オペレーションズ・リサーチ III (5)

田中 俊二

shunji.tanaka@okayama-u.ac.jp

スケジュール

No.	内容
1	導入 (組合せ最適化，グラフ・ネットワーク，整数計画問題)
2	計算複雑さの理論
3	グラフ・ネットワーク 1 (グラフの分類，用語，種々の問題)
4	グラフ・ネットワーク 2 (最短経路問題，動的計画法)
5	グラフ・ネットワーク 3 (最小全域木，最大フロー問題)
6	グラフ・ネットワーク 4 (マッチング)
7	整数計画 (緩和問題，分枝限定法，切除平面法)

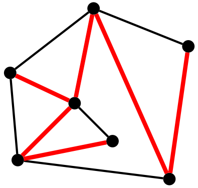
最小全域木

全域木 (spanning tree)

- すべての頂点を含む部分グラフのうち，木となっているもの
- $|E| = |V| - 1$ が成り立つ

最小全域木 (minimum spanning tree)

辺の重みの和が最小の全域木



全域木

最小全域木問題の解法

- クラスカル法 (Kruskal's algorithm)
- プリム法 (Prim's algorithm)

クラスカル法

クラスカル法 (Kruskal's algorithm)

- 閉路ができないよう，重みの小さい辺から順に選ぶ
- 同じ重みの辺の順序は任意

クラスカル法の擬似コード

```
procedure KRUSKAL(V, E, w)
    E の辺を重みの非減少順に並べ替える
    A ← ∅
    for all e ∈ E do                              ▷ 重みの非減少順にチェック
        if A に e を追加しても閉路ができない then
            A ← A ∪ {e}
    return A
```

- 閉路ができたかどうか，うまくチェックする必要がある (詳細略)
- 計算量は辺を並べ替える計算量で決まり，$O(|E| \log |E|)$

クラスカル法の正当性

最小全域木に辺を 1 本追加すると必ず閉路ができることに注意

クラスカル法の正当性

簡単のため，枝の重みはすべて異なるとする．$G' = (V, A')$ を最小全域木，クラスカル法で順に辺を求めるとして，初めて A' に含まれなかった辺を e，辺 e 以前にクラスカル法で求まった辺の集合を A とする．上の性質より，辺 e を G' に追加すると閉路ができる．

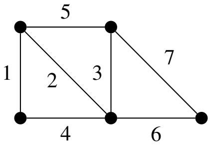
(a) 閉路上には A に含まれない辺 $e' \in A'$ が存在

もし閉路上のすべての辺が A に含まれるなら，クラスカル法で求まる $A \cup \{e\}$ も閉路を持つはず．

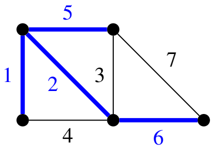
(b) $w(e') > w(e)$ が成り立つ

もし $w(e') < w(e)$ なら，辺 e' はクラスカル法で辺 e より先に候補に挙がったものの，閉路ができるため選ばれなかった辺．$A \subset A'$ だから，辺 e' より前にクラスカル法により選ばれた辺は A' にも含まれる．したがって，G' が閉路を持つことになる．

グラフ G' から辺 e' を取り除き，代わりに辺 e を追加したグラフは全域木で，重み和は G' よりも小さくなる．これは G' が最小全域木であることに矛盾．



A (クラスカル法)



$G' = (V, A')$

クラスカル法の正当性

最小全域木に辺を 1 本追加すると必ず閉路ができることに注意

クラスカル法の正当性

簡単のため，枝の重みはすべて異なるとする．$G' = (V, A')$ を最小全域木，クラスカル法で順に辺を求めるとして，初めて A' に含まれなかった辺を e，辺 e 以前にクラスカル法で求まった辺の集合を A とする．上の性質より，辺 e を G' に追加すると閉路ができる．

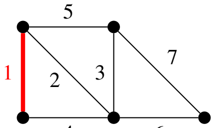
(a) 閉路上には A に含まれない辺 $e' \in A'$ が存在

もし閉路上のすべての辺が A に含まれるなら，クラスカル法で求まる $A \cup \{e\}$ も閉路を持つはず．

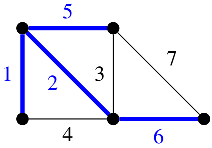
(b) $w(e') > w(e)$ が成り立つ

もし $w(e') < w(e)$ なら，辺 e' はクラスカル法で辺 e より先に候補に挙がったものの，閉路ができるため選ばれなかった辺．$A \subset A'$ だから，辺 e' より前にクラスカル法により選ばれた辺は A' にも含まれる．したがって，G' が閉路を持つことになる．

グラフ G' から辺 e' を取り除き，代わりに辺 e を追加したグラフは全域木で，重み和は G' よりも小さくなる．これは G' が最小全域木であることに矛盾．



A (クラスカル法)



$G' = (V, A')$

クラスカル法の正当性

最小全域木に辺を 1 本追加すると必ず閉路ができることに注意

クラスカル法の正当性

簡単のため，枝の重みはすべて異なるとする．$G' = (V, A')$ を最小全域木，クラスカル法で順に辺を求めるとして，初めて A' に含まれなかった辺を e，辺 e 以前にクラスカル法で求まった辺の集合を A とする．上の性質より，辺 e を G' に追加すると閉路ができる．

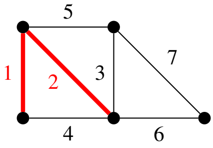
(a) 閉路上には A に含まれない辺 $e' \in A'$ が存在

もし閉路上のすべての辺が A に含まれるなら，クラスカル法で求まる $A \cup \{e\}$ も閉路を持つはず．

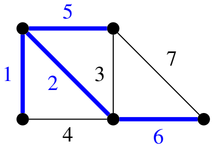
(b) $w(e') > w(e)$ が成り立つ

もし $w(e') < w(e)$ なら，辺 e' はクラスカル法で辺 e より先に候補に挙がったものの，閉路ができるため選ばれなかった辺．$A \subset A'$ だから，辺 e' より前にクラスカル法により選ばれた辺は A' にも含まれる．したがって，G' が閉路を持つことになる．

グラフ G' から辺 e' を取り除き，代わりに辺 e を追加したグラフは全域木で，重み和は G' よりも小さくなる．これは G' が最小全域木であることに矛盾．



A (クラスカル法)



$G' = (V, A')$

クラスカル法の正当性

最小全域木に辺を 1 本追加すると必ず閉路ができることに注意

クラスカル法の正当性

簡単のため，枝の重みはすべて異なるとする．$G' = (V, A')$ を最小全域木，クラスカル法で順に辺を求めるとして，**初めて A' に含まれなかった辺を e，**辺 e 以前にクラスカル法で求まった辺の集合を A とする．上の性質より，辺 e を G' に追加すると閉路ができる．

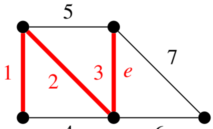
(a) 閉路上には A に含まれない辺 $e' \in A'$ が存在

もし閉路上のすべての辺が A に含まれるなら，クラスカル法で求まる $A \cup \{e\}$ も閉路を持つはず．

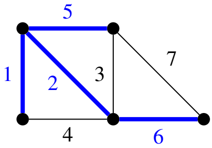
(b) $w(e') > w(e)$ が成り立つ

もし $w(e') < w(e)$ なら，辺 e' はクラスカル法で辺 e より先に候補に挙がったものの，閉路ができるため選ばれなかった辺．$A \subset A'$ だから，辺 e' より前にクラスカル法により選ばれた辺は A' にも含まれる．したがって，G' が閉路を持つことになる．

グラフ G' から辺 e' を取り除き，代わりに辺 e を追加したグラフは全域木で，重み和は G' よりも小さくなる．これは G' が最小全域木であることに矛盾．



A (クラスカル法)



$G' = (V, A')$

クラスカル法の正当性

最小全域木に辺を 1 本追加すると必ず閉路ができることに注意

クラスカル法の正当性

簡単のため，枝の重みはすべて異なるとする．$G' = (V, A')$ を最小全域木，クラスカル法で順に辺を求めるとして，**初めて A' に含まれなかった辺を e**，辺 e 以前にクラスカル法で求まった辺の集合を A とする．上の性質より，**辺 e を G' に追加すると閉路ができる**．

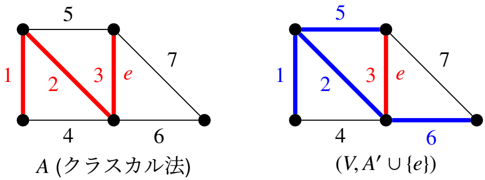
(a) 閉路上には A に含まれない辺 $e' \in A'$ が存在

もし閉路上のすべての辺が A に含まれるなら，クラスカル法で求まる $A \cup \{e\}$ も閉路を持つはず．

(b) $w(e') > w(e)$ が成り立つ

もし $w(e') < w(e)$ なら，辺 e' はクラスカル法で辺 e より先に候補に挙がったものの，閉路ができるため選ばれなかった辺．$A \subset A'$ だから，辺 e' より前にクラスカル法により選ばれた辺は A' にも含まれる．したがって，G' が閉路を持つことになる．

グラフ G' から辺 e' を取り除き，代わりに辺 e を追加したグラフは全域木で，重み和は G' よりも小さくなる．これは G' が最小全域木であることに矛盾．



クラスカル法の正当性

最小全域木に辺を 1 本追加すると必ず閉路ができることに注意

クラスカル法の正当性

簡単のため，枝の重みはすべて異なるとする．$G' = (V, A')$ を最小全域木，クラスカル法で順に辺を求めるとして，**初めて A' に含まれなかった辺を e**，辺 e 以前にクラスカル法で求まった辺の集合を A とする．上の性質より，**辺 e を G' に追加すると閉路ができる**．

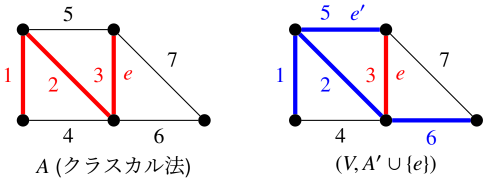
(a) **閉路上には A に含まれない辺 $e' \in A'$ が存在**

もし閉路上のすべての辺が A に含まれるなら，クラスカル法で求まる $A \cup \{e\}$ も閉路を持つはず．

(b) $w(e') > w(e)$ が成り立つ

もし $w(e') < w(e)$ なら，辺 e' はクラスカル法で辺 e より先に候補に挙がったものの，閉路ができるため選ばれなかった辺．$A \subset A'$ だから，辺 e' より前にクラスカル法により選ばれた辺は A' にも含まれる．したがって，G' が閉路を持つことになる．

グラフ G' から辺 e' を取り除き，代わりに辺 e を追加したグラフは全域木で，重み和は G' よりも小さくなる．これは G' が最小全域木であることに矛盾．



クラスカル法の正当性

最小全域木に辺を 1 本追加すると必ず閉路ができることに注意

クラスカル法の正当性

簡単のため，枝の重みはすべて異なるとする．$G' = (V, A')$ を最小全域木，クラスカル法で順に辺を求めるとして，**初めて A' に含まれなかった辺を e**，辺 e 以前にクラスカル法で求まった辺の集合を A とする．上の性質より，**辺 e を G' に追加すると閉路ができる**．

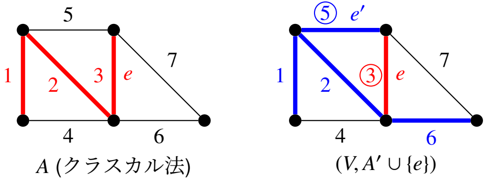
(a) **閉路上には A に含まれない辺 $e' \in A'$ が存在**

もし閉路上のすべての辺が A に含まれるなら，クラスカル法で求まる $A \cup \{e\}$ も閉路を持つはず．

(b) $w(e') > w(e)$ が成り立つ

もし $w(e') < w(e)$ なら，辺 e' はクラスカル法で辺 e より先に候補に挙がったものの，閉路ができるため選ばれなかった辺．$A \subset A'$ だから，辺 e' より前にクラスカル法により選ばれた辺は A' にも含まれる．したがって，G' が閉路を持つことになる．

グラフ G' から辺 e' を取り除き，代わりに辺 e を追加したグラフは全域木で，重み和は G' よりも小さくなる．これは G' が最小全域木であることに矛盾．



クラスカル法の正当性

最小全域木に辺を 1 本追加すると必ず閉路ができることに注意

クラスカル法の正当性

簡単のため，枝の重みはすべて異なるとする．$G' = (V, A')$ を最小全域木，クラスカル法で順に辺を求めるとして，**初めて A' に含まれなかった辺を e**，辺 e 以前にクラスカル法で求まった辺の集合を A とする．上の性質より，**辺 e を G' に追加すると閉路ができる**．

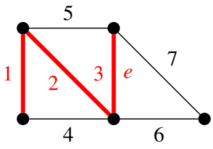
(a) **閉路上には A に含まれない辺 $e' \in A'$ が存在**

もし閉路上のすべての辺が A に含まれるなら，クラスカル法で求まる $A \cup \{e\}$ も閉路を持つはず．

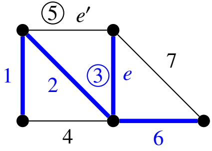
(b) $w(e') > w(e)$ が成り立つ

もし $w(e') < w(e)$ なら，辺 e' はクラスカル法で辺 e より先に候補に挙がったものの，閉路ができるため選ばれなかった辺．$A \subset A'$ だから，辺 e' より前にクラスカル法により選ばれた辺は A' にも含まれる．したがって，G' が閉路を持つことになる．

グラフ G' から辺 e' を取り除き，代わりに辺 e を追加したグラフは全域木で，重み和は G' よりも小さくなる．これは G' が最小全域木であることに矛盾．



A (クラスカル法)



$(V, A' \cup \{e\} \setminus \{e'\})$

プリム法 (Prim's algorithm)

- ダイクストラ法とほぼ同様
- 任意の頂点を始点として，重み最小 ($d(v)$ 最小) の辺が接続する頂点を順次追加

プリム法の擬似コード

procedure PRIM(V, E, w)
  for all $v \in V$ **do**
    $d(v) \leftarrow \infty$ ▷ $d(v)$ を ∞ で初期化
  適当に $s \in V$ を決める ▷ 木の根．始点
  $d(s) \leftarrow 0$ ▷ s から s までの辺の重みは 0
  $(S, A, Q) \leftarrow (\emptyset, \emptyset, \emptyset)$ ▷ 最小全域木の頂点 S・辺集合 A
  while $Q \neq \emptyset$ **do**
    $u \leftarrow$ (Q の頂点で d 最小のもの)
    $S \leftarrow S \cup \{u\}$ ▷ S に u を追加
    $A \leftarrow A \cup \{(p(u), u)\}$ ▷ A に辺 (p(u),u) を追加
    $Q \leftarrow Q \setminus \{u\}$ ▷ Q から u を削除
    for all $(u, v) \in E$ **do**
      if $v \notin S$ かつ $d(v) > w(u, v)$ **then**
        $d(v) \leftarrow w(u, v)$ ▷ 最小重みの更新
        $p(v) \leftarrow u$ ▷ v の直前の頂点
        if $v \notin Q$ **then**
          $Q \leftarrow Q \cup \{v\}$ ▷ u から到達可能な v を Q に追加
  return A

プリム法

プリム法 (Prim's algorithm)

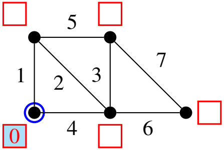
- ダイクストラ法とほぼ同様
- 任意の頂点を始点として，重み最小 ($d(v)$ 最小) の辺が接続する頂点を順次追加

プリム法の擬似コード

procedure Prim(V, E, w)
    for all $v \in V$ **do**
        $d(v) \leftarrow \infty$ ▷ $d(v)$ を ∞ で初期化
    適当に $s \in V$ を決める ▷ 木の根．始点
    $d(s) \leftarrow 0$ ▷ s から s までの辺の重みは 0
    $(S, A, Q) \leftarrow (\emptyset, \emptyset, \emptyset)$ ▷ 最小全域木の頂点 S・辺集合 A
    while $Q \neq \emptyset$ **do**
        $u \leftarrow$ (Q の頂点で d 最小のもの)
        $S \leftarrow S \cup \{u\}$ ▷ S に u を追加
        $A \leftarrow A \cup \{(p(u), u)\}$ ▷ A に辺 (p(u),u) を追加
        $Q \leftarrow Q \setminus \{u\}$ ▷ Q から u を削除
        for all $(u, v) \in E$ **do**
            if $v \notin S$ かつ $d(v) > w(u, v)$ **then**
                $d(v) \leftarrow w(u, v)$ ▷ 最小重みの更新
                $p(v) \leftarrow u$ ▷ v の直前の頂点
                if $v \notin Q$ **then**
                    $Q \leftarrow Q \cup \{v\}$ ▷ u から到達可能な v を Q に追加
    return A

ダイクストラ法の場合
$d(u) + w(u, v)$

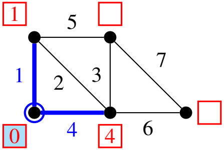
プリム法の例



1 回目の反復

$d(v)$ (四角内の数字) は最短距離ではなく最小重みを表す

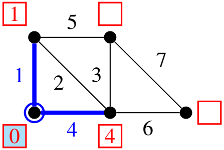
プリム法の例



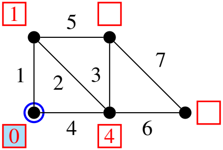
1 回目の反復

$d(v)$ (四角内の数字) は最短距離ではなく最小重みを表す

プリム法の例



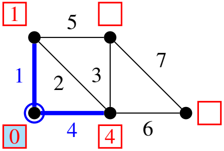
1 回目の反復



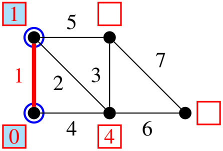
2 回目の反復

$d(v)$ (四角内の数字) は最短距離ではなく最小重みを表す

プリム法の例



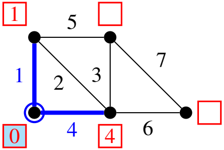
1 回目の反復



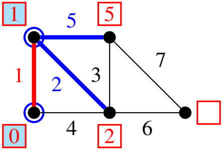
2 回目の反復

$d(v)$ (四角内の数字) は最短距離ではなく最小重みを表す

プリム法の例



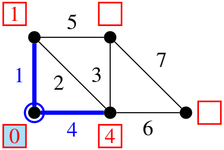
1 回目の反復



2 回目の反復

$d(v)$ (四角内の数字) は最短距離ではなく最小重みを表す

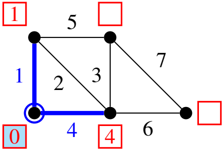
プリム法の例



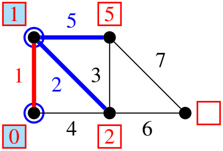
1 回目の反復　2 回目の反復　3 回目の反復

$d(v)$ (四角内の数字) は最短距離ではなく最小重みを表す

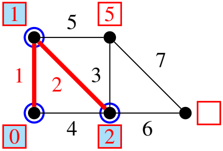
プリム法の例



1 回目の反復



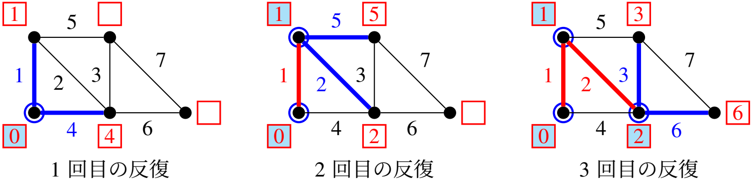
2 回目の反復



3 回目の反復

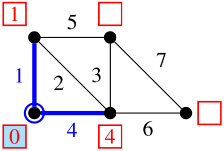
$d(v)$ (四角内の数字) は最短距離ではなく最小重みを表す

プリム法の例

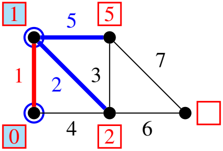


$d(v)$ (四角内の数字) は最短距離ではなく最小重みを表す

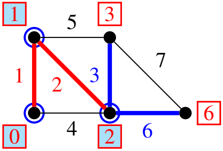
プリム法の例



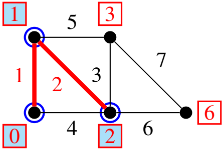
1 回目の反復



2 回目の反復



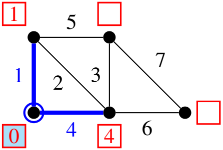
3 回目の反復



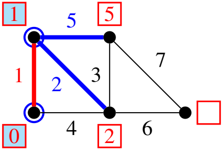
4 回目の反復

$d(v)$ (四角内の数字) は最短距離ではなく最小重みを表す

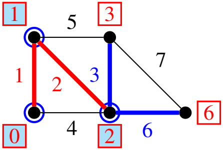
プリム法の例



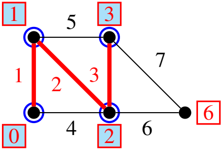
1 回目の反復



2 回目の反復



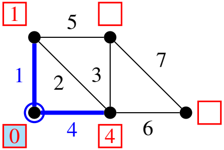
3 回目の反復



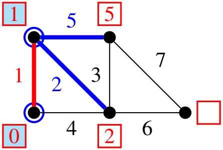
4 回目の反復

$d(v)$ (四角内の数字) は最短距離ではなく最小重みを表す

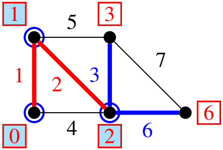
プリム法の例



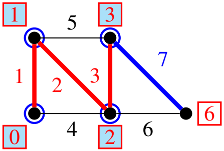
1 回目の反復



2 回目の反復



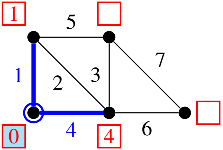
3 回目の反復



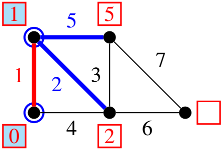
4 回目の反復

$d(v)$ (四角内の数字) は最短距離ではなく最小重みを表す

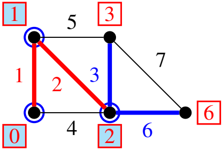
プリム法の例



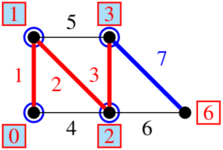
1 回目の反復



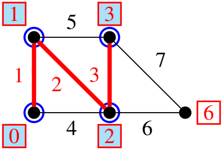
2 回目の反復



3 回目の反復



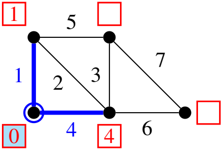
4 回目の反復



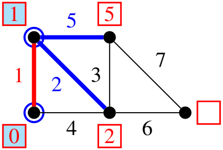
5 回目の反復

$d(v)$ (四角内の数字) は最短距離ではなく最小重みを表す

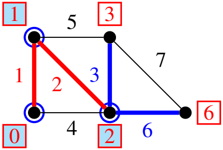
プリム法の例



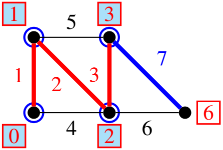
1 回目の反復



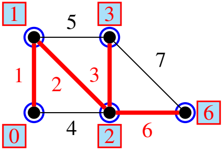
2 回目の反復



3 回目の反復



4 回目の反復



5 回目の反復

d(*v*) (四角内の数字) は最短距離ではなく最小重みを表す

プリム法の正当性

プリム法の正当性

クラスカル法と同様.

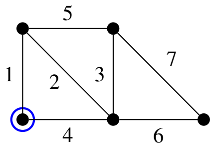
- プリム法で初めて選ばれた，最小全域木 $G' = (V, A')$ と異なる辺 $e = (v, u)$
- e 以前にプリム法で求まった頂点集合 S および辺集合 A

(a) G' の誘導部分グラフ $G'' = (S, A'')$ は，$A \subset A''$ より連結．また，$|A''| > |A| = |S| - 1$ なら閉路ができ，この閉路は G' にも含まれる．したがって，$A'' = A$

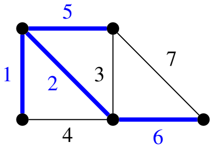
(b) e を G' に追加した際にできる閉路上の辺 $e' = (v', u') \notin A$ で，$v' \in S$ かつ $u' \in V \setminus S$ を満たすものが存在 (プリム法による全域木において，S と $V \setminus S$ を接続する辺は e のみ．$e \notin A'$ だから，G' には S と $V \setminus S$ を接続する $e' \neq e$ が存在する)

(c) $w(e') > w(e)$ ($w(e') < w(e)$ なら，プリム法で e' は e より先に選ばれていたはず)

(d) グラフ G' から辺 e' を取り除き，代わりに辺 e を追加したグラフは全域木で，重み和は G' よりも小さくなる．これは G' が最小全域木であることに矛盾



A (プリム法)



$G' = (V, A')$

プリム法の正当性

プリム法の正当性

クラスカル法と同様.

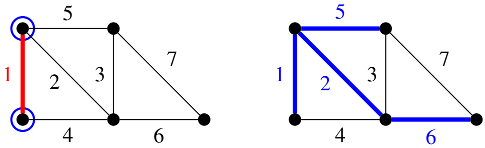
- プリム法で初めて選ばれた，最小全域木 $G' = (V, A')$ と異なる辺 $e = (v, u)$
- e 以前にプリム法で求まった頂点集合 S および辺集合 A

(a) G' の誘導部分グラフ $G'' = (S, A'')$ は，$A \subset A''$ より連結．また，$|A''| > |A| = |S| - 1$ なら閉路ができ，この閉路は G' にも含まれる．したがって，$A'' = A$

(b) e を G' に追加した際にできる閉路上の辺 $e' = (v', u') \notin A$ で，$v' \in S$ かつ $u' \in V \setminus S$ を満たすものが存在 (プリム法による全域木において，S と $V \setminus S$ を接続する辺は e のみ．$e \notin A'$ だから，G' には S と $V \setminus S$ を接続する $e' \neq e$ が存在する)

(c) $w(e') > w(e)$ ($w(e') < w(e)$ なら，プリム法で e' は e より先に選ばれていたはず)

(d) グラフ G' から辺 e' を取り除き，代わりに辺 e を追加したグラフは全域木で，重み和は G' よりも小さくなる．これは G' が最小全域木であることに矛盾



A (プリム法)　　$G' = (V, A')$

プリム法の正当性

プリム法の正当性

クラスカル法と同様.

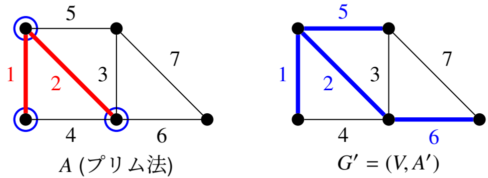
- プリム法で初めて選ばれた，最小全域木 $G' = (V, A')$ と異なる辺 $e = (v, u)$
- e 以前にプリム法で求まった頂点集合 S および辺集合 A

(a) G' の誘導部分グラフ $G'' = (S, A'')$ は，$A \subset A''$ より連結．また，$|A''| > |A| = |S| - 1$ なら閉路ができ，この閉路は G' にも含まれる．したがって，$A'' = A$

(b) e を G' に追加した際にできる閉路上の辺 $e' = (v', u') \notin A$ で，$v' \in S$ かつ $u' \in V \setminus S$ を満たすものが存在 (プリム法による全域木において，S と $V \setminus S$ を接続する辺は e のみ．$e \notin A'$ だから，G' には S と $V \setminus S$ を接続する $e' \neq e$ が存在する)

(c) $w(e') > w(e)$ ($w(e') < w(e)$ なら，プリム法で e' は e より先に選ばれていたはず)

(d) グラフ G' から辺 e' を取り除き，代わりに辺 e を追加したグラフは全域木で，重み和は G' よりも小さくなる．これは G' が最小全域木であることに矛盾



プリム法の正当性

プリム法の正当性

クラスカル法と同様.

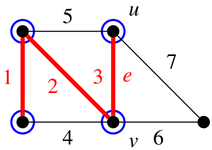
- プリム法で初めて選ばれた，最小全域木 $G' = (V, A')$ と異なる**辺** $e = (v, u)$
- e 以前にプリム法で求まった頂点集合 S および辺集合 A

(a) G' の誘導部分グラフ $G'' = (S, A'')$ は，$A \subset A''$ より連結．また，$|A''| > |A| = |S| - 1$ なら閉路ができ，この閉路は G' にも含まれる．したがって，$A'' = A$

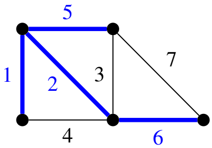
(b) e を G' に追加した際にできる閉路上の辺 $e' = (v', u') \notin A$ で，$v' \in S$ かつ $u' \in V \setminus S$ を満たすものが存在 (プリム法による全域木において，S と $V \setminus S$ を接続する辺は e のみ．$e \notin A'$ だから，G' には S と $V \setminus S$ を接続する $e' \neq e$ が存在する)

(c) $w(e') > w(e)$ ($w(e') < w(e)$ なら，プリム法で e' は e より先に選ばれていたはず)

(d) グラフ G' から辺 e' を取り除き，代わりに辺 e を追加したグラフは全域木で，重み和は G' よりも小さくなる．これは G' が最小全域木であることに矛盾



A (プリム法)



$G' = (V, A')$

プリム法の正当性

プリム法の正当性

クラスカル法と同様.

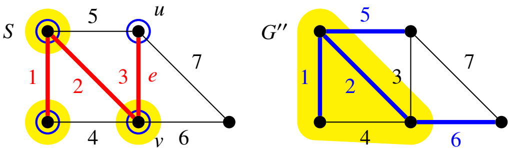
- プリム法で初めて選ばれた，最小全域木 $G' = (V, A')$ と異なる**辺** $e = (v, u)$
- e 以前にプリム法で求まった頂点集合 S および辺集合 A

(a) G' の**誘導部分グラフ** $G'' = (S, A'')$ は，$A \subset A''$ より連結．また，$|A''| > |A| = |S| - 1$ なら閉路ができ，この閉路は G' にも含まれる．したがって，$A'' = A$

(b) e を G' に追加した際にできる閉路上の辺 $e' = (v', u') \notin A$ で，$v' \in S$ かつ $u' \in V \setminus S$ を満たすものが存在 (プリム法による全域木において，S と $V \setminus S$ を接続する辺は e のみ．$e \notin A'$ だから，G' には S と $V \setminus S$ を接続する $e' \neq e$ が存在する)

(c) $w(e') > w(e)$ ($w(e') < w(e)$ なら，プリム法で e' は e より先に選ばれていたはず)

(d) グラフ G' から辺 e' を取り除き，代わりに辺 e を追加したグラフは全域木で，重み和は G' よりも小さくなる．これは G' が最小全域木であることに矛盾



A (プリム法)　　$G'' = (S, A'')$

プリム法の正当性

プリム法の正当性

クラスカル法と同様.

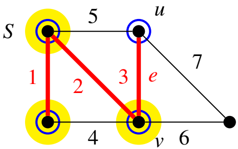
- プリム法で初めて選ばれた，最小全域木 $G' = (V, A')$ と異なる**辺** $e = (v, u)$
- e 以前にプリム法で求まった頂点集合 S および辺集合 A

(a) G' の**誘導部分グラフ** $G'' = (S, A'')$ は，$A \subset A''$ より連結．また，$|A''| > |A| = |S| - 1$ なら閉路ができ，この閉路は G' にも含まれる．したがって，$A'' = A$

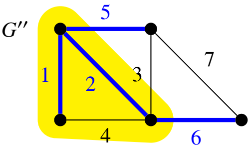
(b) e を G' に追加した際にできる閉路上の辺 $e' = (v', u') \notin A$ で，$v' \in S$ かつ $u' \in V \setminus S$ を満たすものが存在 (プリム法による全域木において，S と $V \setminus S$ を接続する辺は e のみ．$e \notin A'$ だから，G' には S と $V \setminus S$ を接続する $e' \neq e$ が存在する)

(c) $w(e') > w(e)$ ($w(e') < w(e)$ なら，プリム法で e' は e より先に選ばれていたはず)

(d) グラフ G' から辺 e' を取り除き，代わりに辺 e を追加したグラフは全域木で，重み和は G' よりも小さくなる．これは G' が最小全域木であることに矛盾



A (プリム法)



$G'' = (S, A)$

プリム法の正当性

プリム法の正当性

クラスカル法と同様.

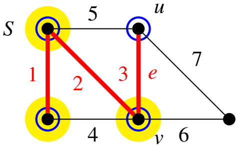
- プリム法で初めて選ばれた，最小全域木 $G' = (V, A')$ と異なる**辺** $e = (v, u)$
- e 以前にプリム法で求まった頂点集合 S および辺集合 A

(a) G' の**誘導部分グラフ** $G'' = (S, A'')$ は，$A \subset A''$ より連結．また，$|A''| > |A| = |S| - 1$ なら閉路ができ，この閉路は G' にも含まれる．したがって，$A'' = A$

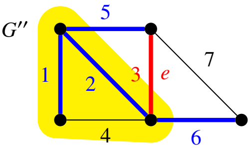
(b) e を G' に追加した際にできる閉路上の辺 $e' = (v', u') \notin A$ で，$v' \in S$ かつ $u' \in V \setminus S$ を満たすものが存在 (プリム法による全域木において，S と $V \setminus S$ を接続する辺は e のみ．$e \notin A'$ だから，G' には S と $V \setminus S$ を接続する $e' \neq e$ が存在する)

(c) $w(e') > w(e)$ ($w(e') < w(e)$ なら，プリム法で e' は e より先に選ばれていたはず)

(d) グラフ G' から辺 e' を取り除き，代わりに辺 e を追加したグラフは全域木で，重み和は G' よりも小さくなる．これは G' が最小全域木であることに矛盾



A (プリム法)



$G'' = (S, A)$

プリム法の正当性

プリム法の正当性

クラスカル法と同様.

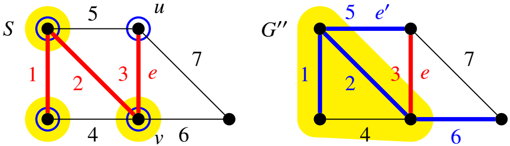
- プリム法で初めて選ばれた，最小全域木 $G' = (V, A')$ と異なる**辺** $e = (v, u)$
- e 以前にプリム法で求まった頂点集合 S および辺集合 A

(a) G' の**誘導部分グラフ** $G'' = (S, A'')$ は，$A \subset A''$ より連結．また，$|A''| > |A| = |S| - 1$ なら閉路ができ，この閉路は G' にも含まれる．したがって，$A'' = A$

(b) e を G' に追加した際にできる閉路上の**辺** $e' = (v', u') \notin A$ で，$v' \in S$ かつ $u' \in V \setminus S$ を満たすものが存在 (プリム法による全域木において，S と $V \setminus S$ を接続する辺は e のみ．$e \notin A'$ だから，G' には S と $V \setminus S$ を接続する $e' \neq e$ が存在する)

(c) $w(e') > w(e)$ ($w(e') < w(e)$ なら，プリム法で e' は e より先に選ばれていたはず)

(d) グラフ G' から辺 e' を取り除き，代わりに辺 e を追加したグラフは全域木で，重み和は G' よりも小さくなる．これは G' が最小全域木であることに矛盾



A (プリム法)　　$G'' = (S, A)$

プリム法の正当性

プリム法の正当性

クラスカル法と同様.

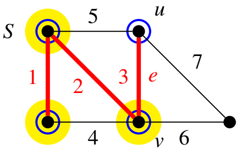
- プリム法で初めて選ばれた，最小全域木 $G' = (V, A')$ と異なる**辺** $e = (v, u)$
- e 以前にプリム法で求まった頂点集合 S および辺集合 A

(a) G' の**誘導部分グラフ** $G'' = (S, A'')$ は，$A \subset A''$ より連結．また，$|A''| > |A| = |S| - 1$ なら閉路ができ，この閉路は G' にも含まれる．したがって，$A'' = A$

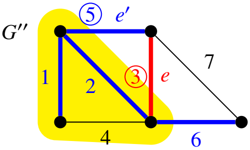
(b) e を G' に追加した際にできる閉路上の**辺** $e' = (v', u') \notin A$ で，$v' \in S$ かつ $u' \in V \setminus S$ を満たすものが存在 (プリム法による全域木において，S と $V \setminus S$ を接続する辺は e のみ．$e \notin A'$ だから，G' には S と $V \setminus S$ を接続する $e' \neq e$ が存在する)

(c) $w(e') > w(e)$ ($w(e') < w(e)$ なら，プリム法で e' は e より先に選ばれていたはず)

(d) グラフ G' から辺 e' を取り除き，代わりに辺 e を追加したグラフは全域木で，重み和は G' よりも小さくなる．これは G' が最小全域木であることに矛盾



A (プリム法)



$G'' = (S, A)$

プリム法の正当性

プリム法の正当性

クラスカル法と同様.

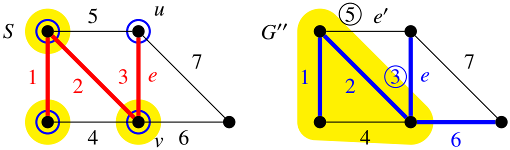
- プリム法で初めて選ばれた，最小全域木 $G' = (V, A')$ と異なる**辺** $e = (v, u)$
- e 以前にプリム法で求まった頂点集合 S および辺集合 A

(a) G' の**誘導部分グラフ** $G'' = (S, A'')$ は，$A \subset A''$ より連結．また，$|A''| > |A| = |S| - 1$ なら閉路ができ，この閉路は G' にも含まれる．したがって，$A'' = A$

(b) e を G' に追加した際にできる閉路上の**辺** $e' = (v', u') \notin A$ で，$v' \in S$ かつ $u' \in V \setminus S$ を満たすものが存在 (プリム法による全域木において，S と $V \setminus S$ を接続する辺は e のみ．$e \notin A'$ だから，G' には S と $V \setminus S$ を接続する $e' \neq e$ が存在する)

(c) $w(e') > w(e)$ ($w(e') < w(e)$ なら，プリム法で e' は e より先に選ばれていたはず)

(d) **グラフ G' から辺 e' を取り除き，代わりに辺 e を追加したグラフ**は全域木で，重み和は G' よりも小さくなる．これは G' が最小全域木であることに矛盾



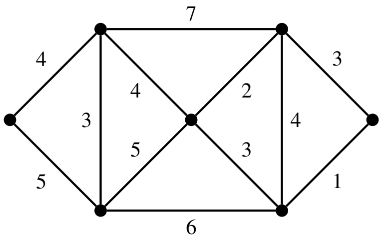
A (プリム法)　　$(V, A' \cup \{e\} \setminus \{e'\})$

プリム法：補足

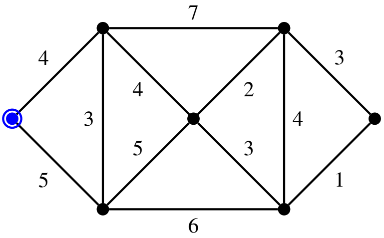
プリム法：補足

- プリム (Robert Clay Prim) が 1957 年に，ダイクストラ法のダイクストラ (Edsger Wybe Dijkstra) が 1959 年に，それぞれ独立にこのアルゴリズムを発表
- それよりも先，1930 年にヤルニク (Vojtěch Jarník) が提案していた
- このため，ヤルニク・プリム法，DJP 法などと呼ばれることもある

最小全域木の練習問題

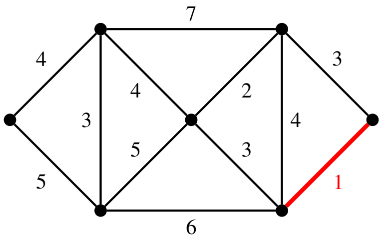


クラスカル法

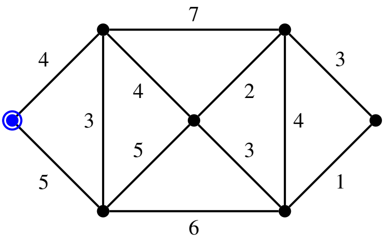


プリム法

最小全域木の練習問題

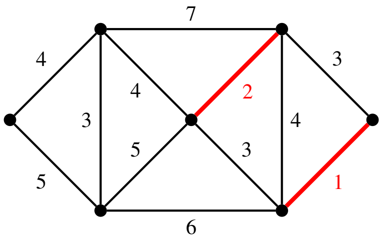


クラスカル法

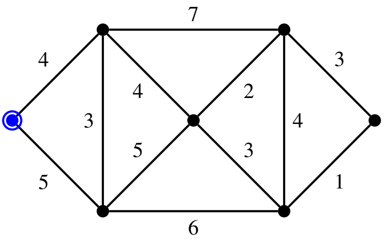


プリム法

最小全域木の練習問題

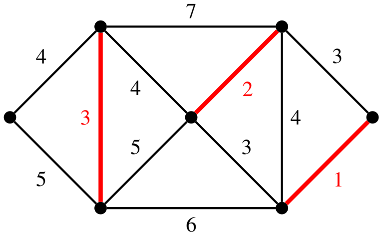


クラスカル法

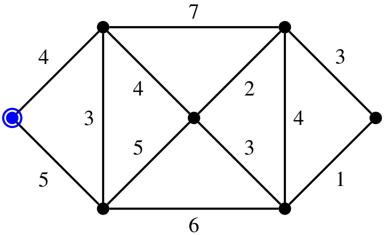


プリム法

最小全域木の練習問題

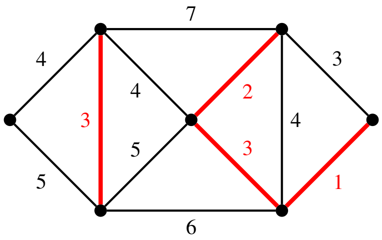


クラスカル法

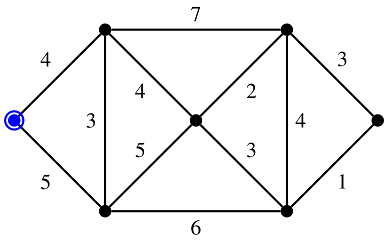


プリム法

最小全域木の練習問題

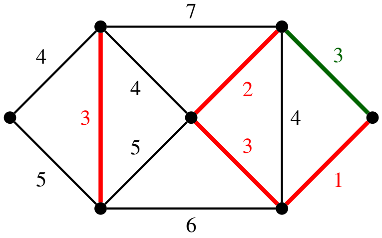


クラスカル法

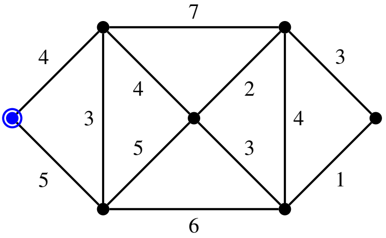


プリム法

最小全域木の練習問題

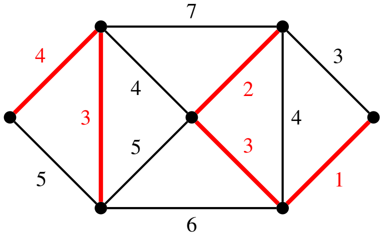


クラスカル法

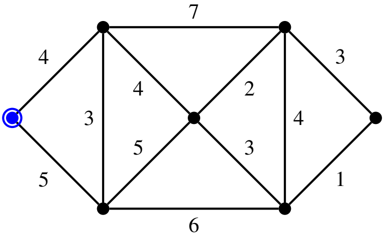


プリム法

最小全域木の練習問題

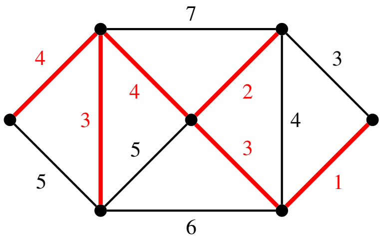


クラスカル法

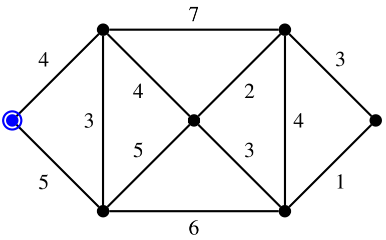


プリム法

最小全域木の練習問題

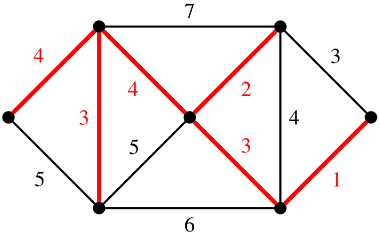


クラスカル法

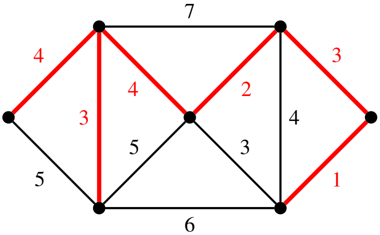


プリム法

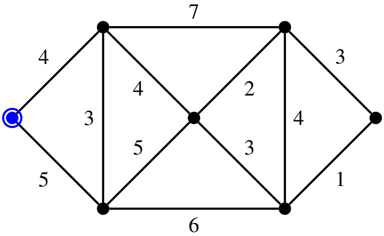
最小全域木の練習問題



クラスカル法

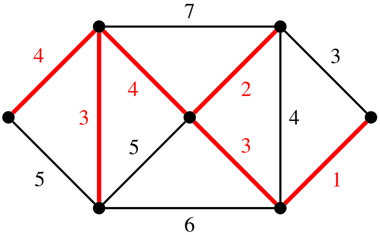


クラスカル法の別解

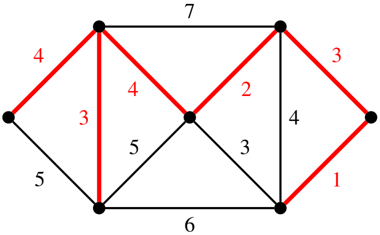


プリム法

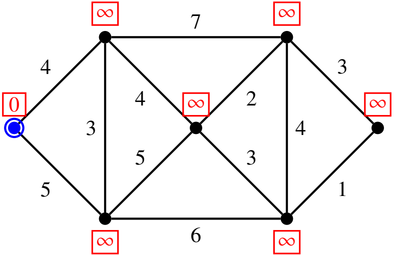
最小全域木の練習問題



クラスカル法

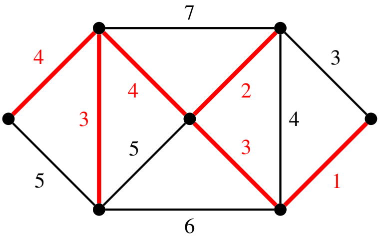


クラスカル法の別解

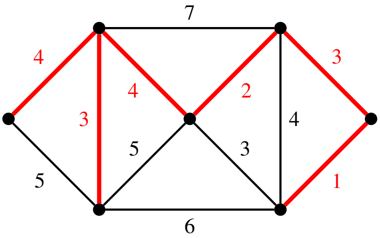


プリム法

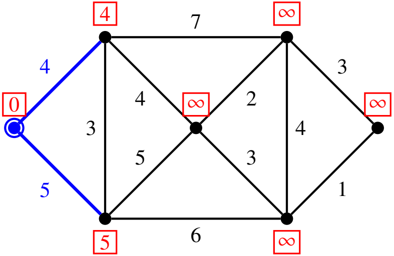
最小全域木の練習問題



クラスカル法

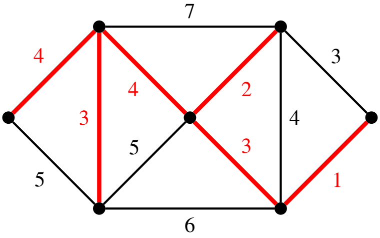


クラスカル法の別解

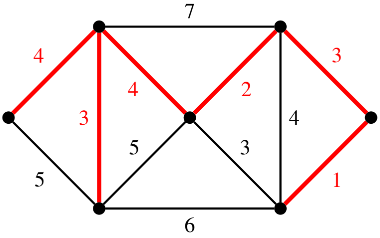


プリム法

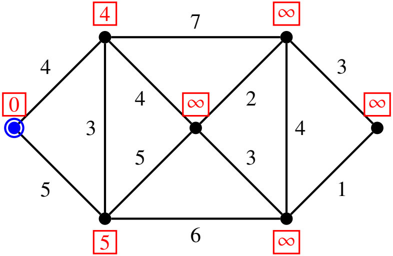
最小全域木の練習問題



クラスカル法

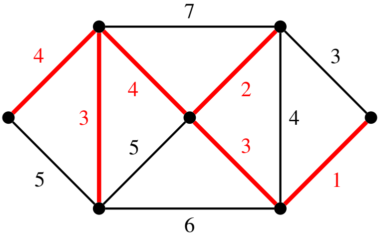


クラスカル法の別解

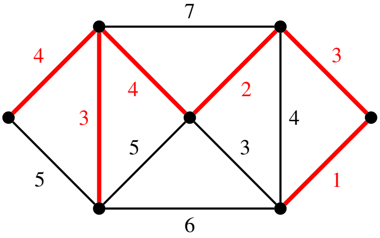


プリム法

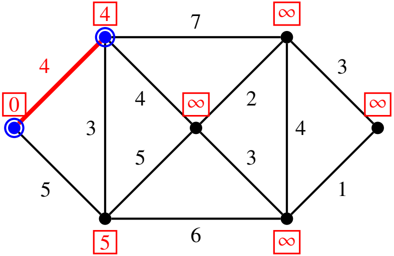
最小全域木の練習問題



クラスカル法

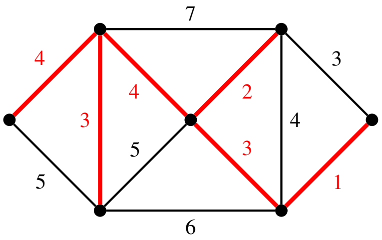


クラスカル法の別解

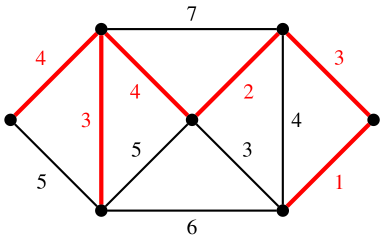


プリム法

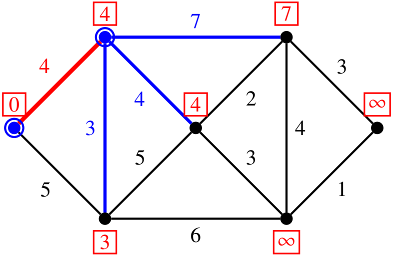
最小全域木の練習問題



クラスカル法

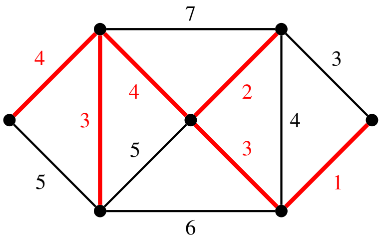


クラスカル法の別解

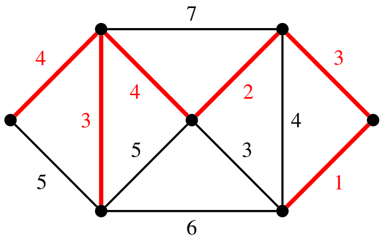


プリム法

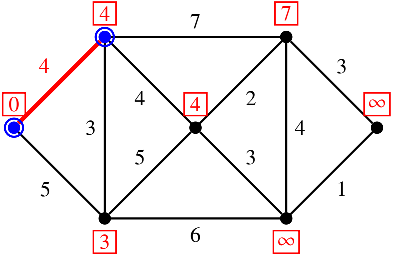
最小全域木の練習問題



クラスカル法

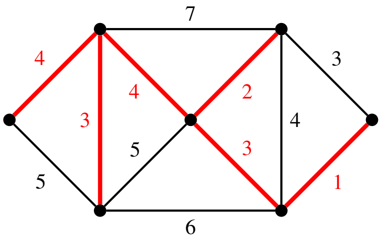


クラスカル法の別解

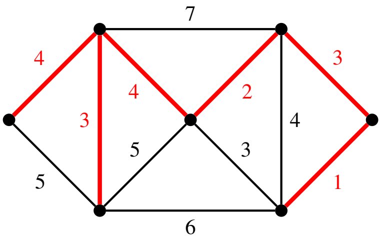


プリム法

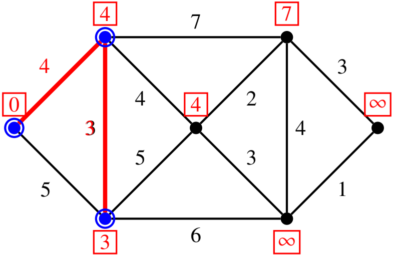
最小全域木の練習問題



クラスカル法

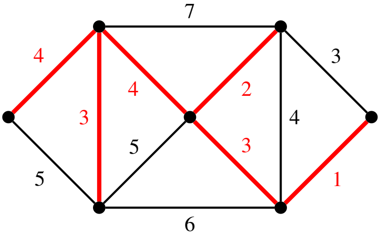


クラスカル法の別解

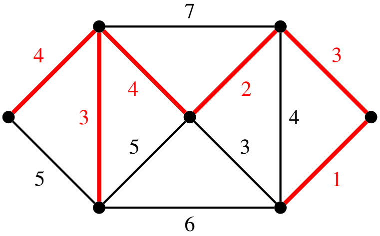


プリム法

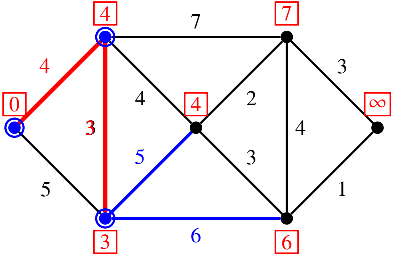
最小全域木の練習問題



クラスカル法

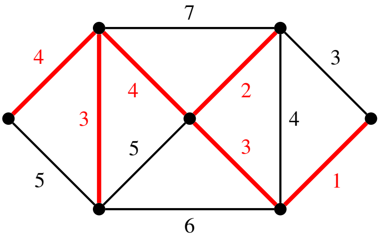


クラスカル法の別解

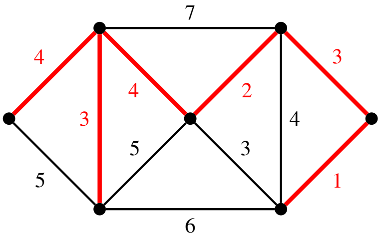


プリム法

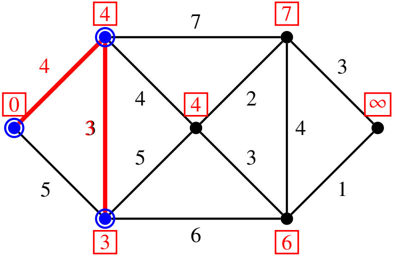
最小全域木の練習問題



クラスカル法

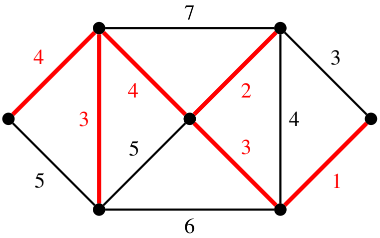


クラスカル法の別解

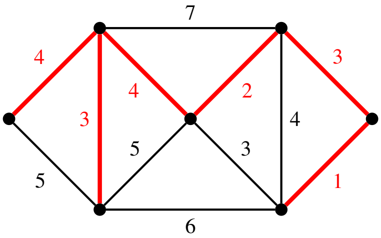


プリム法

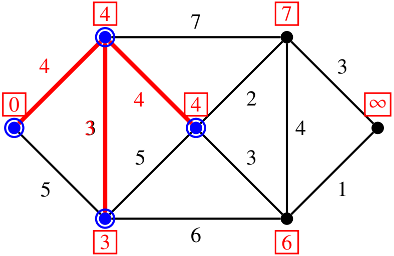
最小全域木の練習問題



クラスカル法

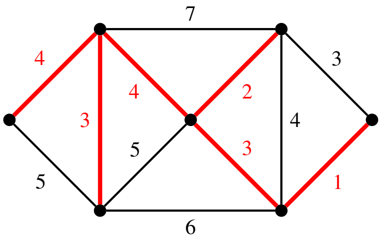


クラスカル法の別解

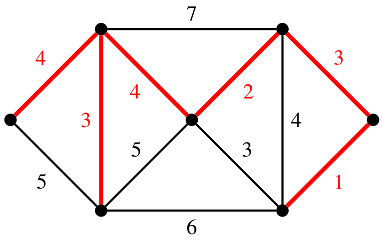


プリム法

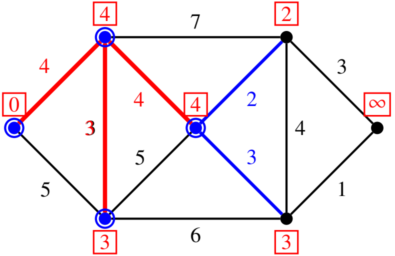
最小全域木の練習問題



クラスカル法

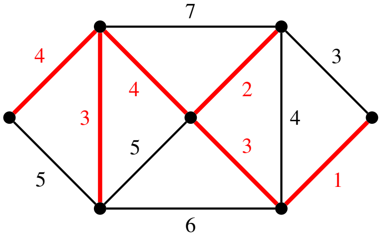


クラスカル法の別解

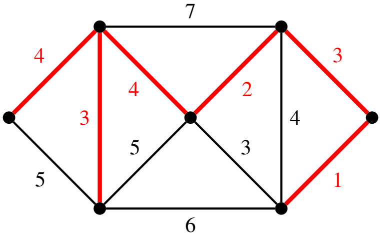


プリム法

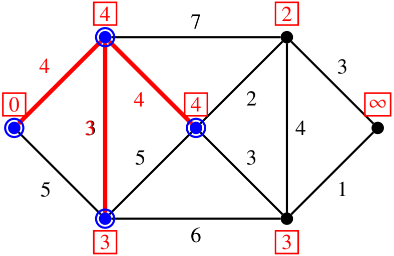
最小全域木の練習問題



クラスカル法

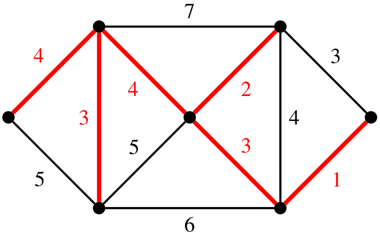


クラスカル法の別解

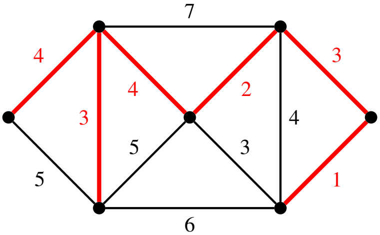


プリム法

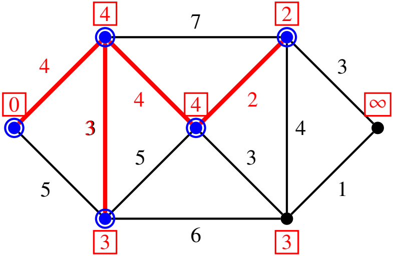
最小全域木の練習問題



クラスカル法

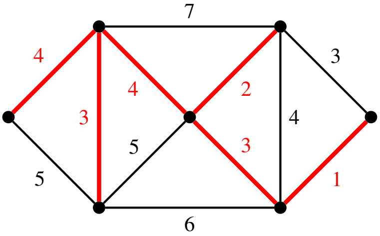


クラスカル法の別解

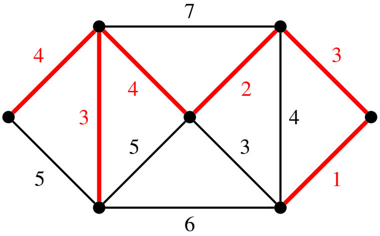


プリム法

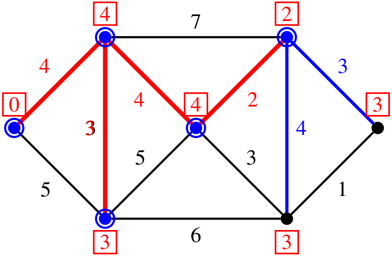
最小全域木の練習問題



クラスカル法

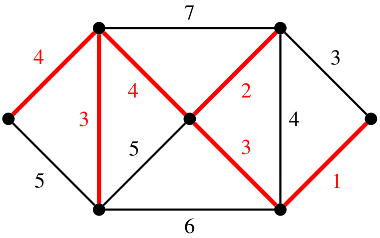


クラスカル法の別解

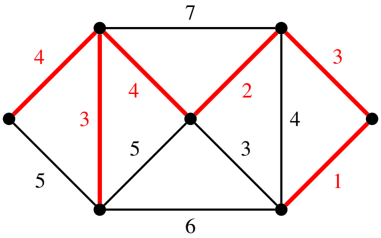


プリム法

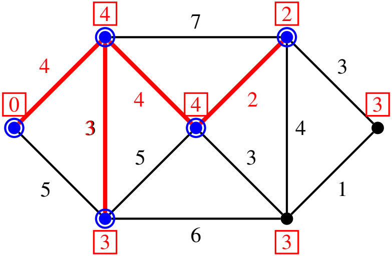
最小全域木の練習問題



クラスカル法

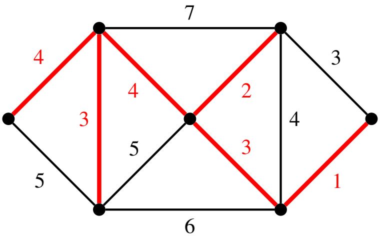


クラスカル法の別解

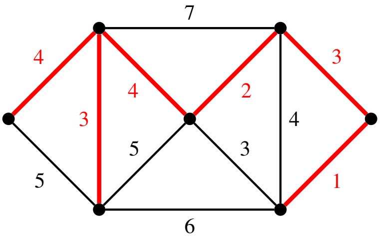


プリム法

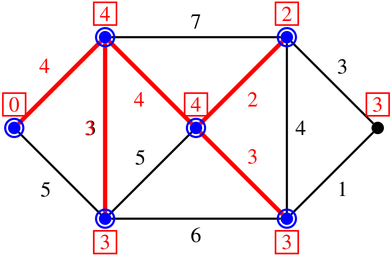
最小全域木の練習問題



クラスカル法

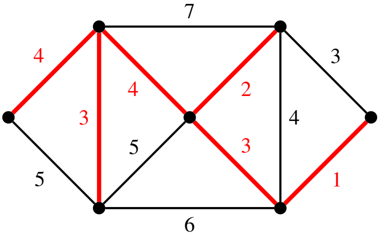


クラスカル法の別解

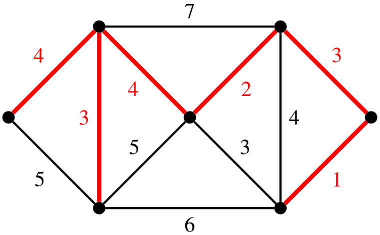


プリム法

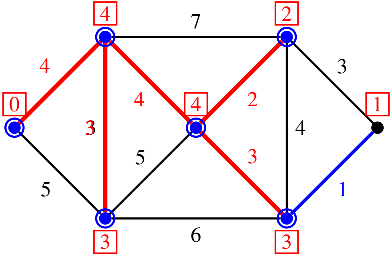
最小全域木の練習問題



クラスカル法

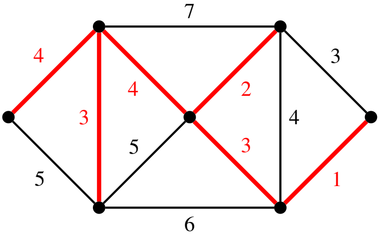


クラスカル法の別解

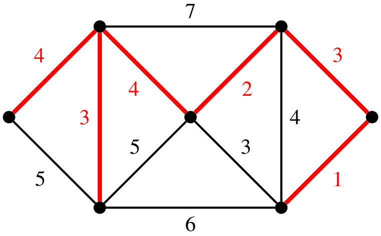


プリム法

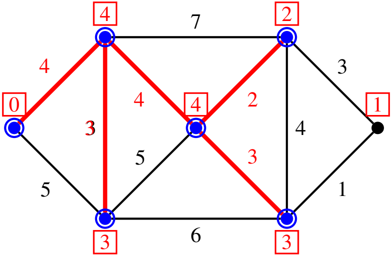
最小全域木の練習問題



クラスカル法

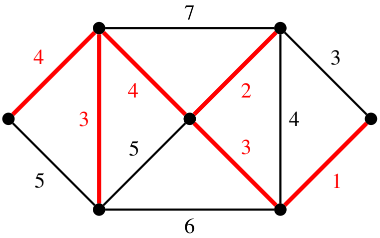


クラスカル法の別解

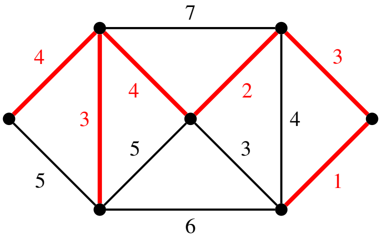


プリム法

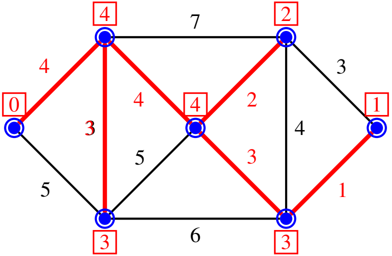
最小全域木の練習問題



クラスカル法

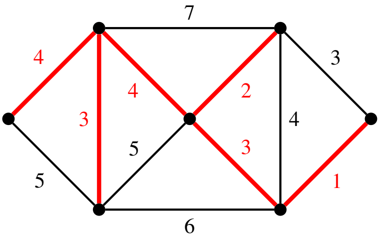


クラスカル法の別解

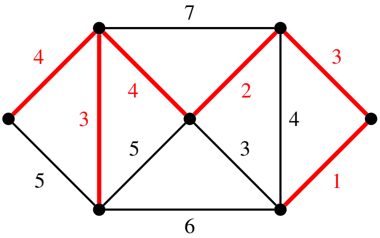


プリム法

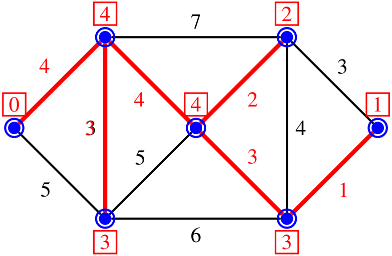
最小全域木の練習問題



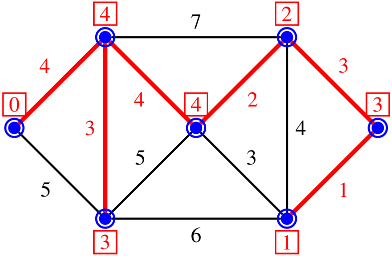
クラスカル法



クラスカル法の別解



プリム法

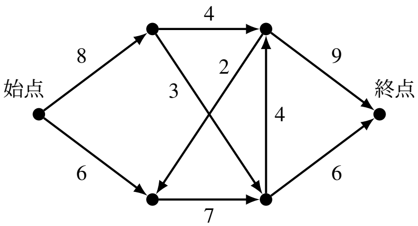


プリム法の別解

最大フロー問題

最大フロー問題 (maximum flow problem)

- 始点 (source) から終点 (sink) までのフロー (モノの流れ) を最大化する問題
- 辺の重み：辺に流せる最大流量・容量 (capacity)

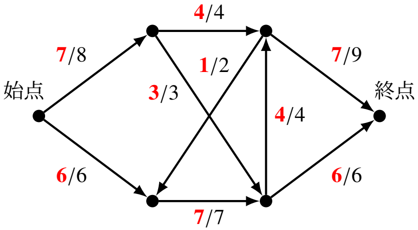


- 有向グラフで考える
- 組合せ最適化問題ではないが，組合せ最適化と密接な関係
- 様々な方法が提案されている．ここでは古典的な方法を紹介
  - フォード・ファルカーソン法 (Ford-Fulkerson algorithm)
  - エドモンズ・カープ法 (Edmonds-Karp algorithm)
  - プッシュ・リラベル法 (push-relabel algorithm)
  - 線形計画法

最大フロー問題

最大フロー問題 (maximum flow problem)

- 始点 (source) から終点 (sink) までのフロー (モノの流れ) を最大化する問題
- 辺の重み：辺に流せる最大流量・容量 (capacity)

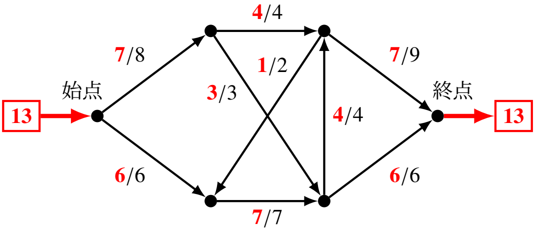


- 有向グラフで考える
- 組合せ最適化問題ではないが，組合せ最適化と密接な関係
- 様々な方法が提案されている．ここでは古典的な方法を紹介
  - フォード・ファルカーソン法 (Ford-Fulkerson algorithm)
  - エドモンズ・カープ法 (Edmonds-Karp algorithm)
  - プッシュ・リラベル法 (push-relabel algorithm)
  - 線形計画法

最大フロー問題

最大フロー問題 (maximum flow problem)

- 始点 (source) から終点 (sink) までのフロー (モノの流れ) を最大化する問題
- 辺の重み：辺に流せる最大流量・容量 (capacity)

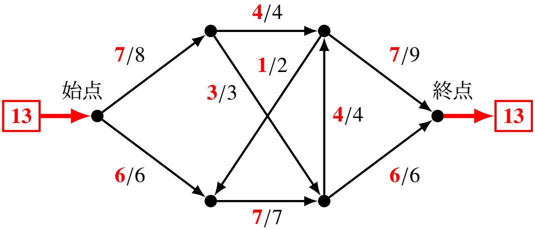


- 有向グラフで考える
- 組合せ最適化問題ではないが，組合せ最適化と密接な関係
- 様々な方法が提案されている．ここでは古典的な方法を紹介
  - フォード・ファルカーソン法 (Ford-Fulkerson algorithm)
  - エドモンズ・カープ法 (Edmonds-Karp algorithm)
  - プッシュ・リラベル法 (push-relabel algorithm)
  - 線形計画法

最大フロー問題

最大フロー問題 (maximum flow problem)

- 始点 (source) から終点 (sink) までのフロー (モノの流れ) を最大化する問題
- 辺の重み：辺に流せる最大流量・容量 (capacity)

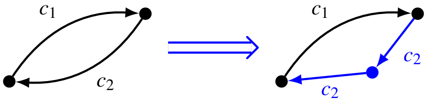


最大フロー：13

- 有向グラフで考える
- 組合せ最適化問題ではないが，組合せ最適化と密接な関係
- 様々な方法が提案されている．ここでは古典的な方法を紹介
  - フォード・ファルカーソン法 (Ford-Fulkerson algorithm)
  - エドモンズ・カープ法 (Edmonds-Karp algorithm)
  - プッシュ・リラベル法 (push-relabel algorithm)
  - 線形計画法

残余ネットワーク

- 有向グラフ $G = (V, E)$，辺 (u, v) の容量 (重み) $c(u, v)$
- $(u, v) \in E$ のときは $(v, u) \notin E$ とする．$(u, v), (v, u) \in E$ のグラフから等価変換可能
- 始点 s，終点 t
- フロー f における辺 (u, v) のフロー $f(u, v)$



残余容量 (residual capacity) $r(u, v)$

辺 (u, v) について，現在のフロー $f(u, v)$ から増やせるフローの量

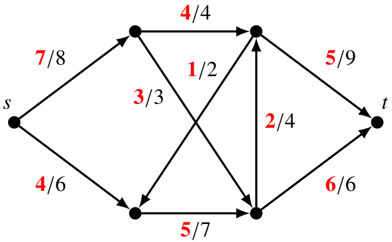
$$r(u, v) = \begin{cases} c(u, v) - f(u, v), & (u, v) \in E \text{ のとき} \\ f(v, u), & (v, u) \in E \text{ のとき} \\ 0, & \text{それ以外} \end{cases}$$

残余ネットワーク (residual network) $G_f = (V, E_f)$

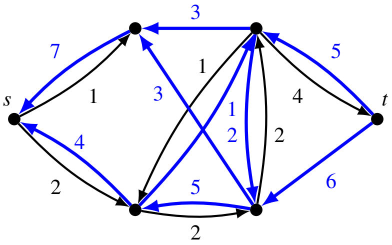
$r(u, v) > 0$ を満たす (u, v) を辺，残余容量 $r(u, v)$ をその容量とするネットワーク

$$E_f = \{(u, v) \mid r(u, v) > 0\}$$

残余ネットワークの例



フロー．辺の数字は「**フロー**/容量」

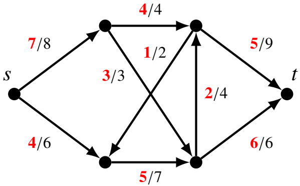


残余ネットワーク

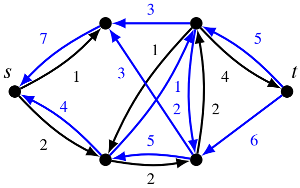
増加路

増加路 (augumenting path)

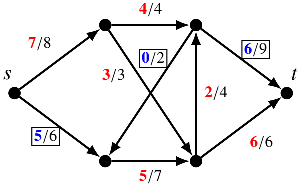
残余ネットワークにおける s から t までの単純路



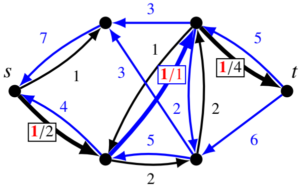
フロー f (11)



残余ネットワーク



フロー f + 増加路 p $(11 + 1 = 12)$



増加路 p

増加路に沿って追加でフローを流すことで，現在のフローが増加

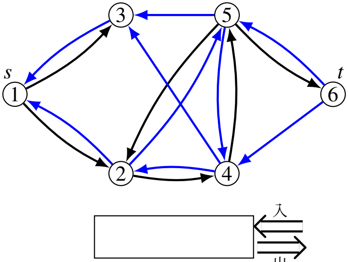
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

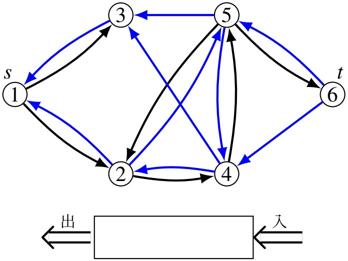
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



深さ優先探索



幅優先探索

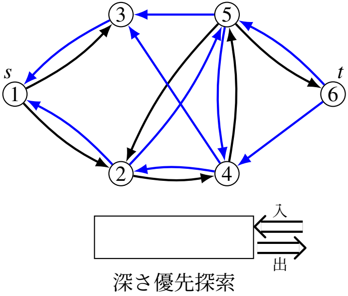
いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

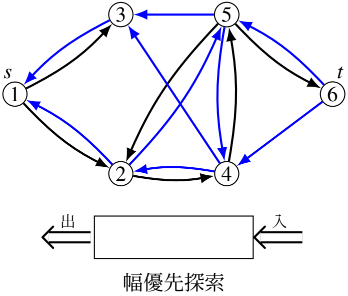
フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など





いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

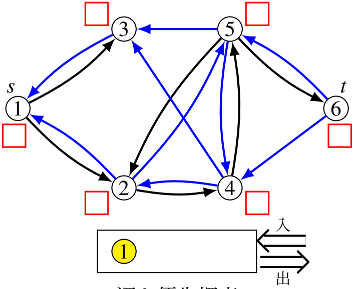
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

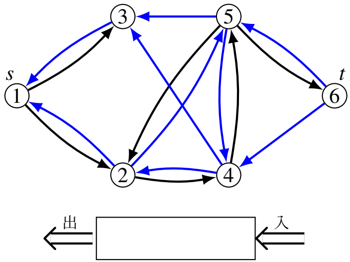
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



深さ優先探索



幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

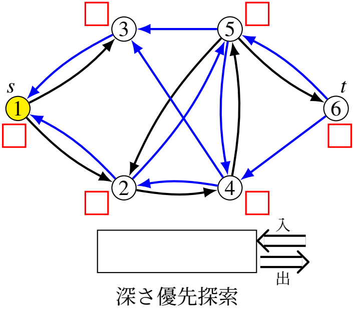
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

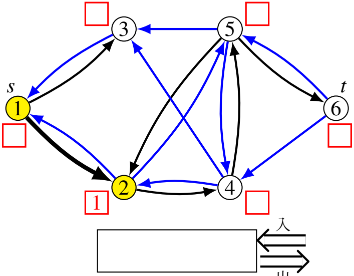
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

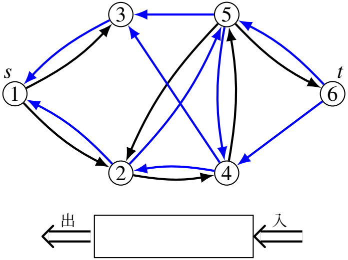
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



深さ優先探索



幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

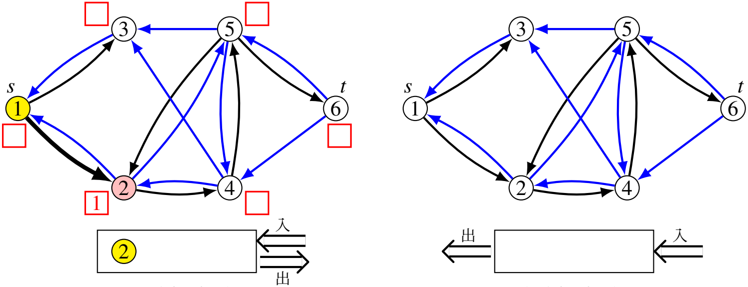
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



深さ優先探索

幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

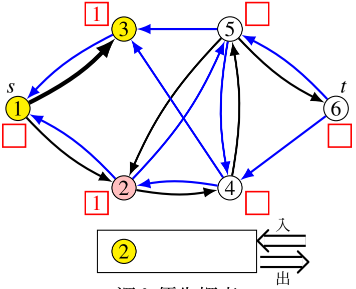
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

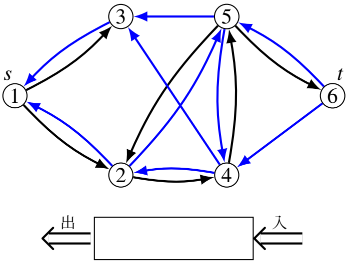
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



深さ優先探索



幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

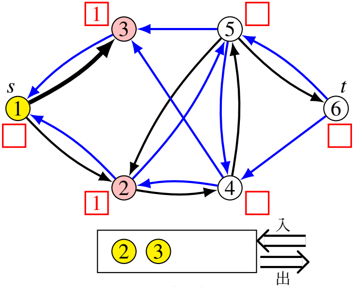
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

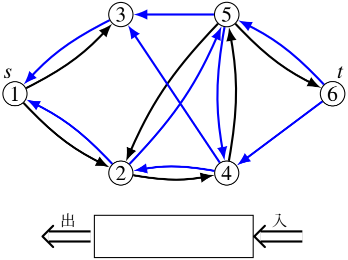
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



深さ優先探索



幅優先探索

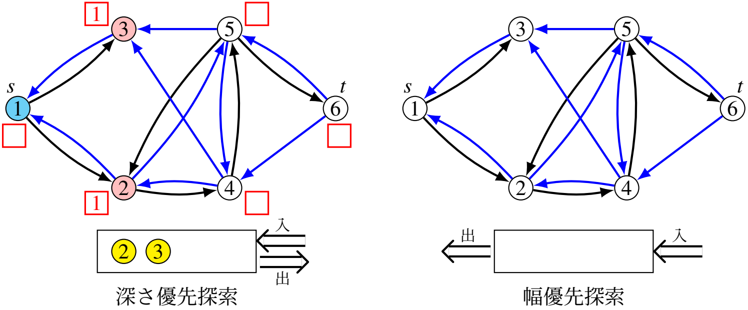
いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

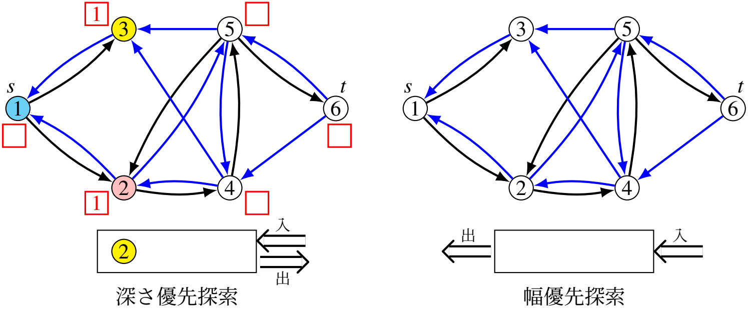
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

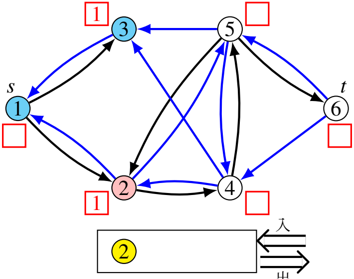
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

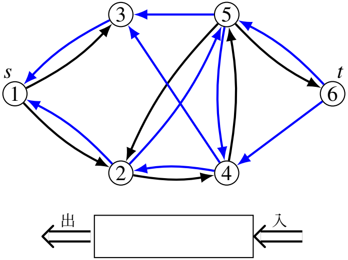
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



深さ優先探索



幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

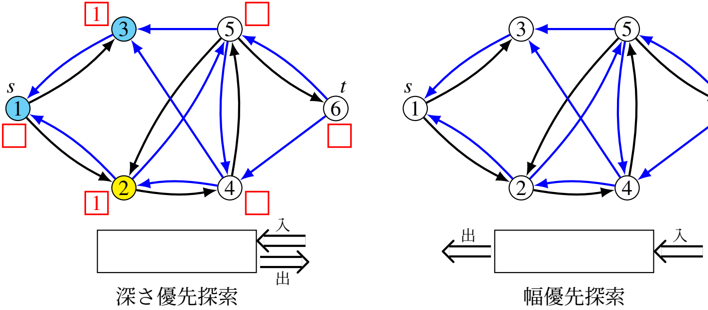
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



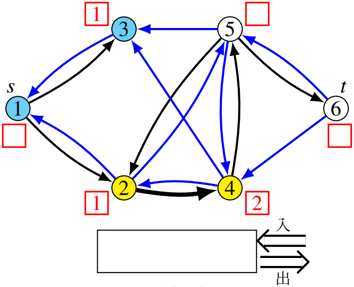
いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

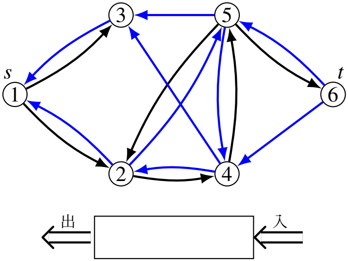
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



深さ優先探索



幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

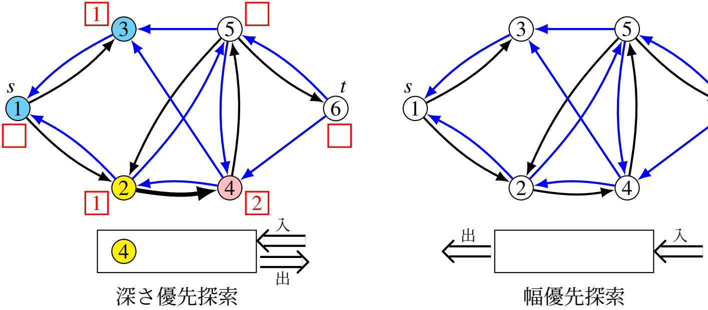
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

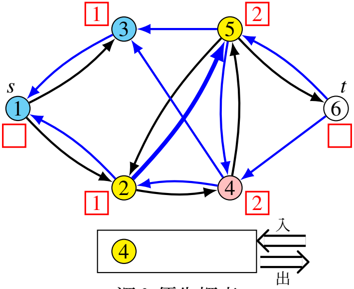
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

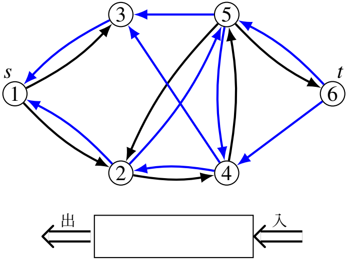
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



深さ優先探索



幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

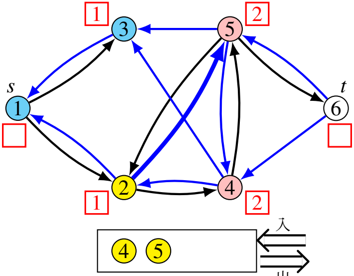
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

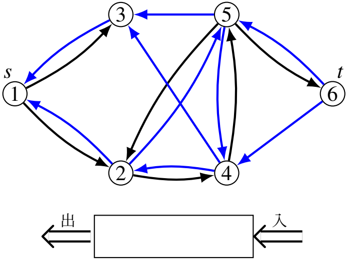
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



深さ優先探索



幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

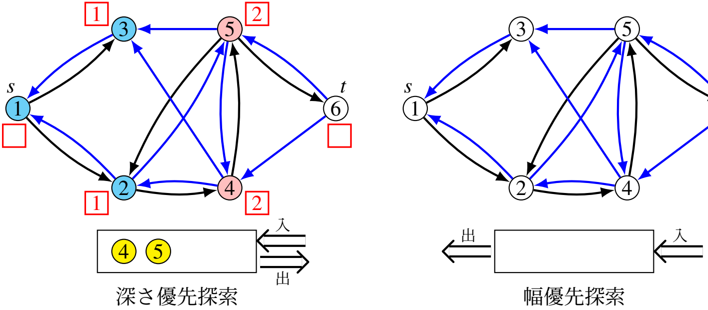
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

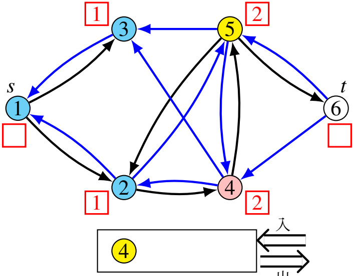
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

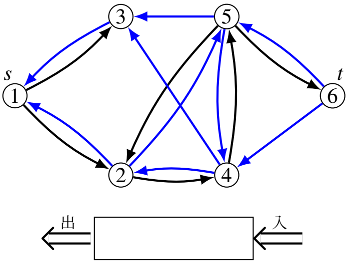
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



深さ優先探索



幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

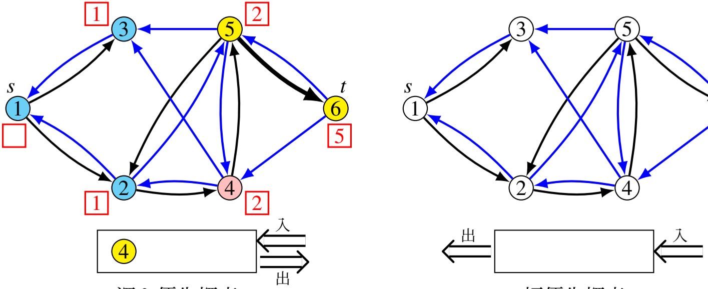
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



深さ優先探索　　幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

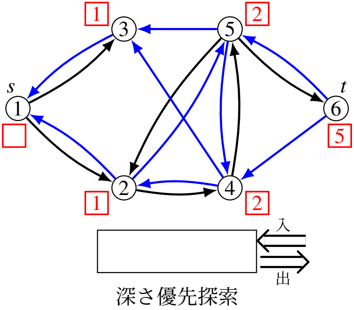
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

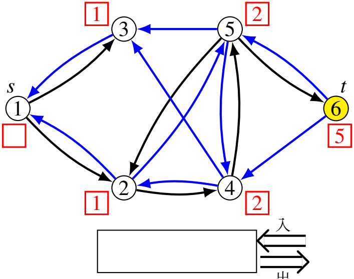
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

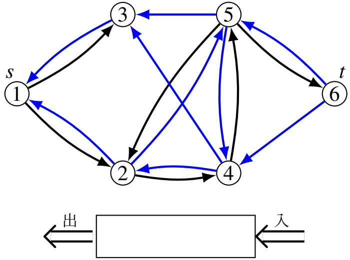
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



深さ優先探索



幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

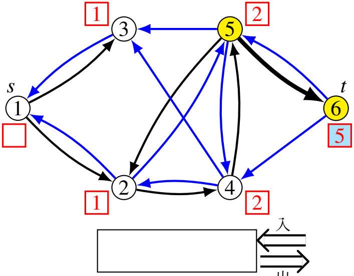
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

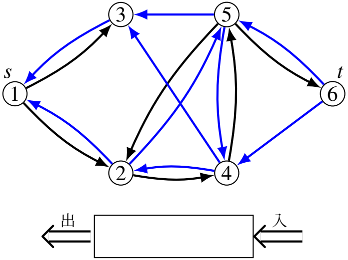
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



深さ優先探索



幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

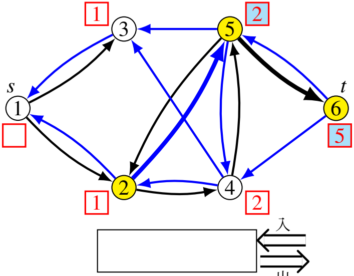
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

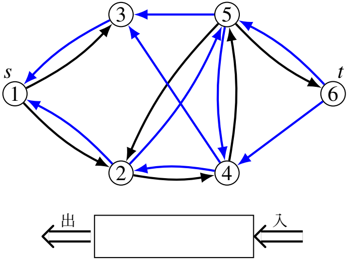
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



深さ優先探索



幅優先探索

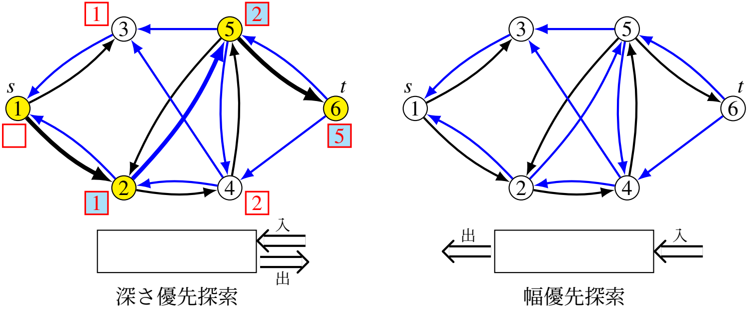
いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や幅優先探索 (breadth-first search) など



いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

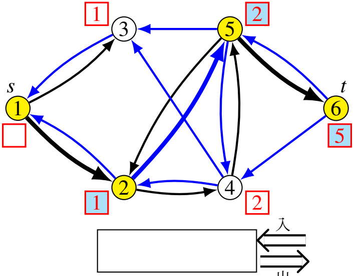
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

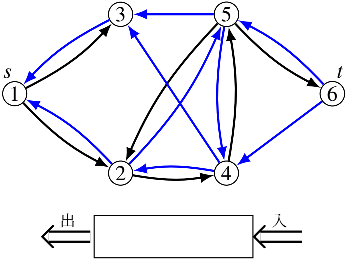
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



深さ優先探索



幅優先探索

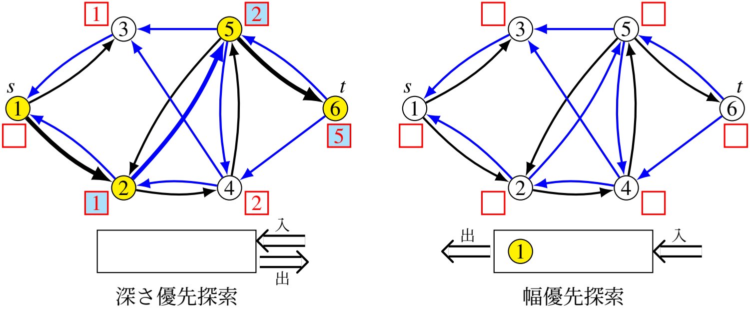
いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

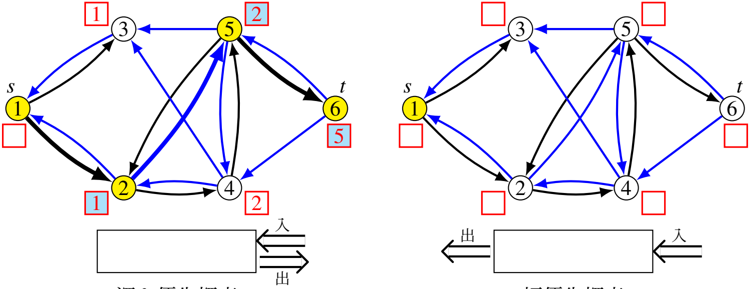
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



深さ優先探索　　幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

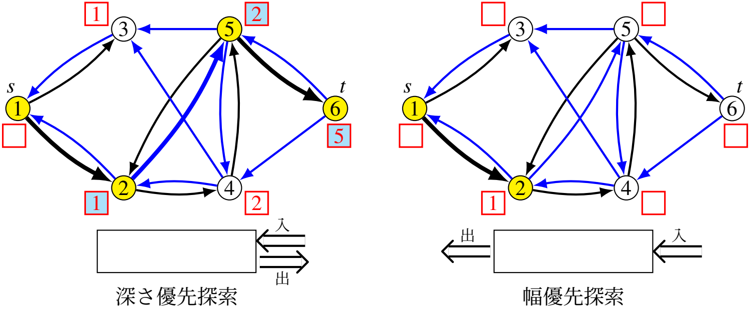
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

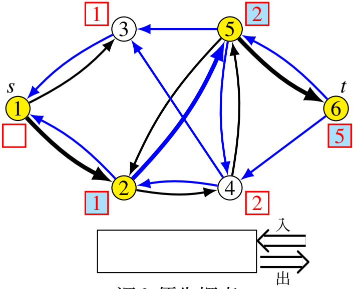
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

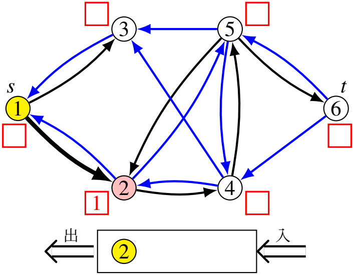
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



深さ優先探索



幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

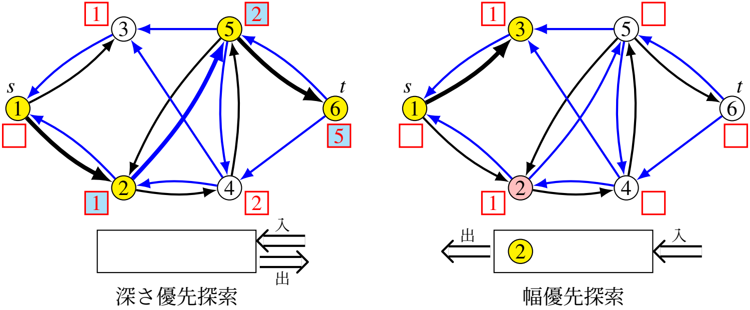
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

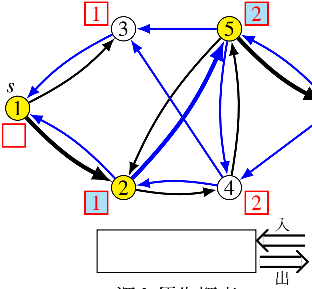
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

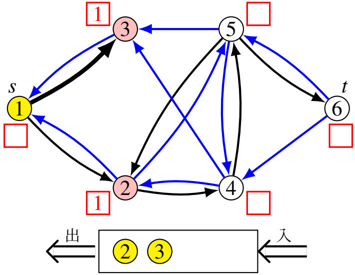
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



深さ優先探索



幅優先探索

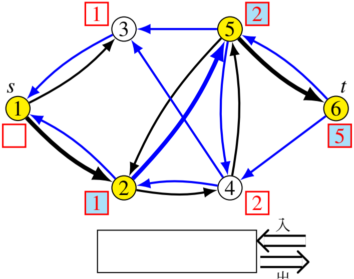
いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

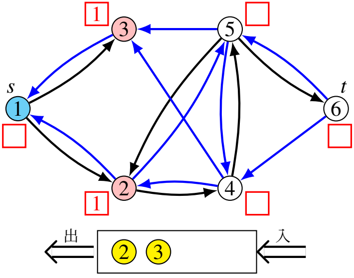
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



深さ優先探索



幅優先探索

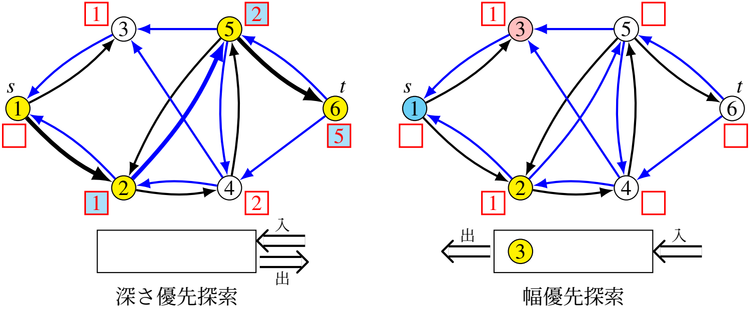
いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

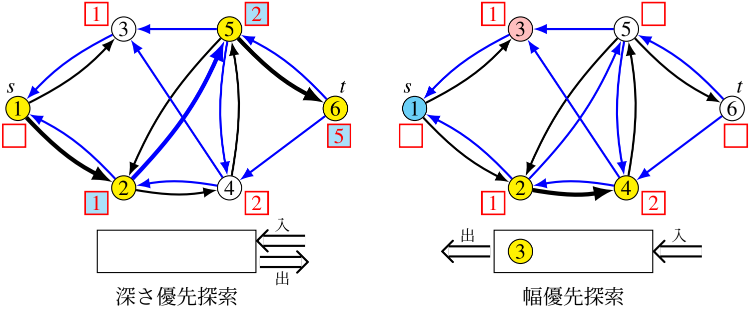
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

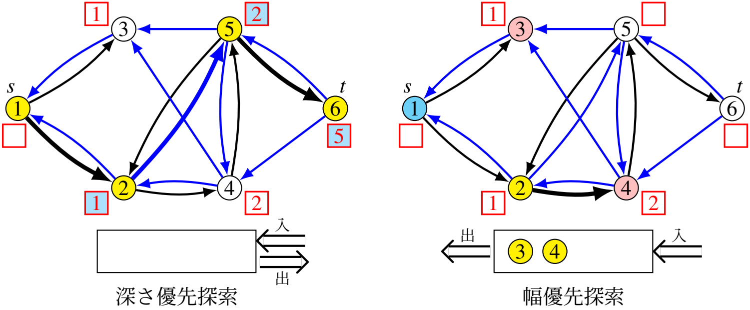
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

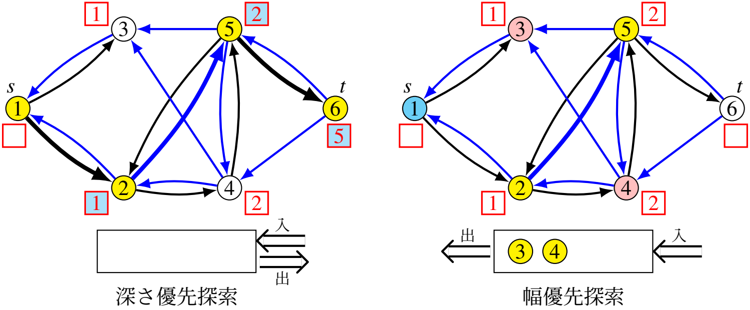
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

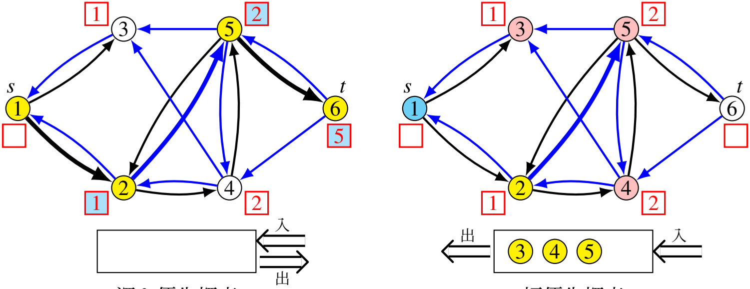
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



深さ優先探索　　幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

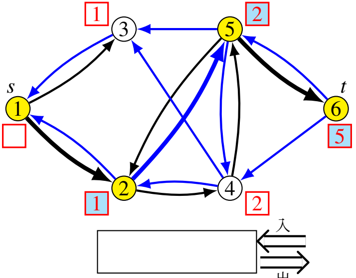
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

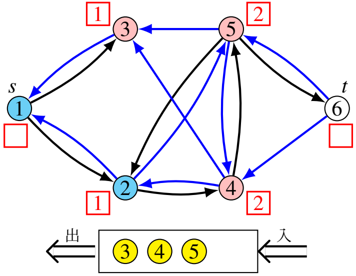
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



深さ優先探索



幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

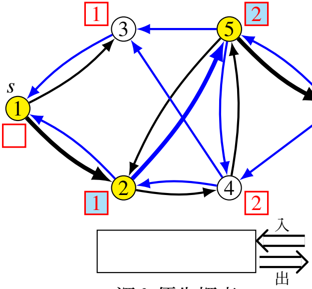
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

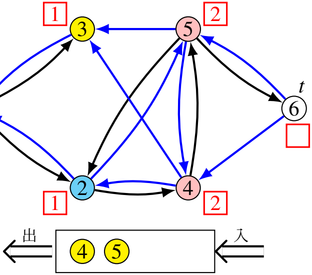
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



深さ優先探索



幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

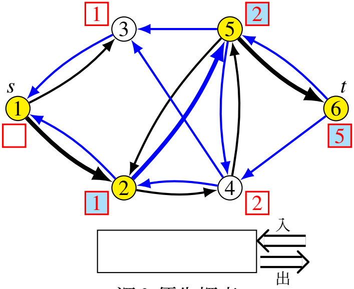
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

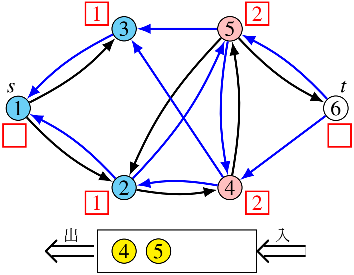
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



深さ優先探索



幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

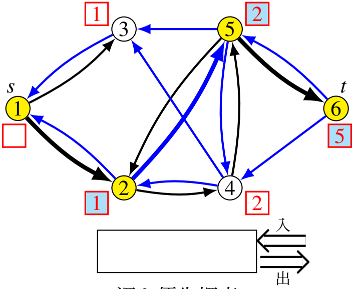
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

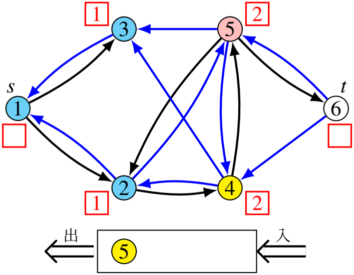
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



深さ優先探索



幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

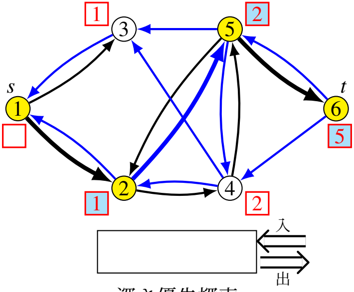
フォード・ファルカーソン法

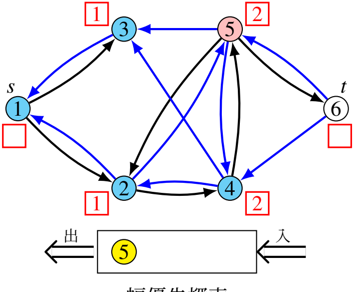
フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など





いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

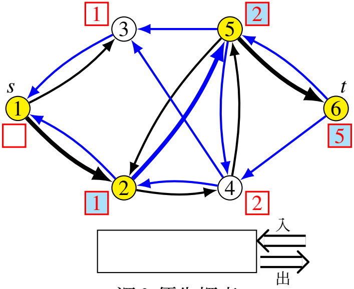
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

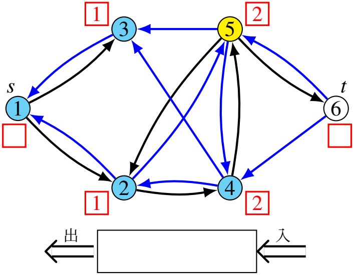
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



深さ優先探索



幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

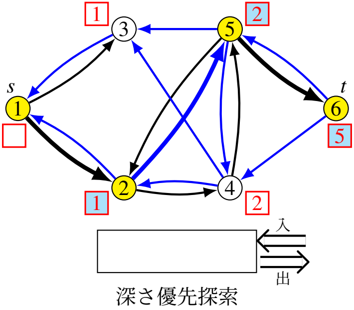
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

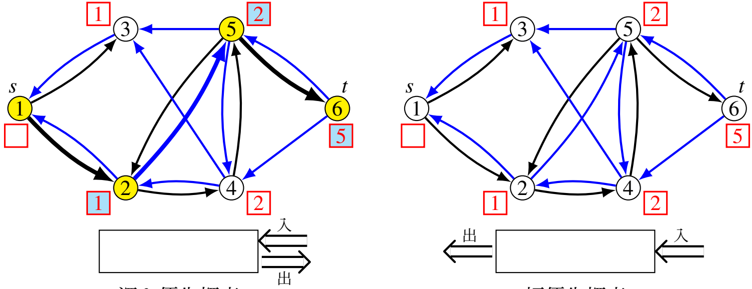
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



深さ優先探索　　幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

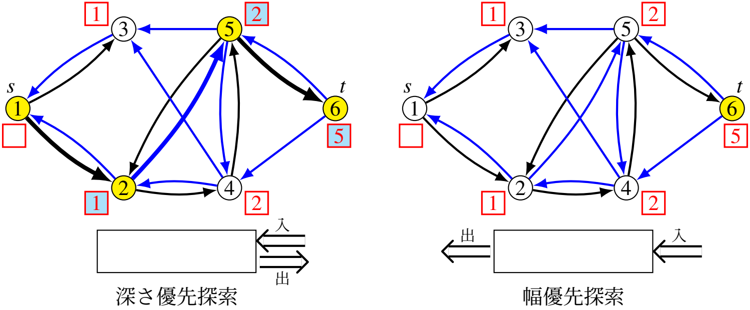
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

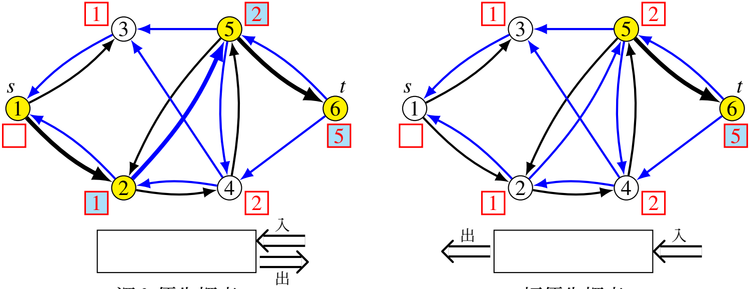
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



深さ優先探索　　　幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

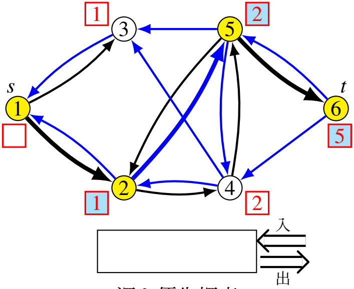
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

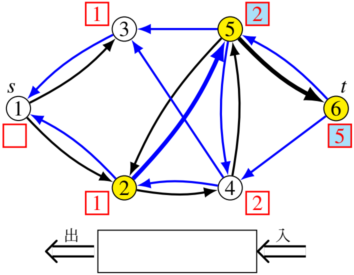
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



深さ優先探索



幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

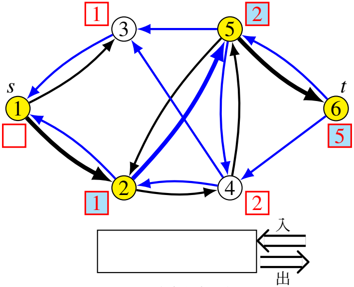
フォード・ファルカーソン法

フォード・ファルカーソン法 (Ford-Fulkerson algorithm)

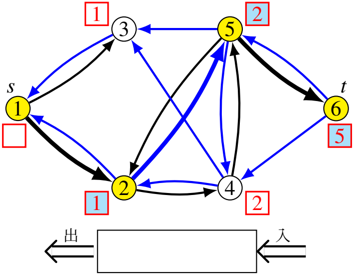
- 増加路を探索してフローを追加することを繰り返す方法
- 最大フローを F とすると，計算量は $O(|E|F)$ (辺容量は整数とする)

増加路の求め方

深さ優先探索 (depth-first search) や**幅優先探索** (breadth-first search) など



深さ優先探索



幅優先探索

いずれも計算量は $O(|E|)$ (各辺を通るのは多くとも 1 回)

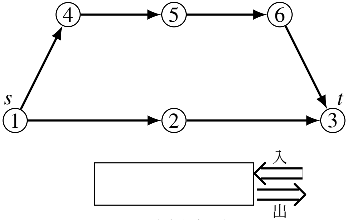
エドモンズ・カープ法

フォード・ファルカーソン法の問題点

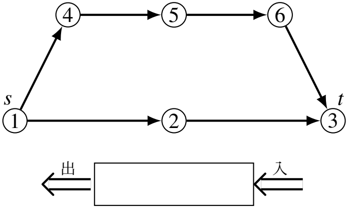
- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) s から t への最短経路が求まる
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)



深さ優先探索



幅優先探索

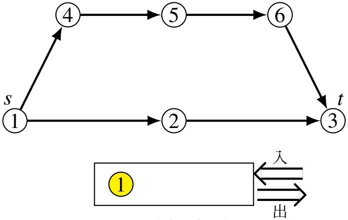
エドモンズ・カープ法

フォード・ファルカーソン法の問題点

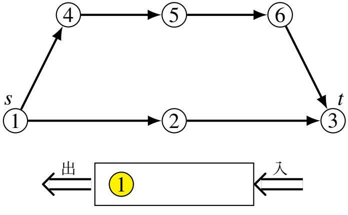
- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) s から t への最短経路が求まる
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)



深さ優先探索



幅優先探索

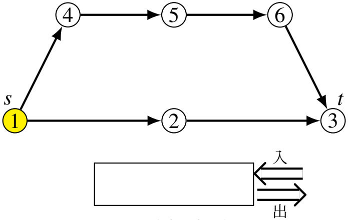
エドモンズ・カープ法

フォード・ファルカーソン法の問題点

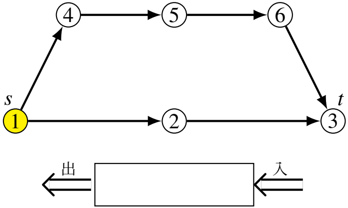
- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) s から t への最短経路が求まる
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)



深さ優先探索



幅優先探索

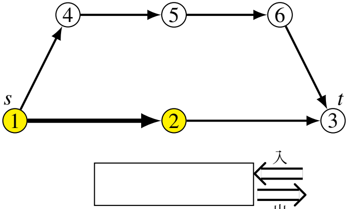
エドモンズ・カープ法

フォード・ファルカーソン法の問題点

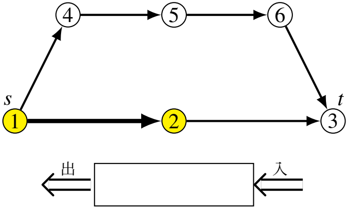
- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) s から t への最短経路が求まる
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)



深さ優先探索



幅優先探索

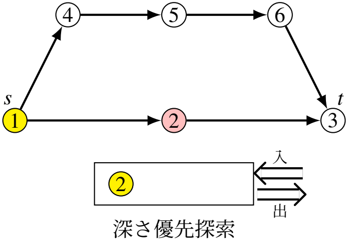
エドモンズ・カープ法

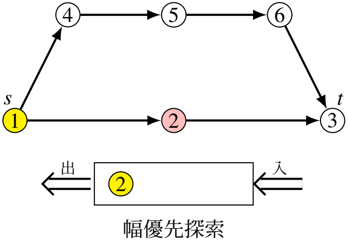
フォード・ファルカーソン法の問題点

- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) s から t への最短経路が求まる
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)





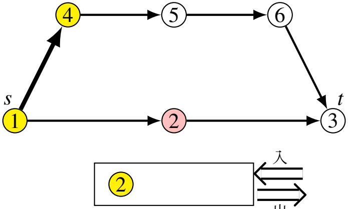
エドモンズ・カープ法

フォード・ファルカーソン法の問題点

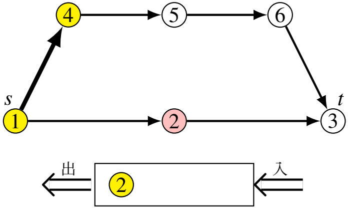
- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) s から t への最短経路が求まる
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)



深さ優先探索



幅優先探索

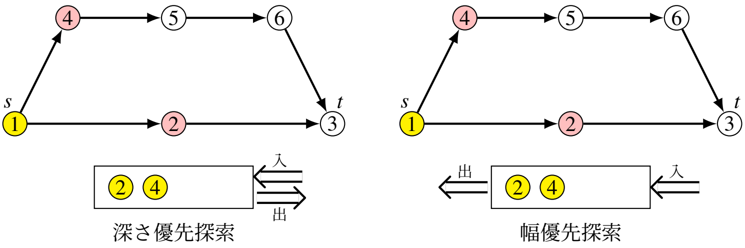
エドモンズ・カープ法

フォード・ファルカーソン法の問題点

- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) s から t への最短経路が求まる
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)



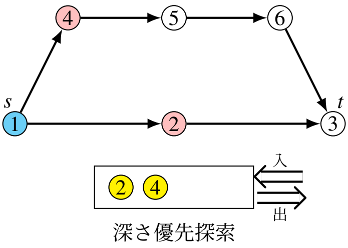
エドモンズ・カープ法

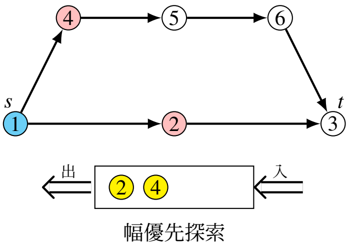
フォード・ファルカーソン法の問題点

- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) s から t への最短経路が求まる
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)





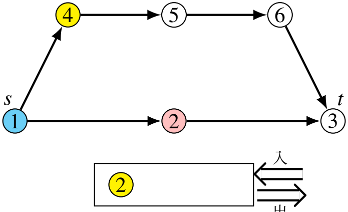
エドモンズ・カープ法

フォード・ファルカーソン法の問題点

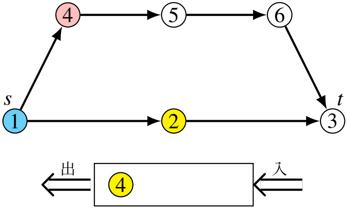
- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) s から t への最短経路が求まる
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)



深さ優先探索



幅優先探索

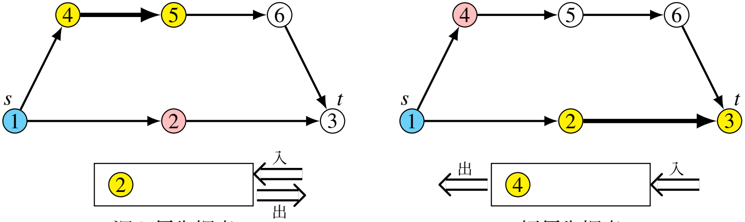
エドモンズ・カープ法

フォード・ファルカーソン法の問題点

- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) s から t への最短経路が求まる
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)



深さ優先探索　　　幅優先探索

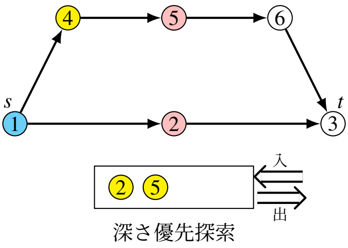
エドモンズ・カープ法

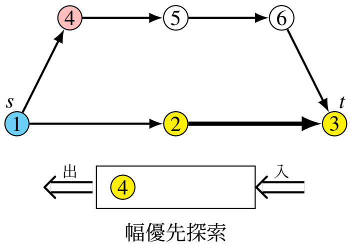
フォード・ファルカーソン法の問題点

- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) s から t への最短経路が求まる
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)





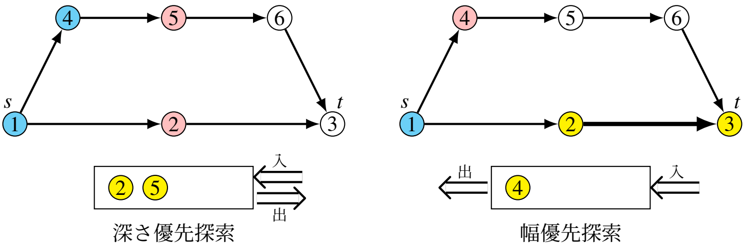
エドモンズ・カープ法

フォード・ファルカーソン法の問題点

- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) s から t への最短経路が求まる
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)



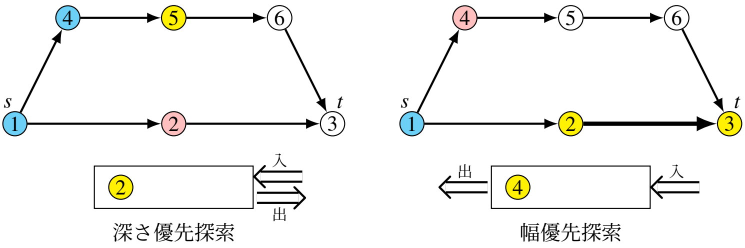
エドモンズ・カープ法

フォード・ファルカーソン法の問題点

- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) s から t への最短経路が求まる
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)



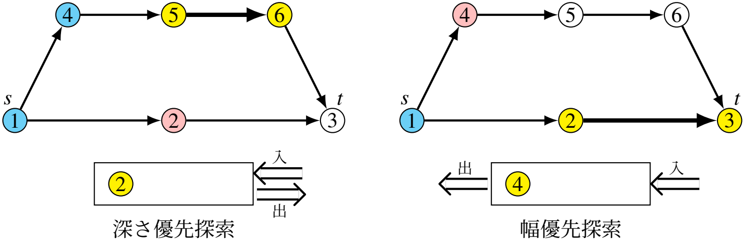
エドモンズ・カープ法

フォード・ファルカーソン法の問題点

- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) s から t への最短経路が求まる
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)



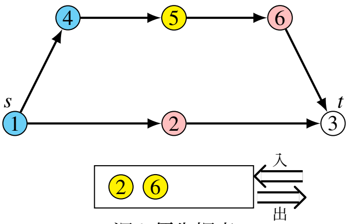
エドモンズ・カープ法

フォード・ファルカーソン法の問題点

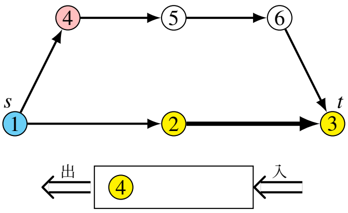
- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) s から t への最短経路が求まる
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)



深さ優先探索



幅優先探索

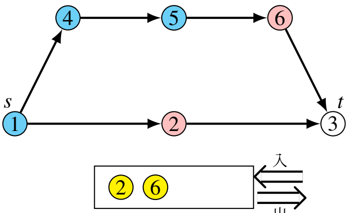
エドモンズ・カープ法

フォード・ファルカーソン法の問題点

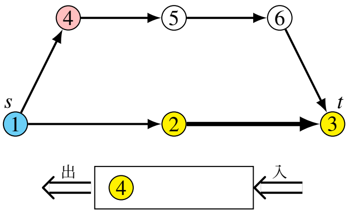
- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) s から t への最短経路が求まる
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)



深さ優先探索



幅優先探索

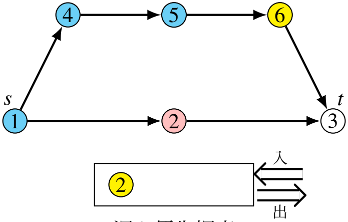
エドモンズ・カープ法

フォード・ファルカーソン法の問題点

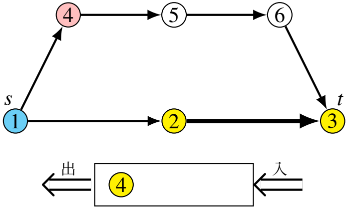
- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) s から t への最短経路が求まる
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)



深さ優先探索



幅優先探索

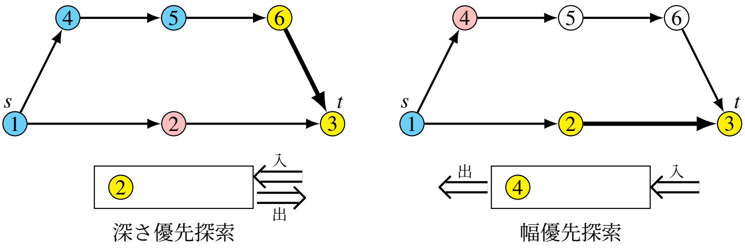
エドモンズ・カープ法

フォード・ファルカーソン法の問題点

- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) s から t への最短経路が求まる
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)



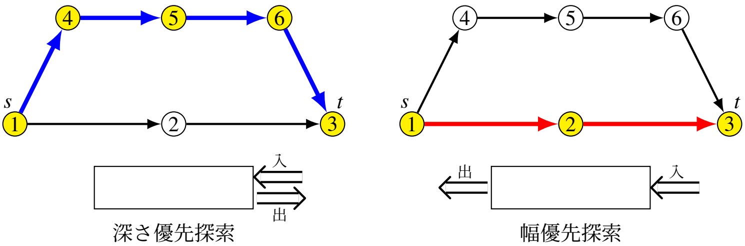
エドモンズ・カープ法

フォード・ファルカーソン法の問題点

- 運が悪いと各反復でフローがあまり増加しない
- 計算量 $O(|E|F)$ に最大フロー F が含まれる (擬多項式時間)

エドモンズ・カープ法 (Edmonds-Karp algorithm)

- フォード・ファルカーソン法において増加路を幅優先探索で求める
  幅優先探索：(各辺の距離を 1 とみなした) **s から t への最短経路が求まる**
- 反復回数が $O(|E||V|)$ に収まるので，計算量 $O(|E|^2|V|)$ (多項式時間)



エドモンズ・カープ法の反復回数

$G_f^{(k)} = (V, E_f^{(k)})$： 第 k 反復における残余ネットワーク

$P^{(k)}(s,v),\ \ d^{(k)}(s,v)$： $G^(k))_f$ における s から v までの最短経路および最短距離

距離の単調増加性

すべての k とすべての $v \in V$ について，$d^{(k)}(s,v) \le d^{(k+1)}(s,v)$

証明

- $d^{(k)}(s,v) > d^{(k+1)}(s,v)$ を満たす v のうち，$d^{(k+1)}(s,v)$ が最小のものを選ぶ
- $P^{(k+1)}(s,v)$ において，v の直前に訪問する頂点を u とする

(a) 部分構造最適性より $d^{(k+1)}(s,v) = d^{(k+1)}(s,u) + 1$．よって $d^{(k+1)}(s,u) < d^{(k+1)}(s,v)$.

(b) (a) と v の定義より $d^{(k)}(s,u) \le d^{(k+1)}(s,u)$.

(c) $(u,v) \notin E_f^{(k)}$．$(u,v) \in E_f^{(k)}$ とすると，$d^{(k)}(s,v) \le d^{(k)}(s,u) + 1$．よって，(b)，(a) より

$$d^{(k)}(s,v) \le d^{(k)}(s,u) + 1 \le d^{(k+1)}(s,u) + 1 = d^{(k+1)}(s,v)$$

これは仮定 $d^{(k)}(s,v) > d^{(k+1)}(s,v)$ に矛盾.

(d) $P^{(k+1)}(s,v)$ は (u,v) を通るので，$(u,v) \in E_f^{(k+1)}$.

(e) (c)，(d) より増加路 $P^{(k)}(s,t)$ は辺 (v,u) を通るため，$d^{(k)}(s,u) = d^{(k)}(s,v) + 1$.

(f) (e)，(b)，(a) より

$$d^{(k)}(s,v) = d^{(k)}(s,u) - 1 \le d^{(k+1)}(s,u) - 1 = d^{(k+1)}(s,v) - 2$$

したがって，$d^{(k)}(s,v) < d^{(k+1)}(s,v)$．これは仮定 $d^{(k)}(s,v) > d^{(k+1)}(s,v)$ に矛盾.

エドモンズ・カープ法の反復回数 (続き)

$G^{(k)} = (V, E_f^{(k)})$： 第 k 反復における残余ネットワーク

$P^{(k)}(s,v)$, $d^{(k)}(s,v)$： $G_f^{(k)}$ における s から v までの最短経路および最短距離

反復回数の上界

エドモンズ・カープ法の反復回数は $O(|V||E|)$.

証明

- 各反復では増加路に沿ってフローを最大限流すので，増加路上のいずれかの辺において (辺のフロー) = (辺の容量) となる．⇒ この辺は残余ネットワークから消える
- 辺 (u,v) は，第 k 反復の操作で残余ネットワークから消え，第 k' 反復の操作で再び残余ネットワークに出現するとする

(a) $P^{(k)}(s,t)$ は (u,v) を通るので，$d^{(k)}(s,v) = d^{(k)}(s,u) + 1$.

(b) $P^{(k')}(s,t)$ は (v,u) を通るので，$d^{(k')}(s,u) = d^{(k')}(s,v) + 1$.

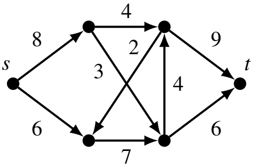
(c) 前ページで示した $d^{(k)}(s,v)$ の単調増加性より，$d^{(k)}(s,v) \leq d^{(k')}(s,v)$.

(d) (b)，(c)，(a) より $d^{(k')}(s,u) = d^{(k')}(s,v) + 1 \geq d^{(k)}(s,v) + 1 = d^{(k)}(s,u) + 2$．したがって，$u$ までの最短距離は，第 k 反復から第 k' 反復の間に少なくとも 2 増加．

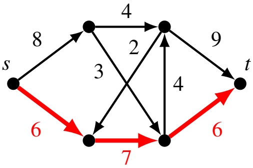
(e) u までの最短距離は $|V| - 1$ 以下なので，(d) より，(u,v) が残余ネットワークから消える回数は $1 + (|V| - 1)/2 = (|V| + 1)/2$ 回以下．

(f) 各反復で少なくとも 1 本の辺が消えるので，反復回数は $|E|(|V| + 1)/2 = O(|E||V|)$.

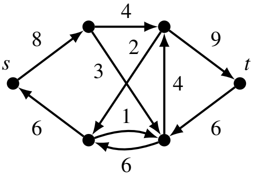
エドモンズ・カープ法の例



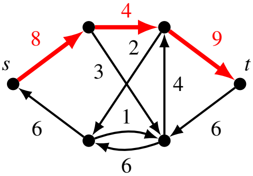
(1) $G_f^{(1)}$ (フロー 0)



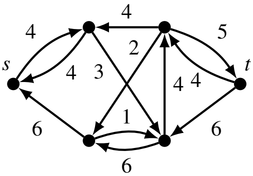
(2) $P^{(1)}(s,t)$



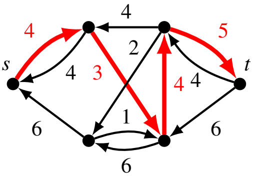
(3) $G_f^{(2)}$ (フロー 6)



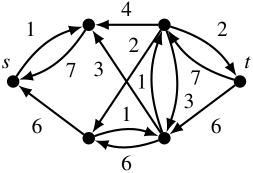
(4) $P^{(2)}(s,t)$



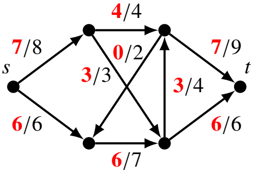
(5) $G_f^{(3)}$ (フロー 10)



(6) $P^{(3)}(s,t)$

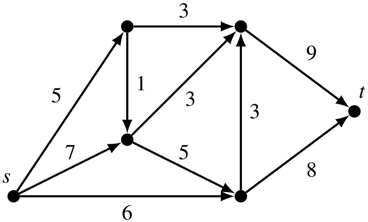


(7) $G_f^{(4)}$ (フロー 13)

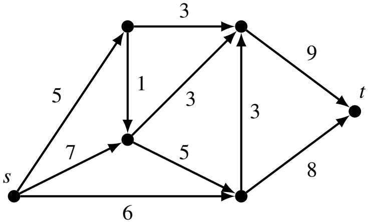


(8) 最大フロー 13

エドモンズ・カープ法の練習問題

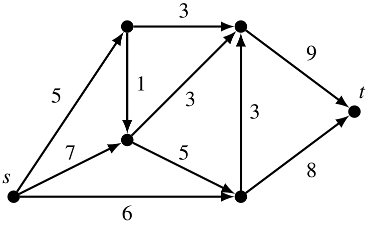


G

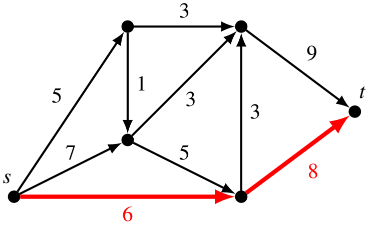


$G_f^{(1)}$

エドモンズ・カープ法の練習問題

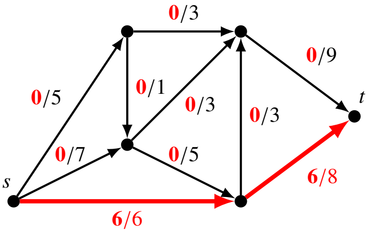


G

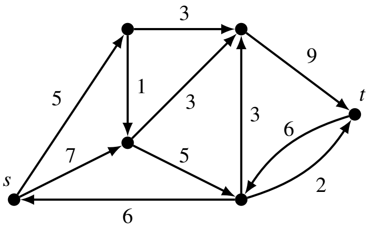


$P^{(1)}(s, t)$

エドモンズ・カープ法の練習問題

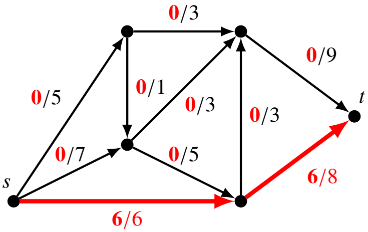


G (フロー 6)

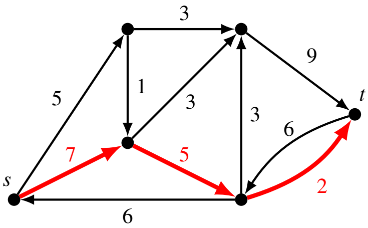


$G_f^{(2)}$

エドモンズ・カープ法の練習問題

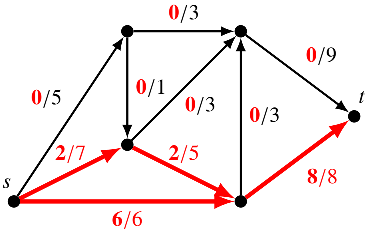


G (フロー 6)

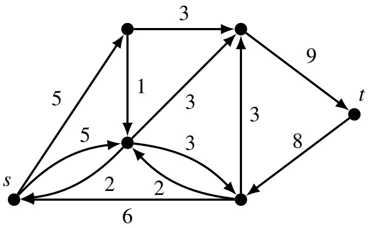


$P^{(2)}(s,t)$

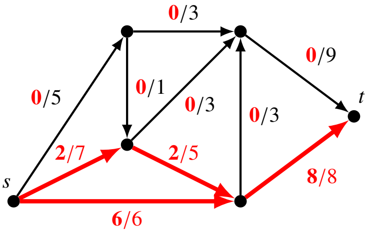
エドモンズ・カープ法の練習問題



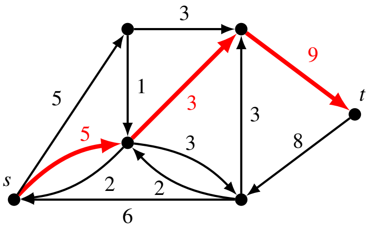
G (フロー 8)



$G_f^{(3)}$

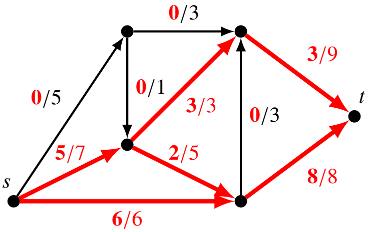


G (フロー 8)

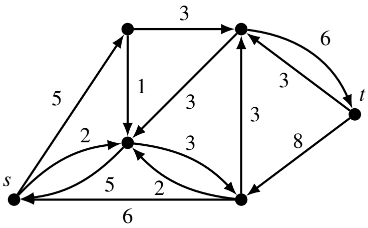


$P^{(3)}(s,t)$

エドモンズ・カープ法の練習問題

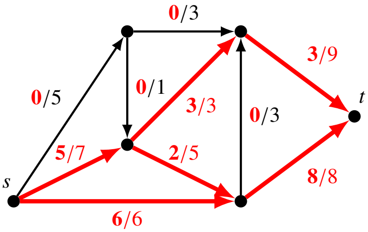


G (フロー 11)

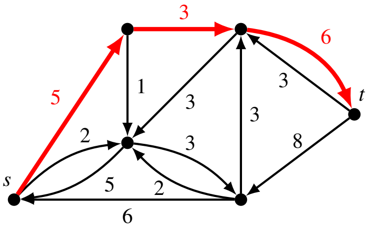


$G_f^{(4)}$

エドモンズ・カープ法の練習問題

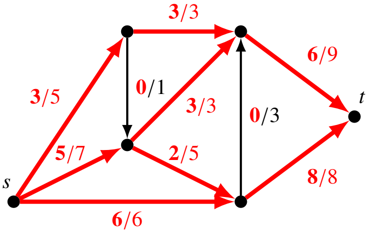


G (フロー 11)

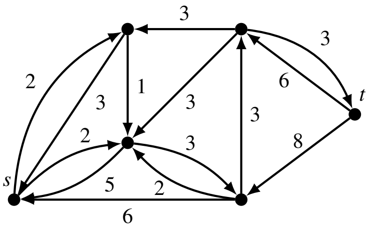


$P^{(4)}(s,t)$

エドモンズ・カープ法の練習問題

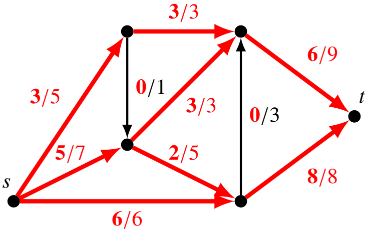


G (フロー 14)

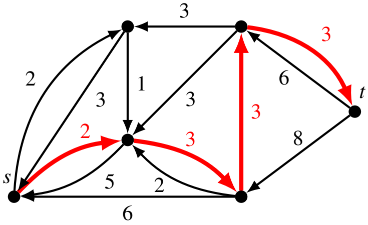


$G_f^{(5)}$

エドモンズ・カープ法の練習問題

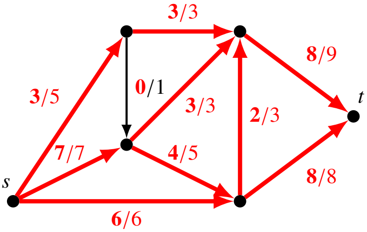


G (フロー 14)

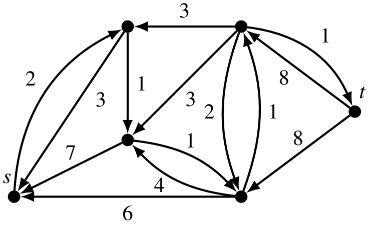


$P^{(5)}(s,t)$

エドモンズ・カープ法の練習問題

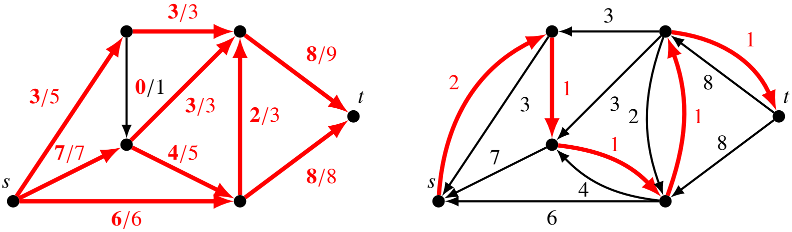


G (フロー 16)



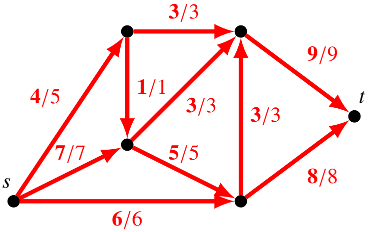
$G_f^{(6)}$

エドモンズ・カープ法の練習問題

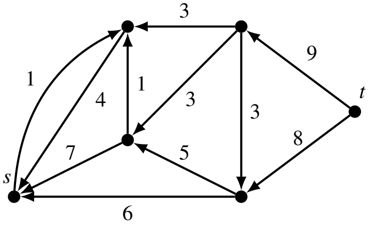


G (フロー 16)　　$P^{(6)}(s,t)$

エドモンズ・カープ法の練習問題



G (フロー 17)



$G_f^{(7)}$

プッシュ・リラベル法

プッシュ・リラベル法 (push-relabel algorithm)

- 終点までの経路にフローを流すのではなく，各頂点ごとにフローを流す
- 残余ネットワーク上で終点からの距離を表すラベルを保持，以下を反復

  プッシュ： 頂点に溜まったフローを終点に近い頂点へ押し出す
  リラベル： 距離のラベルを更新
- プリフロー・プッシュ法 (preflow-push algorithm) とも．計算量 $O(|V|^2|E|)$

活性 (active) 頂点

フローが溜まった始点・終点以外の頂点．$g(u) = (\text{流入フロー}) - (\text{流出フロー}) > 0$

妥当 (valid) な距離ラベル $\tilde{d}(v)$

$(u, v) \in E_f$ のとき $\tilde{d}(u) \leq \tilde{d}(v) + 1$ を満たす

許容 (admissible) 辺

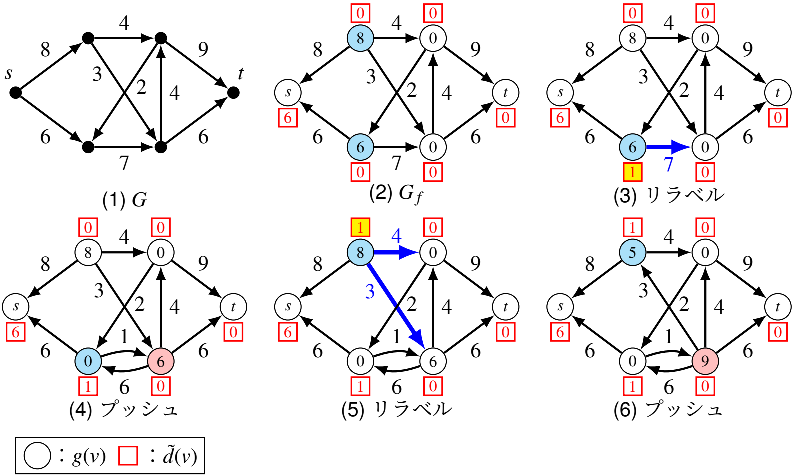
$\tilde{d}(u) = \tilde{d}(v) + 1$ を満たす辺 $(u, v) \in E_f$．活性頂点 u からこの辺にフローをプッシュする

リラベル (relabel)

活性頂点 u が許容辺を持たなければ，距離ラベルを以下で更新

$$\tilde{d}(u) = 1 + \min_{(u,v) \in E_f} \tilde{d}(v)$$

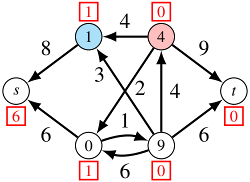
プッシュ・リラベル法の例



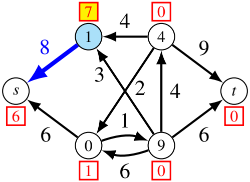
初期フロー： 始点 s に接続する辺に，容量分のフローを流す

初期ラベル： 始点 s は $d(s) = |V|$，それ以外の頂点 v は $d(v) = 0$

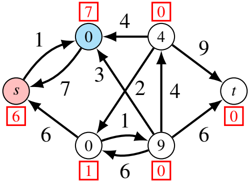
プッシュ・リラベル法の例 (続き)



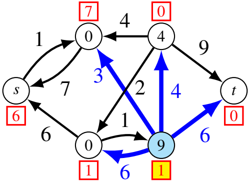
(7) プッシュ



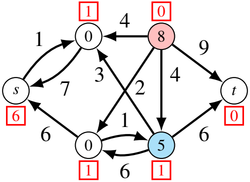
(8) リラベル



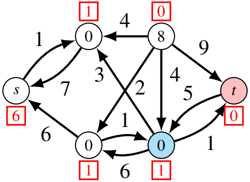
(9) プッシュ



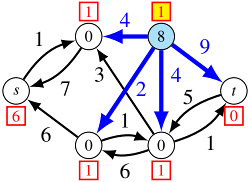
(10) リラベル



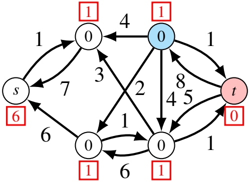
(11) プッシュ



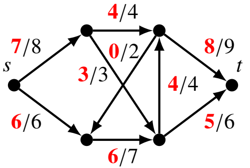
(12) プッシュ



(13) リラベル

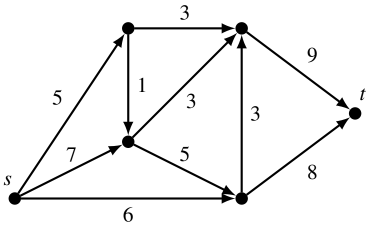


(14) プッシュ

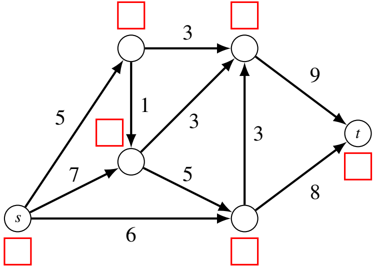


(15) 最大フロー 13

プッシュ・リラベル法の練習問題

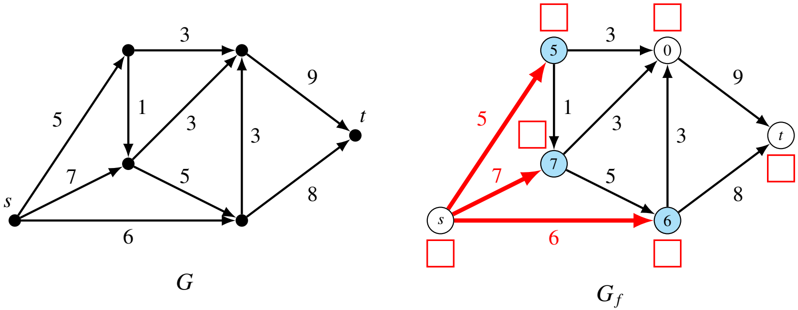


G

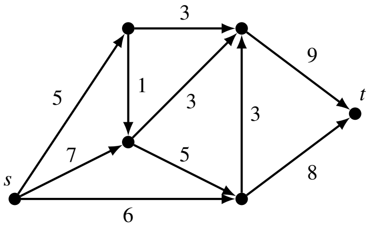


G_f

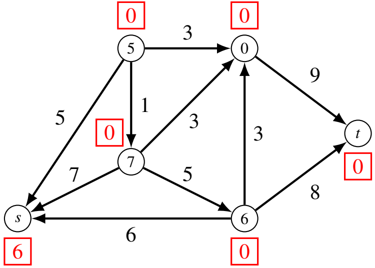
プッシュ・リラベル法の練習問題



プッシュ・リラベル法の練習問題

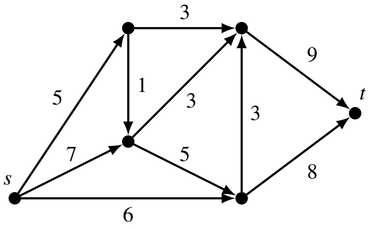


G

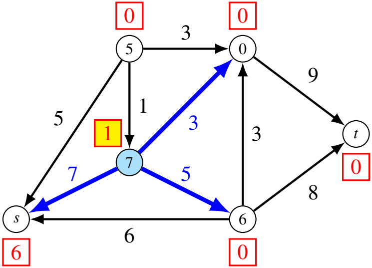


G_f

プッシュ・リラベル法の練習問題

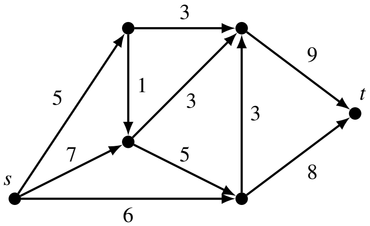


G

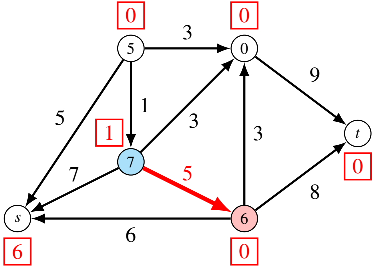


リラベル

プッシュ・リラベル法の練習問題

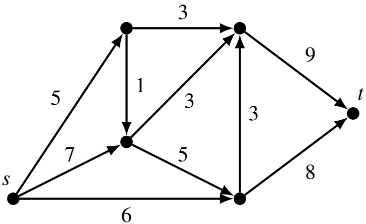


G

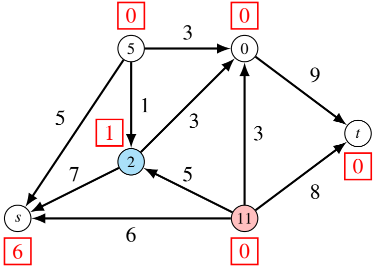


プッシュ

プッシュ・リラベル法の練習問題

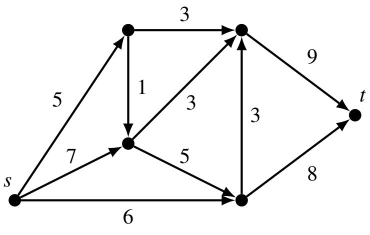


G

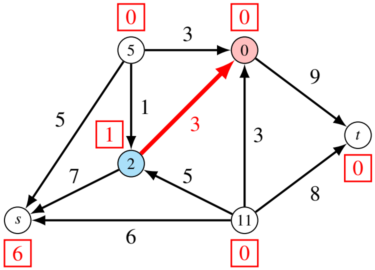


プッシュ

プッシュ・リラベル法の練習問題

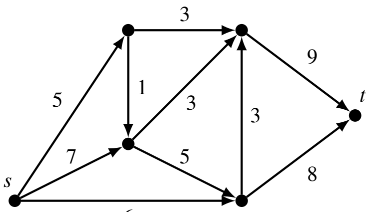


G

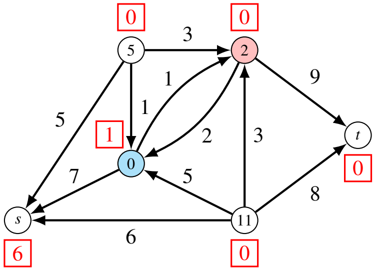


プッシュ

プッシュ・リラベル法の練習問題

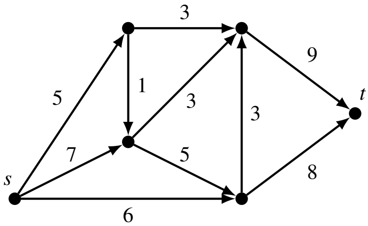


G

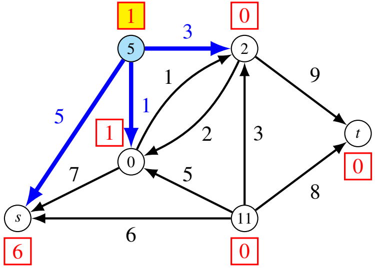


プッシュ

プッシュ・リラベル法の練習問題

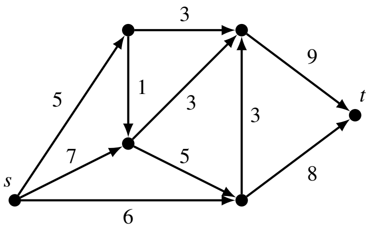


G

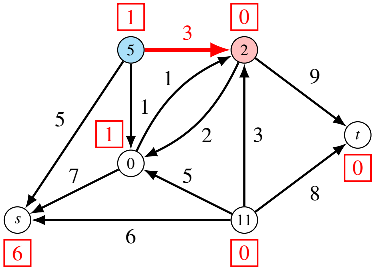


リラベル

プッシュ・リラベル法の練習問題

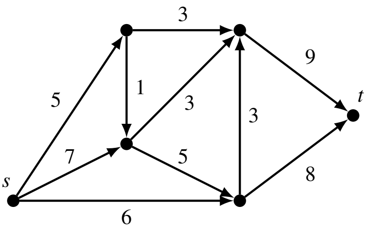


G

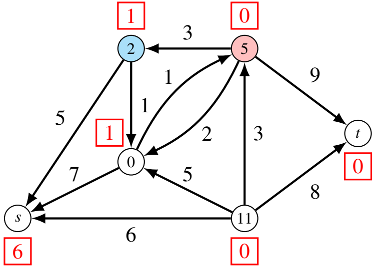


プッシュ

プッシュ・リラベル法の練習問題

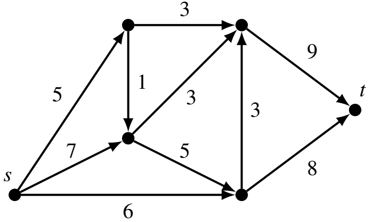


G

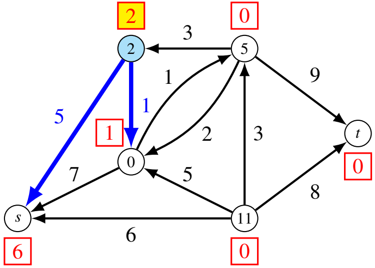


プッシュ

プッシュ・リラベル法の練習問題

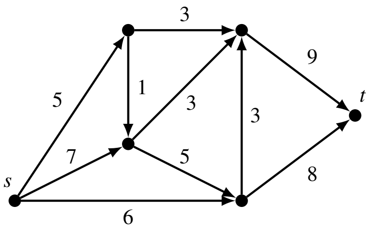


G

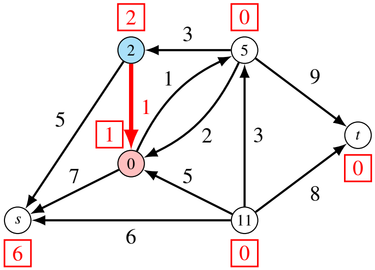


リラベル

プッシュ・リラベル法の練習問題

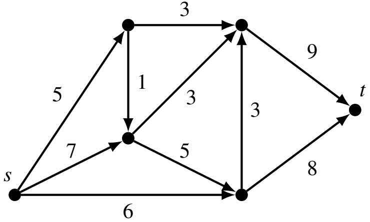


G

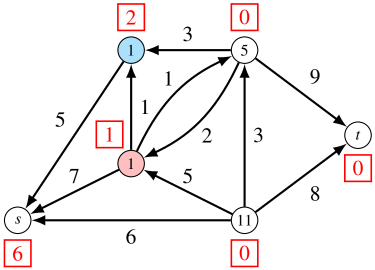


プッシュ

プッシュ・リラベル法の練習問題

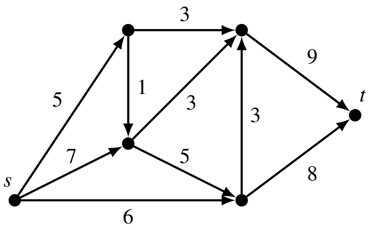


G

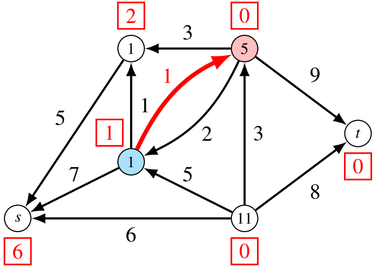


プッシュ

プッシュ・リラベル法の練習問題

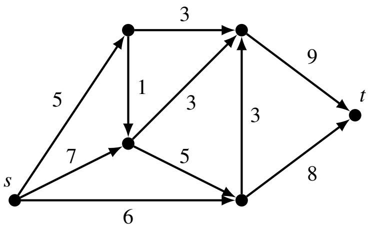


G

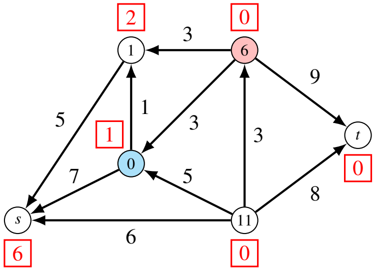


プッシュ

プッシュ・リラベル法の練習問題

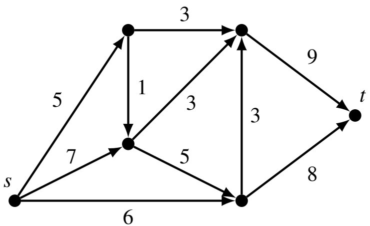


G

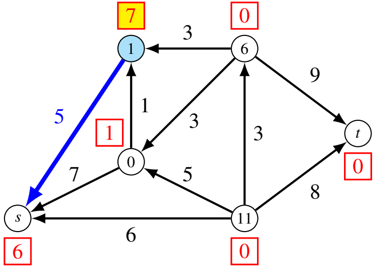


プッシュ

プッシュ・リラベル法の練習問題

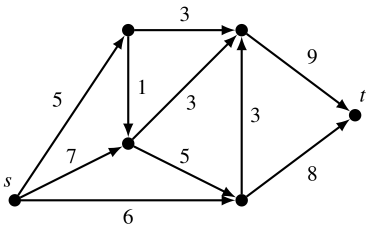


G

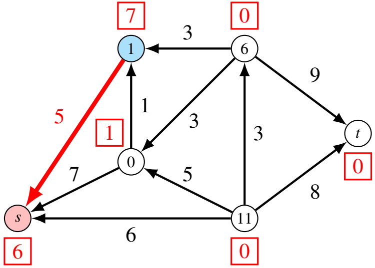


リラベル

プッシュ・リラベル法の練習問題

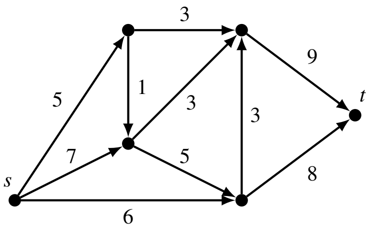


G

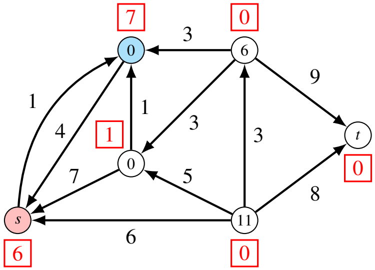


プッシュ

プッシュ・リラベル法の練習問題

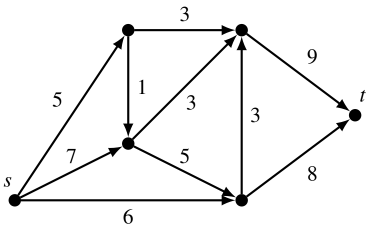


G

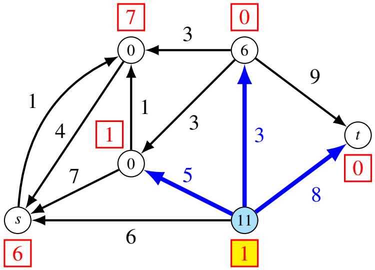


プッシュ

プッシュ・リラベル法の練習問題

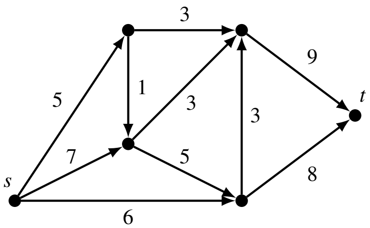


G

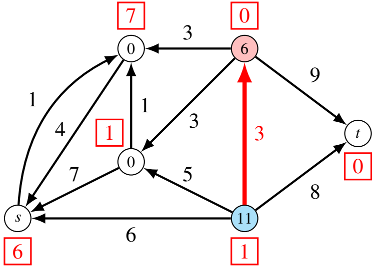


リラベル

プッシュ・リラベル法の練習問題

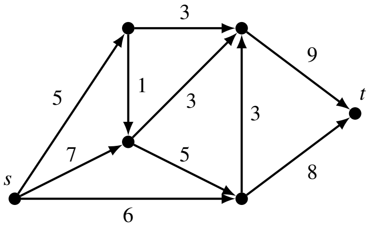


G

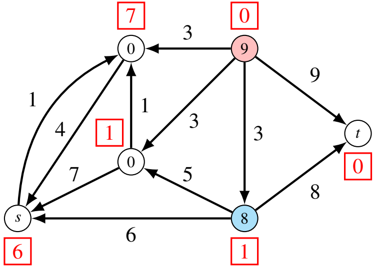


プッシュ

プッシュ・リラベル法の練習問題

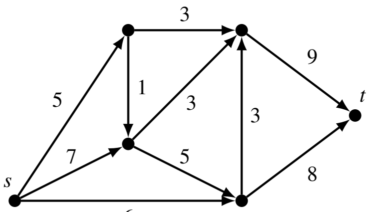


G

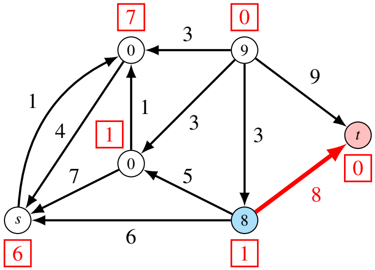


プッシュ

プッシュ・リラベル法の練習問題

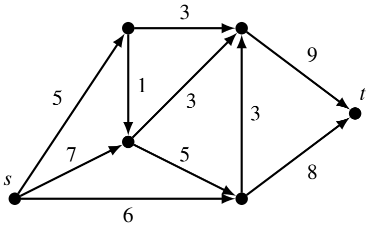


G

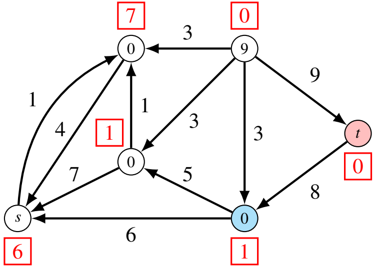


プッシュ

プッシュ・リラベル法の練習問題

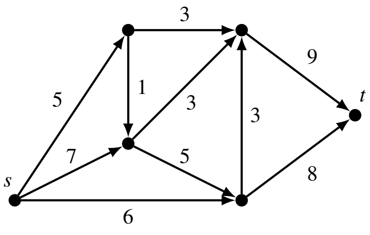


G

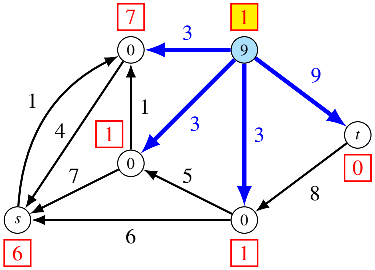


プッシュ

プッシュ・リラベル法の練習問題

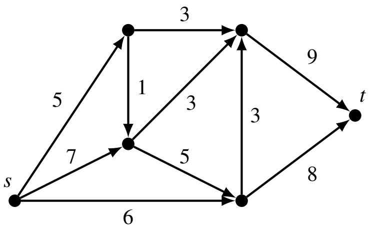


G

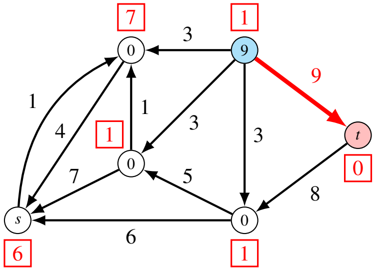


リラベル

プッシュ・リラベル法の練習問題

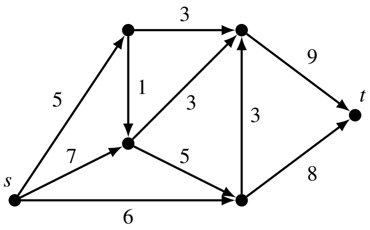


G

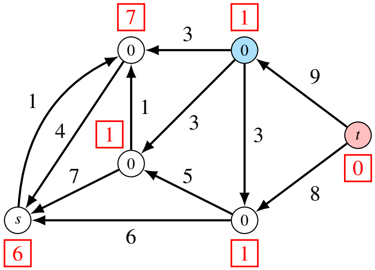


プッシュ

プッシュ・リラベル法の練習問題

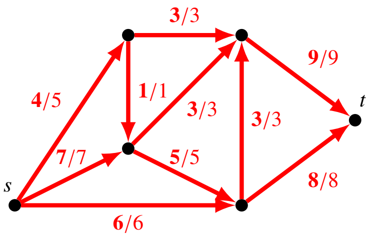


G

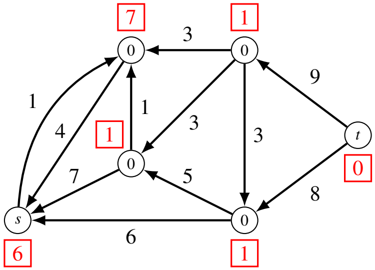


プッシュ

プッシュ・リラベル法の練習問題



G (最大フロー 17)



プッシュ

線形計画法による最大フロー問題の解法

準備

- 始点 $s = 1$，終点 $t = n$ $(n = |V|)$
- $\delta^-(i) = \{(j, i) \in E\}$，$\delta^+(i) = \{(i, j) \in E\}$

決定変数 x_e

辺 e のフロー

線形計画問題としての定式化

$$
\begin{array}{llll}
\text{max} & \displaystyle\sum_{e \in \delta^+(1)} x_e & & \text{(始点から流出するフロー)} \\
\text{s.t.} & \displaystyle\sum_{e \in \delta^+(i)} x_e - \sum_{e \in \delta^-(i)} x_e = 0, & i = 2, 3, \ldots, n-1 & \text{(フロー保存制約)} \\
& x_e \le c_e, & e \in E & \text{(辺容量制約)} \\
& x_e \ge 0, & e \in E & \text{(フロー非負制約)}
\end{array}
$$

フロー保存制約 (flow conservation constraint)

始点・終点以外の頂点では，(フローの流入量) = (フローの流出量)

線形計画法による最大フロー問題の解法

準備

- 始点 $s = 1$，終点 $t = n$ $(n = |V|)$
- $\delta^-(i) = \{(j, i) \in E\}$，$\delta^+(i) = \{(i, j) \in E\}$

決定変数 x_e

辺 e のフロー

線形計画問題としての定式化

$$\begin{array}{llr}
\text{max} & \displaystyle\sum_{e \in \delta^+(1)} x_e \quad \left(= \sum_{e \in \delta^-(n)} x_e \right) & \text{(終点に流入するフロー)} \\
\text{s.t.} & \displaystyle\sum_{e \in \delta^+(i)} x_e - \sum_{e \in \delta^-(i)} x_e = 0, \quad i = 2, 3, \ldots, n-1 & \text{(フロー保存制約)} \\
& x_e \leq c_e, \quad e \in E & \text{(辺容量制約)} \\
& x_e \geq 0, \quad e \in E & \text{(フロー非負制約)}
\end{array}$$

フロー保存制約 (flow conservation constraint)

始点・終点以外の頂点では，(フローの流入量) = (フローの流出量)

接続行列による表現

接続行列 $C = (c_{ij})$

$$c_{ij} = \begin{cases} 1, & e_j = (i,k) \\ -1, & e_j = (k,i) \\ 0, & \text{それ以外} \end{cases}$$

接続行列の第 i 行：頂点 i に入る辺 -1，出る辺 1

準備

$$\boldsymbol{x} = \begin{pmatrix} x_{e_1} \\ x_{e_2} \\ \vdots \\ x_{e_{|E|}} \end{pmatrix}, \quad \boldsymbol{c} = \begin{pmatrix} c_{e_1} \\ c_{e_2} \\ \vdots \\ c_{e_{|E|}} \end{pmatrix}, \quad C = \begin{pmatrix} \boldsymbol{a}_1^\top \\ \boldsymbol{a}_2^\top \\ \vdots \\ \boldsymbol{a}_n^\top \end{pmatrix}$$

線形計画問題としての定式化

$$\begin{aligned} \max \ & \boldsymbol{a}_1^\top \boldsymbol{x} && \text{(始点から流出するフロー)} \\ \text{s.t.} \ & \boldsymbol{a}_i^\top \boldsymbol{x} = 0, \quad i = 2, 3, \ldots, n-1 && \text{(フロー保存制約)} \\ & \boldsymbol{x} \leq \boldsymbol{c} && \text{(辺容量制約)} \\ & \boldsymbol{x} \geq \boldsymbol{0} && \text{(フロー非負制約)} \end{aligned}$$

接続行列による表現

接続行列 $C = (c_{ij})$

$$c_{ij} = \begin{cases} 1, & e_j = (i,k) \\ -1, & e_j = (k,i) \\ 0, & \text{それ以外} \end{cases}$$

接続行列の第 i 行：頂点 i に入る辺 -1，出る辺 1

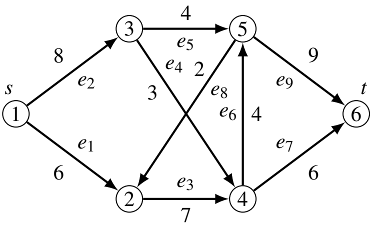
準備

$$\boldsymbol{x} = \begin{pmatrix} x_{e_1} \\ x_{e_2} \\ \vdots \\ x_{e_{|E|}} \end{pmatrix}, \quad \boldsymbol{c} = \begin{pmatrix} c_{e_1} \\ c_{e_2} \\ \vdots \\ c_{e_{|E|}} \end{pmatrix}, \quad C = \begin{pmatrix} \boldsymbol{a}_1^\top \\ \boldsymbol{a}_2^\top \\ \vdots \\ \boldsymbol{a}_n^\top \end{pmatrix}$$

線形計画問題としての定式化

$$\begin{aligned} \text{max} \quad & \boldsymbol{a}_1^\top \boldsymbol{x} \quad (= -\boldsymbol{a}_n^\top \boldsymbol{x}) & & \text{(終点に流入するフロー)} \\ \text{s.t.} \quad & \boldsymbol{a}_i^\top \boldsymbol{x} = 0, \quad i = 2, 3, \ldots, n-1 & & \text{(フロー保存制約)} \\ & \boldsymbol{x} \leq \boldsymbol{c} & & \text{(辺容量制約)} \\ & \boldsymbol{x} \geq \boldsymbol{0} & & \text{(フロー非負制約)} \end{aligned}$$

最大フロー問題の線形計画問題としての定式化の例



$$C = \begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{ccccccccc} e_1 & e_2 & e_3 & e_4 & e_5 & e_6 & e_7 & e_8 & e_9 \end{array} \\ \begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & -1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & -1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 \end{pmatrix} \end{array}$$

線形計画問題としての定式化

$$\begin{array}{ll} \max & x_{e_1} + x_{e_2} \\ \text{s.t.} & -x_{e_1} + x_{e_3} - x_{e_8} = 0 \\ & -x_{e_2} + x_{e_4} + x_{e_5} = 0 \\ & -x_{e_3} - x_{e_4} + x_{e_6} + x_{e_7} = 0 \\ & -x_{e_5} - x_{e_6} + x_{e_8} + x_{e_9} = 0 \\ & x_{e_1} \leq 6 \\ & x_{e_2} \leq 8 \\ & \vdots \\ & x_{e_9} \leq 9 \\ & x_{e_1},\ x_{e_2},\ x_{e_3},\ x_{e_4},\ x_{e_5},\ x_{e_6},\ x_{e_7},\ x_{e_8},\ x_{e_9} \geq 0 \end{array}$$

カット

カット (cut)

- 取り除くと $G = (V, E)$ が 2 分割される辺集合 C
- $S \subset V$ に対し，$C = \{(u, v) \in E \mid u \in S \text{ かつ } v \in V \setminus S\}$

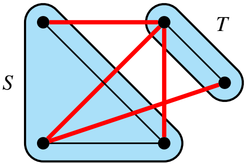
最小カット問題 (minimum cut problem)

重み和 $\displaystyle\sum_{(u,v)\in C} w(u, v)$ が最小となるカット C を見つける問題

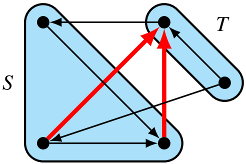
有向カット (directed cut)

$S \subset V$ に対し，$C = \{(u, v) \in E \mid u \in S \text{ かつ } v \in V \setminus S\}$

定義の式は同じだが，有向カットでは S から T へ向かう辺しか考慮しない



カット



有向カット

最大フロー最小カット定理

最大フロー問題

- 有向グラフ $G = (V, E)$
- 辺 (u, v) の容量 $c(u, v)$
- 始点 $s \in V$，終点 $t \in V$

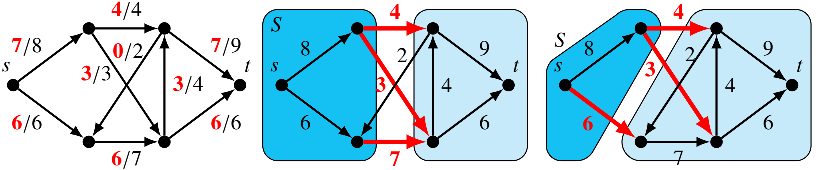
最大フロー最小カット定理 (max-flow min-cut theorem)

(s から t へのフローの最大値) = (s と t を分ける有向カットの容量の最小値)

$s \in S$，$t \notin S$ を満たす $S \subset V$ により定義される有向カット (s-t カット) の容量は，

$$\sum_{u \in S} \sum_{v \in V \setminus S} c(u, v)$$

と表せる．ただし，$(u, v) \notin E$ のとき $c(u, v) = 0$ とする．



最大フロー 13　　s-t カット $(4 + 3 + 7 = 14)$　　最小 s-t カット $(6 + 3 + 4 = 13)$

最大フロー最小カット定理 (続き)

f： 最大フロー

$G_f = (V, E_f)$： f に対応する残余ネットワーク

S： G_f 上で s から到達可能な頂点の集合

証明

(a) G_f に増加路は存在しないので，$t \notin S$．したがって，S は s-t カット C を与える

(b) 最大フローの値 F は S から $V \setminus S$ へ流れるフローの量なので，

$$F = \sum_{u \in S} \sum_{v \in V \setminus S} f(u, v) - \sum_{u \in S} \sum_{v \in V \setminus S} f(v, u)$$

(c) $(u, v) \in E$ ($u \in S$，$v \in V \setminus S$) のとき，$f(u, v) = c(u, v)$.
もし $f(u, v) < c(u, v)$ なら，$r(u, v) = c(u, v) - f(u, v) > 0$ より $(u, v) \in E_f$．したがって，v も s から到達可能であり，$v \notin S$ に反する．

(d) $(v, u) \in E$ ($u \in S$，$v \in V \setminus S$) のとき，$f(v, u) = 0$.
もし $f(v, u) > 0$ なら，$r(u, v) = f(v, u) > 0$ より $(u, v) \in E_f$．v も s から到達可能であり，$v \notin S$ に反する．

(e) (b)，(c)，(d) より，最大フローの量 F は

$$F = \sum_{u \in S} \sum_{v \in V \setminus S} c(u, v)$$

これは s-t カット C の容量の和と一致する．