ネットワークフロー入門

保坂 和宏 (東京大学理学部数学科)

第 14 回 JOI 春合宿
2015/03/19-20

前置き

- ネットワークフローに関する理論的な興味
  - グラフの難しそうな問題が多項式時間で解ける！
  - 応用範囲が広い
  - 賢い高速化もいろいろ
    - 副産物的な発見もある

前置き

- プログラミングコンテストにおけるネットワークフロー
  - 「問題からうまいことグラフを構成してフローを流して解く」というパターンが多い
    - うまいグラフを作るのにひらめき or 経験が重要，ということでコンテストにお似合い
    - フローの部分は皆事前にコードを書いてある (または頭に入っている) という風潮あり
  - 二部グラフへの応用 (最大マッチングなど) はそれだけで強力な道具
    - 複雑な貪欲法を回避できることがしばしばあります
  - ICPC, topcoder, Codeforces 等では容赦なく出題される

前置き

- 情報オリンピック (JOI/IOI) におけるネットワークフロー
  - IOI シラバス (近年の IOI の出題指針)

明確に除外される (Explicitly excluded)

難しいアルゴリズムの項目のうちのいくつかは明確に除外される．競技参加者には，これらの分野の**知識が要求される**課題は出題されないことが保証される．すなわち，全ての競技課題には，これらの項目の知識が無くても作成可能な満点解法が存在する．ここには，主に，難しい教科書のアルゴリズムが含まれる．

Some of the harder algorithmic topics are explicitly marked as excluded. It is guaranteed that there will not be a competition task that *requires* the contestants to know these areas. In other words, each competition task will have a perfect solution that can be produced without the knowledge of these topics. This category mainly contains hard textbook algorithms.

ただし，このシラバスが，競技参加者が課題に取り組む際に適用可能な技術を，いかなる意味においても制限していると解釈すべきではない．

Still, note that the Syllabus must not be interpreted to restrict in any way the techniques that contestants are allowed to apply in solving the competition tasks.

ここには，IOI の範囲から明らかに逸脱した項目も含まれる．

Additionally, this category is used for topics that clearly fall outside the scope of the IOI.

例 (Examples): **最大流アルゴリズム** *(Maximum flow algorithms)* in §5.2 AL3 / **微積分** *(Calculus)* in §4.3

http://people.ksp.sk/~misof/ioi-syllabus/
http://www.ioi-jp.org/ioi/ioi-syllabus_jp.pdf

前置き

- 情報オリンピック (JOI/IOI) におけるネットワークフロー？
  - IOI 2014 Day 2 “Friend”
    - ひとつの小課題の想定解法が二部グラフの最大マッチング
      - 満点解法が思いつかない場合，フローが書ければ 23 点得
  - IOI 2006 Day 1 “Forbidden Subgraph”
    - output-only task
    - 一般のグラフの最大マッチングで解けるテストケースあり
      - フローではないが多少似ていてより高度な話題

前置き

- 情報オリンピック (JOI/IOI) におけるネットワークフロー？

http://www.ioi-jp.org/joi/2014/kyougi.html

前置き (私見)

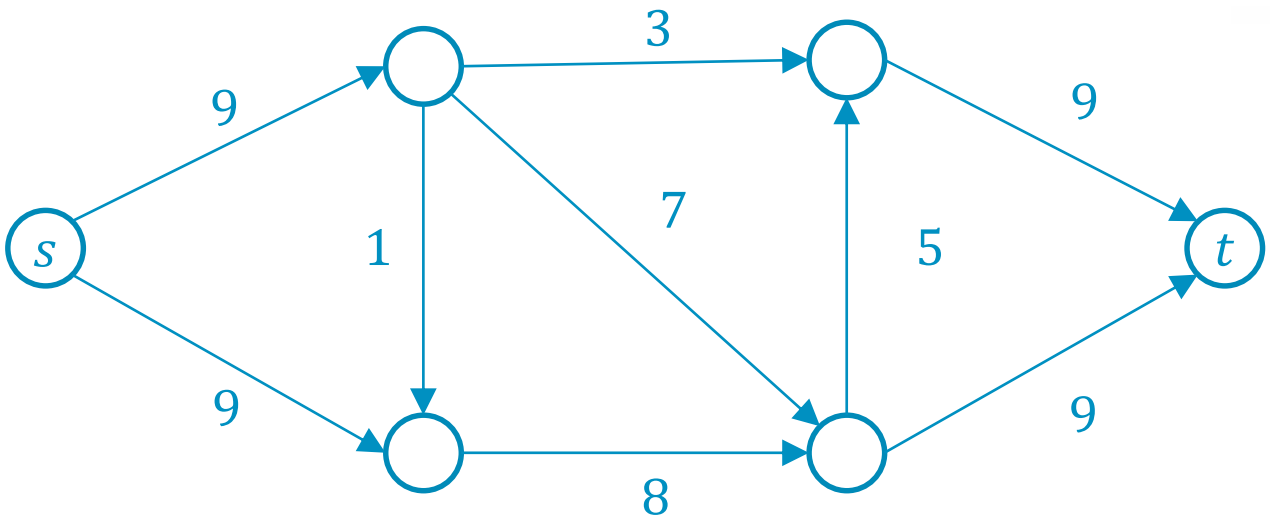
- フローはグラフアルゴリズムにある程度慣れている人向け
  - DFS，BFS，最短路，最小全域木，強連結成分分解，etc.
    - でもその次くらいに知っておくべきかも？
- JOI/IOI において
  - 「情報オリンピックだからフローじゃない」と決めつけるのはいささか危険かもしれません
    - 特に部分点
  - とはいえ「そろそろフロー出題されそうだしフローでしょ」と決めつけるのも危険です
- フローの基本は理解しておいて損はないでしょう

目次

I. 最大流

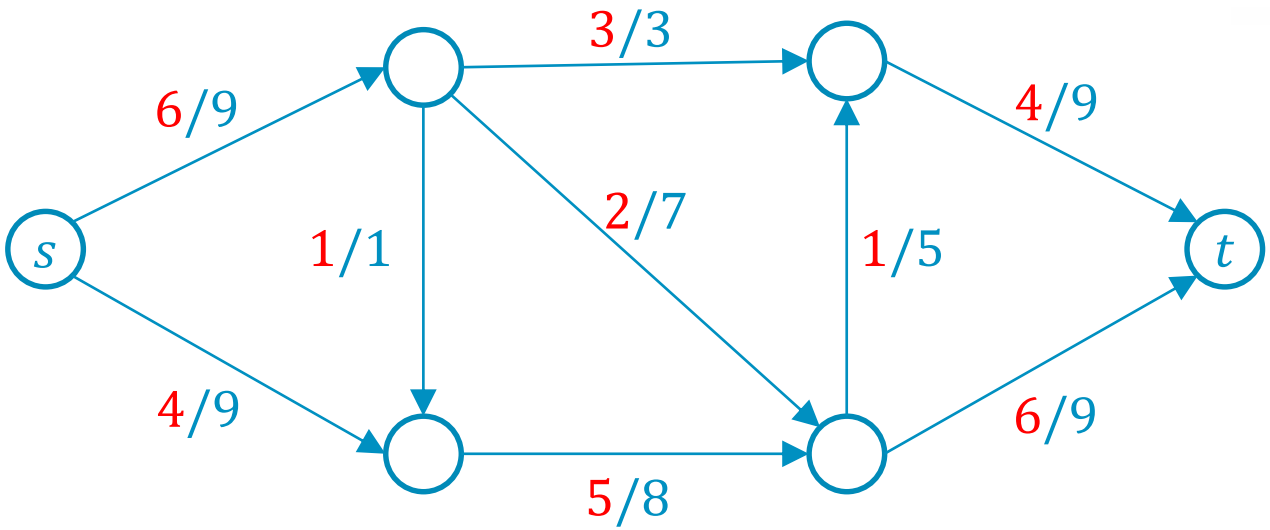
最大流問題

- 各辺には書いてあるぶんまでの水を流せます
- 頂点 s から頂点 t へ水はどのくらい流れるでしょう？



最大流問題

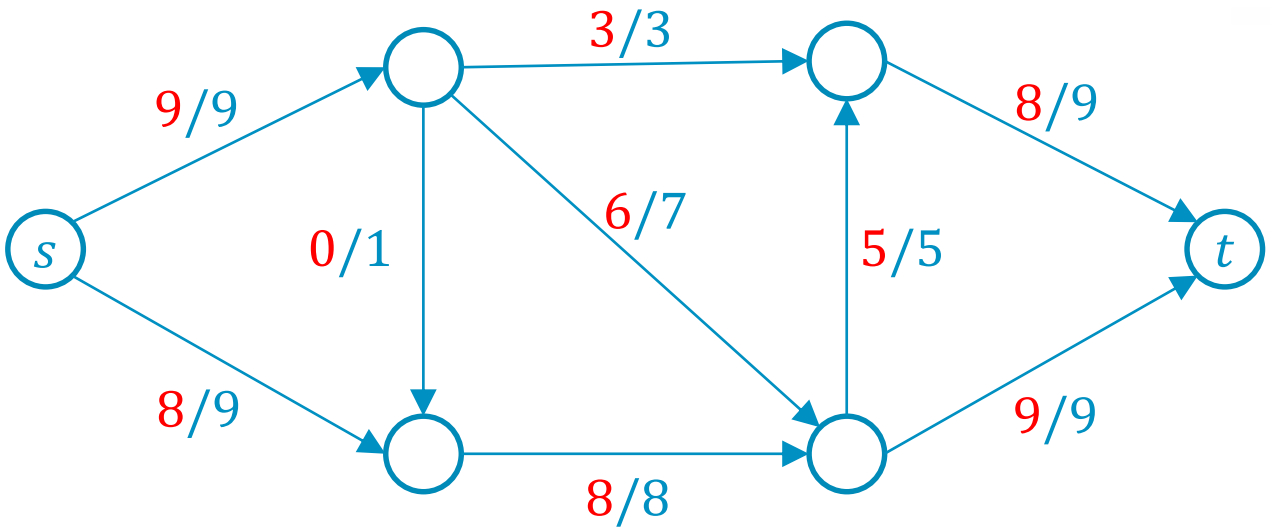
- 各辺には書いてあるぶんまでの水を流せます
- 頂点 s から頂点 t へ水はどのくらい流れるでしょう？



↑ 水を 10 流した様子

最大流問題

- 各辺には書いてあるぶんまでの水を流せます
- 頂点 s から頂点 t へ水はどのくらい流れるでしょう？



↑ うまくやると 17 流せる

最大流問題 (maximum flow problem)

- 与えられるもの：**ネットワーク** (network)
  - 有向グラフ $G = (V, E)$
  - 各辺 $e \in E$ に対して，**容量** (capacity) $u(e) \geq 0$
  - **始点** (source) $s \in V$ と**終点** (sink) $t \in V$ $(s \neq t)$
- 作るもの：**s-t フロー** (s-t flow)
  - 各辺 $e \in E$ に対して $f(e)$ が定まっていて，以下を満たすもの
    - 各辺 $e \in E$ に対して，$0 \leq f(e) \leq u(e)$
    - 始点と終点以外の各頂点 $v \in V \setminus \{s, t\}$ に対して，

$$\sum_{e=*v} f(e) = \sum_{e=v*} f(e)$$

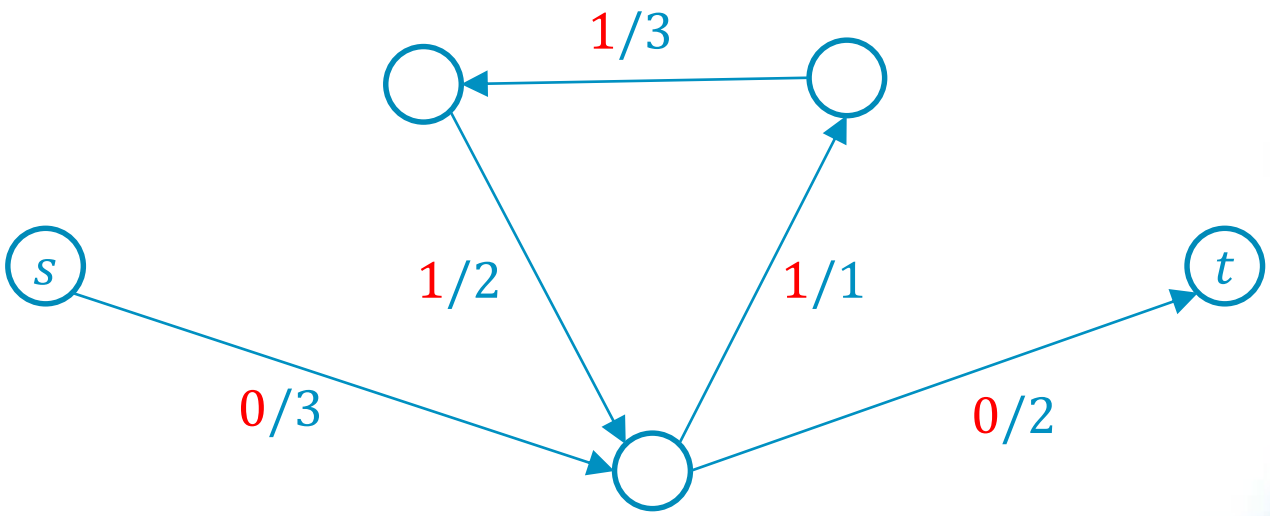
v へ入ってくる辺たち　　v から出ていく辺たち

- 最大化したいもの：フローの**流量** (value)

$$|f| = \sum_{e=s*} f(e) - \sum_{e=*s} f(e)$$

最大流問題

- s-t フローは s-t パスたちとサイクルたちを足したものとして書ける
  - 証明は辺の数に関する帰納法
    - パスとサイクル合わせて $|E|$ 個にできる

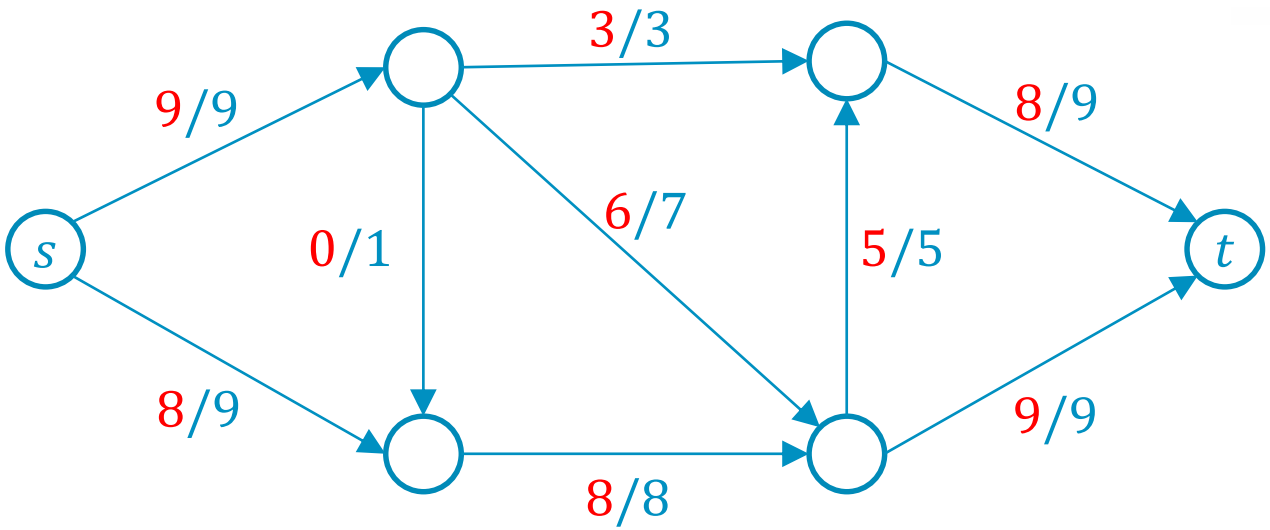


↑ こういうのもれっきとしたフローなので注意

(流量は 0)

最大流問題

- 各辺には書いてあるぶんまでの水を流せます
- 頂点 s から頂点 t へ水はどのくらい流れるでしょう？

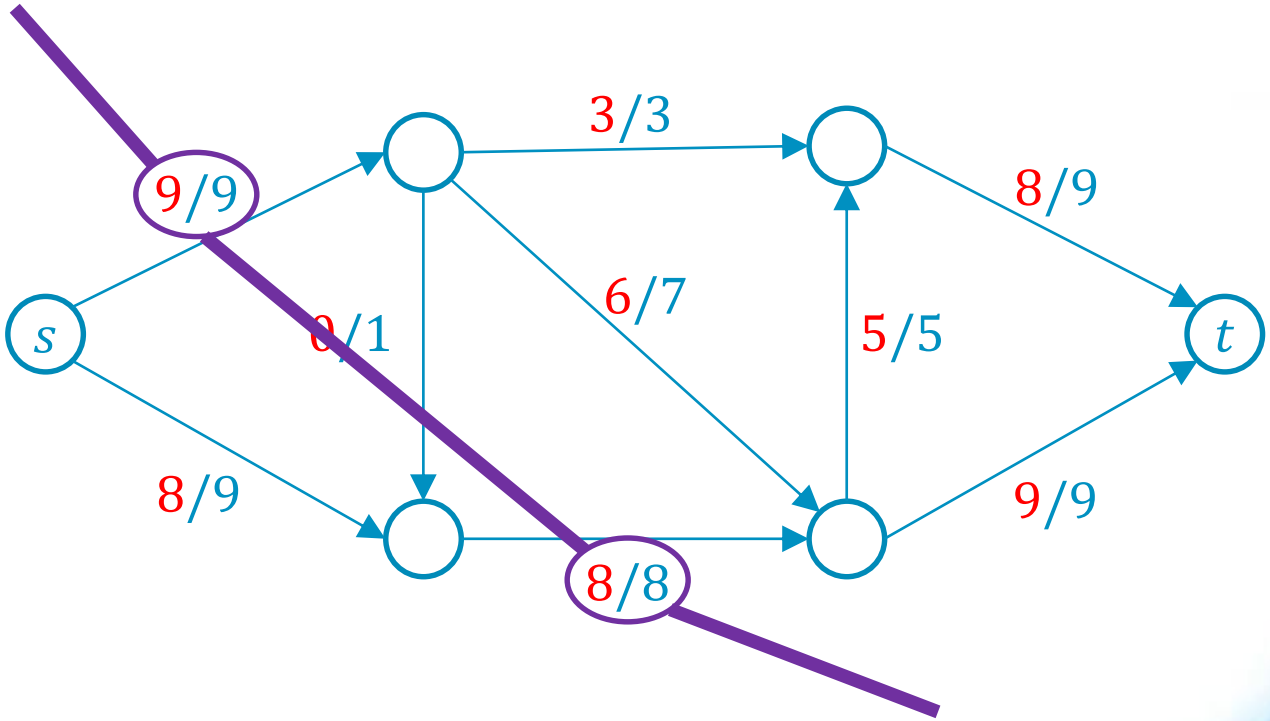


↑ うまくやると 17 流せる

↑ もっと多くは無理？

最大流問題

- 各辺には書いてあるぶんまでの水を流せます
- 頂点 s から頂点 t へ水はどのくらい流れるでしょう？



↑ うまくやると 17 流せる

↑ もっと多くは無理！

最小カット問題

- グラフを s 側と t 側に分断して，s 側から t 側へ向かう辺をの容量を見ると，フローの流量が上から抑えられる
  - さっきの例だと $|f| \leq 17$
    - $|f| = 17$ のフローが実際に構成できていたので，流量の最大値は 17 とわかる

最小カット問題 (minimum cut problem)

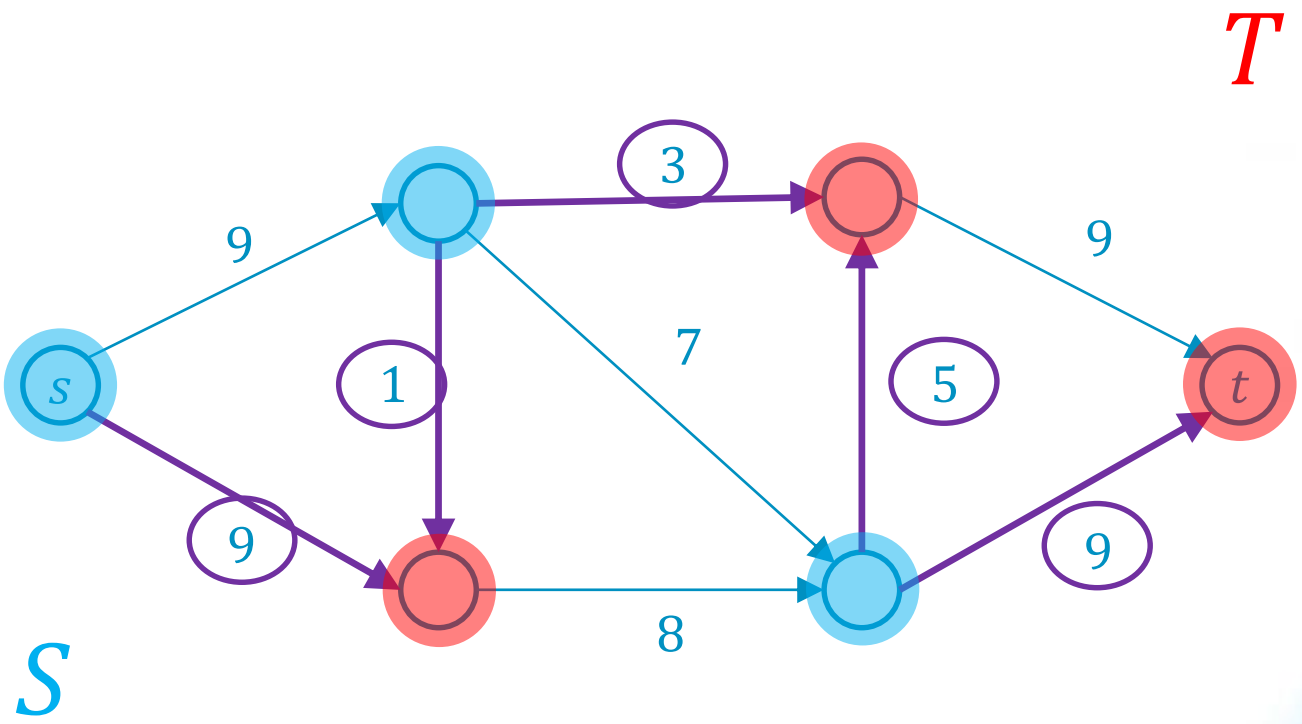
- 与えられるもの
  - 有向グラフ $G = (V, E)$
  - 各辺 $e \in E$ に対して，**容量** (capacity) $u(e) \geq 0$
  - **始点** (source) $s \in V$ と**終点** (sink) $t \in V$ $(s \neq t)$
- 作るもの：**s-t カット** (s-t cut)
  - 頂点集合の分割 $V = S \sqcup T$ であって，$s \in S, t \in T$ なるもの
- 最小化したいもの：カットの**容量** (capacity)

$$u(S, T) = \sum_{e=vw,\, v \in S,\, w \in T} u(e)$$

S から出て T へ入る辺たち

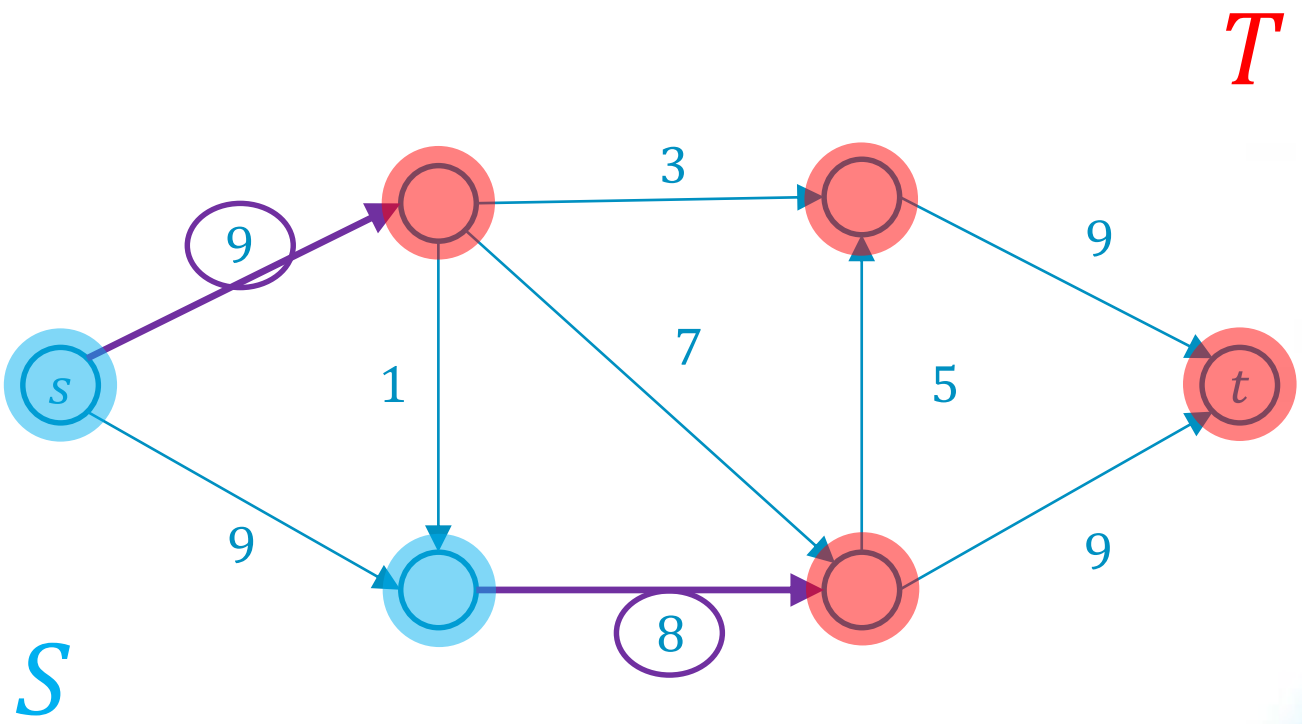
  - 最小 s-t カットの容量は s, t の局所辺連結度 (local edge-connectivity) とも呼ばれる

最小カット問題



↑ 容量 27 のカット

最小カット問題



↑ 容量 17 のカット

最大 s-t フロー ≤ 最小 s-t カット (弱双対性)

- 任意の s-t フロー f と任意の s-t カット (S,T) に対して
$$|f| \leq u(S,T)$$

  - 証明

$$|f| = \sum_{e=s*} f(e) - \sum_{e=*s} f(e) = \sum_{v \in S} \left(\sum_{e=v*} f(e) - \sum_{e=*v} f(e) \right)$$

$$= \sum_{e=vw,\, v \in S,\, w \in T} f(e) - \sum_{e=vw,\, v \in T,\, w \in S} f(e)$$

$$\leq \sum_{e=vw,\, v \in S,\, w \in T} u(e) - \sum_{e=vw,\, v \in T,\, w \in S} 0 = u(S,T)$$

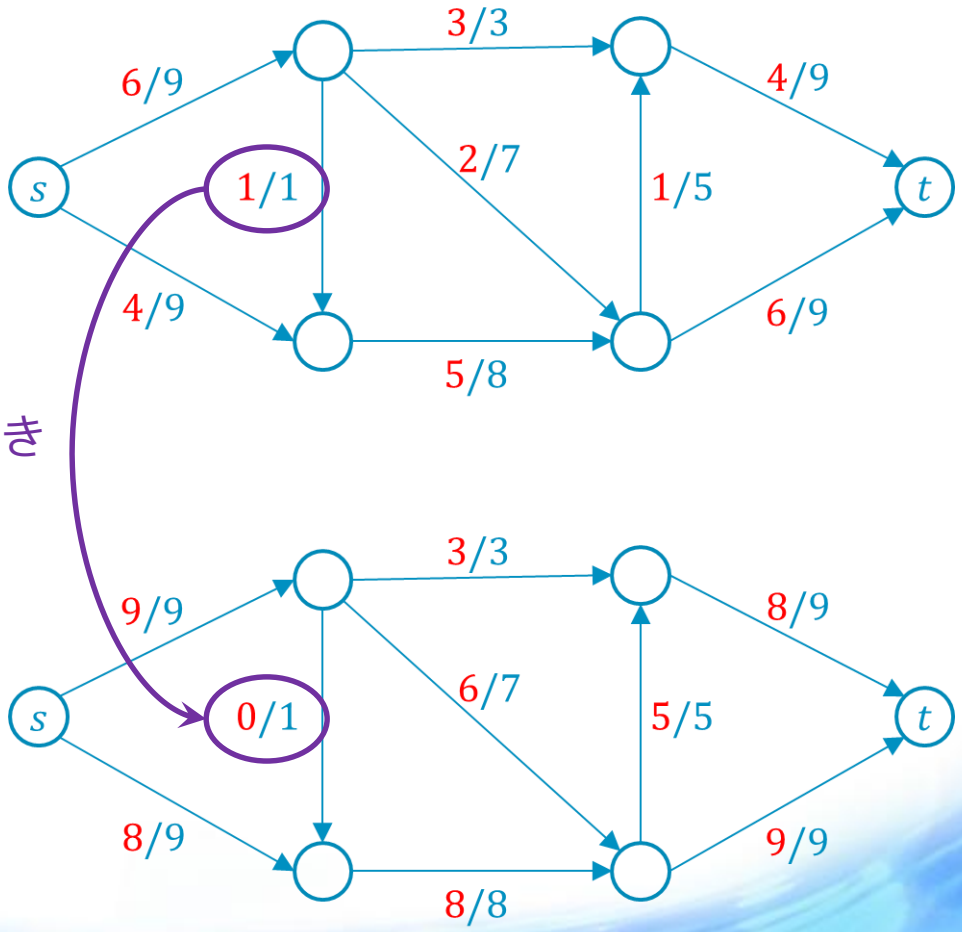
  - 特に，$\max |f| \leq \min\, u(S,T)$

最大 s-t フロー = 最小 s-t カット (強双対性)

- ある s-t フロー f とある s-t カット (S, T) をうまくとると
$$|f| = u(S, T)$$
  - 前の不等号と合わせて，$\max |f| = \min\ u(S, T)$ がわかる
    - 最大フロー最小カット定理 (max-flow min-cut theorem) と呼ばれる
    - 証明も含めて重要

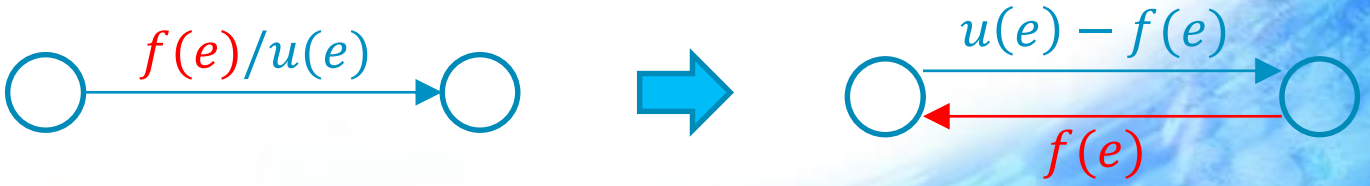
残余グラフ

- フローを改善したいという気持ち
  - 辺を流れる量を増やす
    - ……だけだと最大フローは得られないかもしれない

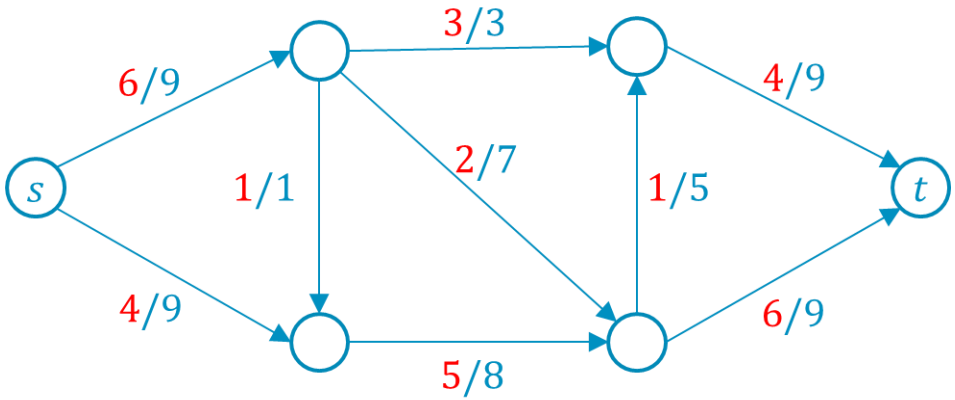


残余グラフ

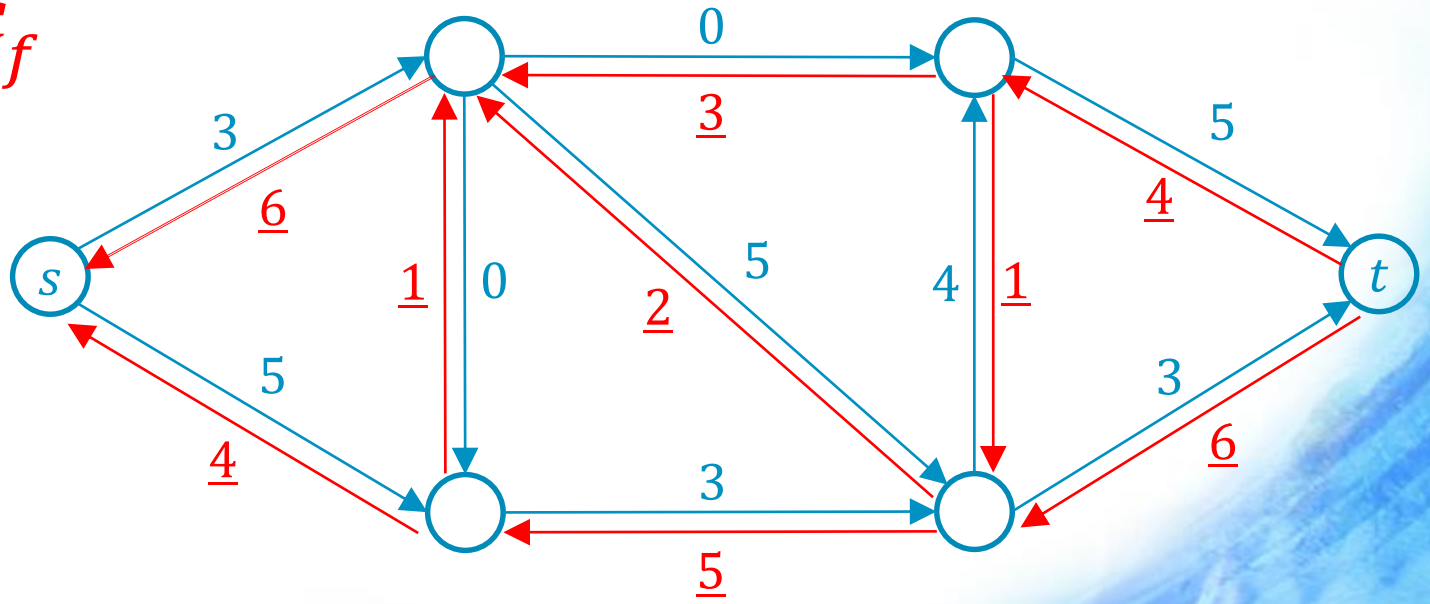
- 与えられるもの
  - ネットワーク (G, u, s, t)
  - s-t フロー f
- **残余グラフ** (residual graph) G_f を以下のように定める
  - 頂点集合は G と同じく V
  - G の各辺 $e = vw \in E$ に対し，次の 2 辺を加える
    - $e = vw$ そのもの
    - $\overleftarrow{e} = wv$： e の**逆辺** (reverse edge)
  - 辺の容量 u_f を次のように定める
    - G の辺 $e \in E$ については $u_f(e) = u(e) - f(e)$
    - G の辺 $e \in E$ の逆辺 $\overleftarrow{e}$ については $u_f(\overleftarrow{e}) = f(e)$



残余グラフ



G_f



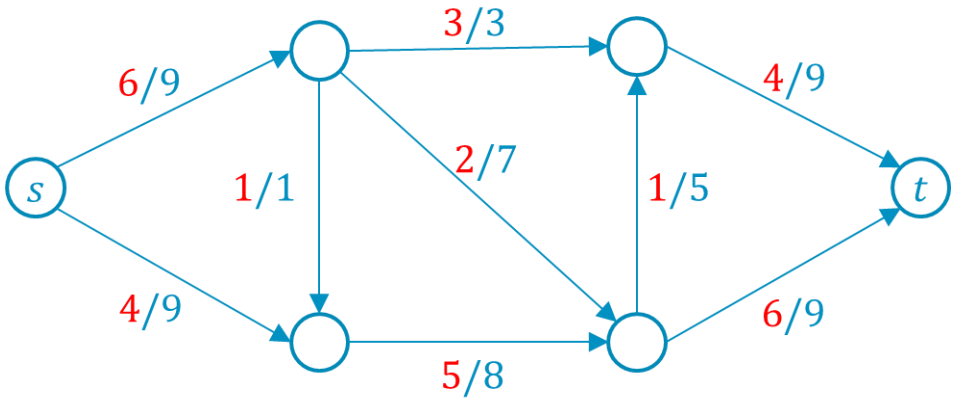
増加路

- G_f の容量正の辺
  - $u_f(e) > 0$ なら e を流れる量をあと $u_f(e)$ 増やせる
  - $u_f(\overleftarrow{e}) > 0$ なら e を流れる量をあと $u_f(\overleftarrow{e})$ 減らせる

  - 容量が 0 の辺は G_f から取り除くという定義もある

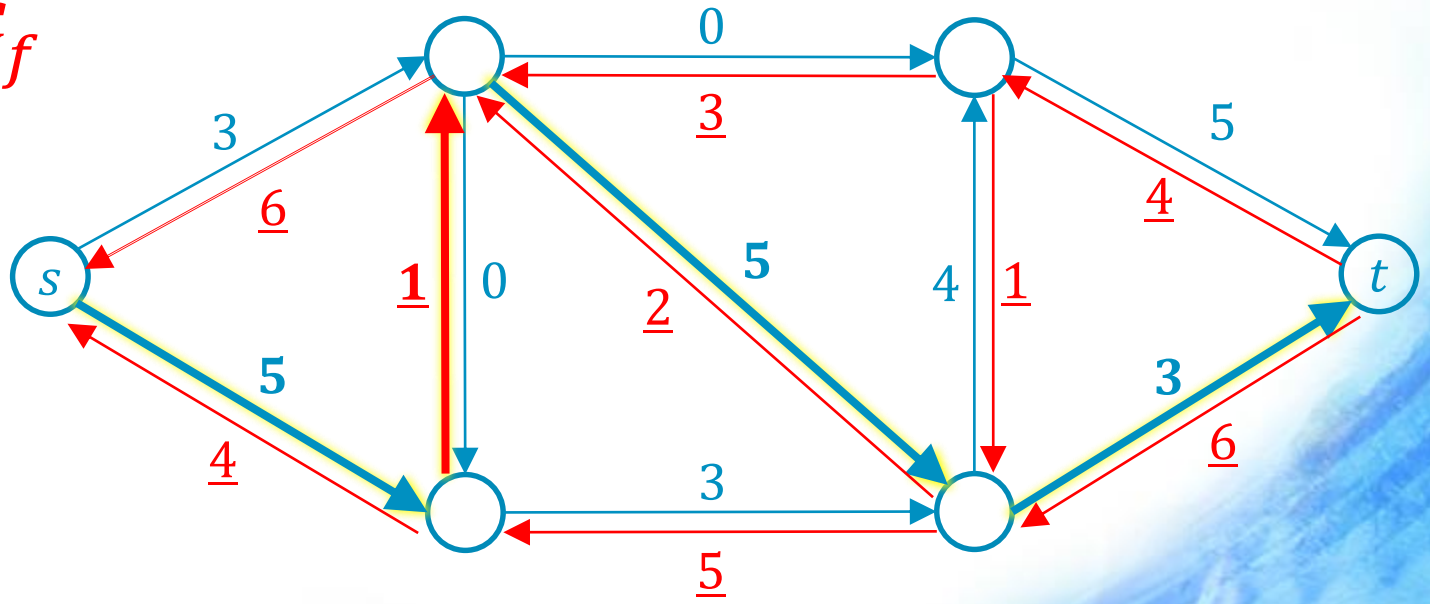
- **増加路** (augmenting path)
  - G_f の容量正の辺からなる s-t パスのこと
    - 増加路に含まれる辺の容量の最小値を δ とすると，増加路に従ってフローを修正すると流量が δ 増える

增加路

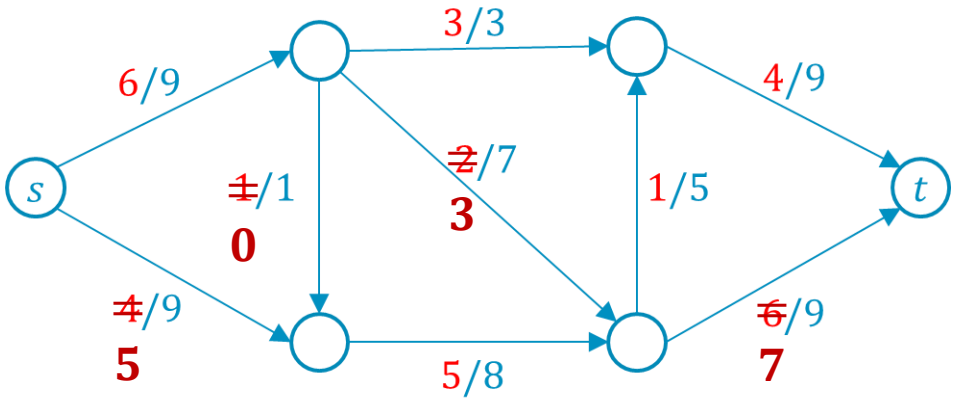


$\delta = 1$

G_f

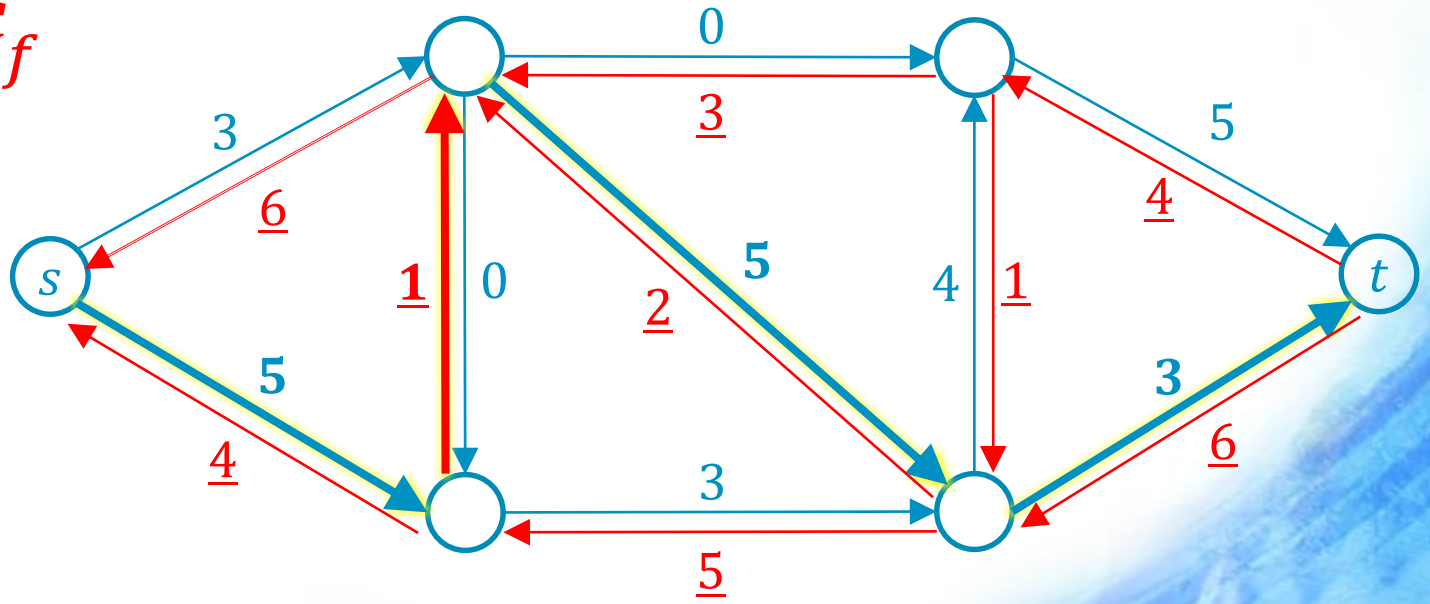


增加路



$\delta = 1$

G_f



残余グラフとカット

- 増加路があるときはフローは最大ではない
- 増加路がないときは？
  - 残余グラフにおいて，容量正の辺のみを用いて s から到達できる頂点の集合を S とし，$T = V \setminus S$ とする
    - 増加路がないので $t \in T$
  - G の辺 $e \in E$ のうち，
    - S から出て T へ入るものは $u_f(e) = 0$ より $f(e) = u(e)$
    - T から出て S へ入るものは $u_f(\overleftarrow{e}) = 0$ より $f(e) = 0$

$$|f| = \sum_{e=s*} f(e) - \sum_{e=*s} f(e) = \sum_{v \in S} \left(\sum_{e=v*} f(e) - \sum_{e=*v} f(e) \right)$$

$$= \sum_{e=vw,\, v \in S,\, w \in T} f(e) - \sum_{e=vw,\, v \in T,\, w \in S} f(e)$$

ここが等号になる

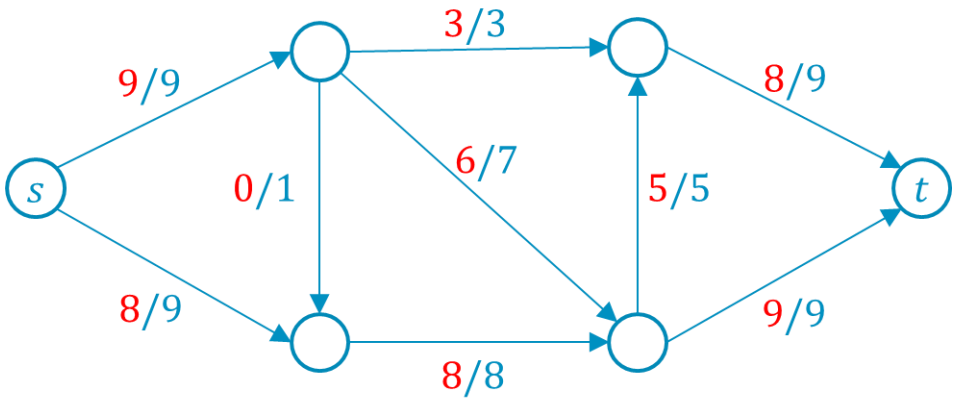
$$\leq \sum_{e=vw,\, v \in S,\, w \in T} u(e) - \sum_{e=vw,\, v \in T,\, w \in S} 0 = u(S,T)$$

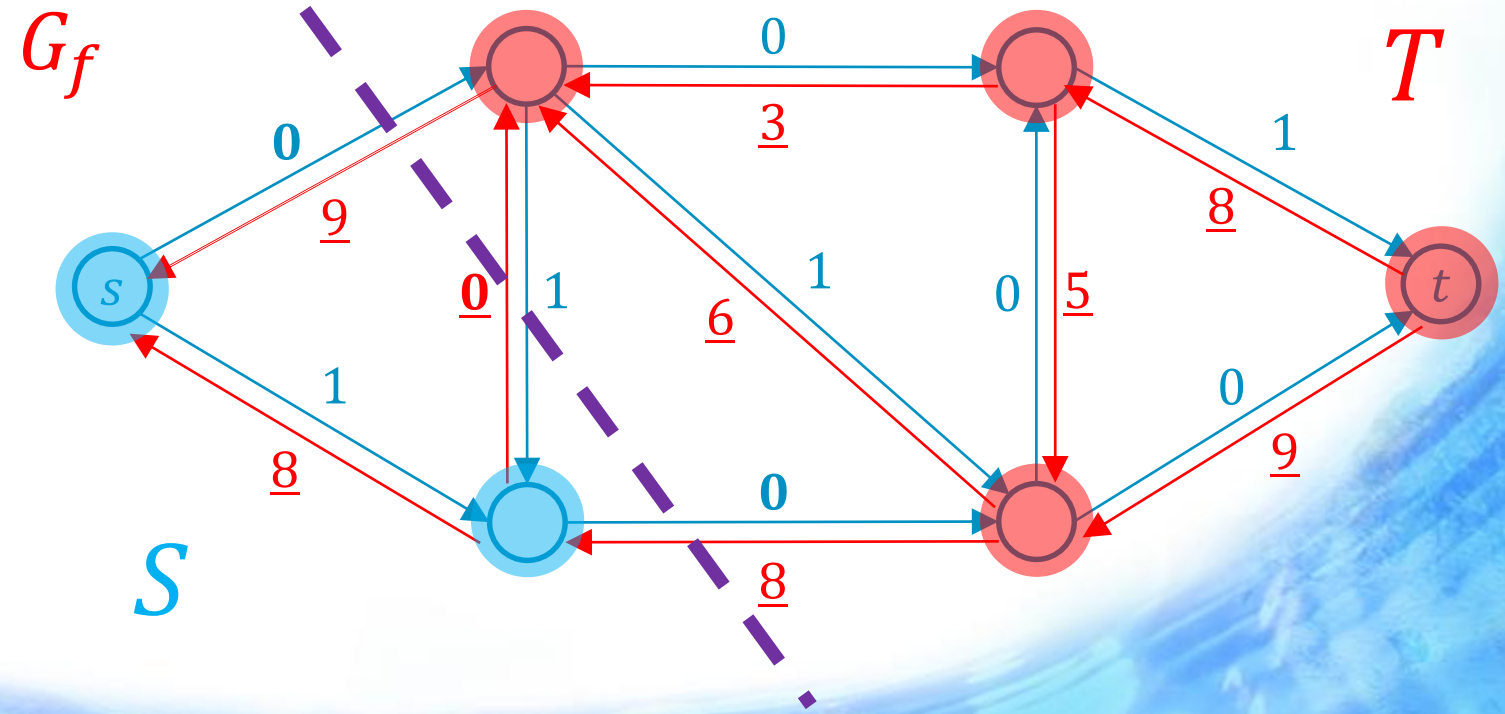
残余グラフとカット

- フローの流量が最大 $\Longrightarrow$ 増加路が存在しない $\Longrightarrow$ フローの流量とカットの容量を等しくできる
  - フローの流量が最大値をとることについて
    - 後述するフローアルゴリズムの停止性を用いれば示せる
    - 解析学の知識を用いるなら，フローの条件は $\mathbb{R}^E$ の有界閉集合 (したがってコンパクト集合) で表されるのでよい

- 示せたこと
  - 最大 s-t フローと最小 s-t カットは等しい
  - 増加路が存在しないことと最大 s-t フローになっていることは同値
    - さらにそのとき，残余グラフで容量正の辺のみを用いて s から到達できるかどうかで最小 s-t カットを得られる

残余グラフとカット





最大流アルゴリズム

- 何も流さない状態から始めて「増加路を見つけてフローを更新」を繰り返す
  - Ford-Fulkerson
    - 増加路を適当に (DFS 等で) 選んでいく
      - 容量が整数でなければ停止しないこともある
  - Edmonds-Karp
    - これから紹介
  - Dinic
  - 容量スケーリング

2^{10} 単位で流せるだけ流す，2^{9} 単位で流せるだけ，2^{8} 単位で，……という感じ

- 他の方法
  - Push/Relabel (Goldberg-Tarjan)
    - 「増加路がないがフローの条件を満たさない」ものから始めて，フローの条件を満たすように修正していく
    - $O\left(|V|^2|E|^{\frac{1}{2}}\right)$ 時間

Edmonds-Karp

Edmonds-Karp のアルゴリズム

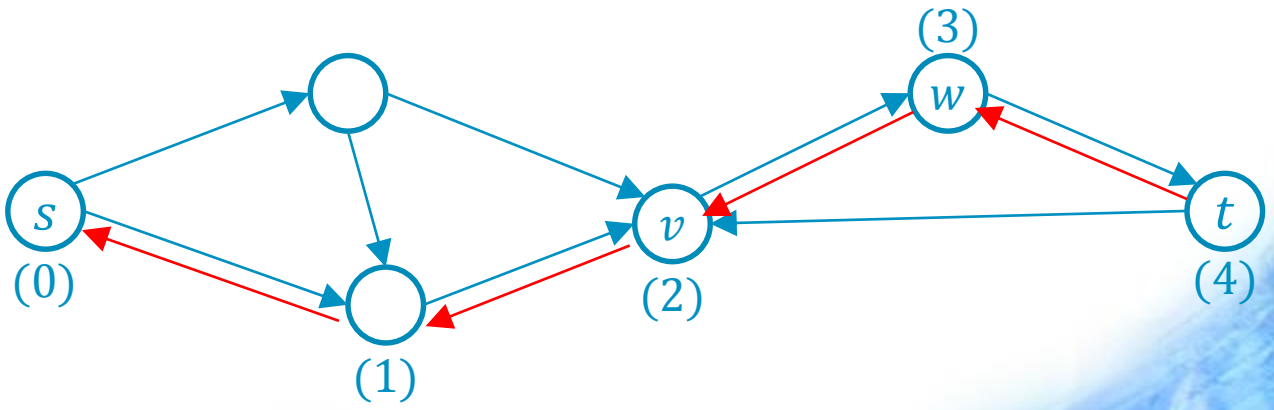
1. すべての $e \in E$ に対し $f(e) = 0$ とおく
2. 以下を繰り返す：
   - ① 残余グラフ G_f で始点 s から BFS を行う
     - G_f の容量正の辺のみ見る
   - ② 終点 t に到達しなければ (増加路が存在しなければ) 終了
   - ③ 辺の個数が最小の増加路を 1 個求める
   - ④ 増加路に従って f を更新する

- 繰り返しの中身は $O(|E|)$ 時間でできる
- 繰り返しの回数は？

Edmonds-Karp

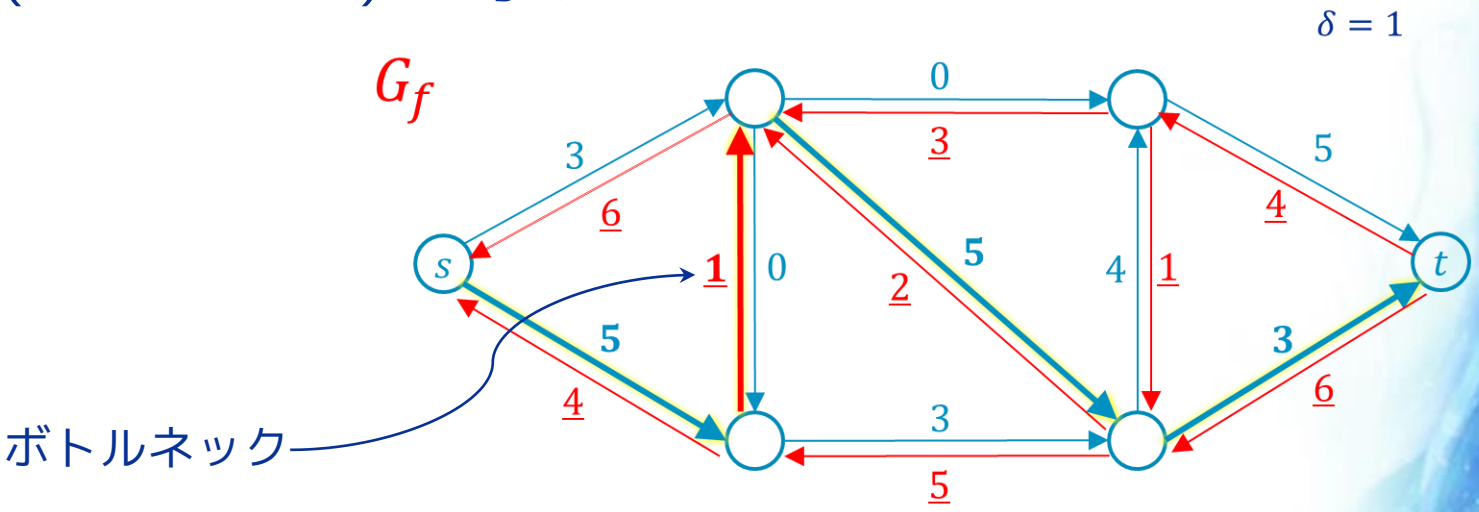
- $v \in V$ に対し，残余グラフの容量正の辺からなる s-v パスの長さの最小値を $d(v)$ とおく

- 残余グラフに容量正の辺が増えるタイミング
  - 辺 wv の容量が 0 から正になる直前，$d(w) = d(v) + 1$
    - 辺 vw が最短増加路に含まれるから
- よって，アルゴリズム中で $d(v)$ は決して減らない



Edmonds-Karp

- 見つけた増加路中の残余グラフでの容量最小の辺を**ボトルネック** (bottleneck) と呼ぶ



- 辺 vw がボトルネックになったとき
  - $d(w) = d(v) + 1$
  - 辺 vw の残余グラフでの容量は正から 0 になる
    - 再び正になる可能性があるのは $d(v) = d(w) + 1$ のとき
      - $d(w)$ は減らないので，$d(v)$ は 2 以上増える必要がある

Edmonds-Karp

- $d(v)$ は $|V|$ 未満または ∞ なので，どの辺についても，それがボトルネックになる回数は高々 $\frac{|V|}{2}$ 回
  - 辺は逆辺を含めて $2|E|$ 本なので，「ある辺がボトルネックになる」ということが起こる回数は高々 $|V||E|$ 回
- 繰り返しでは毎回 1 辺以上がボトルネックになるので，繰り返しは高々 $|V||E|$ 回

- 以上より，Edmonds-Karp アルゴリズムの時間計算量は $O(|V||E|^2)$
  - 多項式時間

Edmonds-Karp

- 実装のヒント
  - 基本的には，グラフを作って BFS して経路復元するだけ
    - 経路復元 (最短路を長さだけでなく具体的に求める) に自信がなければ要練習
  - 隣接行列 vs. 隣接リスト
    - 隣接行列だと，[u][v] の逆辺を [v][u] として簡単にとれる
      - 初めて実装してみるときにおすすめ
      - 多重辺等は扱いにくい
    - 隣接リストだと，逆辺の管理に工夫が必要
      - 例えば，辺を番号で管理し，辺 0 の逆辺が辺 1，辺 2 の逆辺が辺 3，……といった感じにするとよい
      - 計算量的にもよい

Dinic

Dinic のアルゴリズム (概要)

1. すべての $e \in E$ に対し $f(e) = 0$ とおく
2. 以下を繰り返す：
   ① 残余グラフ G_f (容量正の辺のみ見る) で始点 s から BFS を行う
   ② 終点 t に到達しなければ (増加路が存在しなければ) 終了
   ③ $d(w) = d(v) + 1$ となる辺 vw のみを用いた増加路が存在する間，そのような増加路に従って f を更新することを繰り返す (blocking flow)

- 2. の繰り返しの中身は $O(|V||E|)$ 時間でできる
  - 案外難しいです (今回は省略)
- 2. の繰り返しの回数は高々 $|V|$ 回
  - $d(t)$ が毎回 1 以上増えるから

Dinic

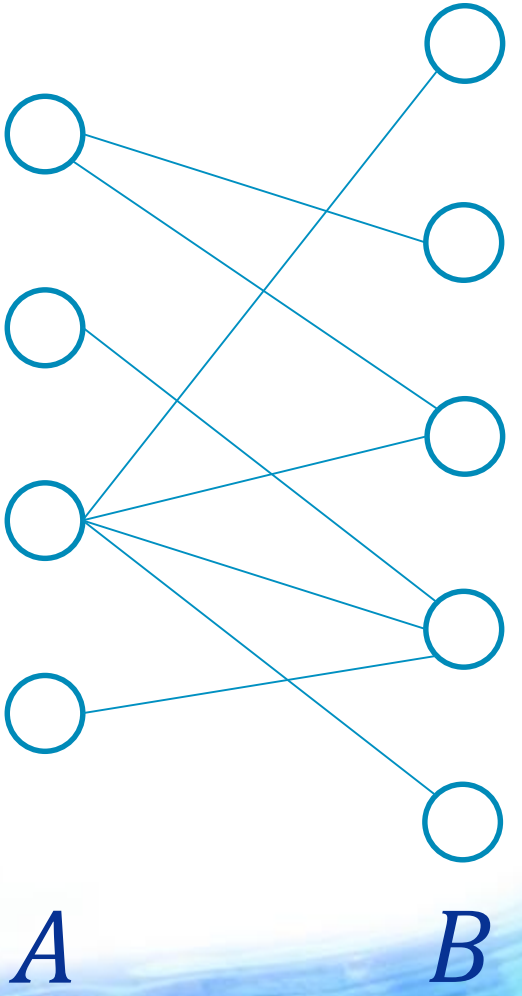
- Dinic 法の時間計算量
  - とりあえず $O(|V|^2|E|)$
    - とはいえ大抵のグラフではとても高速
      - プログラミングコンテストで困ることはまずない
    - 最悪ケースはちゃんと遅いので注意
  - 高速化できる
    - 繰り返しの中身に Link/Cut Tree を使う (Sleator-Tarjan) と $O(|E|\log|V|)$
    - 全体で $O(|V||E|\log|V|)$
      - 定数倍はそこそこある，コンテスト向きではない
  - 特殊なグラフだと速い保証あり
    - すべての辺の容量が 1 なら，繰り返しの回数は $O\left(\min\left\{|V|^{\frac{2}{3}}, |E|^{\frac{1}{2}}\right\}\right)$

二部グラフへの応用

- 無向グラフのいろいろ
  - **マッチング** (matching)
    - 辺の集合であって，どの 2 辺も端点を共有しないもの
  - **頂点被覆** (vertex cover)
    - 頂点の集合であって，すべての辺の端点の少なくとも一方を含むもの
  - **独立集合** or **安定集合** (independent set, stable set)
    - 頂点の集合であって，どの 2 点も辺で結ばれていないもの
    - 補集合が頂点被覆であることと同値
  - いずれも，最大あるいは最小といったら辺や頂点の個数が最大あるいは最小のものを指す

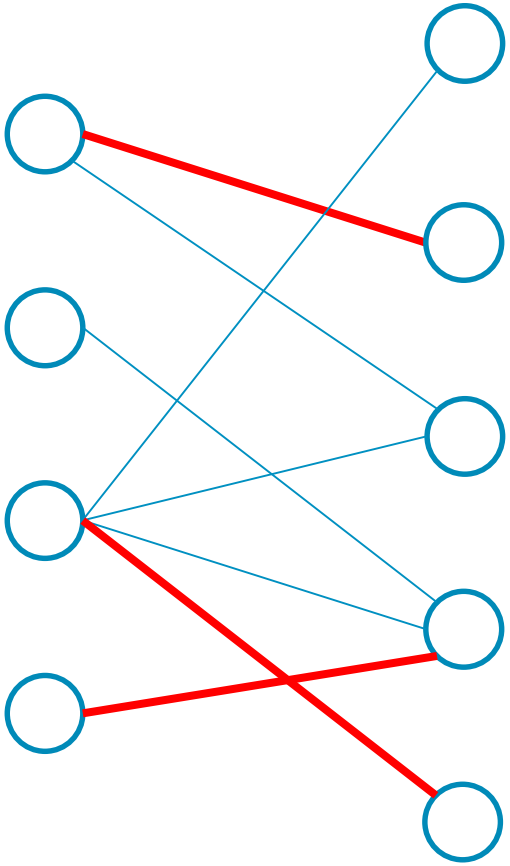
二部グラフへの応用

- **二部グラフ** (bipartite graph)
  - 無向グラフ $G = (V, E)$ であって，すべての辺が A の頂点と B の頂点を結ぶように頂点集合を $V = A \sqcup B$ と分割できるもの



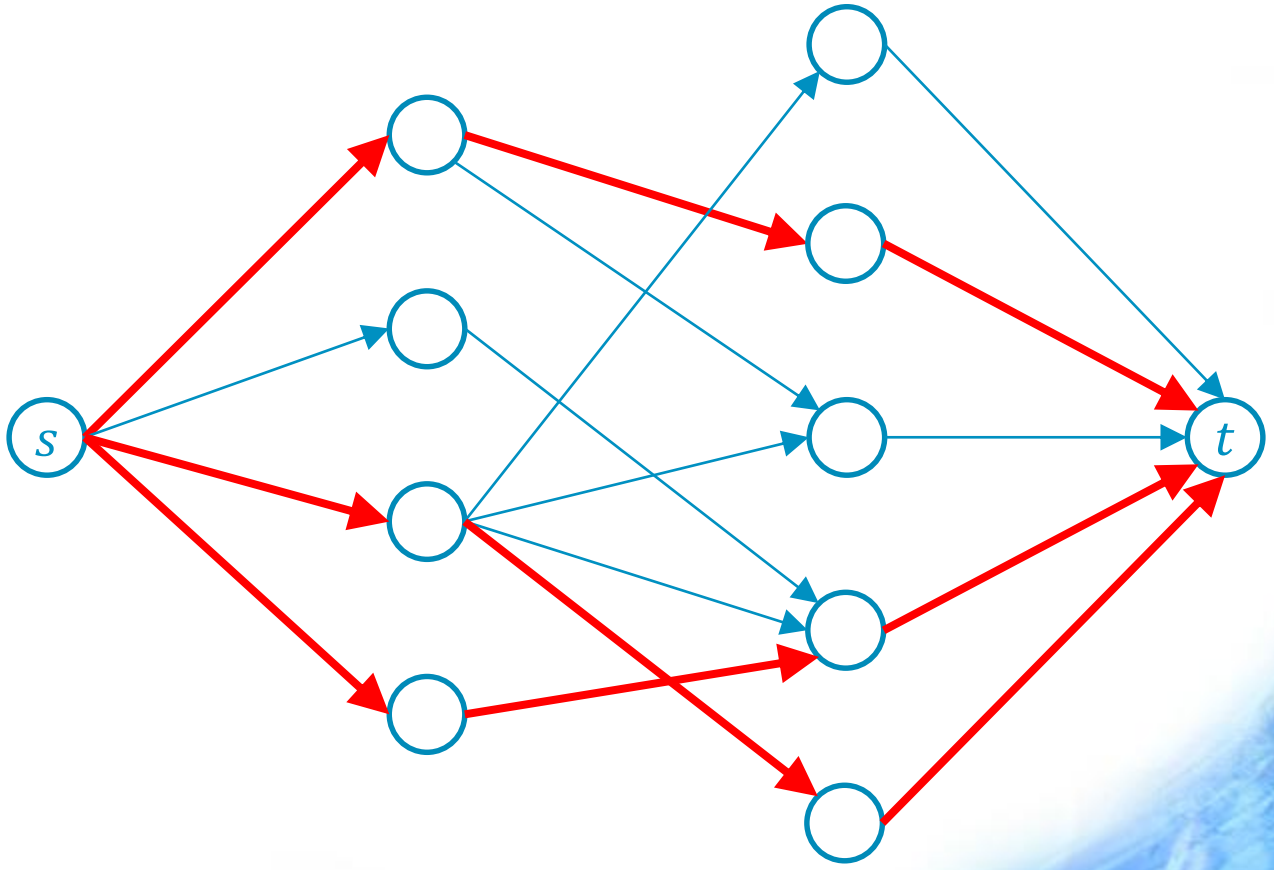
二部グラフへの応用

- 二部グラフの最大マッチングは最大流で求められる



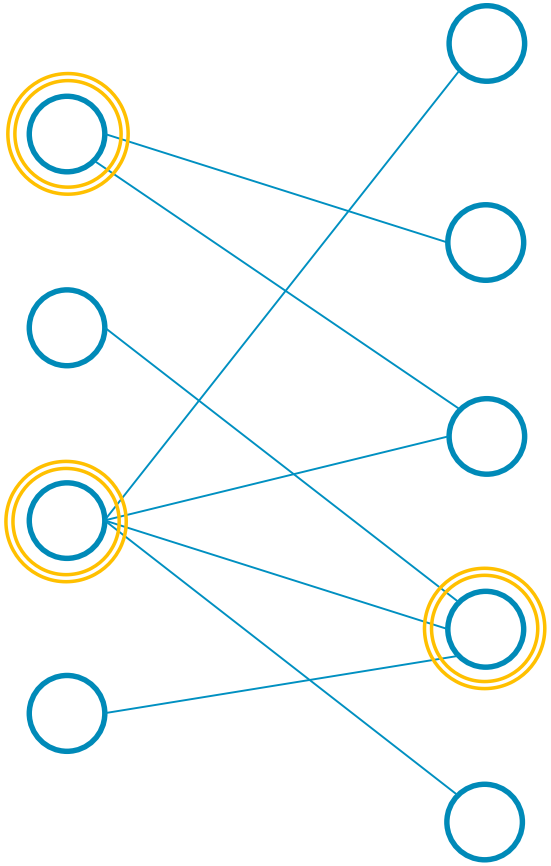
二部グラフへの応用

- 二部グラフの最大マッチングは最大流で求められる
  - 辺の容量はすべて 1



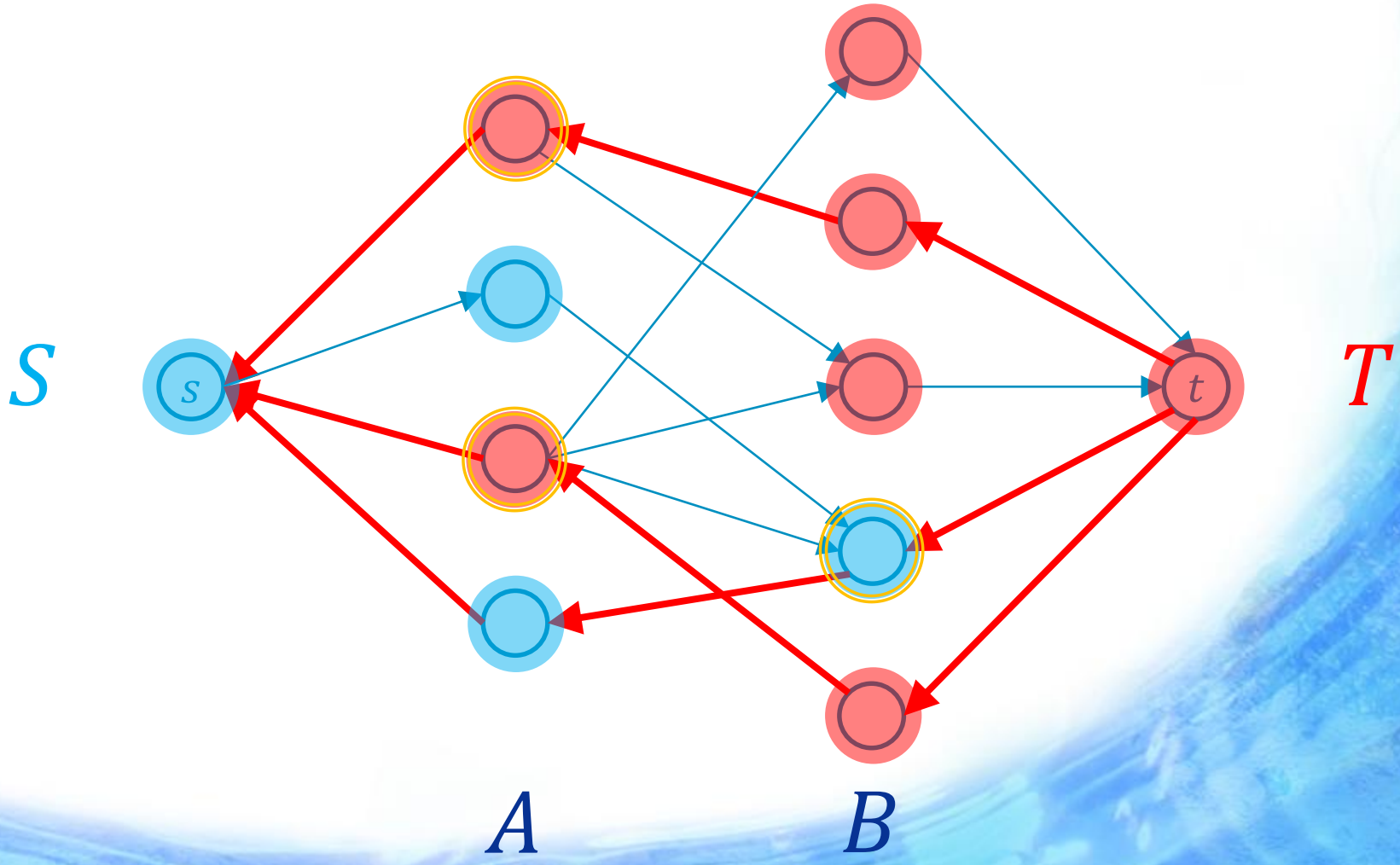
二部グラフへの応用

- 二部グラフの最小頂点被覆は最小カットで求められる



二部グラフへの応用

- 二部グラフの最小頂点被覆は最小カットで求められる
  - $(A \cap T) \cup (B \cap S)$



二部グラフへの応用

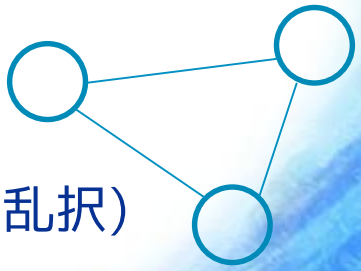
- 二部グラフの最小頂点被覆は最小カットで求められる
  - $(A \cap T) \cup (B \cap S)$
    - 頂点被覆になっている理由
      - $A \cap S$ と $B \cap S$ を結ぶ辺は $B \cap S$ の頂点で被覆
      - $A \cap S$ と $B \cap T$ を結ぶ辺は存在しない
      - $A \cap T$ と $B \cap S$ を結ぶ辺は両端点を選んでいる
      - $A \cap T$ と $B \cap T$ を結ぶ辺は $A \cap T$ の頂点で被覆
    - 最小である理由
      - 最大マッチング以上は必要
      - 最大マッチングの辺は S どうし or T どうしを結ぶので，最大マッチングの各辺をちょうど 1 頂点ずつで被覆できている

二部グラフへの応用

- 最大流アルゴリズムを二部グラフ専用に書き換えておくとコード量的にも計算量の定数倍的にもよい

- グラフの特殊性から漸近計算量が抑えられる
  - 最大流が $O(|V|)$ なので Edmonds-Karp などは $O(|V||E|)$ 時間
  - Dinic ベースなら $O\left(|V|^{\frac{1}{2}}|E|\right)$ (Hopcroft-Karp)

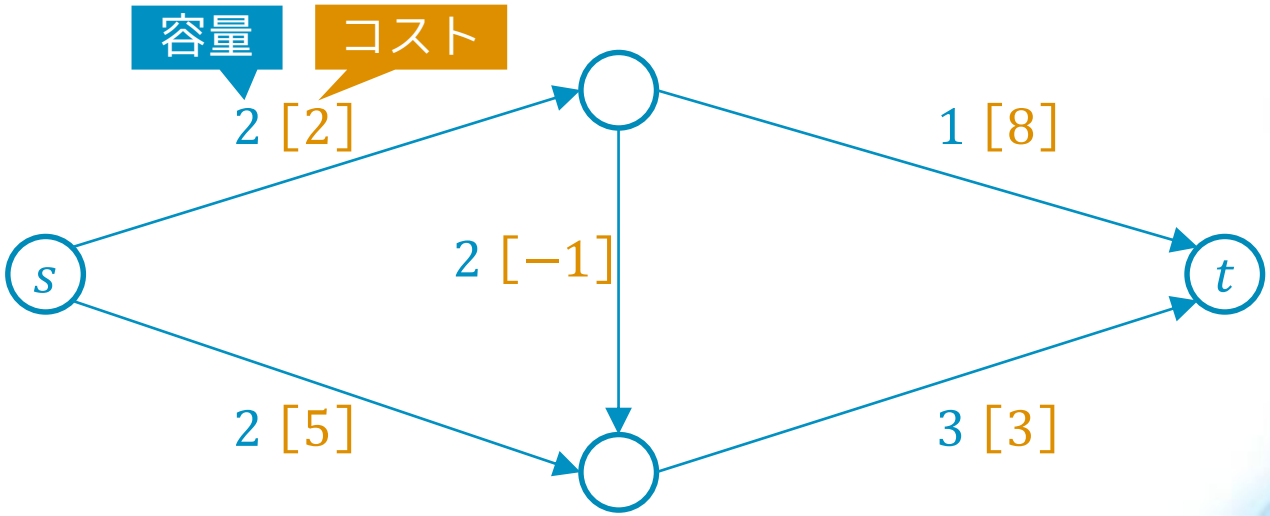
- 一般のグラフでは，最大マッチング ≤ 最小頂点被覆 であるが，等号は必ずしも成り立たない
  - 最大マッチングは $O(|V|^3)$ 時間で求まる
    - Edmonds 法 (難しい) または Tutte 行列 (乱択)
  - 最小頂点被覆問題は NP 困難



II. 最小費用流

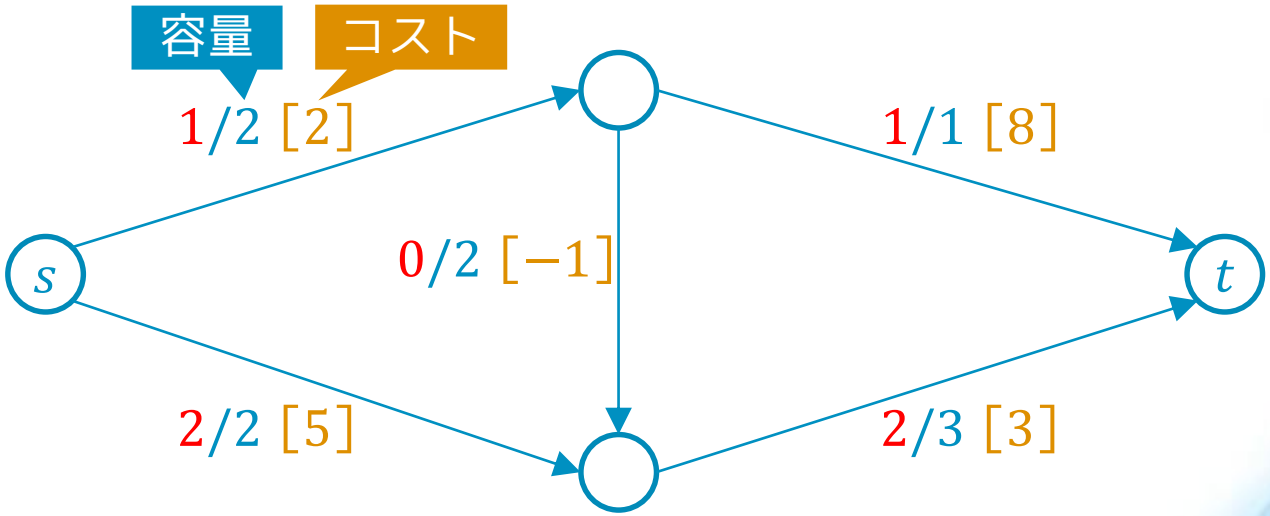
最小費用流問題

- 各辺には次の 2 つの値が定まっています：
  - どのくらい水を流せるか
  - 水を 1 流すごとに費用がいくらかかるか
- 頂点 s から頂点 t へ水を 3 流すとき，最も安くするには？



最小費用流問題

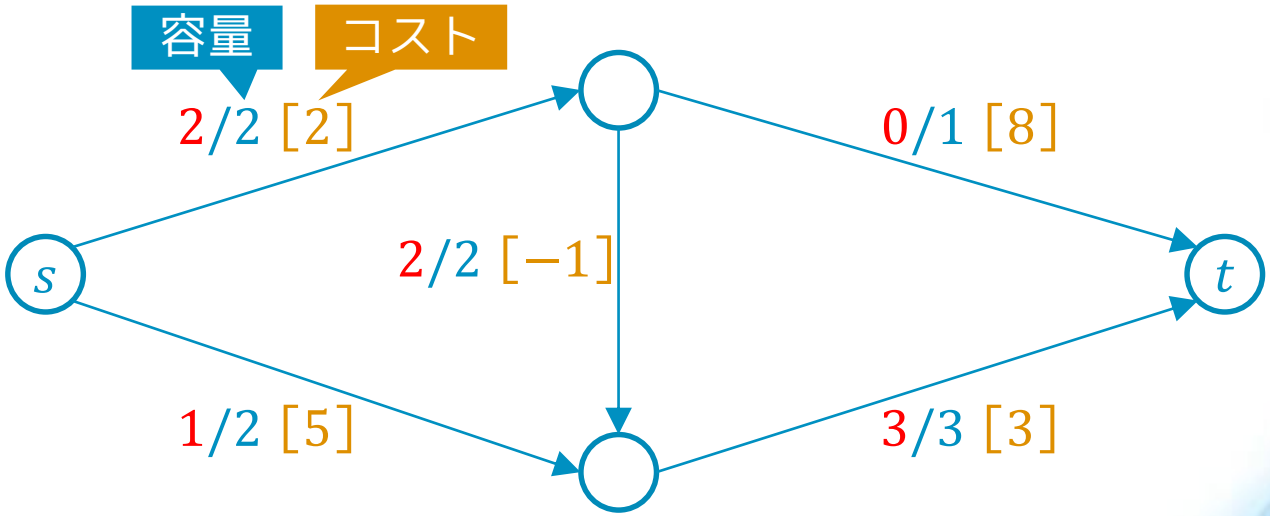
- 各辺には次の 2 つの値が定まっています：
  - どのくらい水を流せるか
  - 水を 1 流すごとに費用がいくらかかるか
- 頂点 s から頂点 t へ水を 3 流すとき，最も安くするには？



↑ 費用の合計 26

最小費用流問題

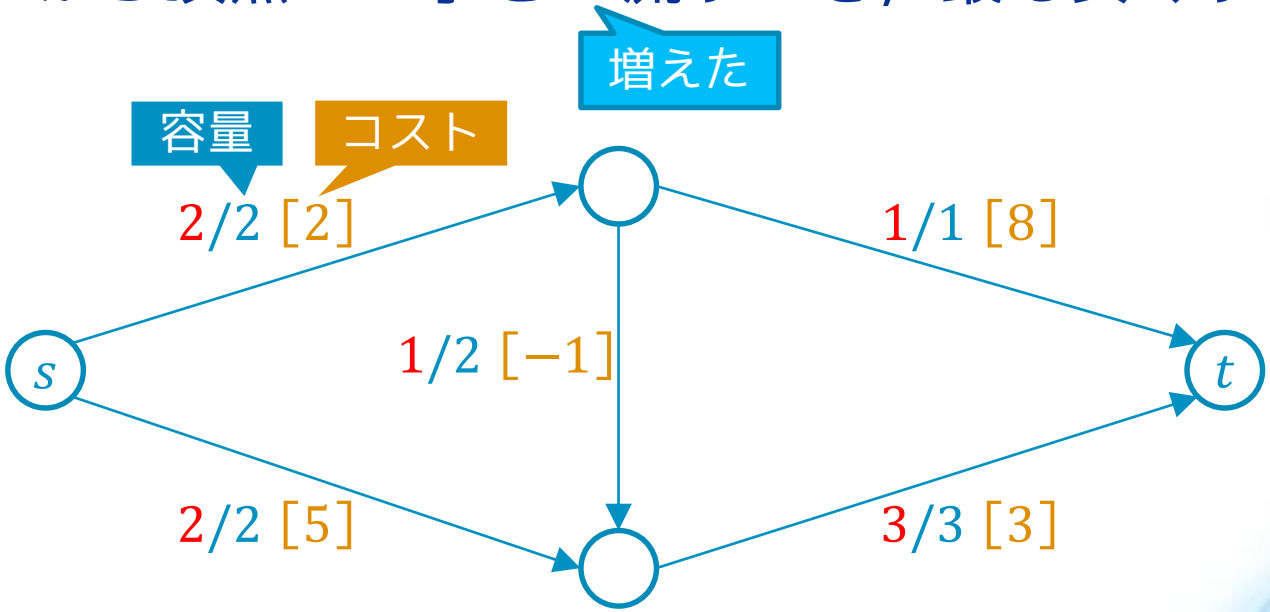
- 各辺には次の 2 つの値が定まっています：
  - どのくらい水を流せるか
  - 水を 1 流すごとに費用がいくらかかるか
- 頂点 s から頂点 t へ水を 3 流すとき，最も安くするには？



↑ 費用の合計 16

最小費用流問題

- 各辺には次の 2 つの値が定まっています：
  - どのくらい水を流せるか
  - 水を 1 流すごとに費用がいくらかかるか
- 頂点 s から頂点 t へ水を 4 流すとき，最も安くするには？



↑ 費用の合計 30

最小費用流問題 (minimum cost flow problem)

- 与えられるもの：
  - (費用つき) ネットワーク
    - 有向グラフ $G = (V, E)$
    - 各辺 $e \in E$ に対して，容量 $u(e) \geq 0$
    - 各辺 $e \in E$ に対して，**費用** (cost) $c(e)$ (負でも OK)
    - 始点 $s \in V$ と終点 $t \in V$ $(s \neq t)$
  - 要求流量 $k \geq 0$

- 作るもの：流量 k の s-t フロー f

- 最小化したいもの：フローの**費用** (cost)

$$c(f) = \sum_{e \in E} c(e) f(e)$$

最小費用流の亜種

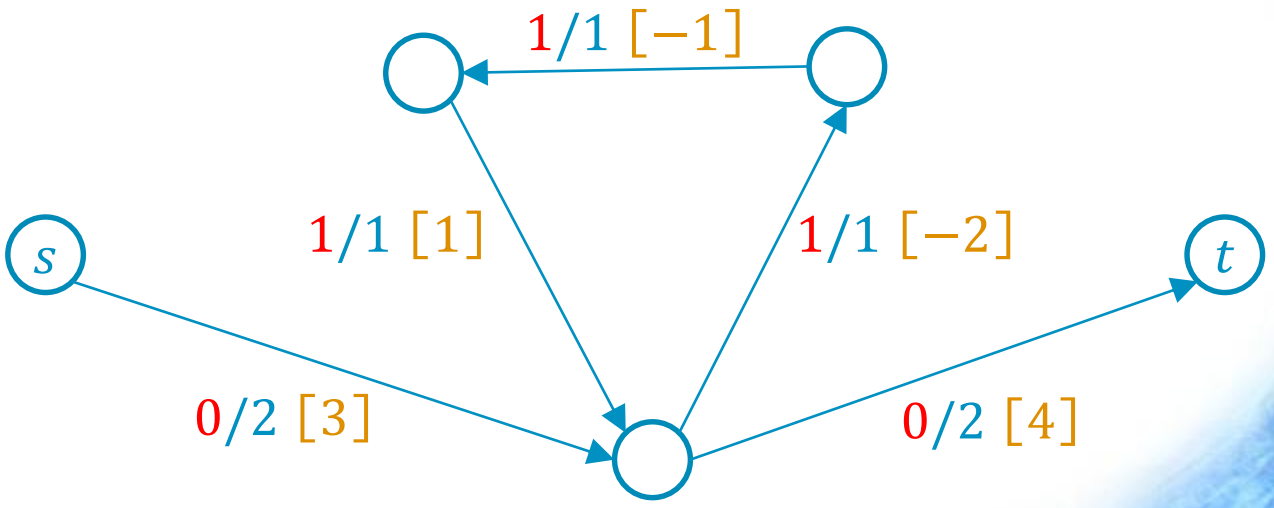
- 流量が特に指定されない場合
  - 負費用辺を有意義に使える限り流す
- 流量が 0 と指定される場合
  - **最小費用循環流** (minimum cost circulation)

最小費用流アルゴリズム

- フローを徐々に増やしていく系
  - 最短路反復
    - これから紹介
    - 擬多項式時間 (k に比例する計算量)
  - 容量スケーリング
    - 多項式時間になる ($\log k$ に比例する計算量)
  - 最小平均長閉路解消
    - 循環流の問題に変形，負閉路を消していく
- Push/Relabel 系
  - 費用スケーリング
    - 状況によってはとても速い

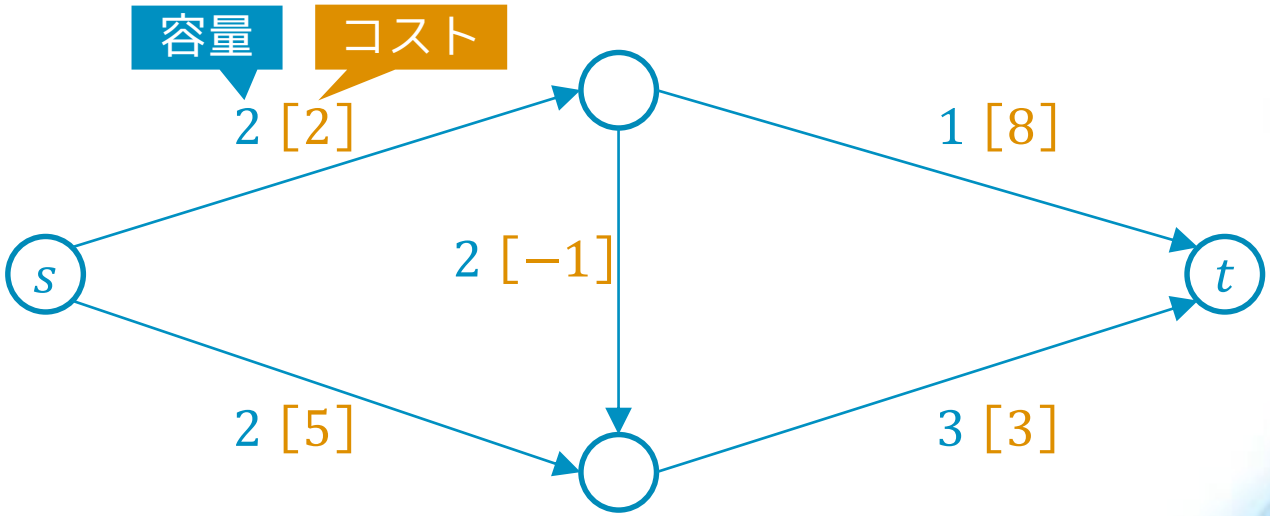
最短路反復

- 入力に対する仮定
  - 辺の容量 $f(e)$ と要求流量 k は整数とする
    - 1 ずつ増やしていくため
  - 費用の合計が負になる閉路は存在しないとする
    - 今後単に「負閉路」と呼ぶ
    - 閉路解消系のアルゴリズムならあっても大丈夫



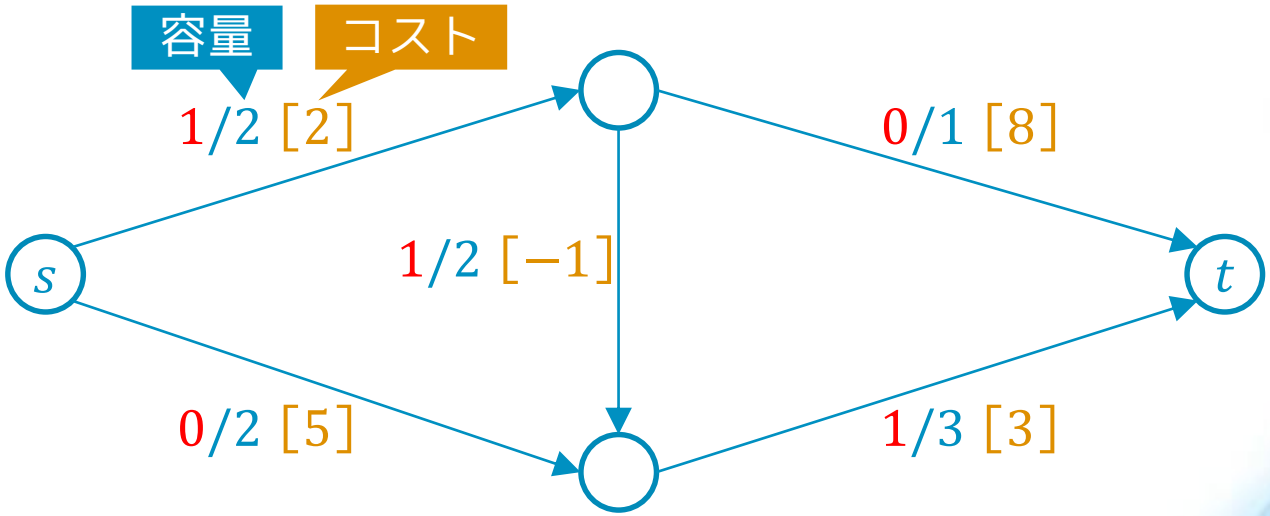
最短路反復

- 流量 1 の最小費用 s-t フローは？



最短路反復

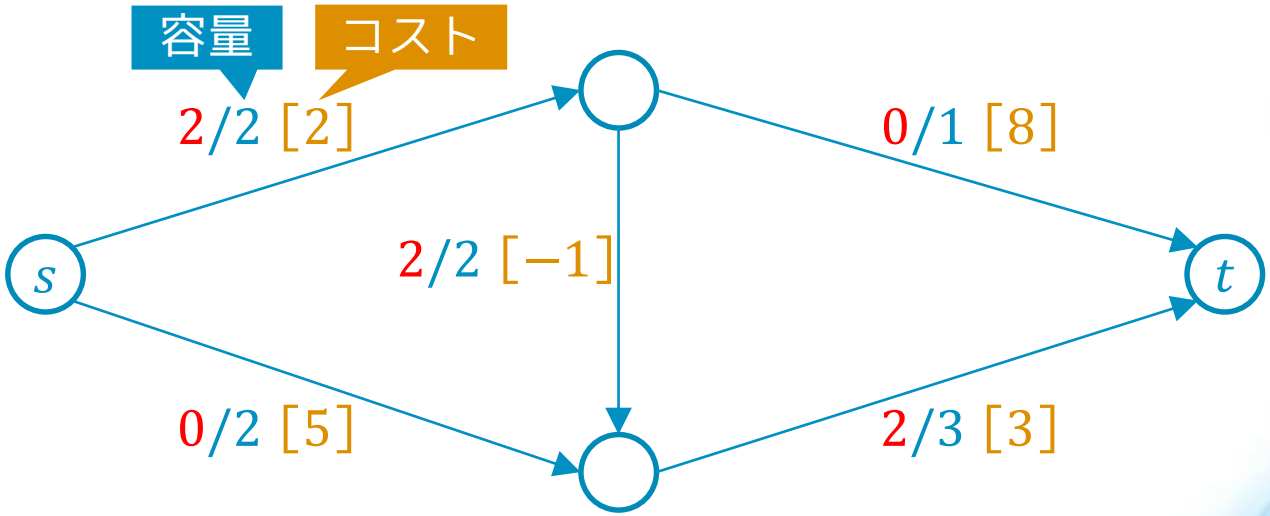
- 流量 1 の最小費用 s-t フローは？
  - (費用についての) 最短路が最適



↑ 費用 4

最短路反復

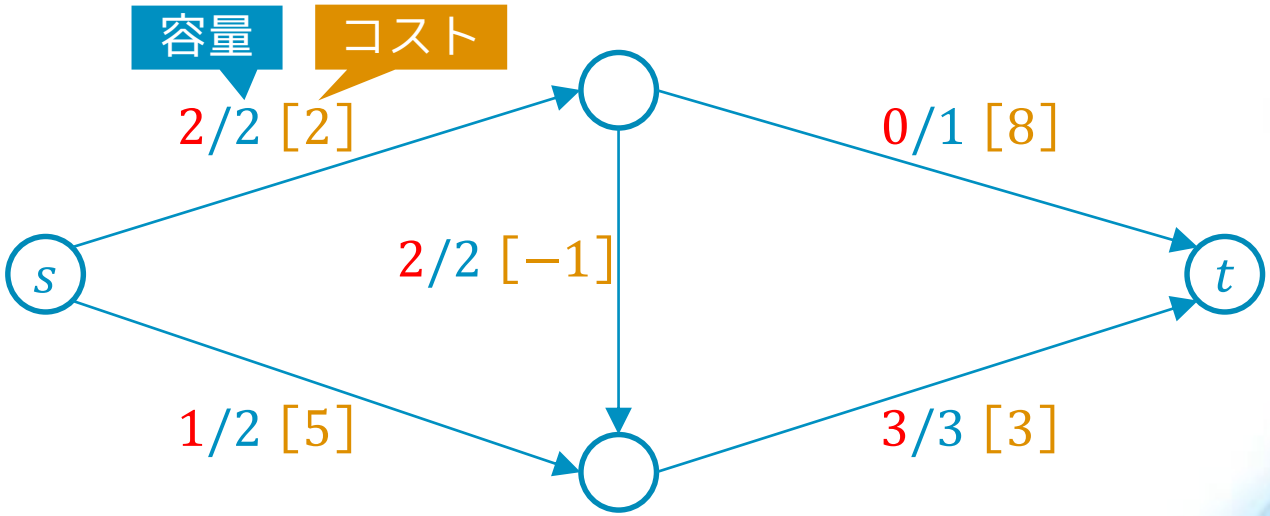
- 流量 2 の最小費用 s-t フローは？



↑ 費用 8

最短路反復

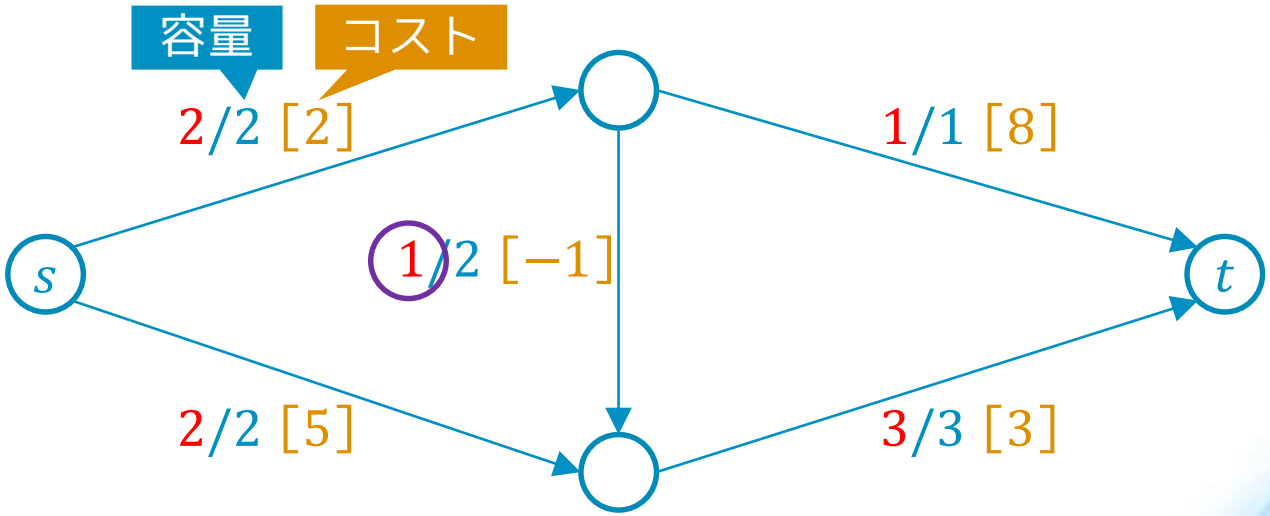
- 流量 3 の最小費用 s-t フローは？
  - 「残っている辺で最短路」を繰り返す？



↑ 費用 16

最短路反復

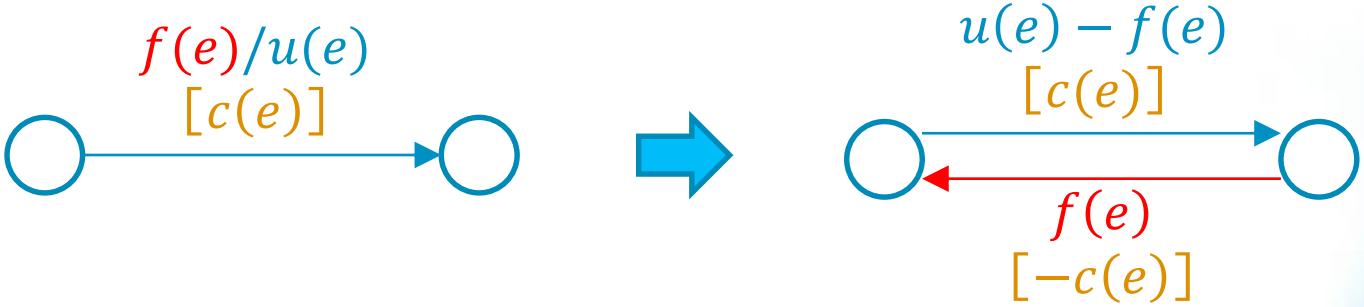
- 流量 4 の最小費用 s-t フローは？
  - フローを減らすべき辺があるかもしれない



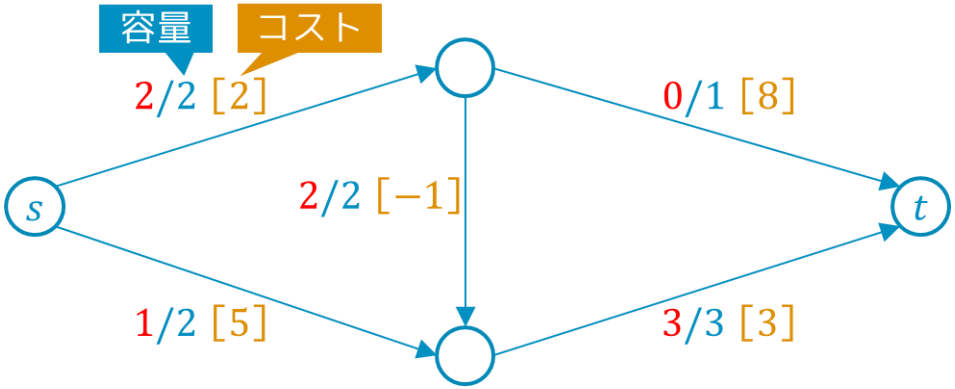
↑ 費用 30

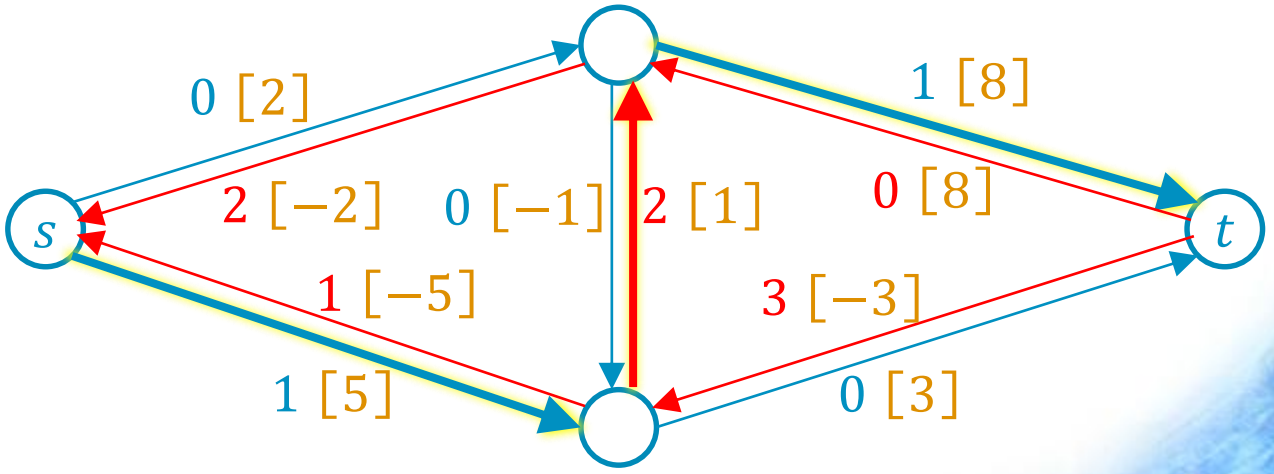
残余グラフ

- 残余グラフも費用つきにする
  - G の辺 $e \in E$ に対する $c(e)$ はそのまま
  - G の辺 $e \in E$ の逆辺 $\overleftarrow{e}$ の費用を $c(\overleftarrow{e}) = -c(e)$ で定める



残余グラフ





最短路反復

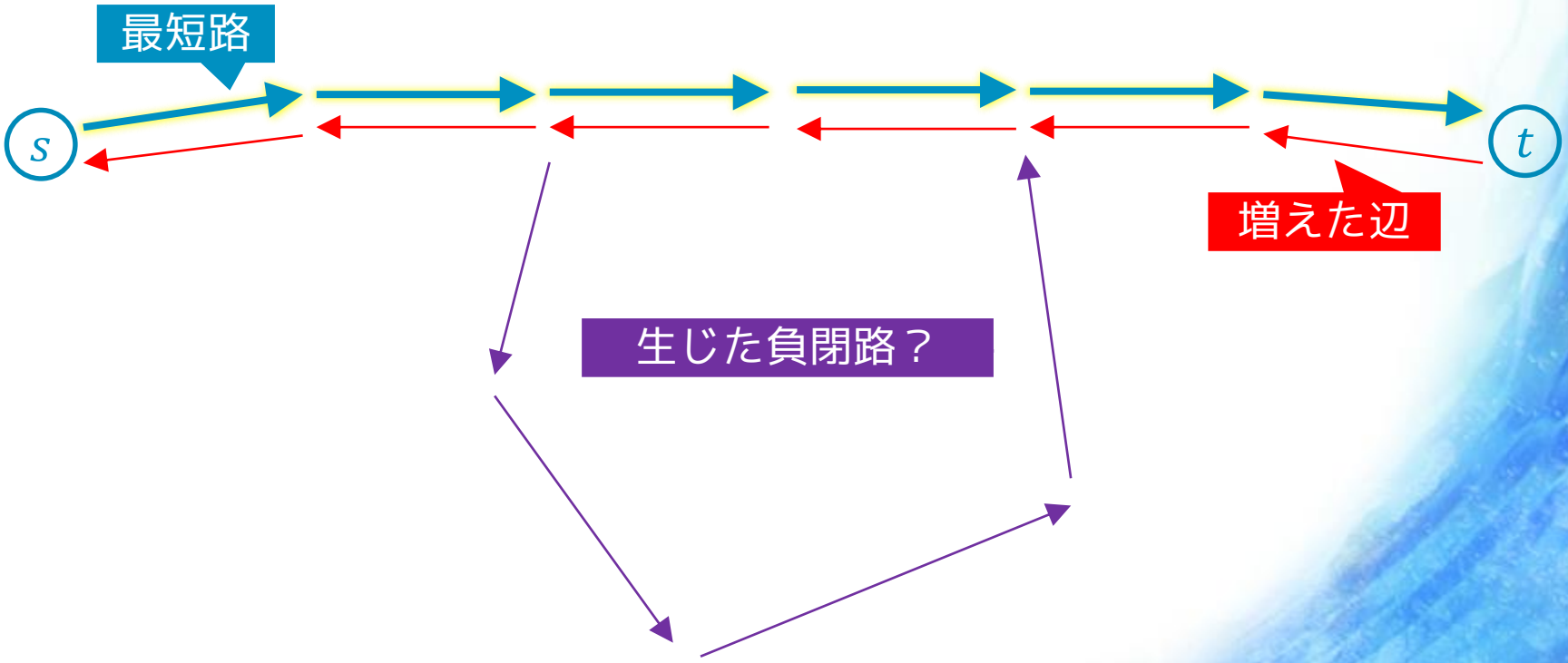
最短路反復法

1. すべての $e \in E$ に対し $f(e) = 0$ とおく
2. フローの流量が k 未満の間，以下を繰り返す：
   ① 残余グラフ G_f (容量正の辺のみ見る) で始点 s から終点 t までの費用についての最短路を求める
   ② 終点 t に到達しなければ終了 (この場合流量 k は不可能)
   ③ 最短路の 1 つに従ってフローを更新

- 最短路は Bellman-Ford で $O(|V||E|)$ 時間
  - 全体で $O(k|V||E|)$ 時間
- 正当性は？
  - 残余グラフに負閉路が生じないこと
  - ちゃんと最小費用になっていること

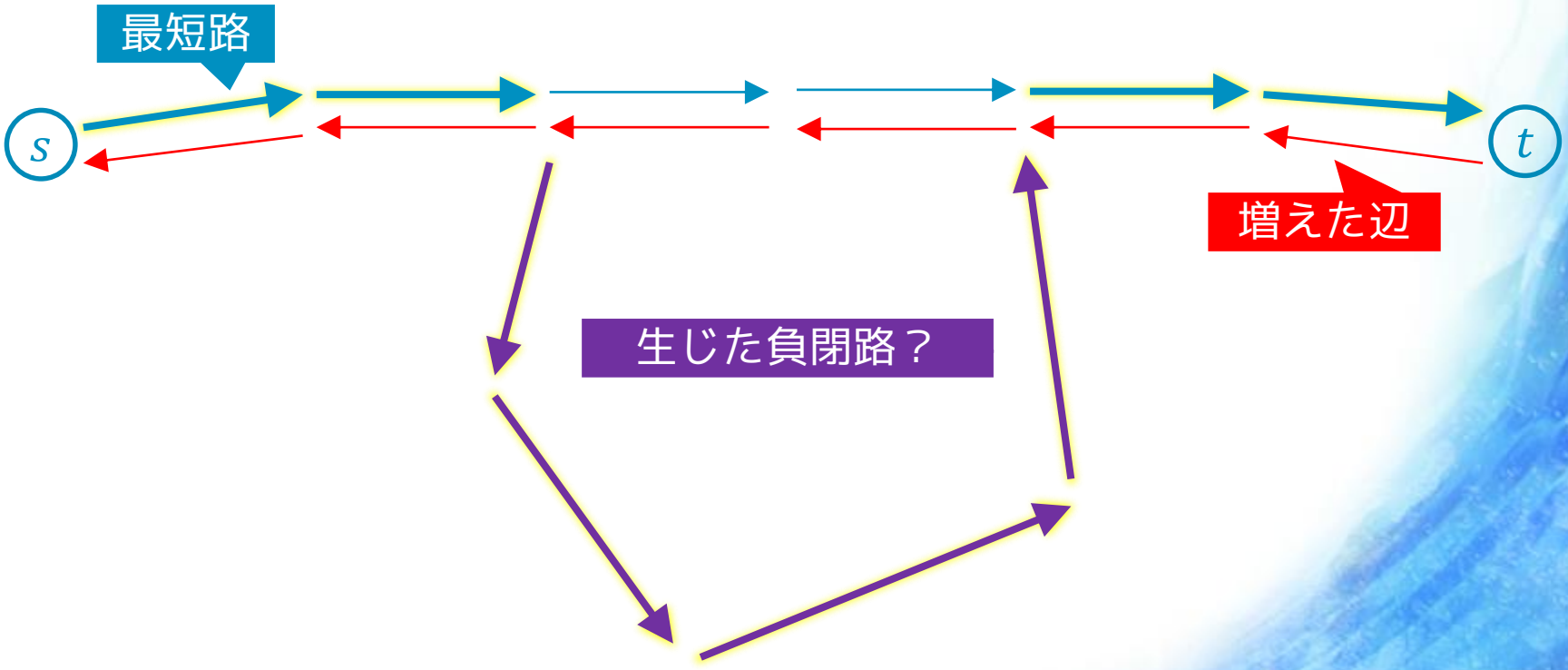
最短路反復

- フロー更新時，残余グラフに容量正の負閉路は生じない？



最短路反復

- フロー更新時，残余グラフに容量正の負閉路は生じない！
  - もっと短い s-t パスがあることになり矛盾



最短路反復

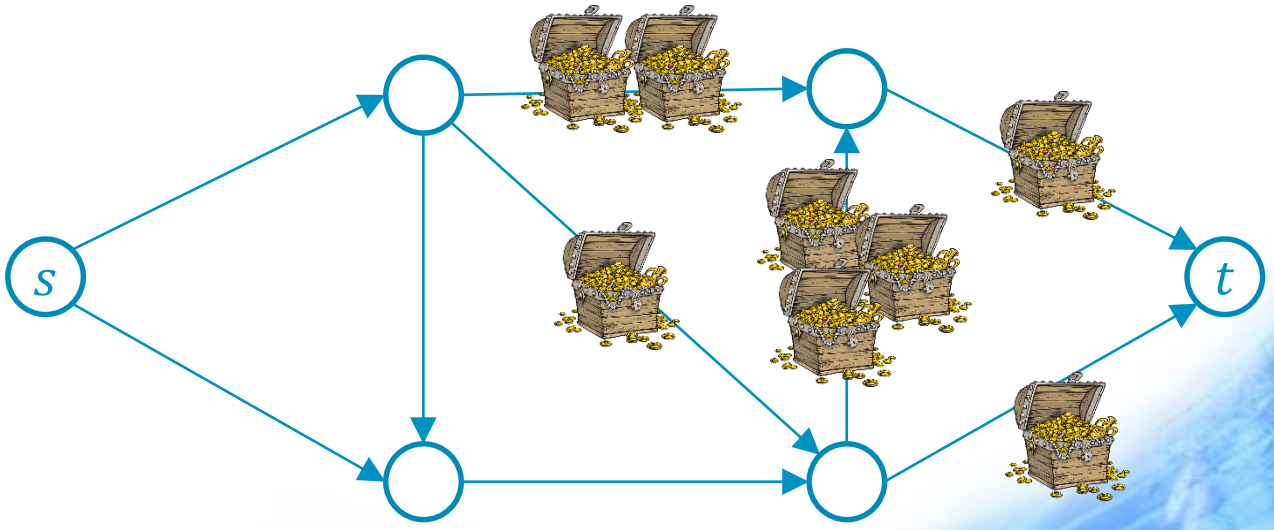
- 残余グラフに容量正の負閉路が存在しないなら，その流量での最小費用 s-t フローになっている
  - 証明の概要：
    - 今の s-t フロー f より費用が小さく流量が同じフロー f' が存在したとする
    - f' と f の差をとってうまく調節すると費用が負の循環流を得る
    - 循環流閉路に分割できて，そのうちいずれかが負閉路

最短路反復

- 最短路反復法の高速化
  - Bellman-Ford を最初の 1 回だけで済ます方法がある
    - うまくポテンシャルをとって Dijkstra できる形にする
    - 全体で $O(|V||E| + k|E|\log|V|)$ 時間
    - もしもともと負辺がなければ最初から Dijkstra のみでよく，$O(k|E|\log|V|)$ 時間

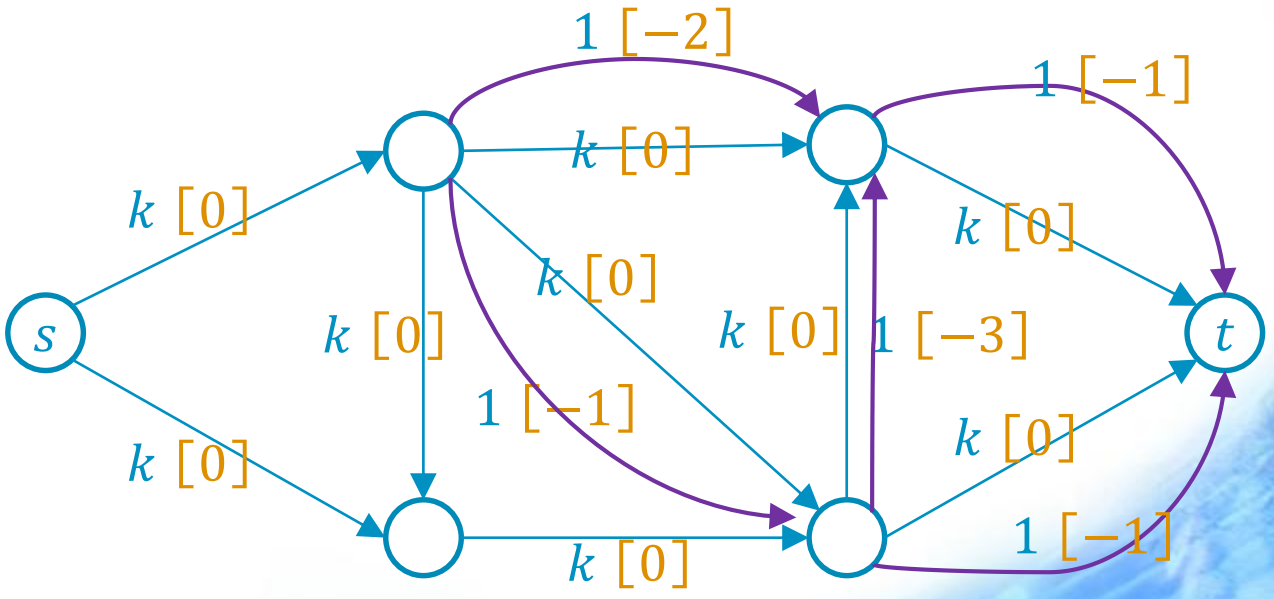
最小費用流の問題例

- 閉路がない有向グラフがある
- 始点 s と終点 t が指定されている
- いくつかの辺には宝が置いてありそれぞれ価値がある
- k 人がそれぞれ s から t へ向かうとき，回収できる宝の価値の合計を最大化せよ
  - 宝は置かれている辺を最初に通った人だけが回収できる



最小費用流の問題例

- 元のグラフの各辺ごとに次の 2 辺を作る
  - 宝をとれないけど進める：容量 k，費用 0
  - 1 人だけ宝をとれる：容量 1，費用 $-$(宝の価値)
- 最小費用 s-t フローで流量 k のものが答え (の -1 倍)



最小費用流の問題例

- 負閉路があると，人が通れない場所にフローが流れてしまってまずいことがある
- 元のグラフに閉路がある場合もうまい処理をしてやると解けます
  - 考えてみてください

III. 線型計画法

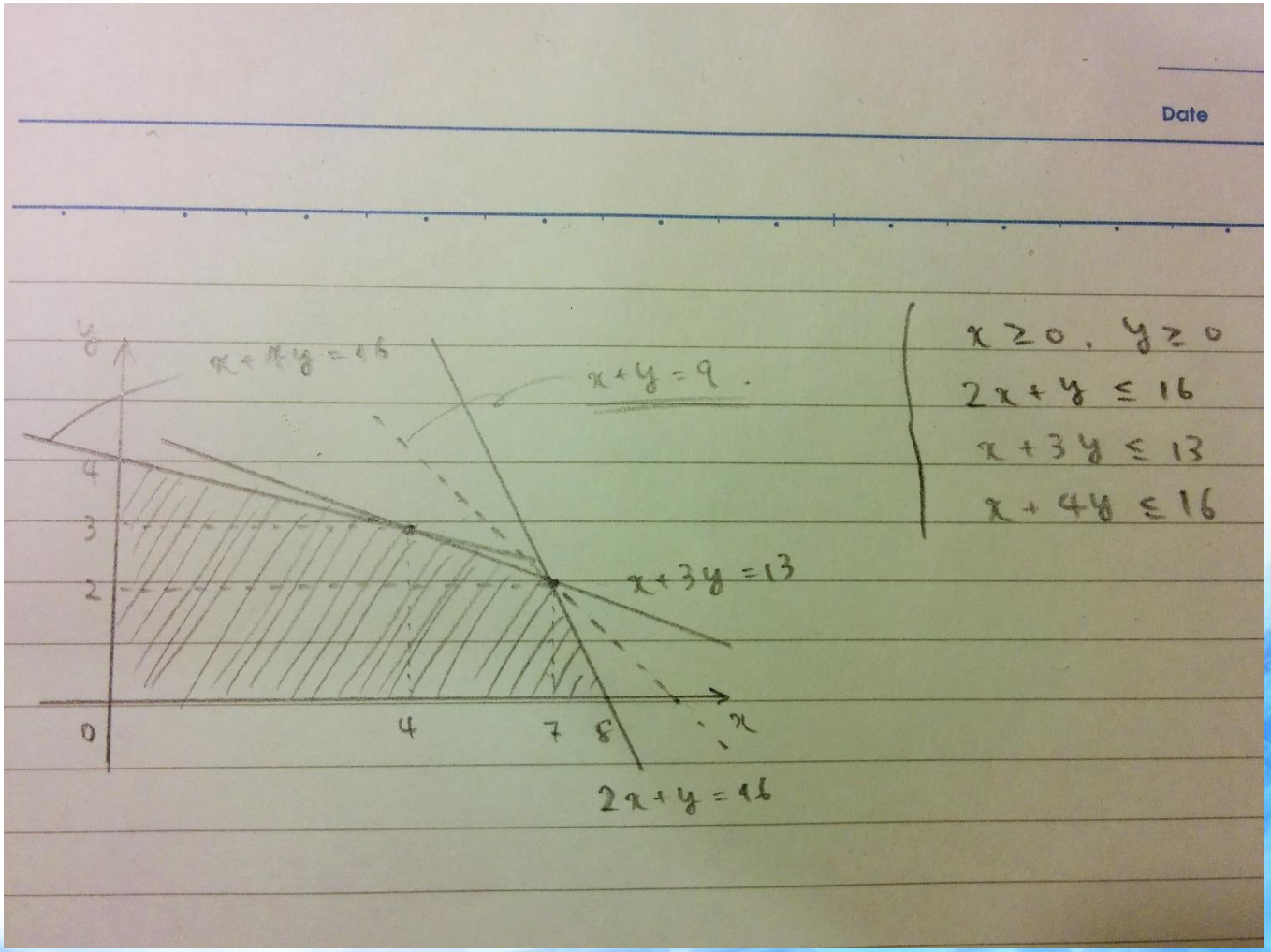
線型計画問題

- 実数を動く変数がいくつか
- 制約は変数たちの 1 次式
- 最大化したい値も変数たちの 1 次式

- 例
  - $x \geq 0$，$y \geq 0$，$2x + y \leq 16$，$x + 3y \leq 13$，$x + 4y \leq 16$ のとき，$x + y$ の最大値は？

線型計画問題

- 真面目なひとの解法
  - 変数が動く領域を図示してがんばる



線型計画問題

- 天才な人の解法
  - $2x + y \leq 16$ より $\frac{4}{5}x + \frac{2}{5}y \leq \frac{32}{5}$
  - $x + 3y \leq 13$ より $\frac{1}{5}x + \frac{3}{5}y \leq \frac{13}{5}$
  - 足すと $x + y \leq 9$ がわかる
  - $(x, y) = (7, 2)$ で実際に $x + y = 9$ となるので，これが最大

線型計画問題

- 天才な人の真似をしようとした人の解法
  - $2x + y \leq 16$ より $2px + py \leq 16p$
  - $x + 3y \leq 13$ より $qx + 3qy \leq 13q$
  - $x + 4y \leq 16$ より $rx + 4ry \leq 16r$
  - 足すと $(2p + q + r)x + (p + 3q + 4r)y \leq 16p + 13q + 16r$ がわかる
  - $x + y \leq \bullet$ を示したいので，$2p + q + r \geq 1$，$p + 3q + 4r \geq 1$ が成り立ってほしく，$16p + 13q + 16r$ はできるだけ小さくなってほしい
  - …… $p = \frac{1}{5}$，$q = \frac{2}{5}$，$r = 0$ とするとよさそう！！！

線型計画問題 (linear programming)

- 与えられるもの
  - A ： $m \times n$ 行列
  - b ： m 次元縦ベクトル
  - c ： n 次元横ベクトル
- 問題

全成分非負　　成分ごと比較

$$\max \{ cx \mid x \geq 0,\ Ax \leq b \}$$

- 次のいずれであるかを判定する
  - **実行不可能** (infeasible)： $x \geq 0$ かつ $Ax \leq b$ を満たす n 次元縦ベクトル x は存在しない
  - **実行可能** (feasible)
    - **非有界** (unbounded)： $x \geq 0$ かつ $Ax \leq b$ を満たす x であって，cx がいくらでも大きいものが存在する
    - **有界** (bounded)
      - このときはさらに，cx の最大値も求める

線型計画問題

- 問題

$$\max \{ cx \mid x \geq 0,\ Ax \leq b \}$$

- 最大値を上から評価したい
- $Ax \leq b$ の i 番目の式を y_i 倍して足すと $yAx \leq yb$ となる
  - y は m 次元横ベクトル

- 以下が成り立ってほしい
  - $y \geq 0$ でないとダメ (不等式を負数倍すると逆になる！)
  - $yA \geq c$ でないと cx の評価にならない
  - yb はできるだけ小さいほうが厳しい評価になって嬉しい
- ということで次の問題が生まれる

$$\min \{ yb \mid y \geq 0,\ yA \geq c \}$$

双対 LP

- $\max\{cx \mid x \geq 0,\ Ax \leq b\}$ に対して $\min\{yb \mid y \geq 0,\ yA \geq c\}$ は**双対 LP** (dual LP) と呼ばれる
  - 対応して，元の LP は**主 LP** (primal LP) と呼ばれる

- 双対の双対は元の主 LP
  - $\min\{yb \mid y \geq 0,\ yA \geq c\} =$
    $-\max\{-b^{\mathrm{T}}y^{\mathrm{T}} \mid y^{\mathrm{T}} \geq 0,\ -A^{\mathrm{T}}y^{\mathrm{T}} \leq -c^{\mathrm{T}}\}$

- LP の式の形には $x \geq 0$ が入ったり入らなかったり，$Ax \leq b$ が $Ax = b$ だったりいろいろなパターンがある
  - 上記の形だと双対をとるとき綺麗
  - 他の形のときは変数を置き換えるなどして対処できる

弱双対性 (weak duality)

- $\max \{ cx \mid x \geq 0,\ Ax \leq b \} \leq \min \{ yb \mid y \geq 0,\ yA \geq c \}$
  - ただし，両方が実行不可能な場合を除く

- 証明：
  - $x \geq 0$，$Ax \leq b$，$y \geq 0$，$yA \geq c$ とすると $cx \leq yAx \leq yb$

強双対性 (strong duality)

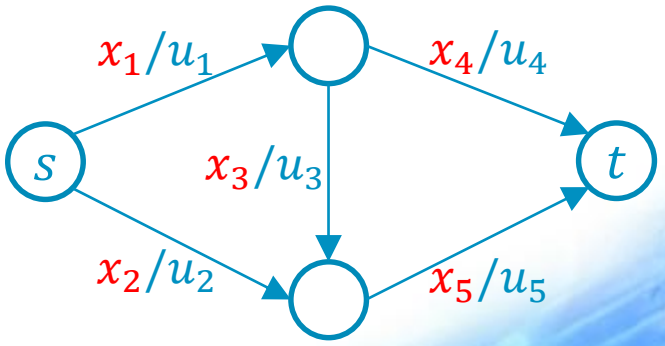
- $\max \{ cx \mid x \geq 0,\ Ax \leq b \} = \min \{ yb \mid y \geq 0,\ yA \geq c \}$
  - ただし，両方が実行不可能な場合を除く

- 証明：
  - 単体法 (simplex algorithm) を経由して示せる (そこそこ大変)

最大流と双対

- 最大流問題は LP で書ける

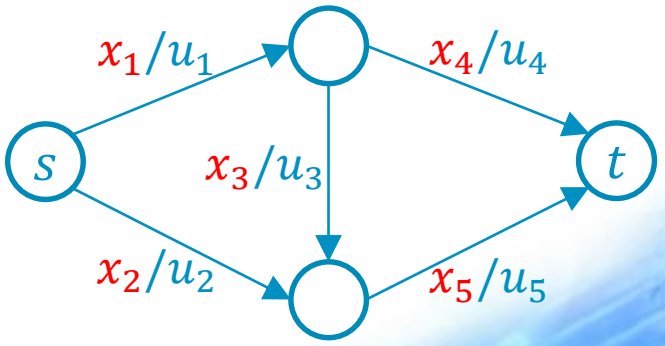
- maximize: $x_1 + x_2$
- subject to:
  - $x_1, x_2, x_3, x_4, x_5 \geq 0$
  - $x_1 \leq u_1$
  - $x_2 \leq u_2$
  - $x_3 \leq u_3$
  - $x_4 \leq u_4$
  - $x_5 \leq u_5$
  - $-x_1 + x_3 + x_4 = 0$
  - $-x_2 - x_3 + x_5 = 0$



最大流と双対

- 最大流問題は LP で書ける

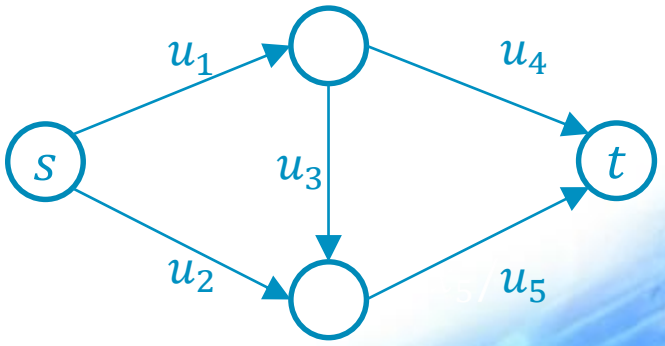
$$\max\left\{ \begin{pmatrix}1 & 1 & 0 & 0 & 0\end{pmatrix}x \;\middle|\; x \geq 0,\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 1 & 1 & 0 \\ 1 & 0 & -1 & -1 & 0 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 1 & 1 & 0 & -1 \end{pmatrix} x \leq \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \right\}$$



最大流と双対

- 双対をとってみる

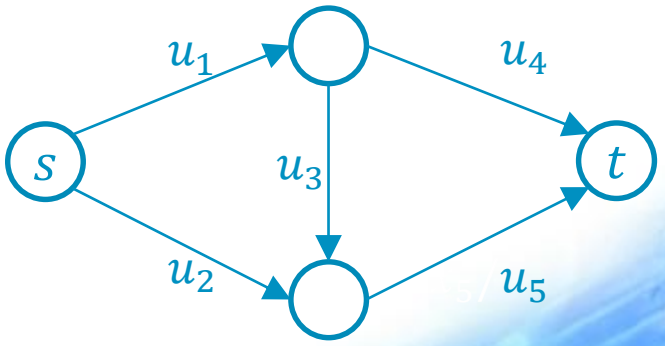
$$\min\left\{ y \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \middle| y \geq 0,\ y \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ -1 & 0 & 1 & 1 & 0 \\ 1 & 0 & -1 & -1 & 0 \\ 0 & -1 & -1 & 0 & 1 \\ 0 & 1 & 1 & 0 & -1 \end{pmatrix} \geq (1 \quad 1 \quad 0 \quad 0 \quad 0) \right\}$$



最大流と双対

- 双対をとってみる

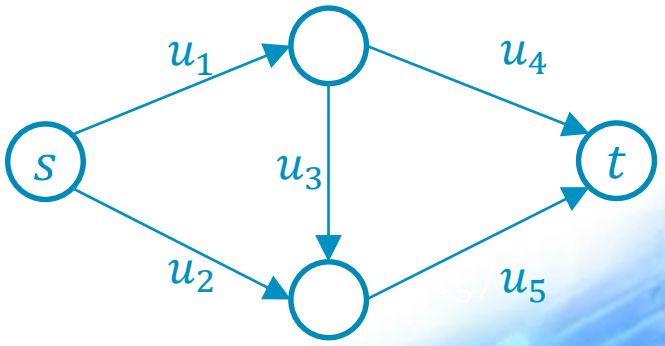
- minimize: $u_1y_1 + u_2y_2 + u_3y_3 + u_4y_4 + u_5y_5$
- subject to:
  - $y_1, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9 \geq 0$
  - $y_1 - y_6 + y_7 \geq 1$
  - $y_2 - y_8 + y_9 \geq 1$
  - $y_3 + y_6 - y_7 - y_8 + y_9 \geq 0$
  - $y_4 + y_6 - y_7 \geq 0$
  - $y_5 + y_8 - y_9 \geq 0$



最大流と双対

- 変数を置き換えてみる ((非負) – (非負) は任意の実数を動く)

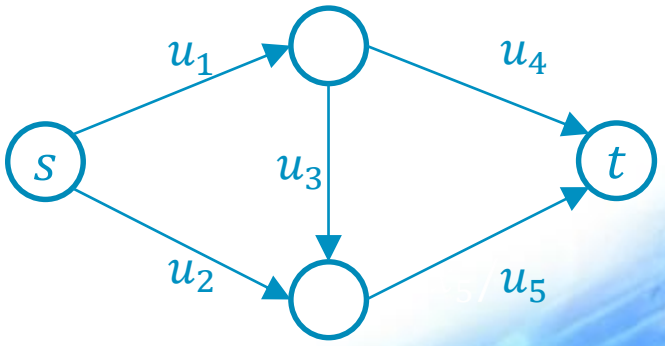
- minimize: $u_1y_1 + u_2y_2 + u_3y_3 + u_4y_4 + u_5y_5$
- subject to:
  - $y_1, y_2, y_3, y_4, y_5 \geq 0$
  - $y_1 \geq 1 - z_2$
  - $y_2 \geq 1 - z_3$
  - $y_3 \geq z_2 - z_3$
  - $y_4 \geq z_2 - 0$
  - $y_5 \geq z_3 - 0$



最大流と双対

- これはいったい？

- 各頂点 v に実数 z_v を決める
  - ただし $z_s = 1,\ z_t = 0$
- 各辺 $e = vw$ に対しては $y_e \geq 0$ かつ $y_e \geq z_v - z_w$ なる実数 y_e を決める
  - コストに $u_e y_e$ が加えられる
    - $y_e = \max\{0,\ z_v - z_w\}$ にしないと損
    - $1 \geq z_v \geq 0$ になっていないと損



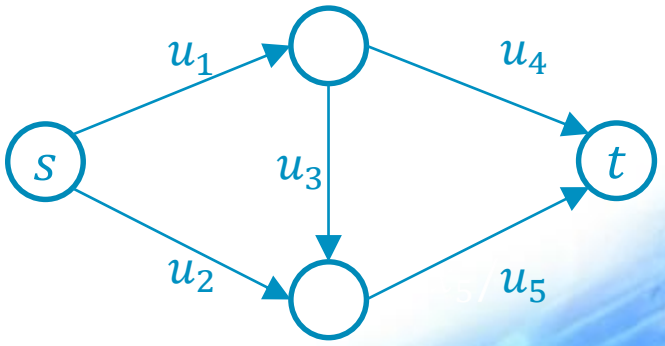
最大流と双対

- これはいったい？

- $z_v = 1$ または $z_v = 0$ になっているとしてよい
  - もし $1 > z_v > 0$ だとすると，z_v 以外の変数を固定して z_v を動かすと目標の値は 1 次式で変わるので，端で最適になる

- 各頂点に 1 か 0 を書く，始点は 1，終点は 0
- 1 から 0 への辺にはコストがかかる

→ これは最小カット！



最大流と双対

- 「s-t フローの流量が s-t カットの容量で上から抑えられる」は弱双対性に他ならない
- 最大フロー最小カット定理は強双対性に他ならない

- 最小カット問題に対応する LP は，必ず整数解を持つ例
  - 行列が完全ユニモジュラー

最小費用流の双対

- 最小費用流も LP で書ける
  - 最短路も最小費用流の特別な場合 (容量 1・要求流量 1) なので LP で書ける
  - 双対 LP をとってみると面白そうな謎の問題になります

- 逆に，一見よくわからない問題の双対 LP をとったら最小費用流になっていて解ける！みたいなこともあります

ネットワークフロー？と線型計画法

- よくわからない問題に出会った！
- とりあえず数式で条件を書いてみた！

- どうがんばっても 1 次式にならなさそうだったら最大流・最小費用流ではどうがんばっても解けません
- LP になっていたら
  - 最大流・最小費用流で解けるかもしれません
  - 双対をとってみると綺麗になるかもしれません

おわりに

演習問題：多項式時間で解いてください (有名問題)

1. 長方形のマス目があり，いくつかのマスには障害物が置かれている．将棋の飛車の駒を，互いに攻撃しあわないように，障害物のないマスに最大何個置けるか？ (飛車どうしは，同じ行か同じ列にあって間に障害物がないときに攻撃しあう)

2. 無向グラフが与えられる． 1 頂点以上からなる部分グラフであって，$\frac{(\text{辺数})}{(\text{頂点数})}$ が最大となるものを求めよ．

3. 連結な重み付き無向グラフ G と G のある全域木 T が与えられる．G の各辺の重みを修正して，T が最小全域木となるようにしたい．各辺の修正量の絶対値の和を最小化せよ．

演習問題のヒント (薄文字)

1. 二部グラフの最大マッチングに帰着させてください.

2. 実数 r に対して，$\frac{(\text{辺数})}{(\text{頂点数})} \geq r$ である部分グラフがとれるかどうか判定する問題を解いてください.

3. T の各辺が T に含まれない辺にとってかわられてしまうことがないという条件を LP で書いて双対をとってみてください.

参考文献

- Lecture Notes on Network Flow (Spring 2004)
  - David P. Williamson
  - http://people.orie.cornell.edu/dpw/orie633/
    - Cornell University の講義録 (1 学期間ずっとフロー)
    - 理論面の内容が非常に豊富
    - 特にスケーリングアルゴリズムについて詳しい

- プログラミングコンテストチャレンジブック
  - 秋葉拓哉，岩田陽一，北川宜稔
  - マイナビ，2012 (第 2 版)
    - 皆さんご存じ蟻本
    - 本講義と互いに補完という感じで参照してください

参考文献

- http://topcoder.g.hatena.ne.jp/Mi_Sawa/20140311
  - Dinic 法の計算量や注意点についての詳しい解説
- アルゴリズムデザイン
  - J. Kleinberg, E. Tardos
  - 共立出版，2008
    - 分厚くて大きい
    - フローの演習問題が豊富
- 組合せ最適化 理論とアルゴリズム
  - B. コルテ，J. フィーゲン
  - 丸善出版，2012 (第 2 版)
    - 黄色い
    - 特に線型計画法について詳しい

目次

http://hos.ac/slides/