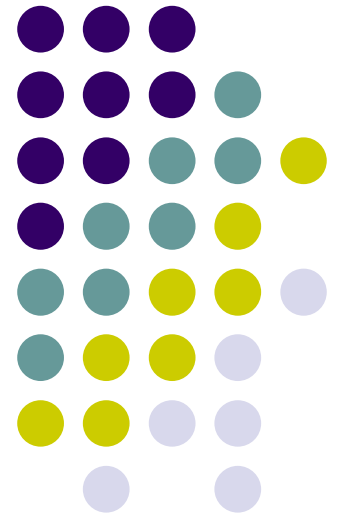


数理計画法 第7回

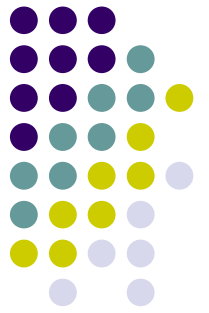
ネットワーク計画 最大流問題とフロー増加法

担当： 塩浦昭義
(情報科学研究科 准教授)
shioura@dais.is.tohoku.ac.jp



中間試験について

- 日時: 11月29日(木) 13:00~14:30
- 場所: 総合研究棟110講義室(この部屋)
- 11/22までにレポートを一度も出していない場合, 受験不可
- **手書きのA4用紙一枚のみ**持ち込み可(**印刷やコピーは不可**)
 - これも採点の対象, 試験終了後に回収します
- 教科書, ノート等の持ち込みは不可
- 座席はこちらで指定
- 試験内容: 11/15(第6回目)までの講義で教えたところ
 - 様々な数理計画モデル
 - 線形計画問題の標準形, 単体法, 各種定理
 - 組合せ計画問題(分枝限定法)
- 50点満点, **29点以下は原則として不合格**



今後の予定

12月は講義室が頻繁に変わります！

- 11/29 第8回目 --- 中間試験
- 12/6 第9回目 --- ネットワーク計画その2
 - 情報科学研究科棟大講義室
- 12/13 第10回目 --- ネットワーク計画その3
 - 青葉記念会館4階
- 12/20 第11回目 --- 非線形計画その1
 - 総合研究棟110講義室

ネットワーク計画問題

- (無向、有向)グラフ

- 頂点 (vertex, 接点、点) が枝 (edge, 辺、線) で結ばれたもの

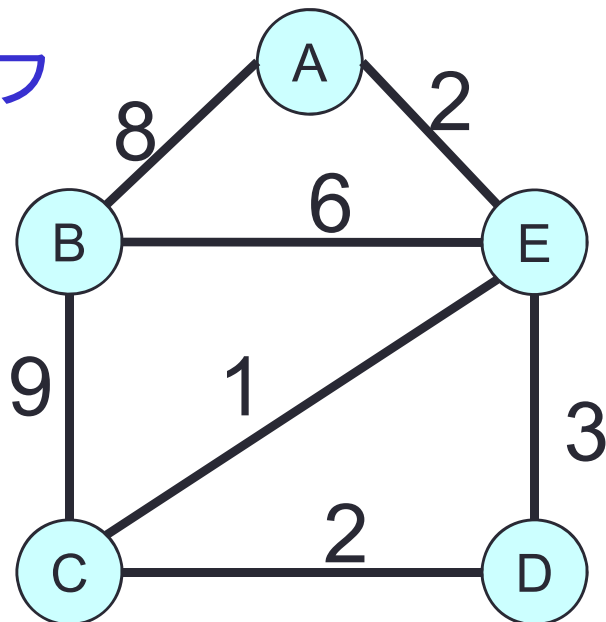
- ネットワーク

- 頂点や枝に数値データ (距離、コストなど) が付加されたもの

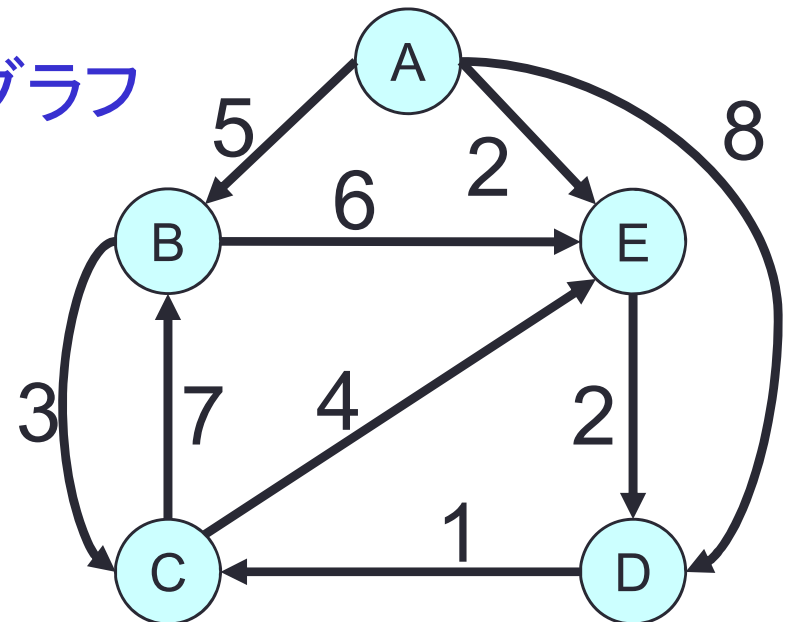
- ネットワーク計画問題

- ネットワークを使って表現される数理計画問題

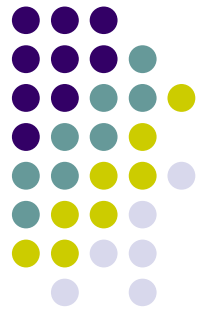
無向グラフ



有向グラフ



ネットワーク計画問題の例



「ネットワーク」に関する数理計画問題(最適化問題)

例: 最小木問題

(minimum spanning tree prob.)

最短路問題

(shortest path prob.)

最大流問題

(maximum flow prob.)

最小費用流問題

(minimum cost flow prob.)

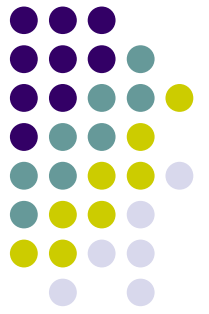
割当問題

(assignment prob.)

他の講義で扱う
「アルゴリズムとデータ構造」
「情報数学」

この授業で扱う

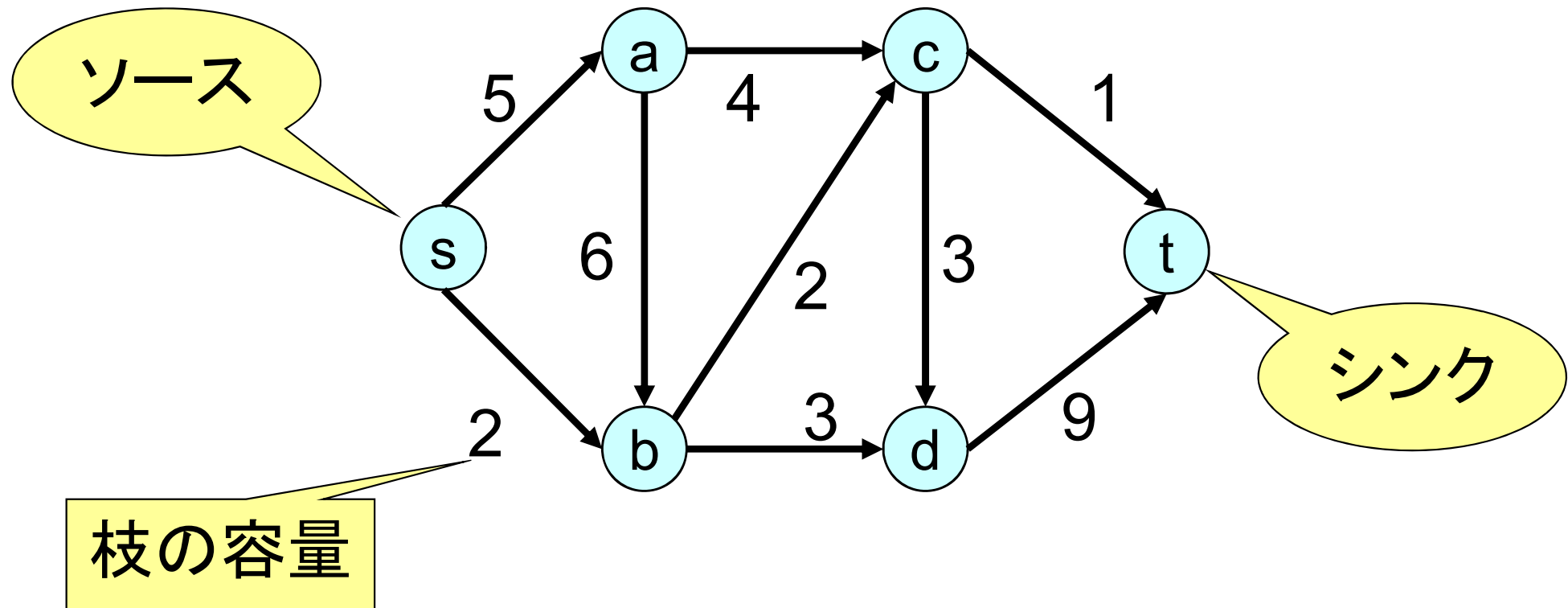
最大流問題の定義(その1)



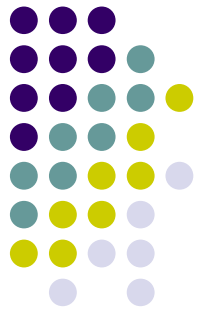
入力: 有向グラフ $G = (V, E)$

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 $u_{ij} \geq 0$



最大流問題の定義(その2)



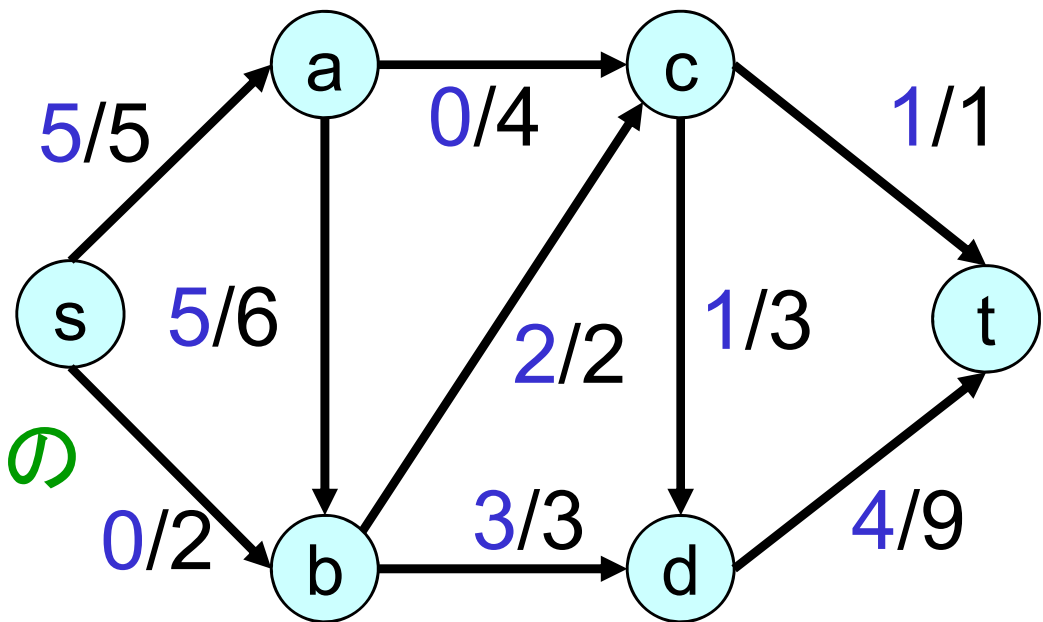
目的: ソースからシンクに向けて、枝と頂点を経由して
「もの」を出来るだけたくさん流す

条件1 (容量条件, capacity constraint):

$0 \leq$ 各枝を流れる「もの」の量 \leq 枝の容量

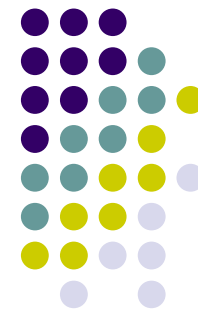
条件2 (流量保存条件, flow conservation constraint):

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 : ソースからシンクへの総流量

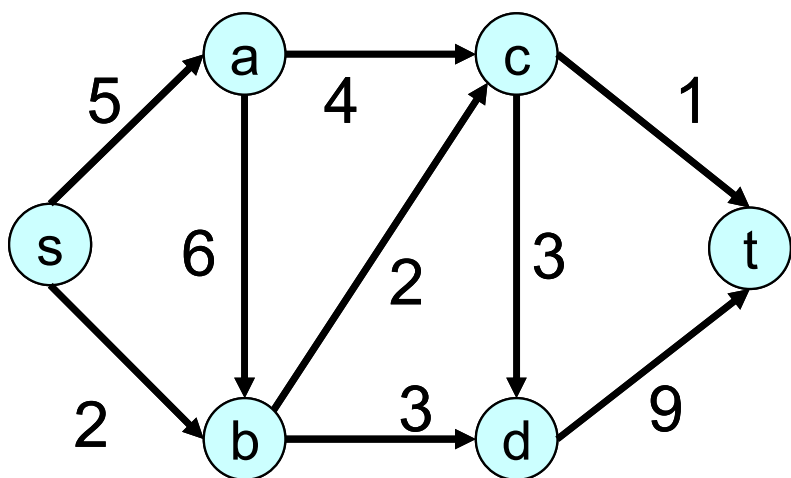
= シンクに流れ込む「もの」の量 = ソースから流れ出す「もの」の量

目的: ソースからシンクに「もの」をたくさん流したい

⇒ 目的関数 f → 最大

容量条件: $0 \leq$ 各枝を流れる「もの」の量 \leq 枝の容量

⇒ $0 \leq x_{ij} \leq u_{ij} \quad ((i, j) \in E)$



具体例

目的: 最大化 f

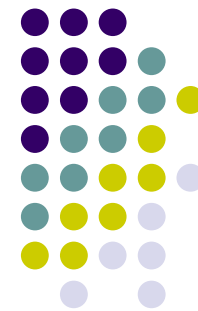
容量条件:

$0 \leq x_{sa} \leq 5, 0 \leq x_{sb} \leq 2, 0 \leq x_{ab} \leq 6,$

$0 \leq x_{ac} \leq 4, 0 \leq x_{bc} \leq 2,$

...

最大流問題の定式化: 流れ保存則



流れ保存則 (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

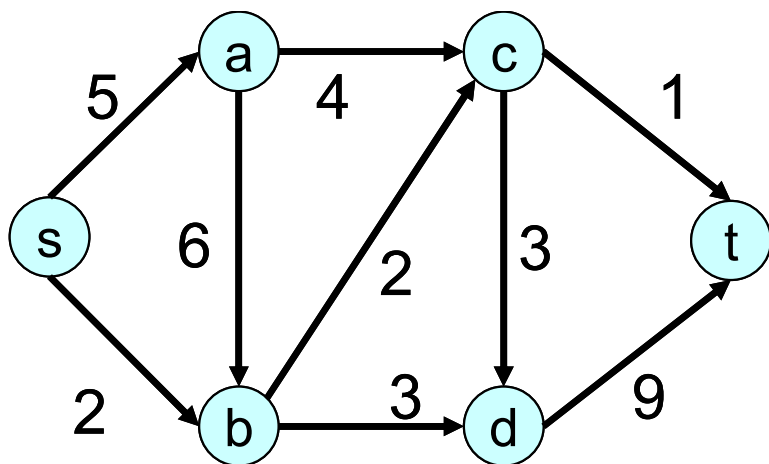
⇒ $\sum\{x_{kj} \mid \text{枝}(k,j) \text{ は頂点 } k \text{ から出る}\}$

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

ソースとシンクに関する条件:

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

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



流れ保存則の例:

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

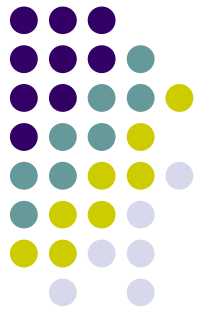
$$x_{bc} + x_{bd} - x_{ab} - x_{sb} = 0$$

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

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

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

最大流問題の定式化:まとめ



目的関数 $f \rightarrow$ 最大

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

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

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

$\sum\{x_{sj} \mid (s,j) \text{ は } s \text{ から出る}\}$

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

$\sum\{x_{tj} \mid (t,j) \text{ は } t \text{ から出る}\}$

$-\sum\{x_{it} \mid (i,t) \text{ は } t \text{ に入る}\} = -f$

この問題の許容解 x_{ij} --- フロー(flow)
フローの目的関数値 f --- 流量

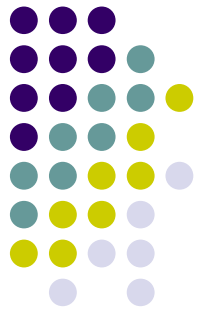


最大流問題の応用例

- 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



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

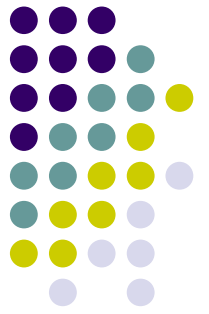
⇒ 単体法で解くことが可能！

最大流問題は良い(数学的な)構造をもつ

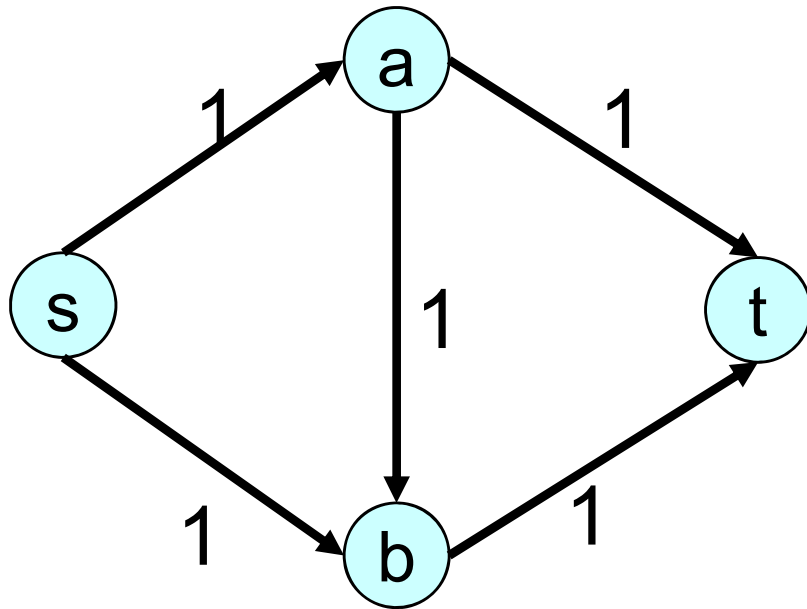
⇒ この問題専用の解法(フロー増加法など)

を使うと, より簡単, かつより高速に解くことが可能

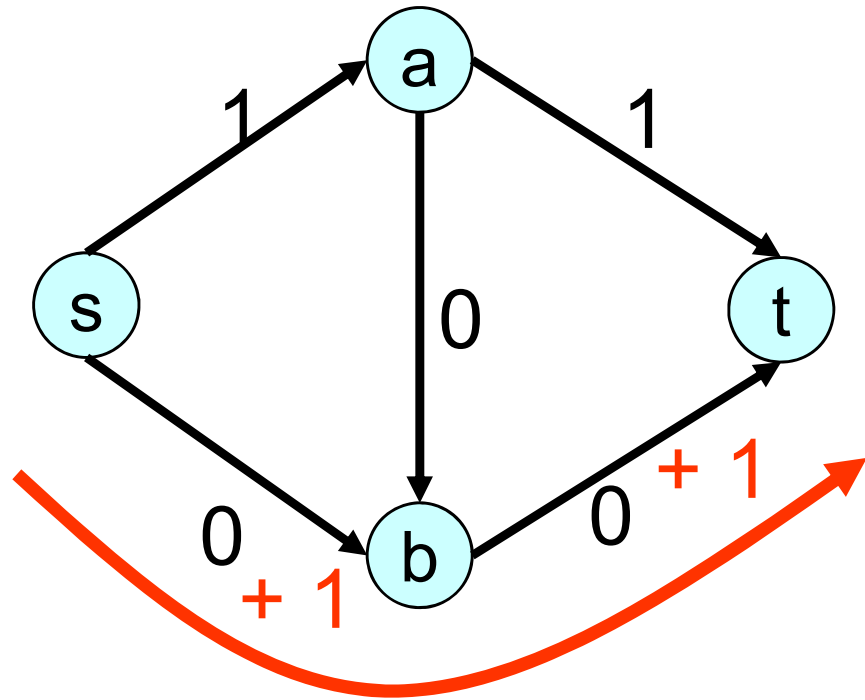
最大流の判定



問題の例

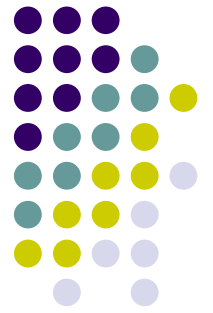


フローの例1: **最大?**

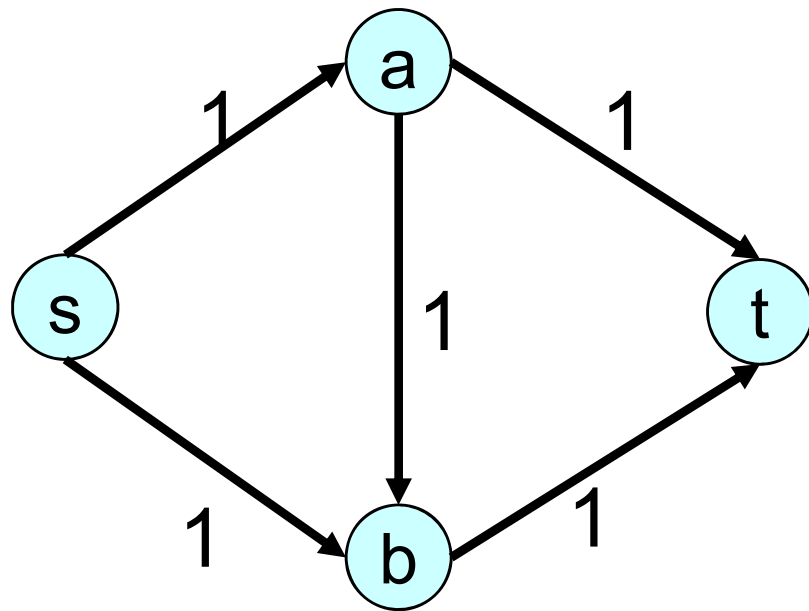


最大流ではない

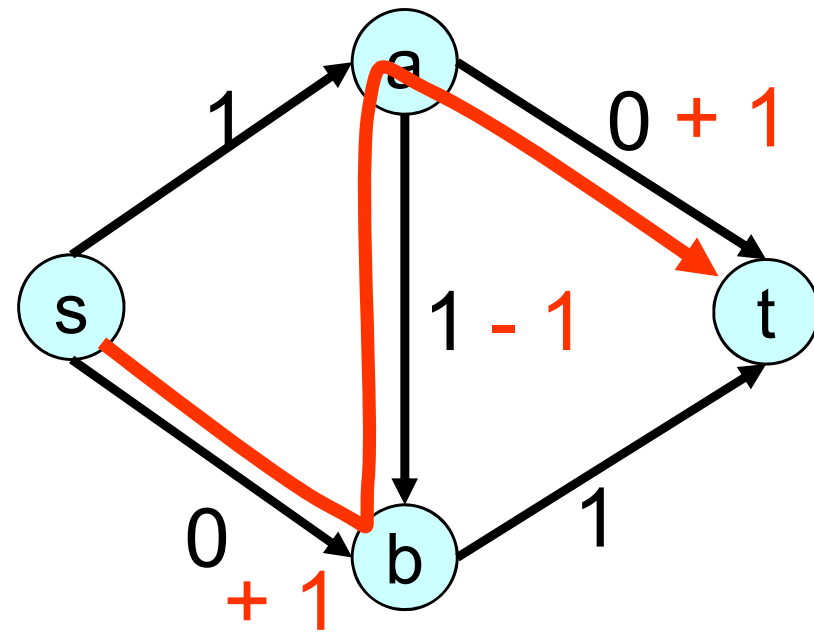
最大流の判定



問題の例



フローの例2: **最大?**

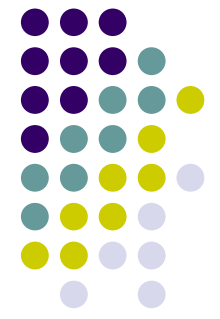


最大流ではない

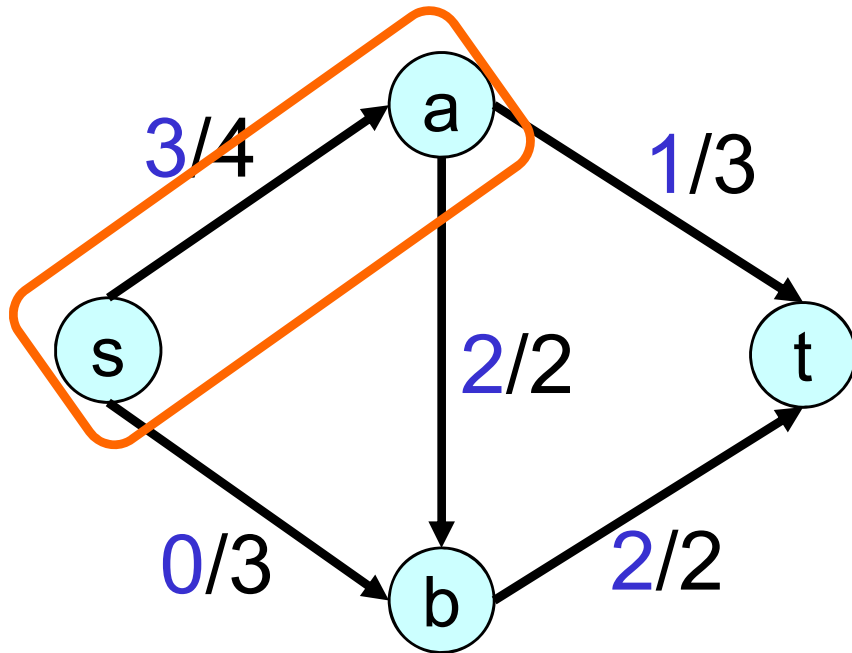
最大流であることの判定を効率よく行うには？

⇒ **残余ネットワーク(residual network)**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

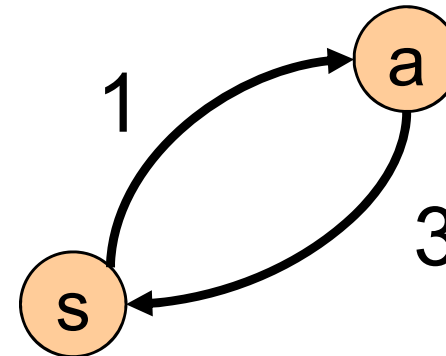


問題例とフロー
各枝のデータは
(**フロー量**/容量)

枝(s,a)において

☆さらに $4 - 3 = 1$ だけフロー
を流せる

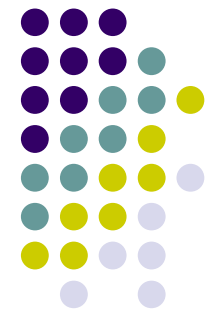
⇒ 残余ネットワークに
容量1の枝(s,a)を加える



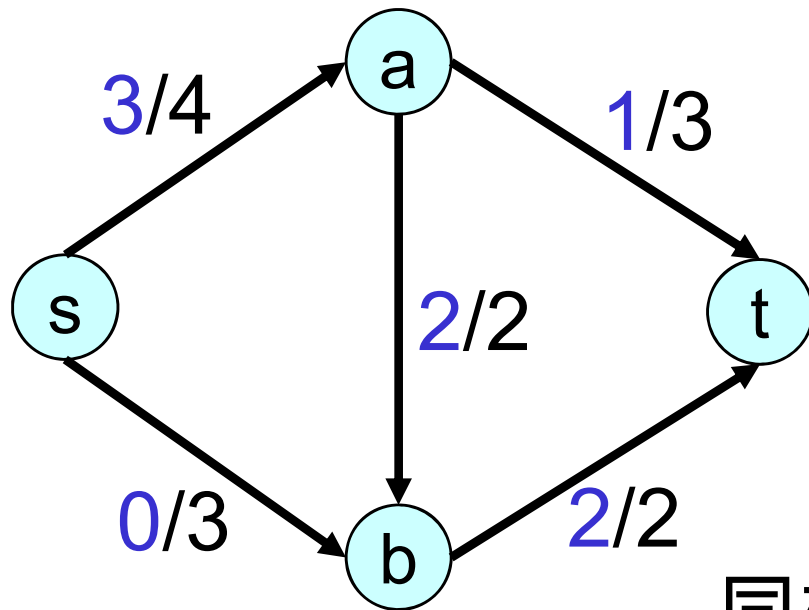
☆現在のフロー3を逆流させて
0にすることが出来る

⇒ 容量3の枝(a,s)を加える

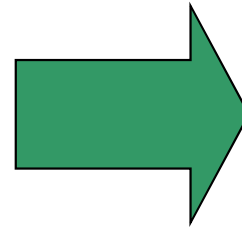
残余ネットワークの定義



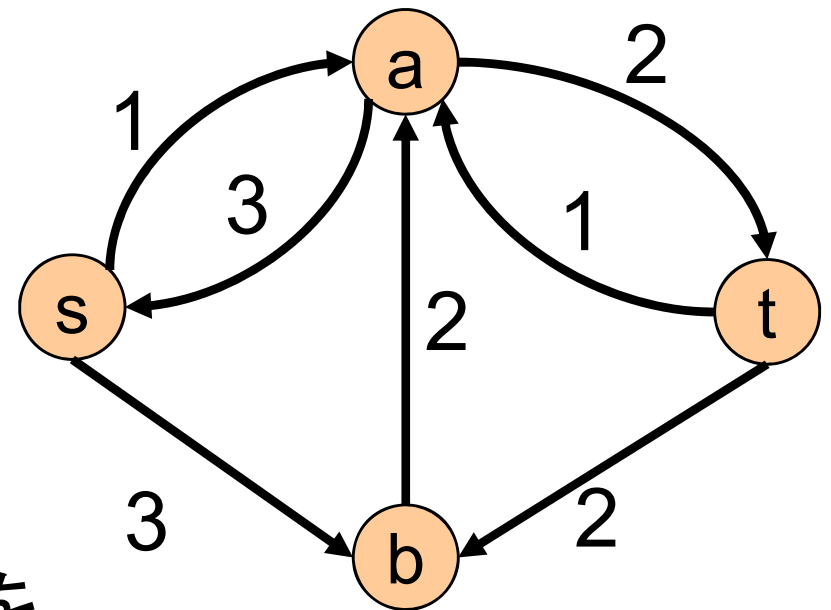
残余ネットワークの作り方



問題例とフロー

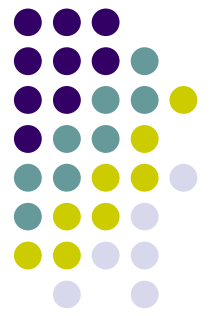


同様の操作を
各枝に行う



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)

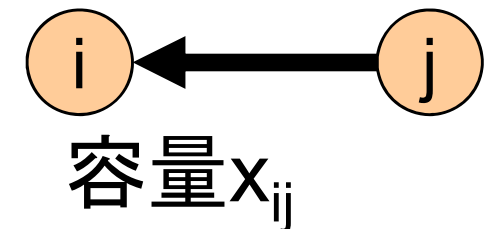
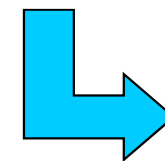
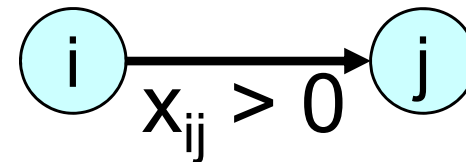
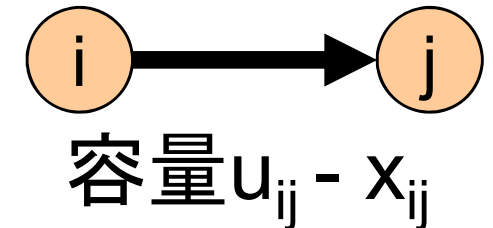
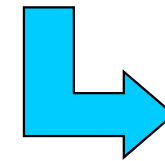
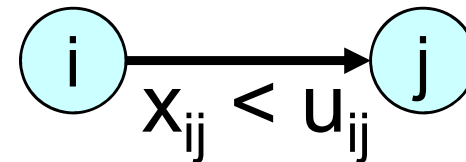


$x = (x_{ij} \mid (i,j) \in E)$: 現在のフロー

→ フロー x に関する残余ネットワーク $G^x = (V, E^x)$
 $E^x = F^x \cup R^x$

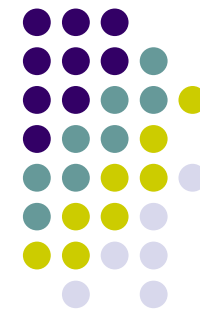
順向きの枝集合

$F^x = \{ (i, j) \mid (i, j) \in E, x_{ij} < u_{ij} \}$
各枝の容量 $u^x_{ij} = u_{ij} - x_{ij}$



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



フロー増加路: 残余ネットワークでのソースからシンクへのパス(路)

定理 1: 残余ネットワークに **フロー増加路が存在する**
→ 現在のフローは**増加可能**

定理 2: 残余ネットワークに **フロー増加路が存在しない**
→ 現在のフローは**最大流**

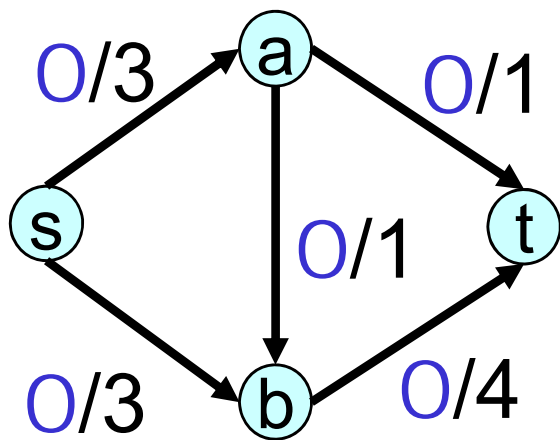
定理1の例



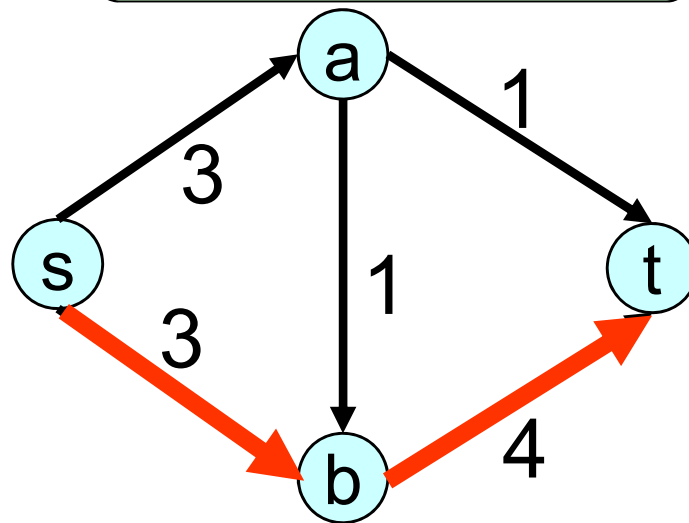
定理1 : 残余ネットワークに s-t パスが存在する
→ 現在のフローは増加可能

証明: s-t パスを使うことで, 実際にフローを増加させることができる

与えられた問題と
現在のフロー x

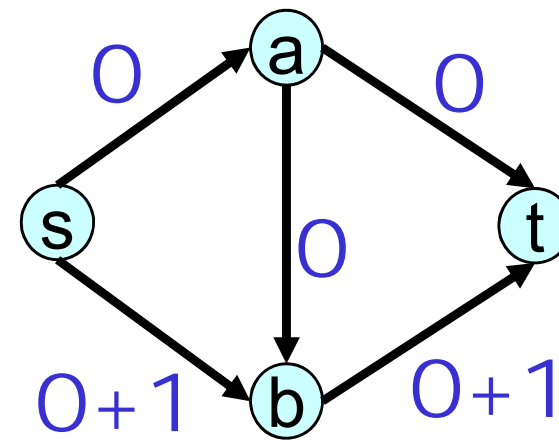


残余ネットワーク



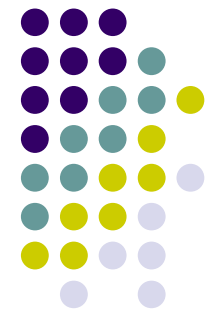
s-t パスが存在

新しいフロー x'



フロー値が
1増えた

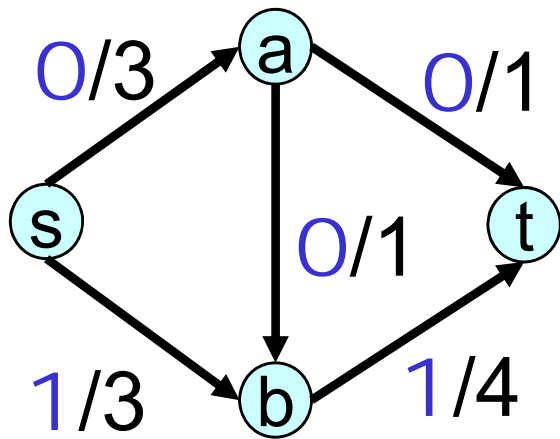
定理1の例



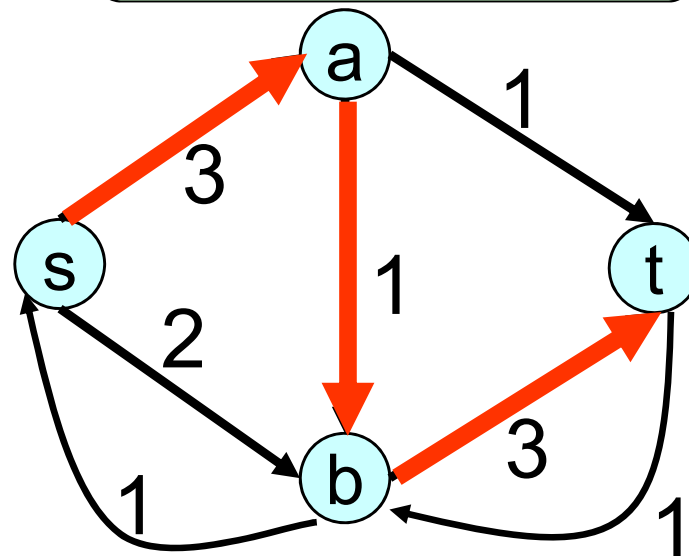
定理1 : 残余ネットワークに s-t パスが存在する
→ 現在のフローは増加可能

証明: s-t パスを使うことで, 実際にフローを増加させることができる

与えられた問題と
現在のフロー x

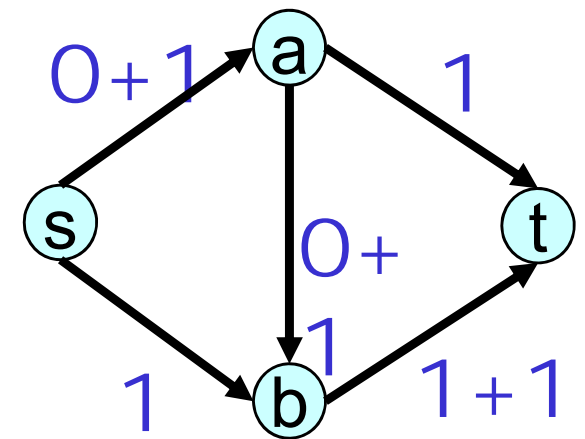


残余ネットワーク



s-t パスが存在

新しいフロー x'



フロー値が
1増えた

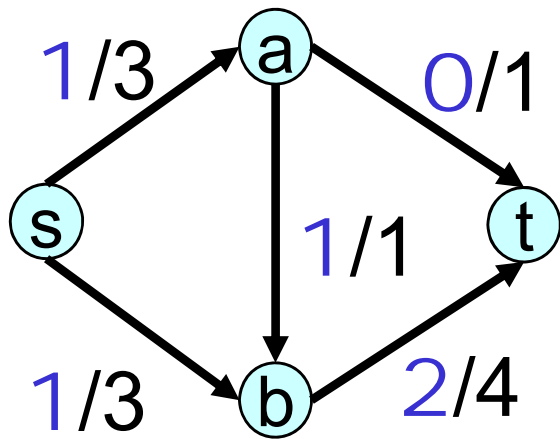
定理1の例



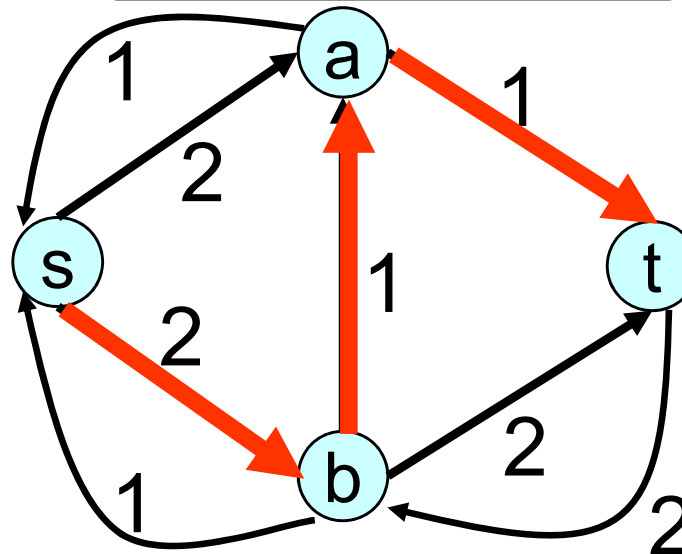
定理 1 : 残余ネットワークに s-t パスが存在する
→ 現在のフローは増加可能

証明: s-t パスを使うことで, 実際にフローを増加させることができる

与えられた問題と
現在のフロー x

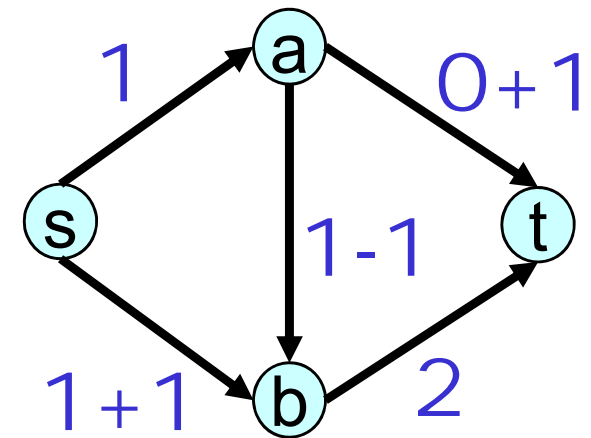


残余ネットワーク



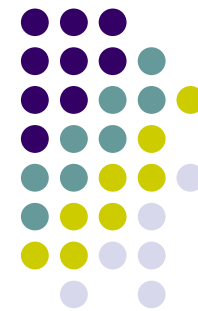
s-t パスが存在

新しいフロー x'



フロー値が
1 増えた

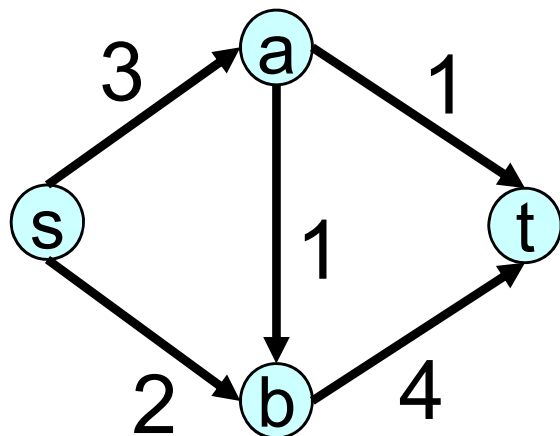
定理2の例



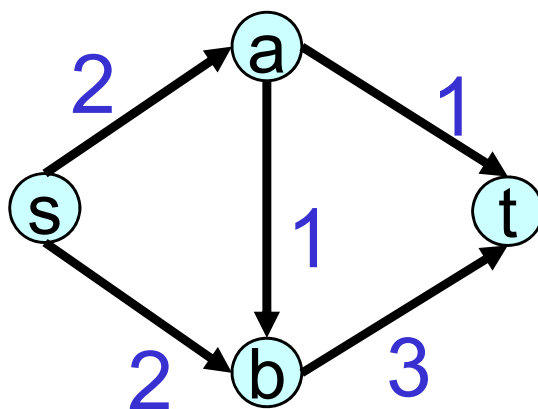
定理2: 残余ネットワークに s-t パスが存在しない
→ 現在のフローは最大流

証明は次回

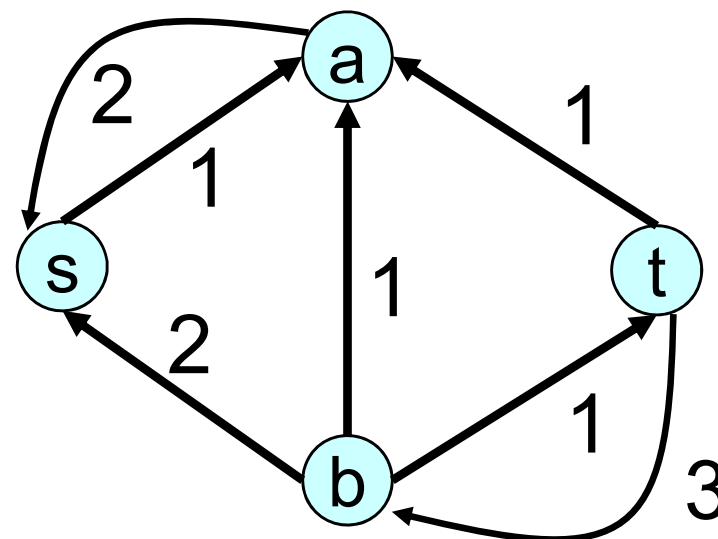
与えられた問題



現在のフロー

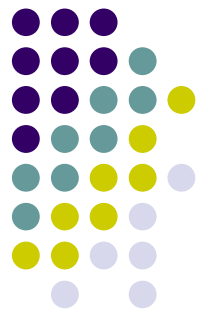


残余ネットワーク



s-t パスがない
→ 現在のフローは最適!

フロー増加法 flow augmenting algorithm



最大流を求めるためのアルゴリズム

ステップ0: 初期フローとして、全ての枝のフロー量を0とする

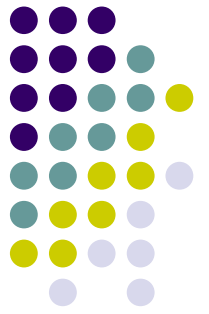
ステップ1: 現在のフローに関する残余ネットワークを作る

ステップ2: 残余ネットワークにフロー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フロー増加路をひとつ求め、それ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

フロー増加法の計算時間



※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

各反復においてフローが1以上増加

→ 反復回数 \leq 最大流量 $\leq m U$

各反復での計算時間

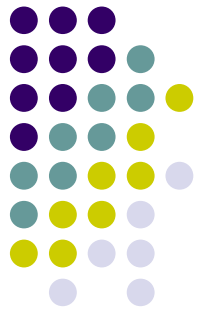
= 残余ネットワークのフロー増加路を求める時間

→ 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)

フロー増加法の改良



フロー増加法の反復回数を少なくしたい

→ 各反復でのフロー増加路の選び方を工夫する

(改良法1) 各反復でのフロー増加量を大きくする

→ 各反復で**容量最大のフロー増加路**を選ぶ

→ 反復回数 $O(m \log (n U))$, 計算時間 $O(m^2 \log (n U))$

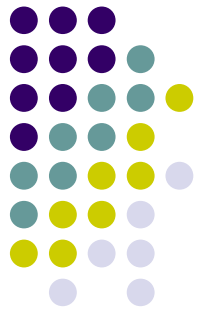
(改良法2) 各反復で**最短(枝数最小)のフロー増加路**を選ぶ

→ 反復回数 $O(m n)$, 計算時間 $O(m^2 n)$

※この他にも、フロー増加法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 → プリフロープッシュ法

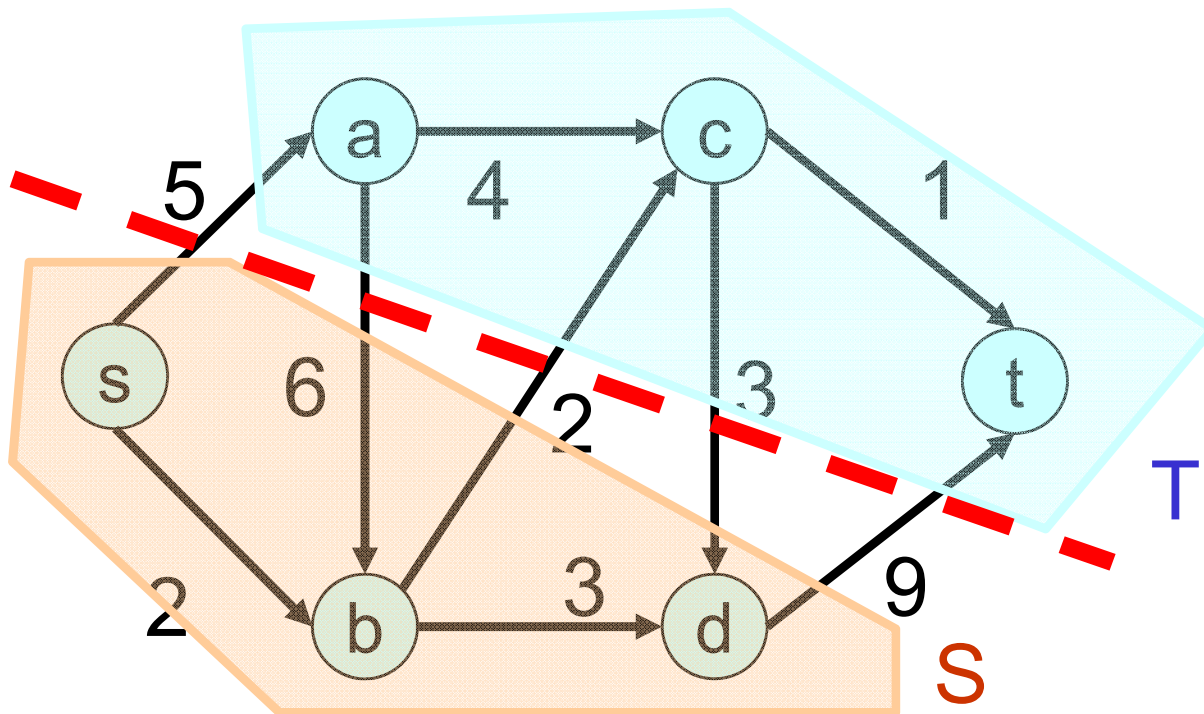
カット



フローを流すとき、ネットワークのボトルネックはどこにあるか？

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



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

レポート問題



問1: 次の2つの最大流問題に対する定式化を書きなさい

問2: 次の2つの最大流問題に対して、フロー増加法で最大流を求めよ(各反復での残余ネットワークやフローも省略せずに書くこと)

提出日: 次回講義(12/6)

