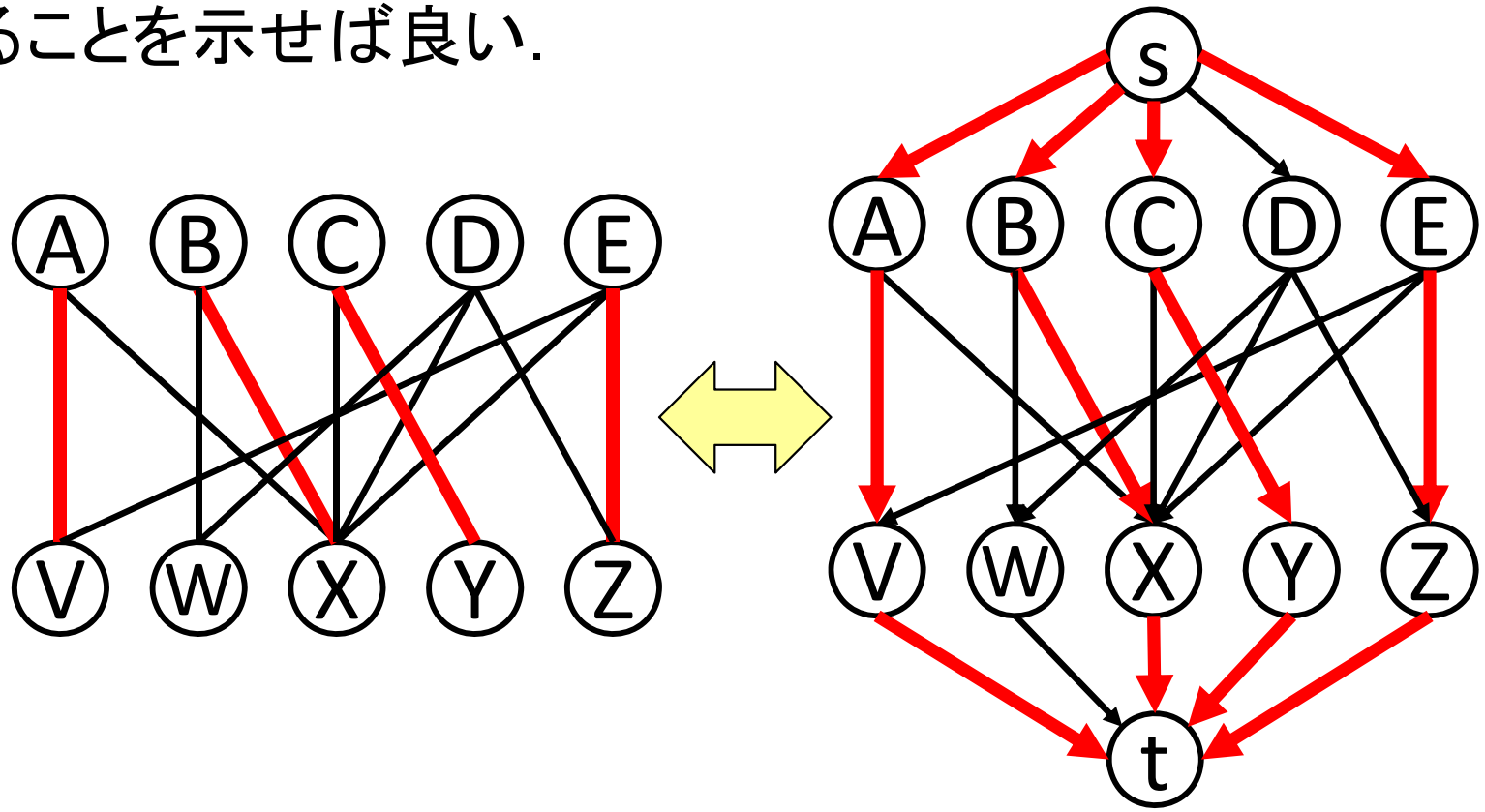


二部グラフの最大マッチングとの関係

二部グラフの最大マッチング問題は最大流問題に帰着可能

命題: この方法で得られる最大流問題の最大フローは
最大マッチングに対応する.

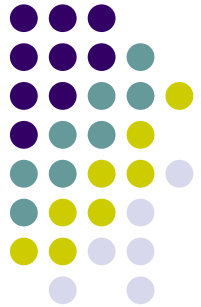
（証明の概略）任意の実行可能フローと二部グラフのマッチング
が一対一対応することを示せば良い.



最大流問題の応用例

- 物流
- シーズン途中でのプロ野球チームの優勝可能性判定
  - 残り試合全勝しても優勝の可能性がないかどうか？
- 画像処理における物体の切り出し
  - 画像内の物体のみ取り出す
- その他多数：後ほど詳しく説明

最大流問題の解法

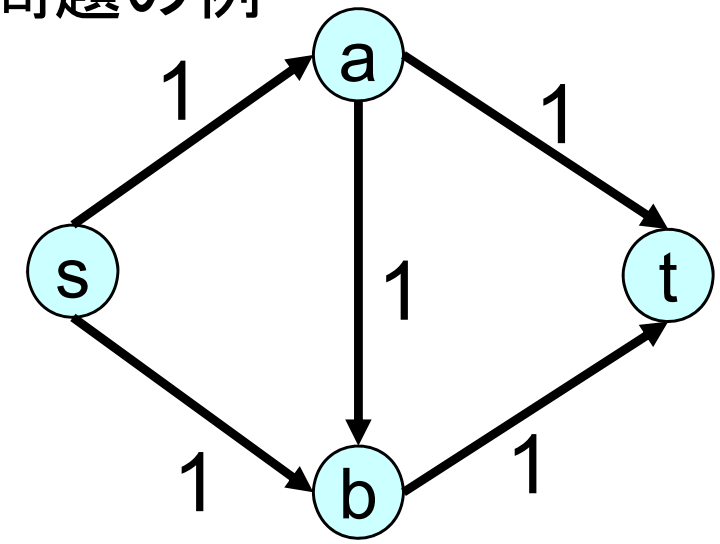


最大流問題は線形計画問題の特殊ケース
⇒　単体法（シンプレックス法）などの解法で解くことが可能

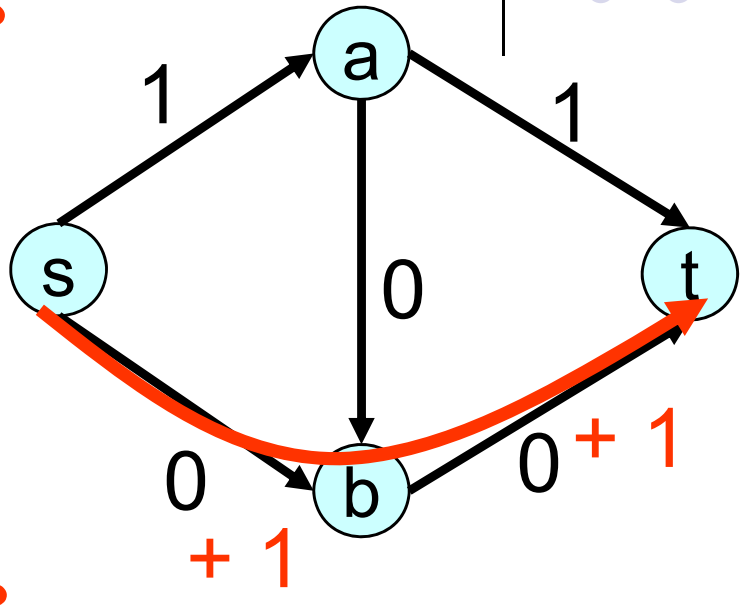
最大流問題は良い（数学的な）構造をもつ
　⇒　この問題専用の解法（増加路アルゴリズムなど）
　　を使うと，より簡単＆より高速に解くことが可能

最大フローの判定

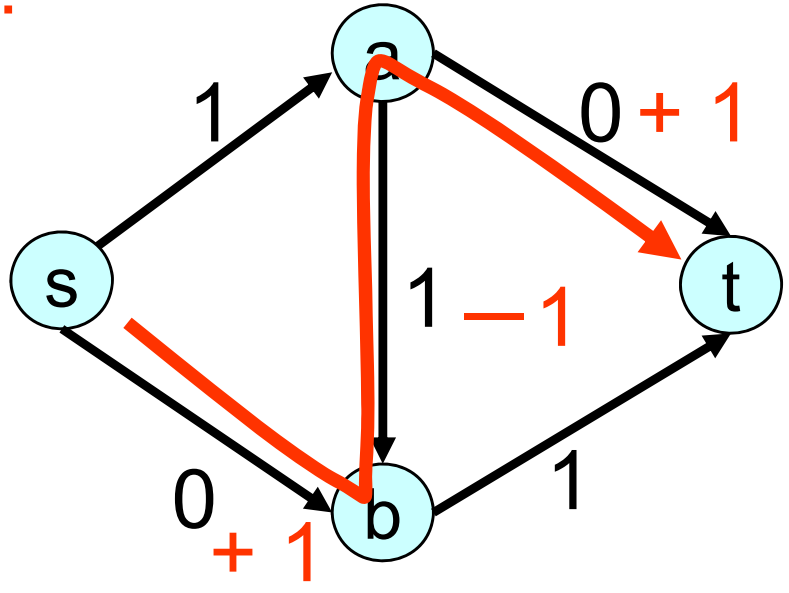
問題の例



フロー例1：最大？
最大ではない



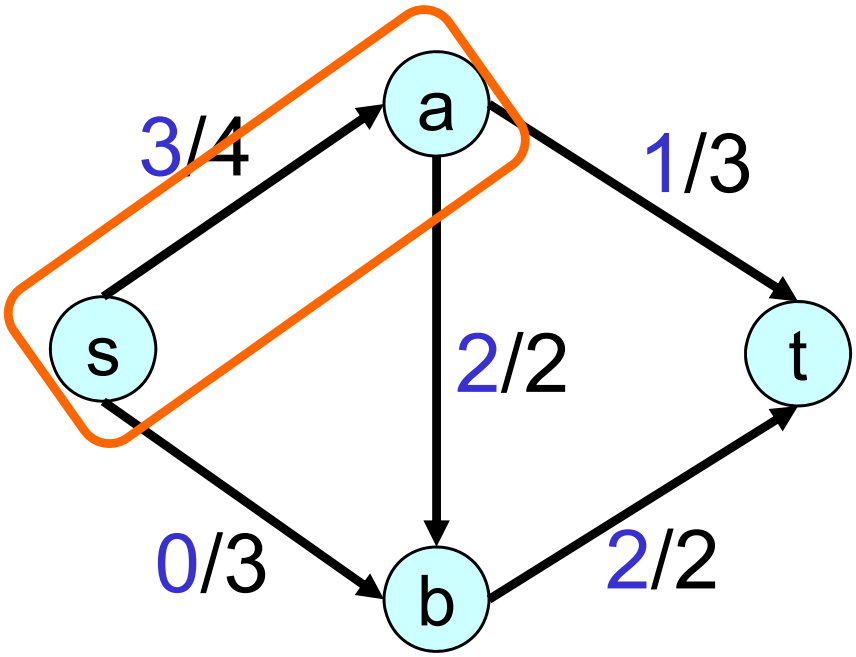
フロー例2：最大？
最大ではない



最大フローであることの判定を
効率よく行うには？
⇒　残余ネットワークを利用

残余ネットワークの定義

残余ネットワークの作り方



問題例とフロー

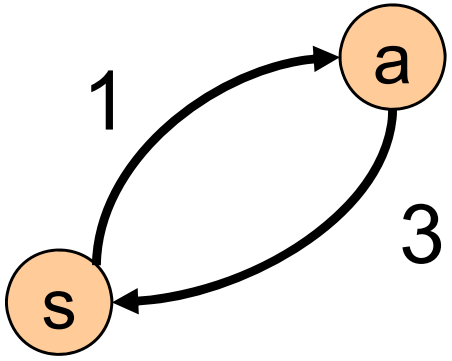
各枝のデータは

(フロー量/容量)

枝(s,a)において

☆さらに4－3＝1だけフローを流せる

⇒　残余ネットワークに
容量1の枝(s,a)を加える

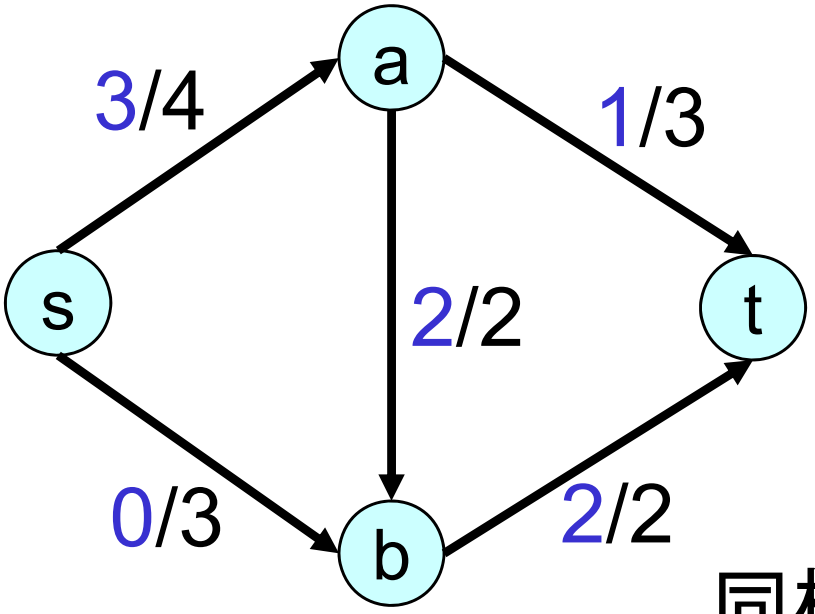


☆現在のフロー3を逆流させて0にすることが出来る

⇒　容量3の枝(a,s)を加える

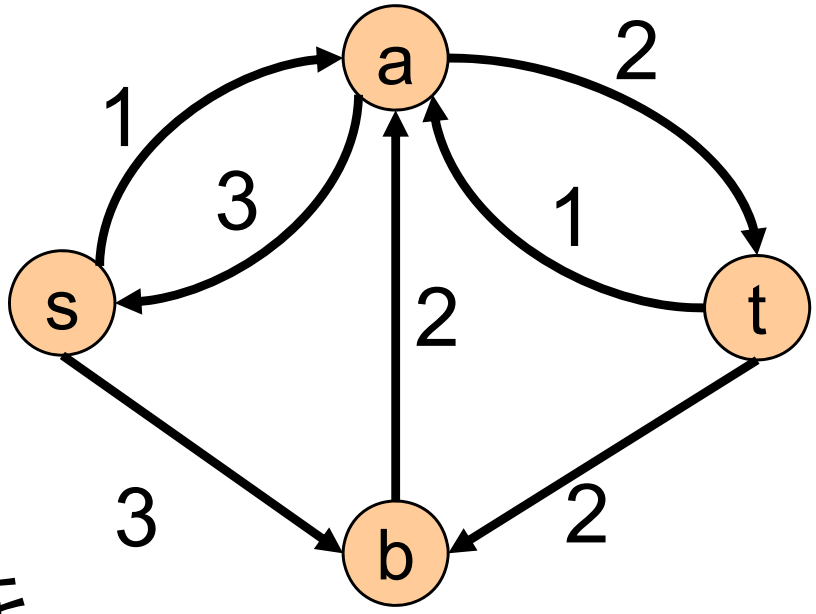
残余ネットワークの定義

残余ネットワークの作り方



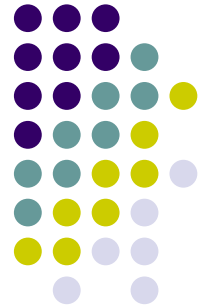
問題例とフロー

同様の操作を
各枝に行う

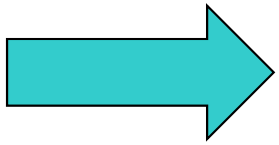


残余ネットワーク
の完成

残余ネットワークの定義（まとめ）



$x = (x_{ij} \mid (i,j) \in E)$: 現在のフロー



フロー x に関する残余ネットワーク $G^x = (V, E^x)$

$E^x = F^x \cup R^x$

順向きの枝集合

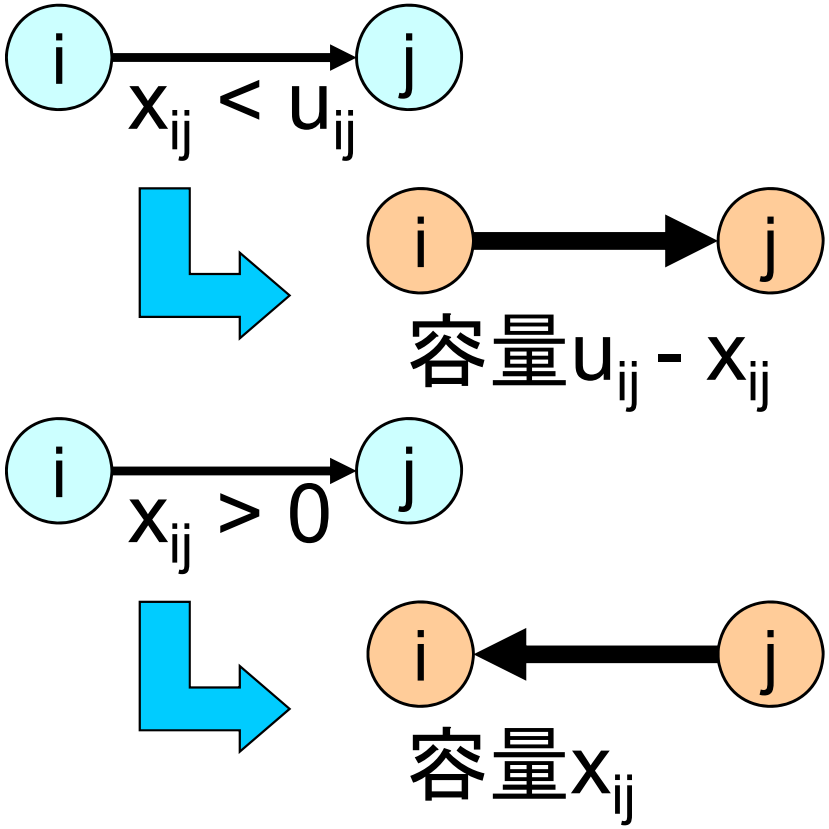
$F^x = \{ (i, j) \mid (i, j) \in E, x_{ij} < u_{ij}\}$

各枝の容量 $u^x_{ij} = u_{ij} - x_{ij}$

逆向きの枝集合

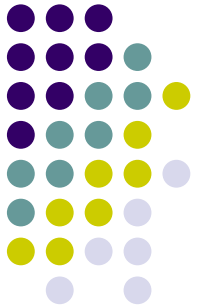
$R^x = \{ (j, i) \mid (i, j) \in E, x_{ij} > 0\}$

各枝の容量 $u^x_{ji} = x_{ij}$



注意！：現在のフローが変わると残余ネットワークも変わる

残余ネットワークに関する定理



定義：
増加路：残余ネットワークでの
ソース s からシンク t へのパス（路・みち）
増加路 P の容量 ＝ P の枝の容量の最小値

定理：残余ネットワークに 増加路が存在する
➔ 現在のフローの総流量は増加可能

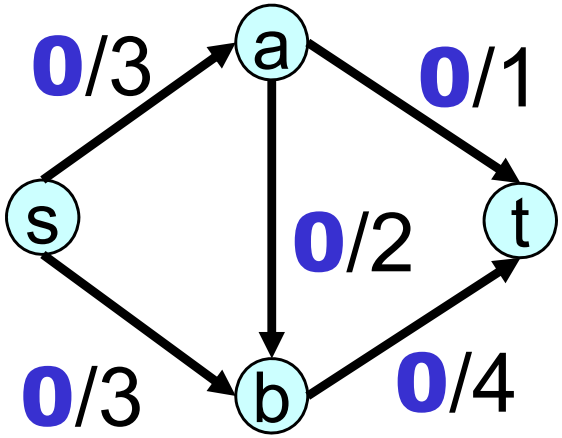
定理：残余ネットワークに 増加路が存在しない
➔ 現在のフローは最大フロー

増加可能なフローの例

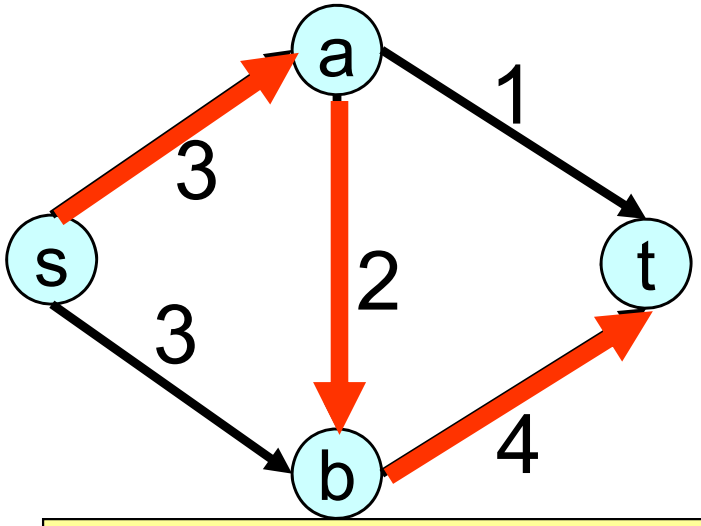
定理：残余ネットワークに増加路が存在する
➔ 現在のフローの総流量は増加可能

証明：増加路（s-tパス）を使うと，総流量を増加できる

現在のフロー**x**

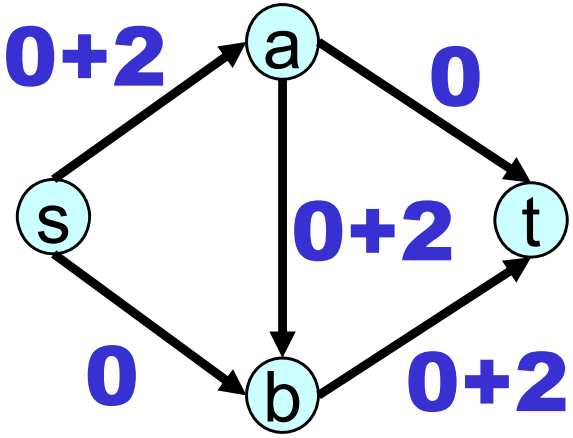


残余ネットワーク



増加路が存在．容量=2

新しいフロー**x’**



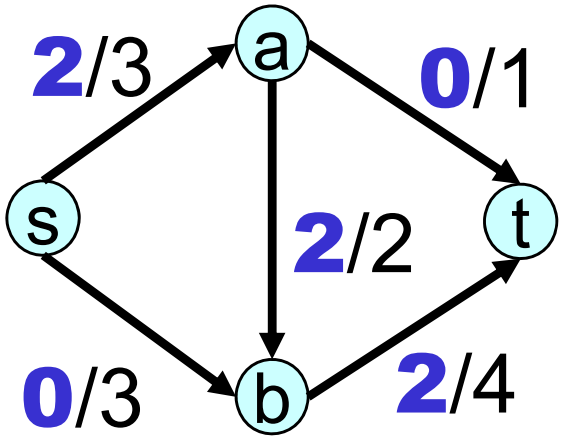
総流量が
2増えた

定義：増加路 P の容量
＝ P の枝の容量の最小値

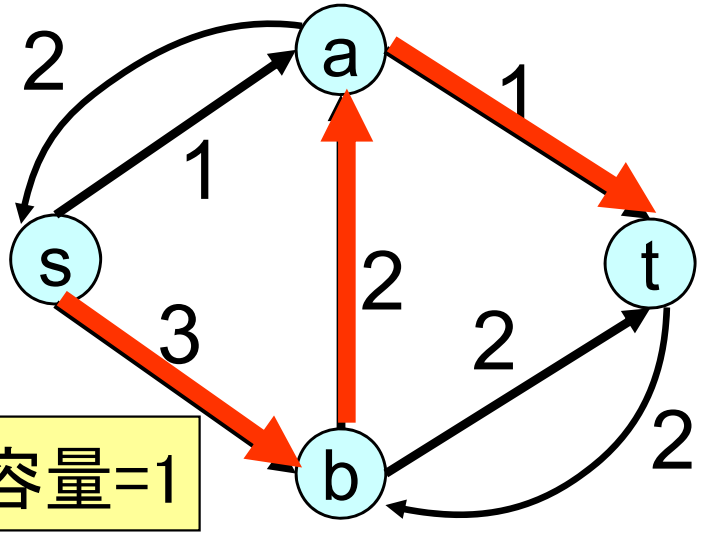
増加路の容量分，
総流量が増加

増加可能なフローの例

現在のフローx

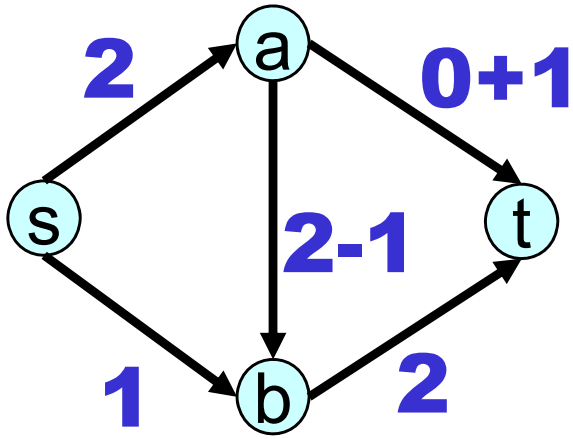


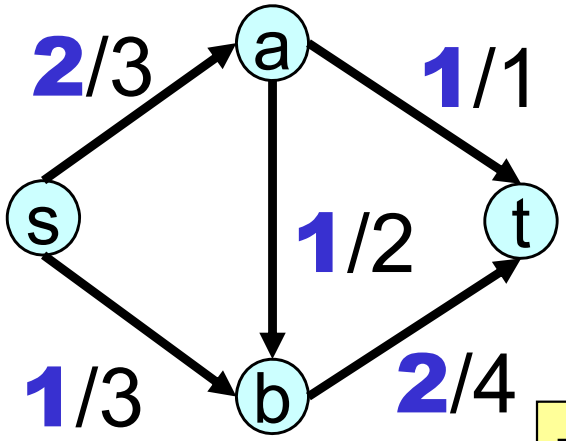
残余ネットワーク

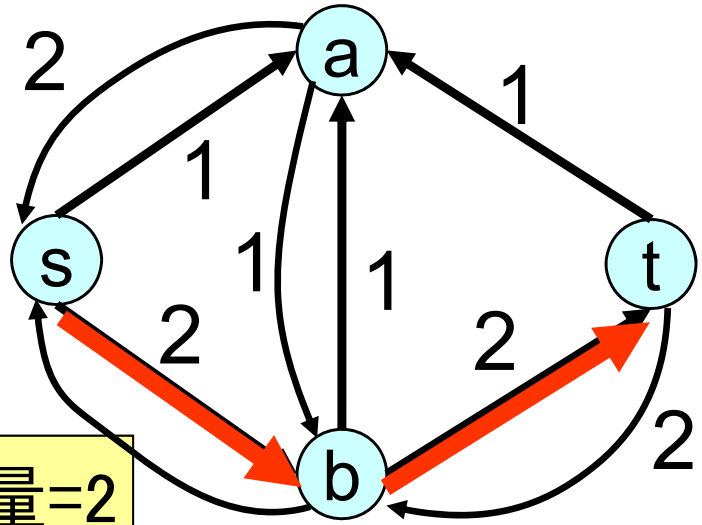


容量=1

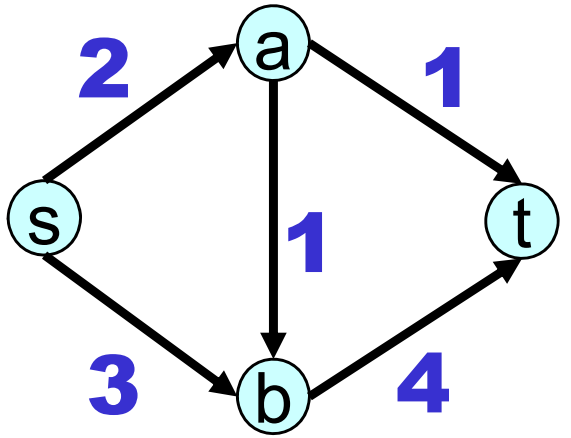
新しいフローx’





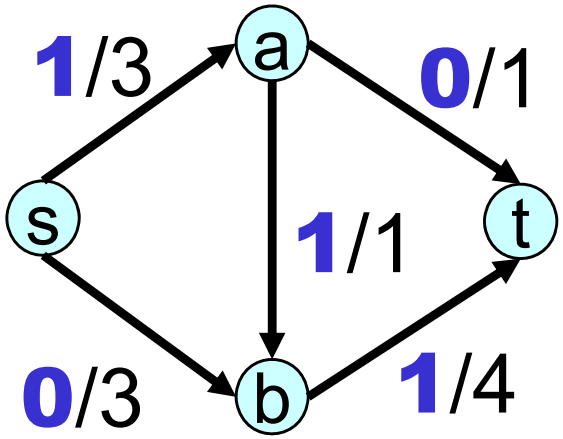


容量=2

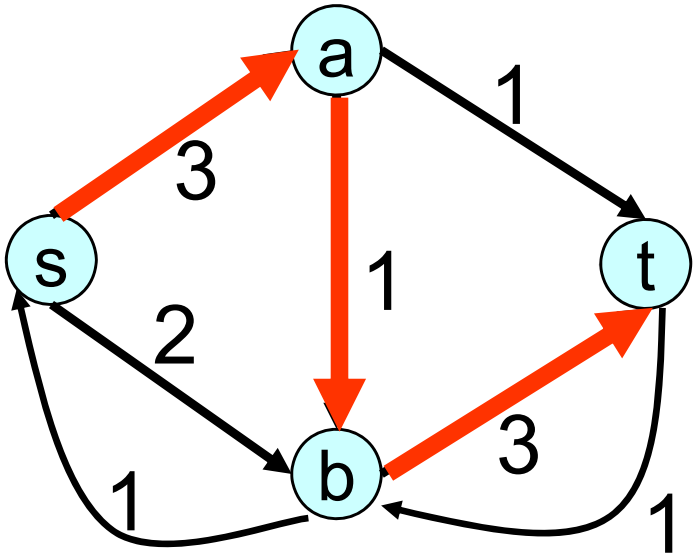


増加可能なフローの例

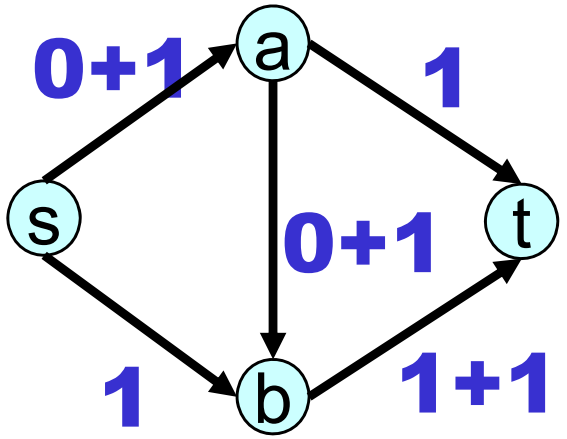
現在のフローx

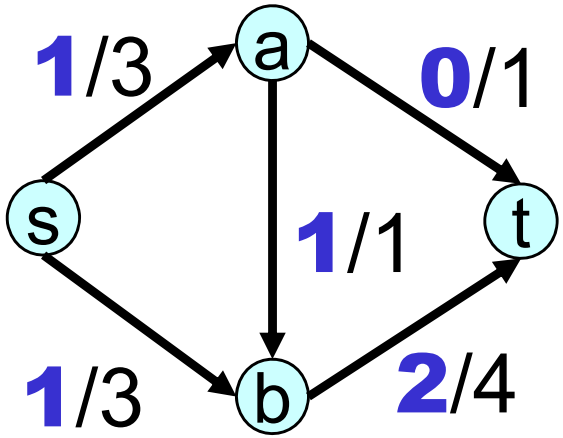


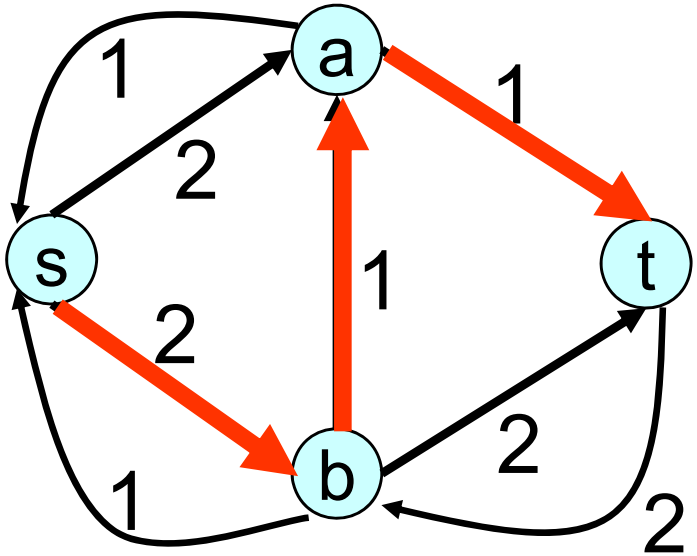
残余ネットワーク

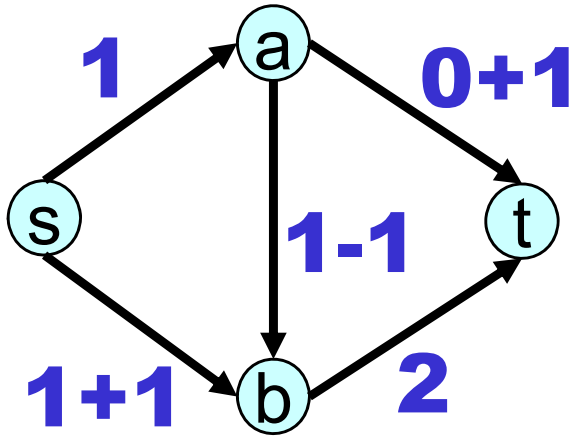


新しいフローx'







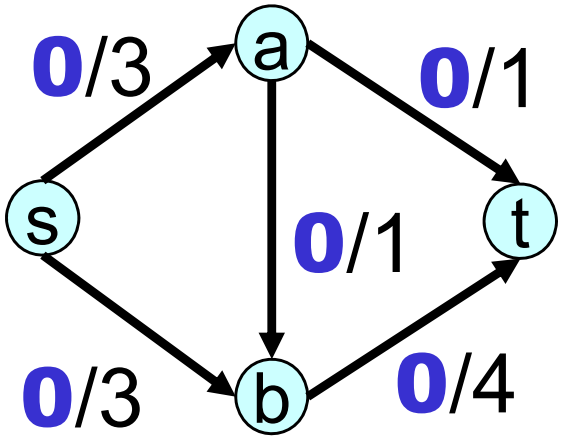


増加可能なフローの例

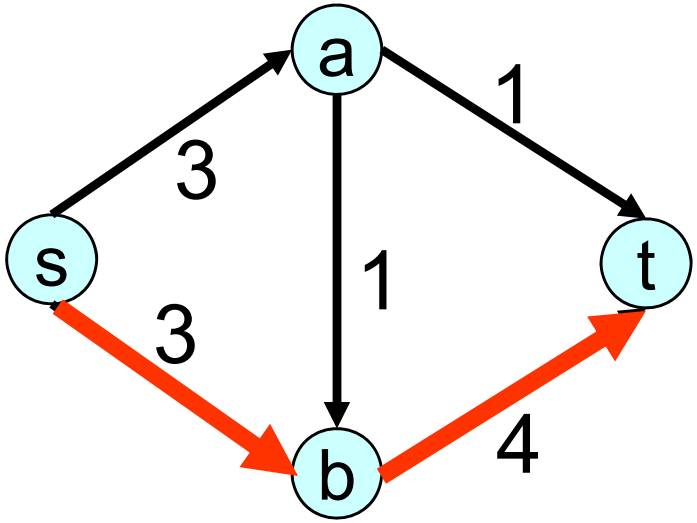
定理：残余ネットワークに増加路が存在する
➔ 現在のフローの総流量は増加可能

証明：増加路（s-tパス）を使うと，本当に総流量を増加できる

現在のフロー**x**

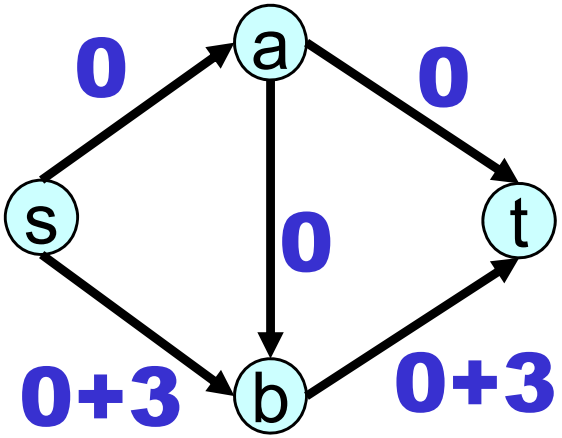


残余ネットワーク



増加路が存在．容量=3

新しいフロー**x’**



総流量が
3増えた

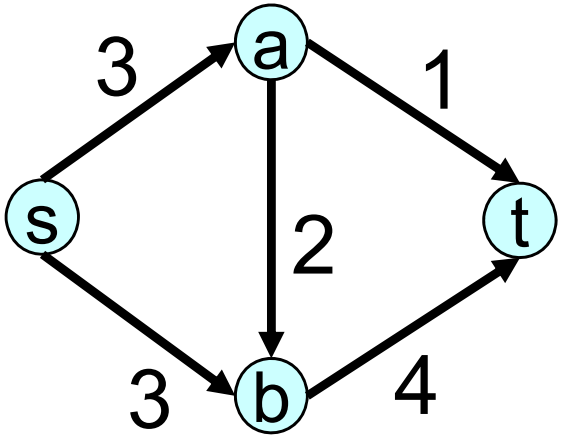
定義：増加路 P の容量
＝ P の枝の容量の最小値

増加不可能なフローの例

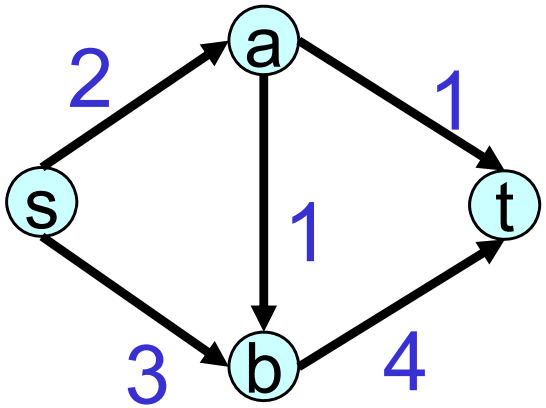
定理：残余ネットワークに s-t パスが存在しない
➔ 現在のフローは最大フロー

証明は後ほど

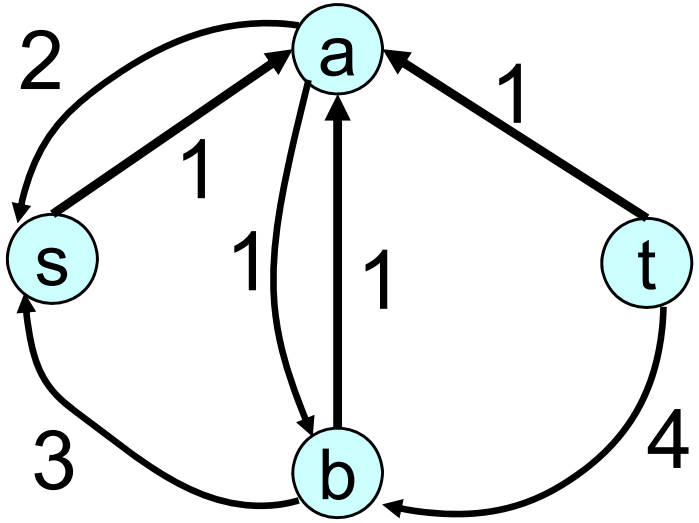
与えられた問題



現在のフロー

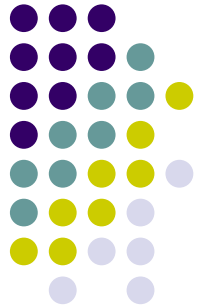


残余ネットワーク



s-t パスがない
➔現在のフローは最適！

増加路アルゴリズム

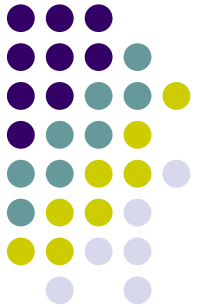


最大フローを求めるアルゴリズム

ステップ０：初期の実行可能フローとして，
　　全ての枝のフロー量を0とする
ステップ１：現在のフローに関する残余ネットワークを作る
ステップ２：残余ネットワークに 増加路が存在しない　⇒　終了
ステップ３：残余ネットワークの増加路 P をひとつ求める.
　　P の容量 α を計算する.
　　P と α を用いて現在のフローを更新する
ステップ４：ステップ1へ戻る

このアルゴリズムを最大マッチング問題に対して適用
　　➔ 最大マッチング問題に対する増加路アルゴリズム

増加路アルゴリズムの性質



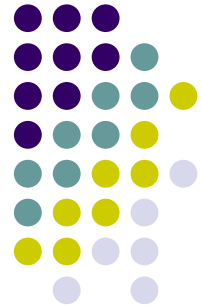
定理：残余ネットワークに s-t パスが存在しない
➔ 現在のフローは最大フロー

この定理より，増加路アルゴリズムの正当性が得られる

定理：増加路アルゴリズムは最大フローを求める．
とくに，枝容量がすべて整数のとき，整数値の最大フローを求める．

（後半の証明）アルゴリズムの開始前のフローは全て0（整数）．
各反復では，フローが整数値ならば，
残余ネットワークでの枝の容量は全て整数．
ゆえに，増加路の容量も整数．
したがて，フローの変化量も整数．
よって，アルゴリズムの終了時のフローも整数値． ■

増加路アルゴリズムの計算時間



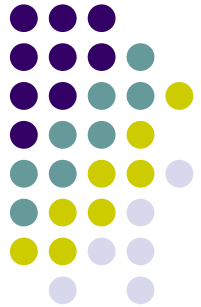
※各枝の容量は整数と仮定
　U = 容量の最大値
　m = 枝の数，n = 頂点の数

各反復において総流量が1以上増加
➔反復回数 ≦ 総流量の最大値 ≦ m U

各反復での計算時間
　＝ 残余ネットワークの増加路を求める時間
➔深さ優先探索，幅優先探索などを使うと O(m + n) 時間

∴ 計算時間は O((m+n) m U)
　（入力サイズは m + n + log U なので，指数時間）

増加路アルゴリズムの改良



反復回数を少なくしたい
➔ 各反復での増加路の選び方を工夫する

（改良法１）各反復での総流量の増加量を大きくする
➔各反復で容量最大の増加路を選ぶ
➔反復回数 $O(m \log (n\, U))$，計算時間 $O(m^2 \log (n\, U))$

（改良法2）各反復で最短（枝数最小）の増加路を選ぶ
➔反復回数 $O(m\, n)$，計算時間 $O(m^2 n)$

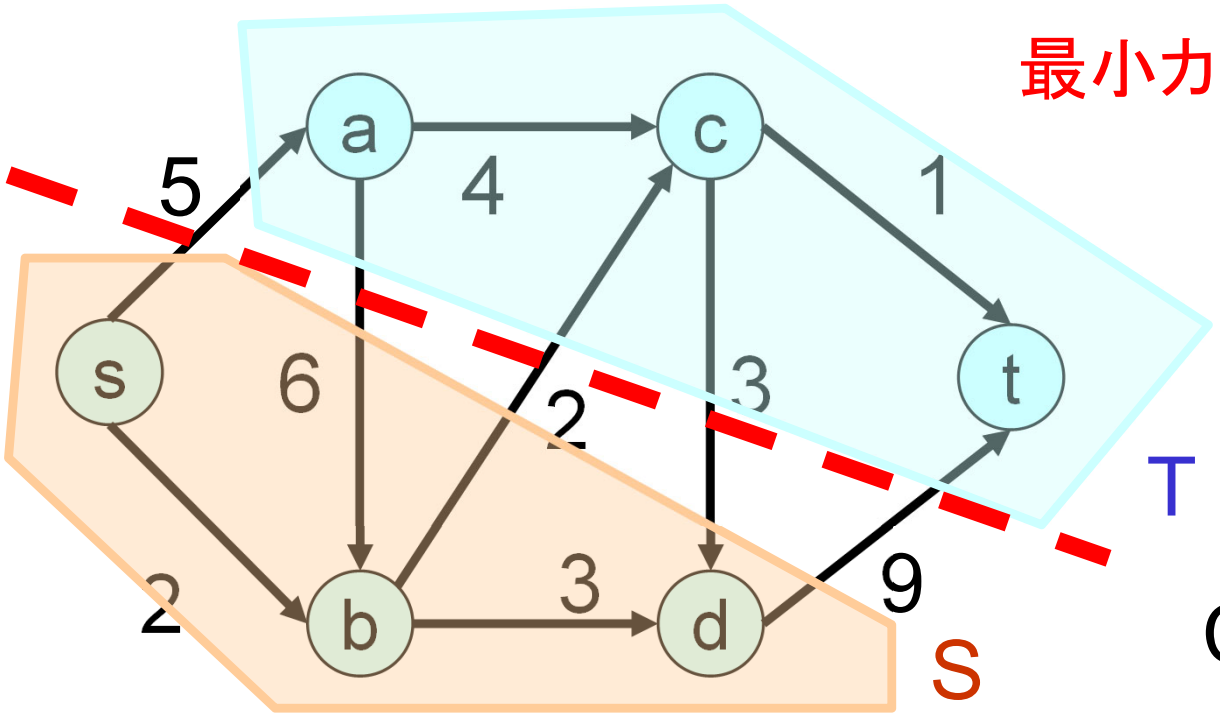
※この他にも，増加路アルゴリズムの計算時間を短縮するための
様々なテクニックが存在
全く違うアイディアのアルゴリズム：「プリフロー」を利用

カット

フローを流すとき，ネットワークのボトルネックはどこ？

カット (S, T): S, T は頂点集合Vの分割($S \cap T = \emptyset, S \cup T = V$)
S はソース s を含む，Tはシンク t を含む

カット (S, T) の容量C(S,T) ＝ SからTへ向かう枝の容量の和



最小カット：容量が最小のカット

C(S,T)=5+2+9=16

カットの性質（その1）

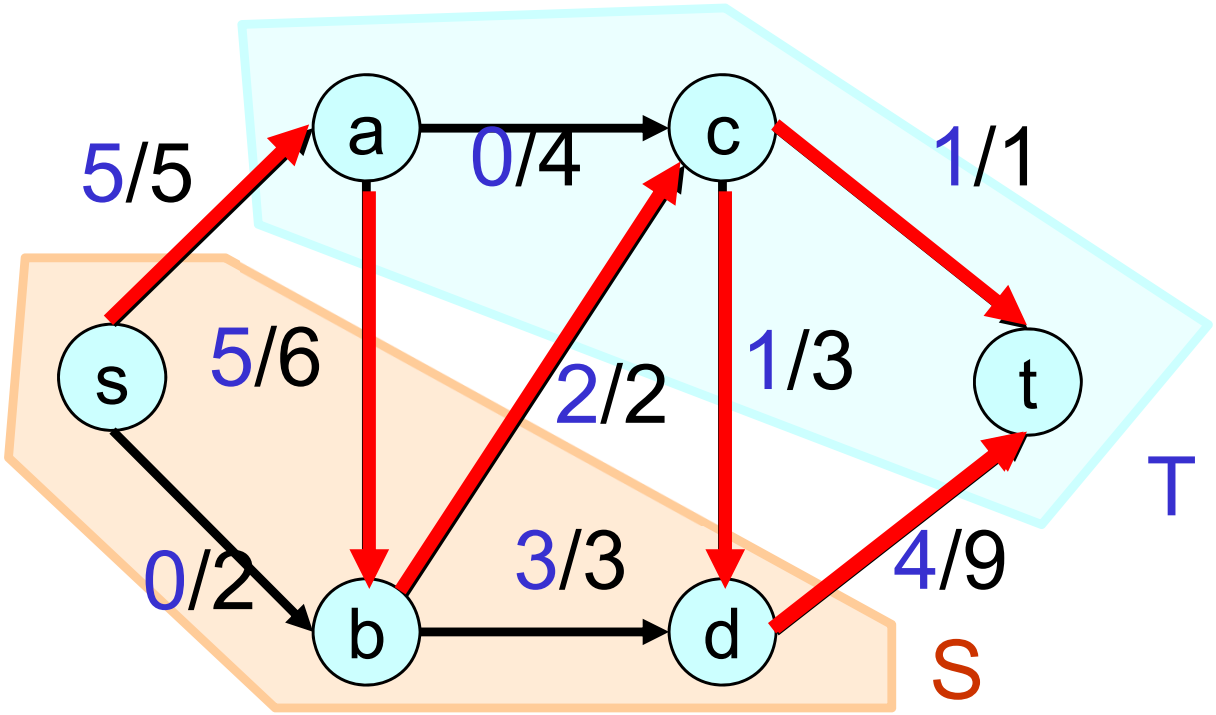
性質：

任意のカット(S, T) と任意の実行可能フロー $(x_{ij} \mid (i,j) \in E)$ に対し

SからTへの枝のフローの和 x(S,T)

－ TからSへの枝のフローの和 x(T,S)

＝ フローの総流量 f



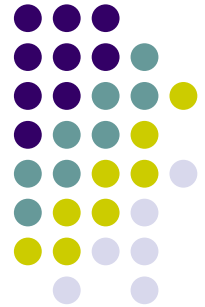
f = 1 + 4 = 5

x(S, T) = 5 + 2 + 4 = 11

x(T, S) = 5 + 1 = 6

f = 11 – 6 = 5

カットの性質（その1）



一般の場合の証明：下記の制約式を足し合わせる

$$\sum\{x_{kj} \mid (k,j) \text{ は } k \text{ から出る}\} - \sum\{x_{ik} \mid (i,k) \text{ は } k \text{ に入る}\} = 0 \qquad (k \in S - \{s\})$$

$$\sum\{x_{sj} \mid (s,j) \text{ は } s \text{ から出る}\} - \sum\{x_{is} \mid (i,s) \text{ は } s \text{ に入る}\} = f$$

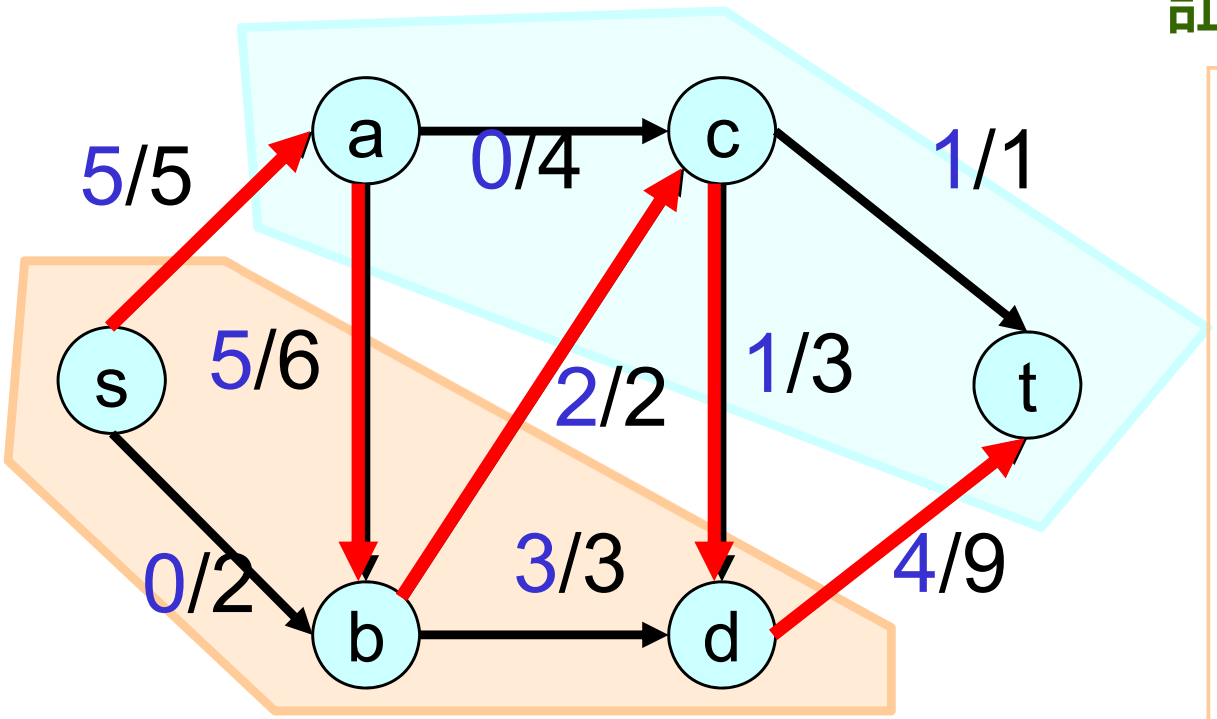
左辺の和をとる

- SからTへの枝 の変数 x_{ij} は係数が＋1
- TからSへの枝 の変数 x_{ij} は係数が－1
- SからSへの枝 の変数 x_{ij} は打ち消される
- TからTへの枝 の変数 x_{ij} は登場しない

⇒ 左辺＝ $x(S, T) - x(T, S)$

カットの性質（その2）

性質：任意のカット(S, T) とフロー $(x_{ij} \mid (i,j) \in E)$ に対し
フローの総流量 f ≦ カットの容量 C(S,T)



f = 5 ≦ 16 = C(S, T)

証明：

f = x(S, T) – x(T,S)
（性質１）
x(S, T) ≦ C(S, T)
（容量条件）
x(T, S) ≧ 0
（フローは非負）
∴ f ≦ C(S, T) – 0
= C(S, T)

最小カット問題

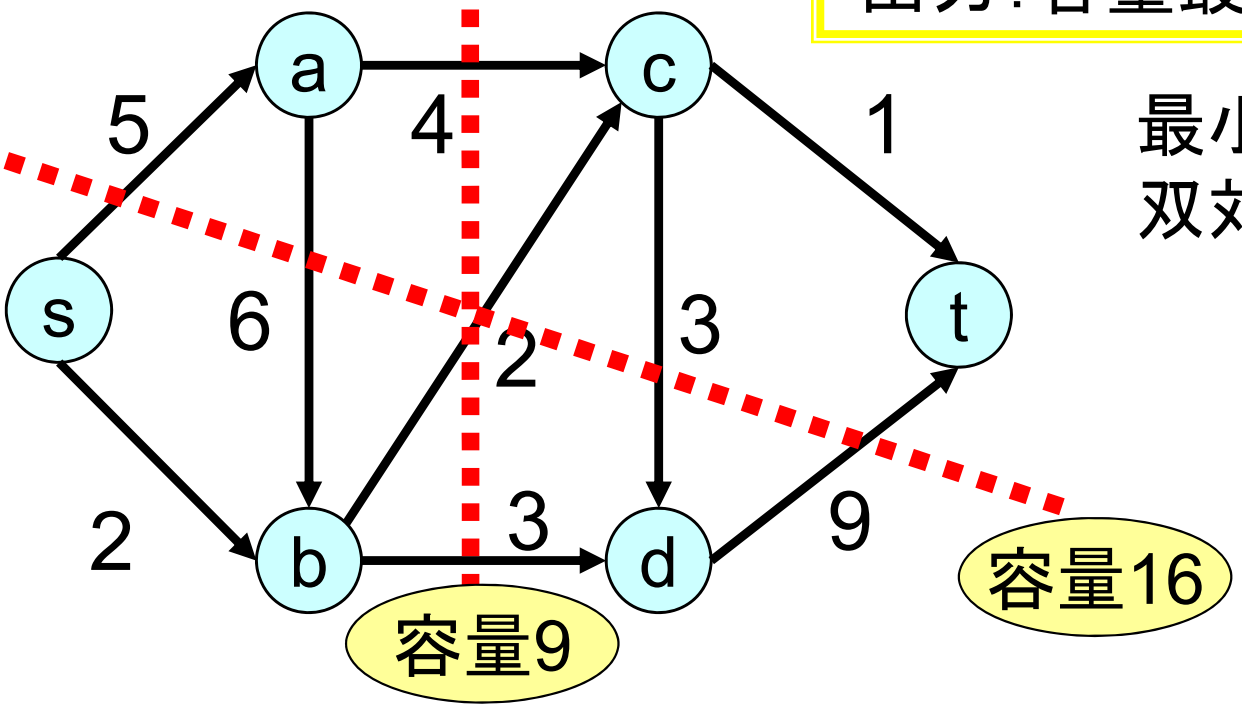
性質：任意のカットとフローに対し
フローの総流量 ≦ カットの容量

LPの弱双対定理に対応

⇒ カットの容量は，最大フローの総流量に対する上界

より良い上界を求めたい⇒

最小カット問題
入力：グラフ G = (V, E), 頂点 s, t ∈V
出力：容量最小の s-t カット（最小カット）



最小カット問題は最大流問題の双対問題

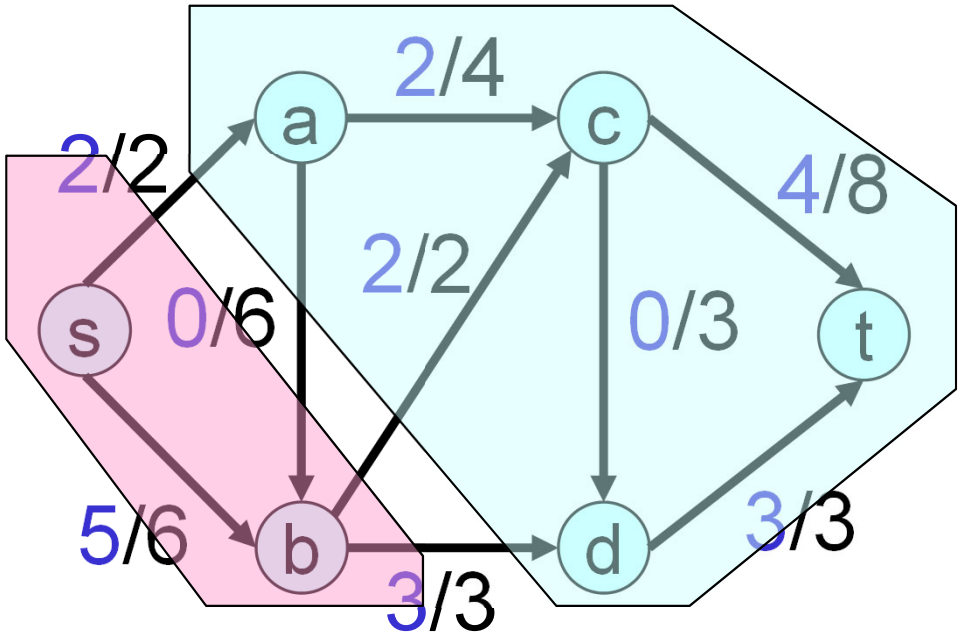
カットの性質（その3）

増加路アルゴリズムの正当性の証明に使用

性質：任意のカット(S, T) と実行可能フロー $(x_{ij} \mid (i,j) \in E)$ に対し
フローの総流量 f ＝ カットの容量 C(S,T) が成り立つ

➔ 現在のフローは最大フロー，カットは最小カット

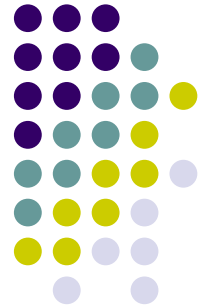
[証明] 性質2を使えば，LPに対する弱双対定理の系と同様に示せる．



f = 7，　C(S, T) = 7

➔ 現在のフローは最大フロー，
カットは最小カット

最大フロー最小カット定理



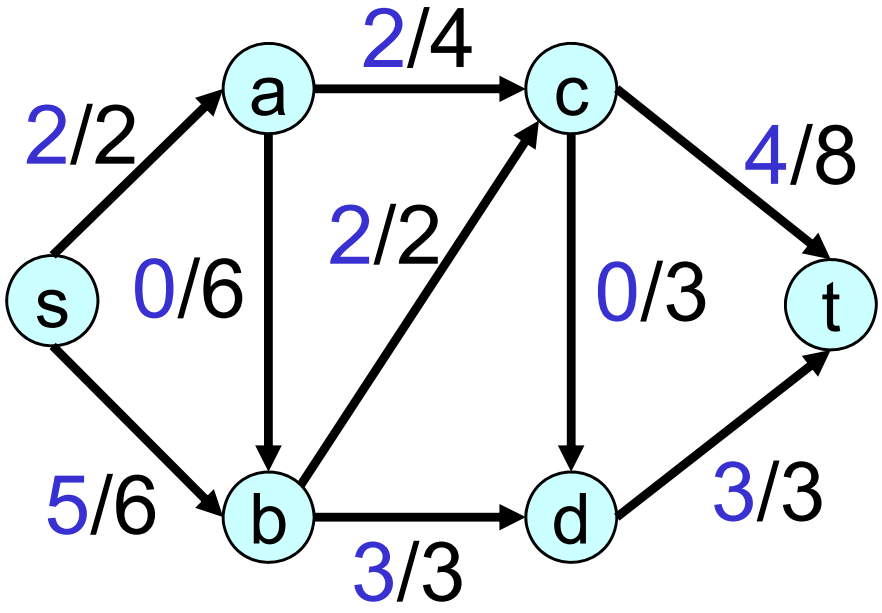
増加路アルゴリズムの正当性の証明

定理：増加路アルゴリズムは最大フローを求める.
また,
S = 残余ネットワークで s より到達可能な頂点集合
T = V – S
とすると、(S, T) は 最小カット .
さらに, f = C(S,T) が成立

この定理より「最大フローの総流量＝最小カットの容量」が成立

最大フロー最小カット定理：
最大フロー $(x_{ij} \mid (i,j) \in E)$ と最小カット(S,T)に対し
$f = C(S,T)$

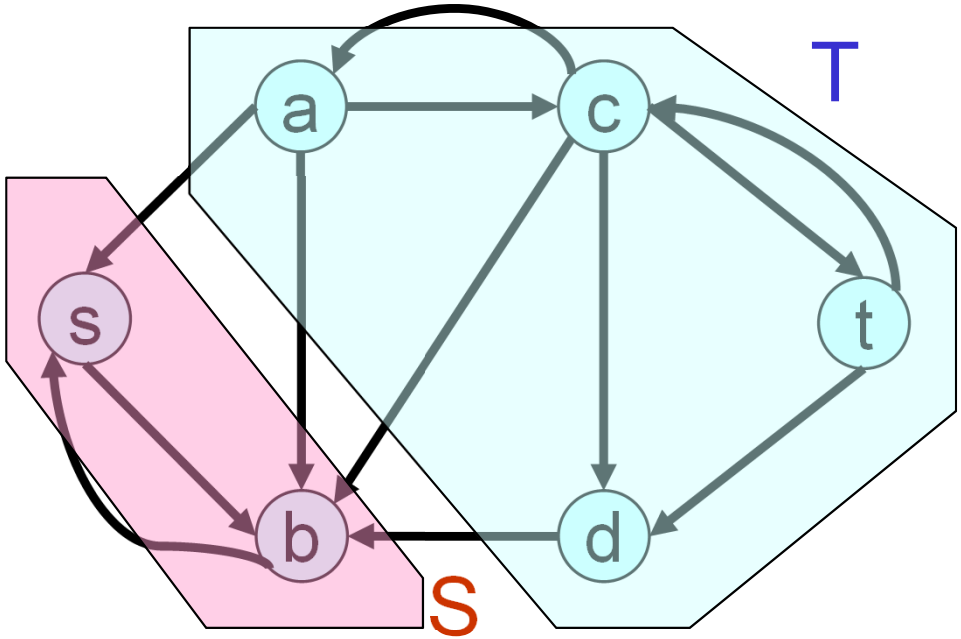
増加路アルゴリズムの正当性(その1)



目標:アルゴリズム終了時のフローに対し, f = C(S,T)を満たすカット(S,T)を見つける➔性質3より最大フロー

アルゴリズム終了時のフローに対して残余ネットワークを作る

残余ネットワークには増加路がない

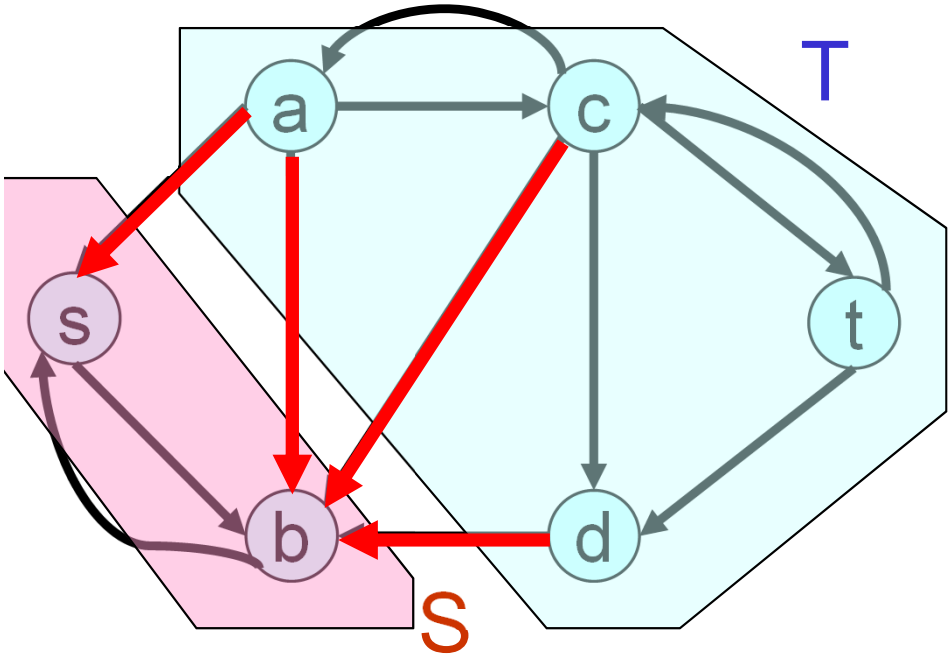


S = 残余ネットワークにおいて
s から到達可能な頂点集合

T = V – S

に対し、(S, T) はカット

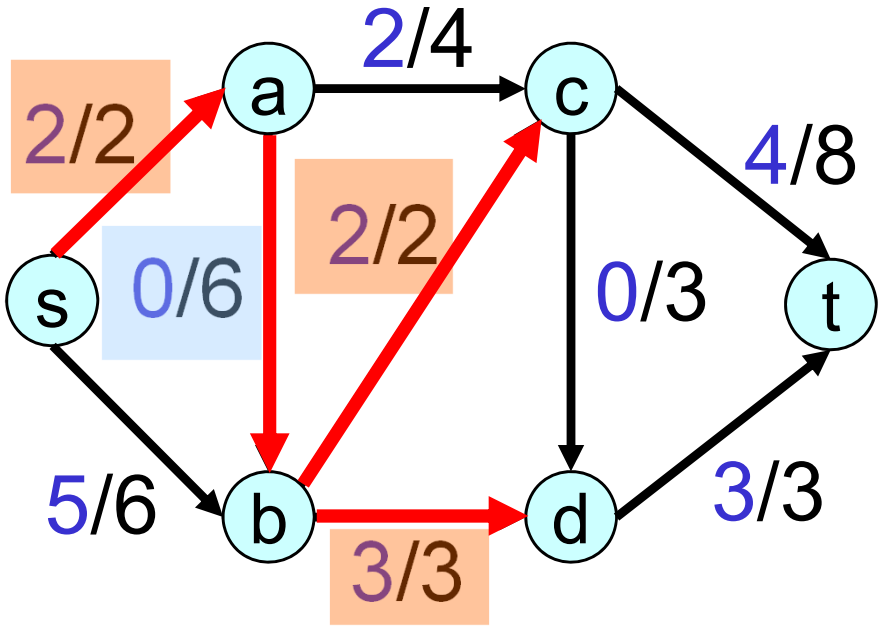
増加路アルゴリズムの正当性(その2)



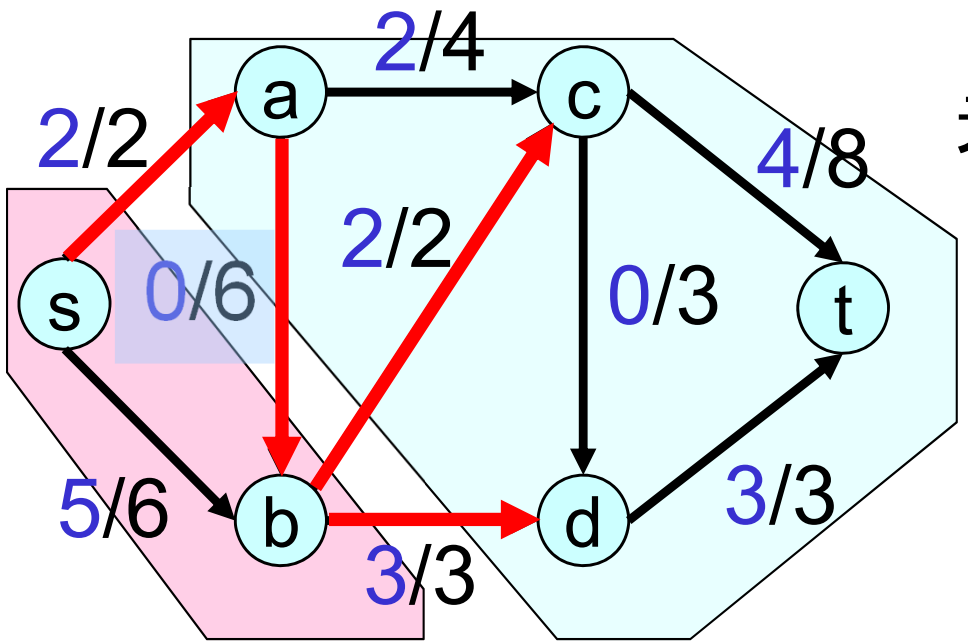
S = s から到達可能な頂点集合
T = V – S

残余ネットワークにおいて
SからTに向かう枝は存在しない

元のネットワークにおいて
SからTに向かう枝では $x_{ij} = u_{ij}$
TからSに向かう枝では $x_{ij} = 0$



増加路アルゴリズムの正当性（その3）



元のネットワークにおいて
SからTに向かう枝では $x_{ij} = u_{ij}$
TからSに向かう枝では $x_{ij} = 0$

$x(S, T) = \sum\{x_{ij} \mid (i,j)$ はSからTへ向かう枝$\}$
$= \sum\{u_{ij} \mid (i,j)$ はSからTへ向かう枝$\} = C(S, T)$
$x(T, S) = \sum\{x_{ij} \mid (i,j)$ はTからSへ向かう枝$\} = 0$
$\therefore\ x(S, T) - x(T, S) = C(S, T)$

性質1より $f = x(S,T) - x(T, S)$

$\therefore\ f = C(S, T)$ （証明終わり）