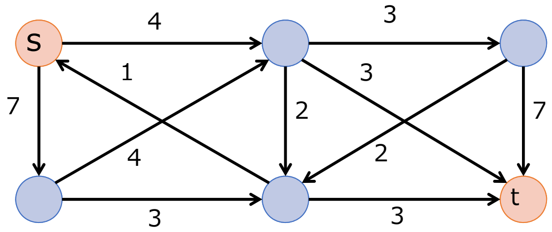
データ構造とアルゴリズム (第12回)

グラフのアルゴリズム(3)

グラフ上のs-tフロー

- 始点(s)と終点(t)を持つ連結な重みつき有向グラフ
  - 始点は蛇口，終点は排水口
  - 各辺は水道管(向き，容量付き)

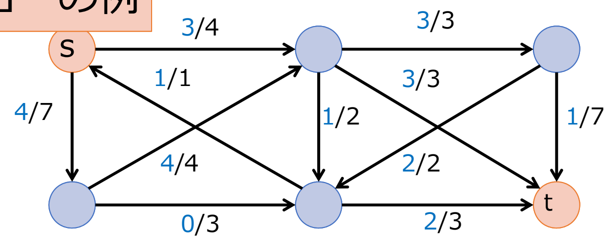


  - フロー：水道管を破裂させないようなsからtへの水の流し方
  - 流量：sからtへと流れる(単位時間当たりの)の量

グラフ上のs-tフロー

- 始点(s)と終点(t)を持つ重みつき有向グラフ
  - 始点は蛇口，終点は排水口
  - 各辺は水道管(向き，容量付き)

フローの例



各辺について 流量/容量 の形式で書くのが一般的

s-tフロー：形式的な定義

- 入力：ネットワーク(V, E, c, s, t)
  - (V, E)　連結有向グラフ
  - $c: E \to \mathbb{R}^+$　容量関数　(授業では非負の整数のみ考える)
  - $s, t\ (\in V)$　始点(source)と終点(sink)
- フロー：辺への流量割り当て関数$f: E \to \mathbb{R}$
  - ただし以下の条件を満たす必要がある

(条件1) $\forall v \in V \setminus \{s, t\} : \sum_{e \in \partial^+(v)} f(e) - \sum_{e \in \partial^-(v)} f(e) = 0$

- $\partial^+(v), \partial^-(v)$: vから出る(に入る)辺の集合

(条件2) $\forall \mathrm{e} \in E$: $0 \le f(e) \le c(e)$

s-tフロー：形式的な定義

- 入力：ネットワーク(V,E,c,s,t)
  - (V,E)　連結有向グラフ
  - $c: E \to \mathbb{R}^+$　容量関数　(授業では非負の整数のみ考える)
  - $s,t\ (\in V)$　始点(source)と終点(sink)
- フロー：辺への流量割り当
  - ただし以下の条件を満た

s,tを除き，各頂点において
流入量=流出量

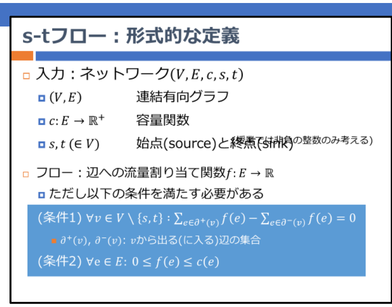
(条件1) $\forall v \in V \setminus \{s,t\} : \sum_{e\in\partial^+(v)} f(e) - \sum_{e\in\partial^-(v)} f(e) = 0$

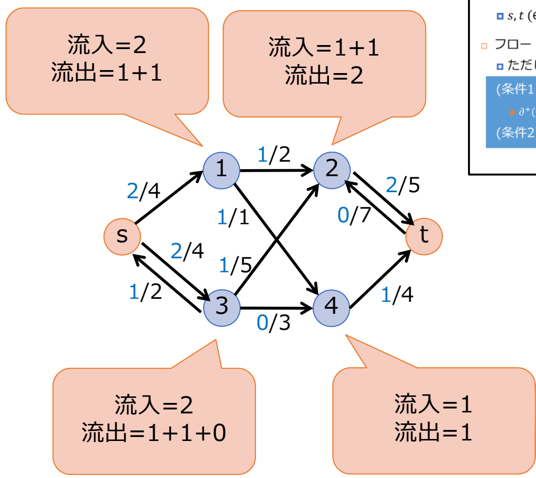
- $\partial^+(v)$, $\partial^-(v)$: vから出る(に入る)辺の集合　(フロー保存則)

(条件2) $\forall e \in E$: $0 \le f(e) \le c(e)$　(容量制約)

フロー:確認

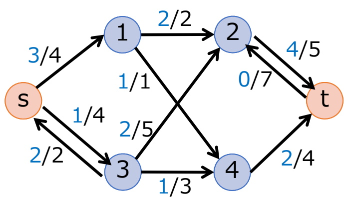
- 条件1

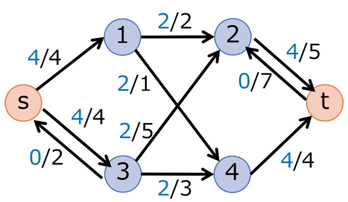




条件2については，どの辺についても流量≤容量であるので明らかに成り立っている

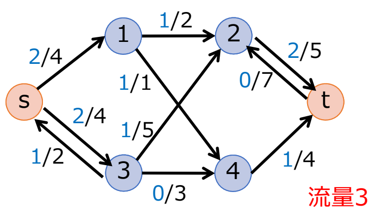
クイズ：これはフローか？



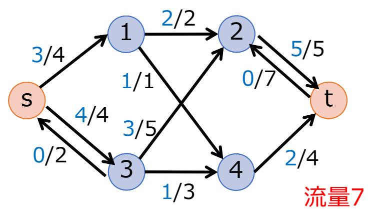


s-tフローの流量

- フローの流量：sから"正味"出ていく流量
  - より正確には「sから出ていく流量-sへ入る流量」
  - 実際にはs,t以外はすべてフロー保存則が成り立つので「tに(正味)入る流量」としても同じである
- フローfの流量を$|f|$で表す
  - すなわち，$|f| = \sum_{e\in\partial^+(s)} f(e) - \sum_{e\in\partial^-(s)} f(e)$



流量3



流量7

最大フロー問題

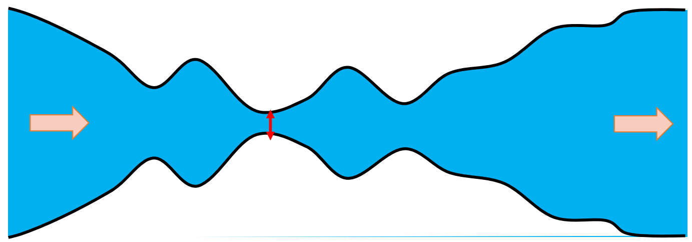
- 自然な問題として
「流量最大のフローfを求めよ」を定義できる

⇨ 最大フロー問題(maximum flow problem)

- 最小全域木問題と同じく，最適化問題の一種
  - 特に，線形計画法(Linear Programming:LP)と
    呼ばれる問題の一種

直感的な観察

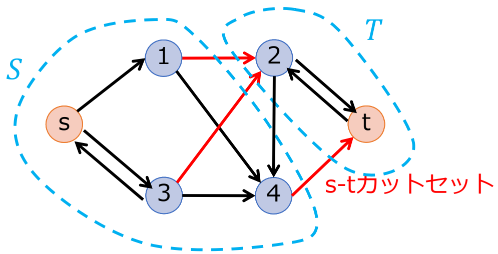
- 太さが変わる管に水を流すことを考える
  - 最もくびれているところが最大流量を決める



  - グラフにおける「くびれ」の概念は何か？

s-tカット

- ネットワーク(V,E,c,s,t)のs-tカット(S,T)
  - $s \in S$, $t \in T$を満たすVの分割$(S \cup T = V, S \cap T = \emptyset)$
  - 実際のところ，Sを決めたら$T = V \setminus S$と自動的に
    決まるので,Sだけで定義する場合もある
  - $C(S,T) = \{(i,j) \in E | i \in S, j \in T\}$をs-tカットセットと呼ぶ



s-tカット

- s-tカットSの容量:s-tカットセット中の辺の容量の和

$$\kappa(S) = \sum_{e=(v,w):v\in S, w\in T} c(e)$$

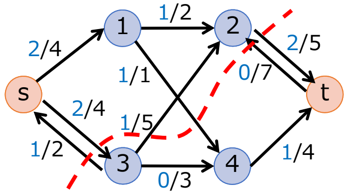
補題

任意のs-tフローの流量≤任意のs-tカットの容量

  - 証明は省略するが，直感的にはほぼ明らかであろう

s-tカットの例

- クイズ：赤線のs-tカットの容量はいくらだろうか？



補題の意味するところ

1. 流量xのフローfを発見
2. 容量xのカットSを発見

$\Rightarrow$ fは最大，Sは最小

- なぜなら，補題より，任意のフローの流量はSの容量，すなわちx以下($|f| \leq x$)
- なぜなら，補題より，任意のカットの容量は fの 流量以上，すなわちx以上($\kappa(S) \geq x$)

最大フロー最小カット定理

- このようなf, Sの対は必ず存在する
(そして，それらは各々最大フロー・最小カットである)

定理(最大フロー最小カット定理)
最大s-tフローの流量＝最小s-tカットの容量

- Ford-Fulkersonの定理とも呼ばれる

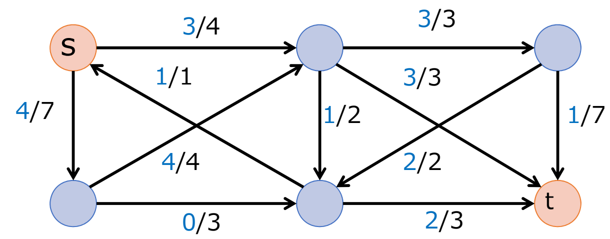
定理の証明

証明

- 証明は構成的に示す．具体的には以下の2つを示す
  1. あるフローを構成するアルゴリズムA
  2. アルゴリズムAの出力fに対し，流量$|f|$となるカットSが必ず構成できる
- 補題から，このアルゴリズムは最大フローを求めるアルゴリズムであることがわかる
- と同時に，最大フロー最小カット定理の証明にもなっている

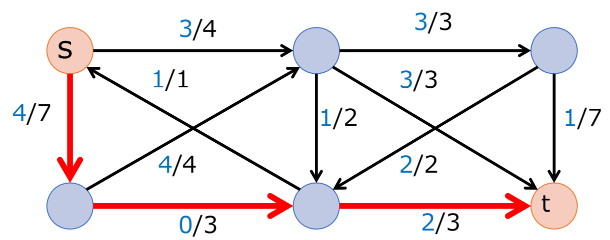
証明：アルゴリズムAの構成

- 与えられたフローに対して，流量をさらに増やせるかどうかを判定する方法を検討してみる



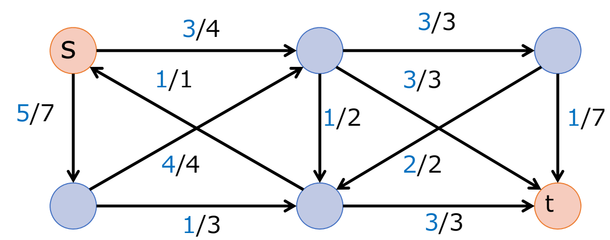
証明：アルゴリズムAの構成

- 増加道
  - それに沿って0より大きいフローを流すことで 流量を上げられるs-tパス



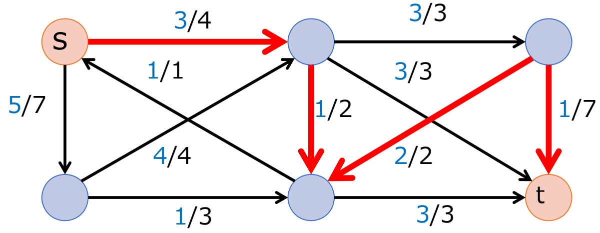
証明：アルゴリズムAの構成

- 増加道がない場合は最大フローか？



証明：アルゴリズムAの構成

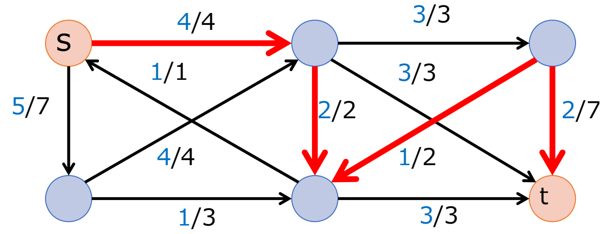
- 増加道がない場合は最大フローか？



証明：アルゴリズムAの構成

- 増加道がない場合は最大フローか？

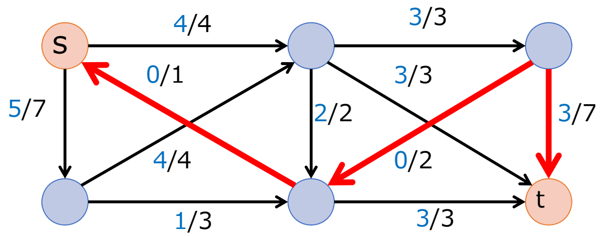
  →逆流によりフロー量を増やせる



証明：アルゴリズムAの構成

- 増加道がない場合は最大フローか？

  →逆流によりフロー量を増やせる

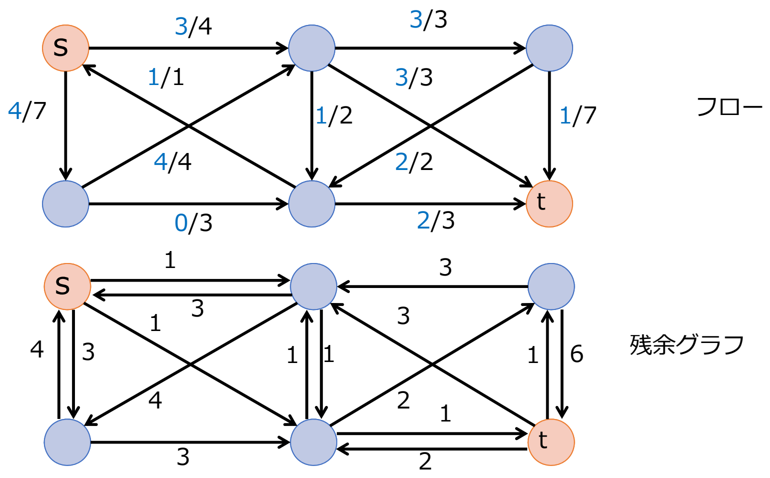


残余グラフと増加道

- (V, E, c, s, t)上のフローfに対する
  残余グラフ(residual graph)(V, E_f, c_f, s, t)
  - $E_f = E_f^+ \cup E_f^-$
  - $E_f^+ = \{e | e \in E, f(e) < c(e)\}$
  - $E_f^- = \{\bar{e} | e \in E, f(e) > 0\}$　$\bar{e}$はeの逆向き辺
  - $c_f(e) = \begin{cases} c(e) - f(e) & \text{if } e \in E_f^+ \\ f(e) & \text{if } e \in E_f^- \end{cases}$
- fの残余ネットワークにおけるs-tパスを
  増加道と呼ぶ(これが真の定義)

残余グラフの例

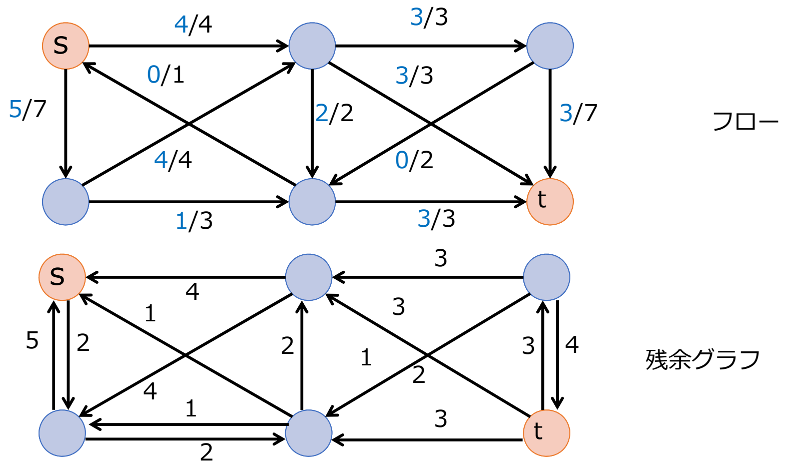
- 最初のフローに対する残余グラフ



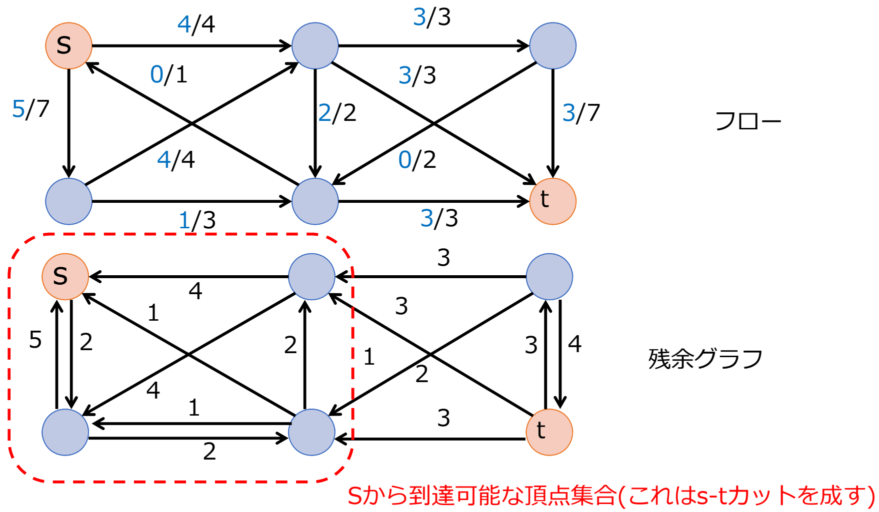
アルゴリズムA

- Ford-Fulkersonのアルゴリズム:
  1. 残余グラフを構成する
  2. 残余グラフ上におけるs-tパス(増加道)を一つ見つける
     - 発見できなければアルゴリズム終了
     - 発見したs-tパスの上での辺重みの最小値cに対し，パスに沿ってフローをcだけ流す（すなわち，パス中の辺に対応するネットワークの辺の流量をcだけ増加させる）
  3. 残余グラフを更新して，2に戻る

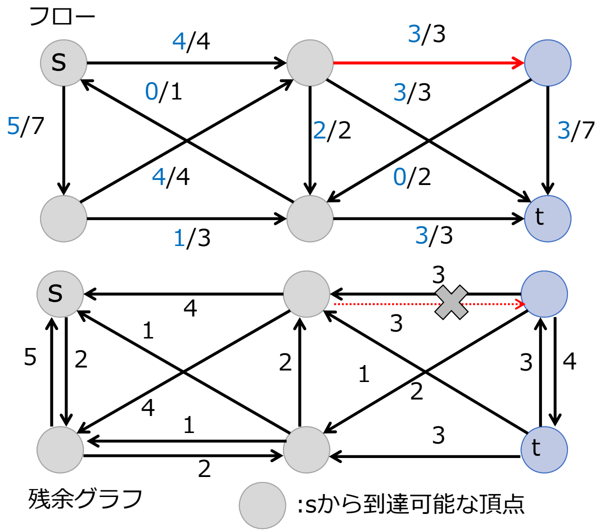
アルゴリズム終了時の例



最大フローであることの証明



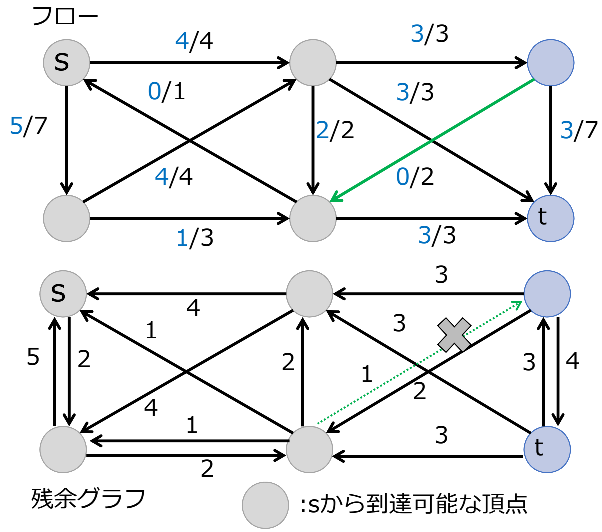
最大フローであることの証明



ネットワークの構成辺
$(i,j) \in E\,(i \in S)$に対して
対応する辺が残余グラフに
存在しない
→(i,j)は流せるだけ
流れている
（飽和しているという）

ネットワークの構成辺
$(j,i) \in E\,(i \in S)$に対して
対応する逆辺が残余グラフに
存在しない
→(j,i)は一切流れていない

最大フローであることの証明



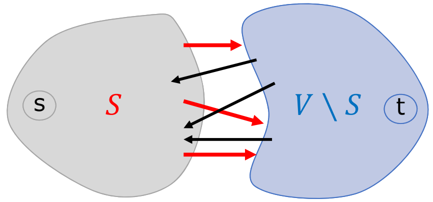
ネットワークの構成辺
$(i,j) \in E\,(i \in S)$に対して
対応する辺が残余グラフに
存在しない
→(i,j)は流せるだけ
流れている
（飽和しているという）

ネットワークの構成辺
$(j,i) \in E\,(i \in S)$に対して
対応する逆辺が残余グラフに
存在しない
→(j,i)は一切流れていない

最大フローであることの証明

- 前述の議論から，以下のことが分かる

残余グラフにおいてsから到達可能な頂点集合がなす
カットをSとすると，その容量はフロー量に等しい



- S側から$V \setminus S$側へとカットセット中の辺を通って
送られた流れは一切S側へと「返却」されていない
→全量がtで吸収されている．すなわち$|f| = \kappa(S)$

Ford-Fulkersonのアルゴリズムの実現とマッチングへの応用

Ford-Fulkersonのアルゴリズム

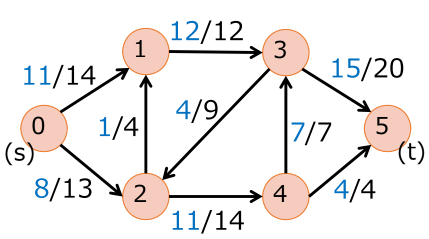
1. 残余グラフを構成する
2. 残余グラフ上におけるs-tパス(増加道)を一つ見つける(BFS or DFS)
   - 発見できなければアルゴリズム終了
   - 発見したs-tパスの上での辺重みの最小値cに対し，パスに沿ってフローをcだけ流す（すなわち，パス中の辺に対応するネットワークの辺の流量をcだけ増加させる）
3. 残余グラフを更新して，2に戻る

実装上の工夫

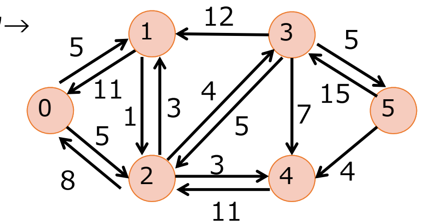
- 通常アルゴリズムは「フロー量」は管理しないで残余グラフのみ管理する
  - 最終的なフローは残余グラフから計算できる
- 残余グラフの更新は，ゼロから作り直すのではなく更新のあったところだけ変更する
  - 見つけたs-tパスに沿って更新する
- パスの発見から残余グラフの更新まで(詳細)
  1. sを始点とした(有向)全域木Tを計算する
  2. T上のs-tパスに沿って，容量の最小値を求める
  3. T上のs-tパスに沿って，残余グラフを更新する
     - 逆辺を(隣接リスト中で)効率良く見つける必要あり

残余グラフの隣接リストに対する工夫(1)

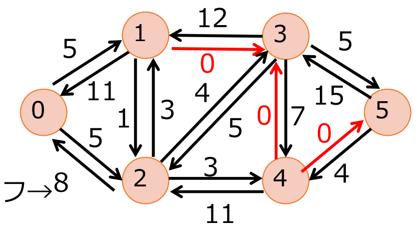
- 辺の追加，削除の代わりに「容量0の辺」を用いる
  - BFS or DFS実行時「容量0」の辺はないものとして扱う



通常の定義に従った場合の残余グラフ→

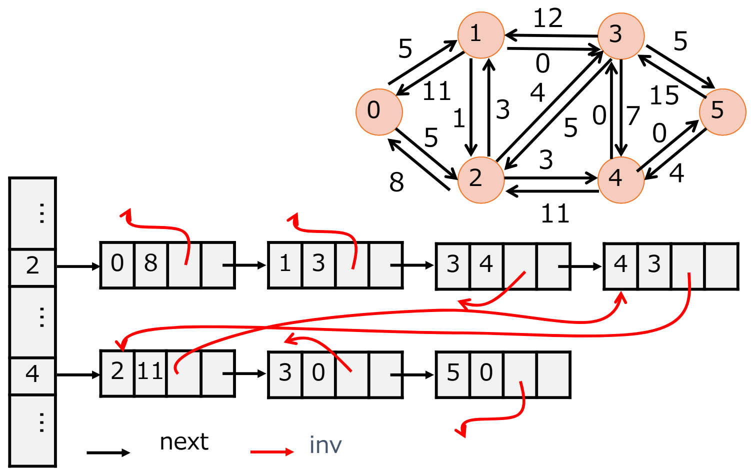


重みゼロの辺を使った場合の残余グラフ→



残余グラフの隣接リストに対する工夫(2)

- 各辺に「逆辺へのポインタ」の情報を付加する



残余グラフ更新アルゴリズム

- 工夫(1),(2)を取り入れると，辺数kの増加道に対する残余グラフの更新が$O(k)$時間で終わる

```
T =  bfs(s) or dfs(s)  // 有向全域木を構成
(T[0, n-1]：幅優先木を記録する配列)
v = t;
cmin = ∞
while(v ≠ s ) {
    cmin = min (cmin, c(T[v], v))        容量最小値の発見
    v = T[v];
}
v = t;
while (v≠s ) {
    辺(T[v], v)の重みをcmin減らす
    辺(T[v], v) の逆辺の重みをcmin増やす    残余グラフの更新
    v = T[v];
}
```

実行時間

- 残余グラフの構成:$O(m)$
- パスの発見：$O(m)$時間
- 残余グラフの更新：$O(m)$時間
- パス発見の反復回数：O(最大流量)
  - 整数フローの場合 (1回で少なくとも1増加するため)

➡ $O(m|f|)$時間

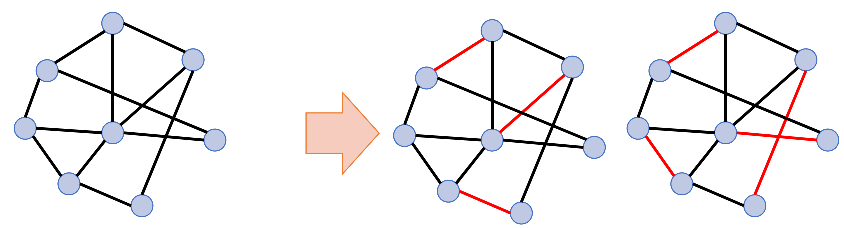
- 注意：$|f|$はmやnの多項式では収まらない可能性がある (弱多項式時間アルゴリズム)

実行時間

- BFSを使った場合の反復回数:$O(mn)$
  - フロー量によらない　(証明は略)
- トータル$O(m^2n)$時間
  - 実用的にはもっと速いことも多い
  - 強多項式時間アルゴリズム
    ↑ 実行時間が問題中の最大値によらない
  - このアルゴリズムはEdmonds-Karpのアルゴリズムと呼ばれる
  - もう少し工夫して$O(n^2m)$にすることもできる

（重みなし）最大マッチング

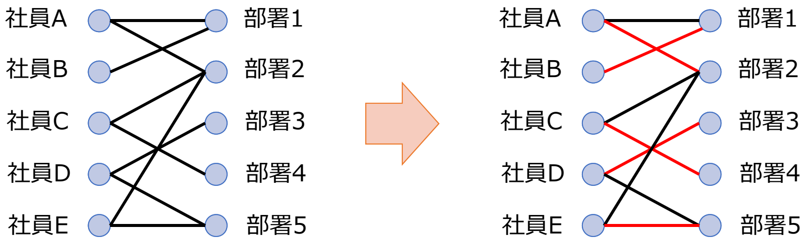
- マッチング(matching)：端点を共有しない辺集合
- 重みなし最大マッチング問題
  - 辺数最大のマッチングを求める



(最大マッチング)

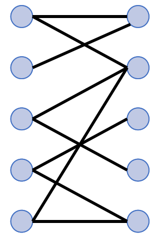
2部グラフの最大マッチング

- 割当て問題(Assignment problem)
  - e.g. 希望割り当て問題
    - n_1人の新入社員/n_1個の配属部署
    - 各部署には1人づつ社員を配属
    - 各社員は就職部署を指定



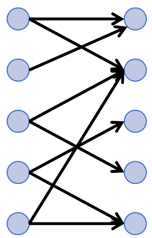
最大フローへの帰着

- 2部グラフの最大マッチングは最大s-tフローを用いて解ける



最大フローへの帰着

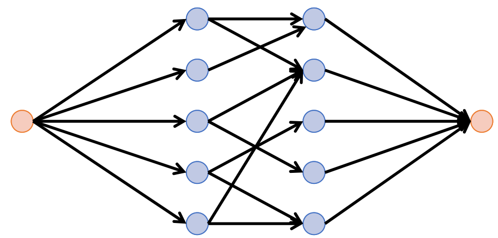
- 2部グラフの最大マッチングは最大s-tフローを用いて解ける



容量１の有向辺にする
（容量はすべて等しいので省略）

最大フローへの帰着

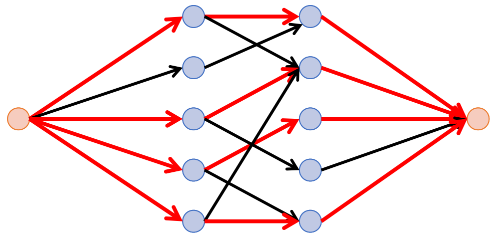
- 2部グラフの最大マッチングは最大s-tフローを用いて解ける



s,tに相当するスーパーノードを追加して
各頂点に辺を引く（容量は１）

最大フローへの帰着

- 2部グラフの最大マッチングは最大s-tフローを用いて解ける

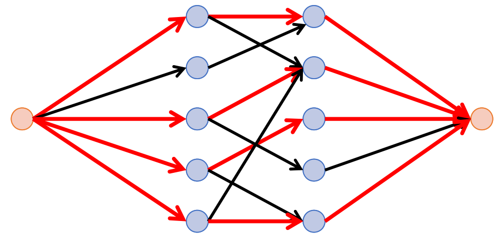


s,tに相当するスーパーノードを追加して
各頂点に辺を引く（容量は１）

最大フローへの帰着

- 2部グラフの最大マッチングは最大s-tフローを用いて解ける

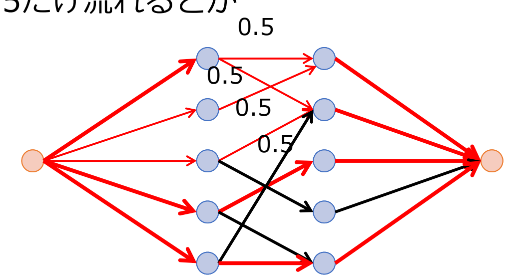
このグラフの最大整数フロー
において飽和する辺は
最大マッチング



s,tに相当するスーパーノードを追加して
各頂点に辺を引く（容量は１）

最大フローへの帰着

- 疑問：こんなことは起こらないか？
  - 0.5だけ流れるとか



  - 答え：起こらない(起こさないようにできる)

フロー整数性定理

定理(フロー整数性定理)

全ての辺容量が整数のネットワークはすべての辺の
流量が整数の最大フローを持つ

- 証明はFord-Fulkersonのアルゴリズムからほぼ明らか
  - 辺容量が整数で，流量がすべて整数のフローから
    得られる残余グラフの辺重みはすべて整数
    →増加道により増やされる流量もやはり整数
    →アルゴリズムの実行中フローの整数性を保ち続ける

フロー整数性定理のご利益

- 「必ず整数の解が手に入る」のは，組み合わせ的な問題を解くときの強力な武器になる
  - 組み合わせ的な問題：何らかの(部分)集合を答えとして取るような問題
    - マッチング：（端点を共有しない)辺の部分集合
  - このような問題では

  「流量が1の辺」=「対応する要素を取る」
  「飽和している辺」=「対応する要素を取る」

  のような関係性に基づく帰着を使って問題を解くことがあり，フロー整数性定理はそのため重要なツールになる