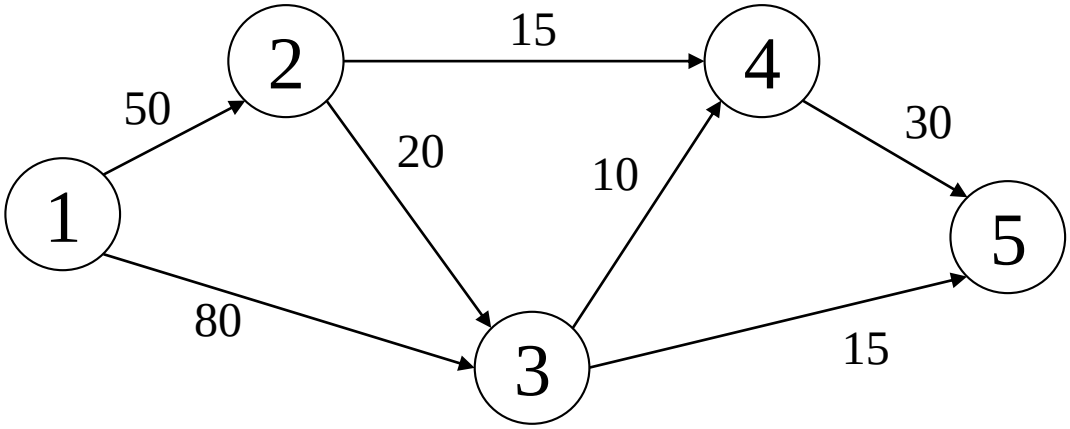
算法数理工学
第9回

定兼　邦彦

グラフ

- グラフ $G = (V, E)$
  - V: 頂点 (節点) 集合 $\{1,2,\ldots,n\}$
  - E: 枝集合, $E \subseteq V \times V = \{(u,v) \mid u, v \in V\}$

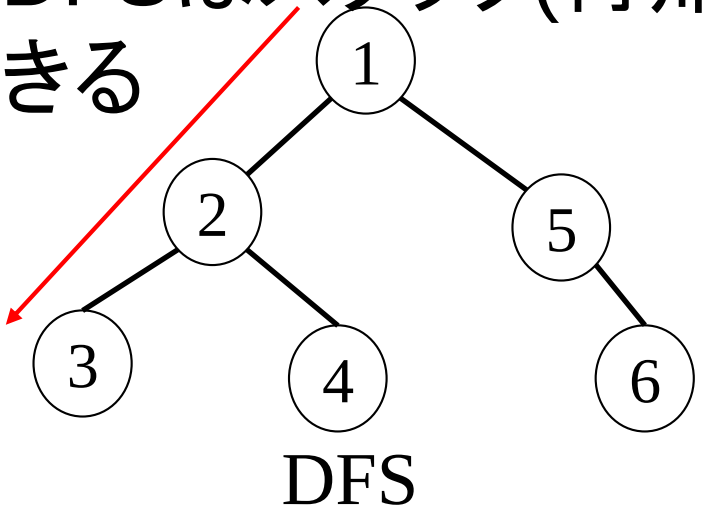
- 無向グラフ
  - 枝は両方向にたどれる



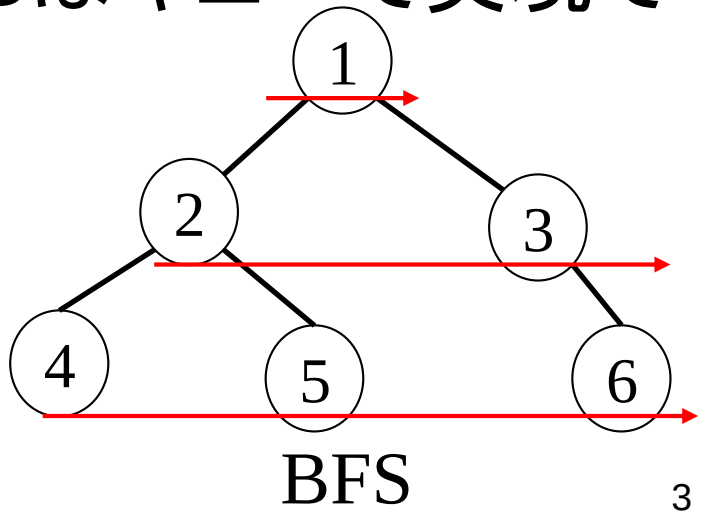
- 有向グラフ
  - 枝 (u,v) は u から v の方向のみたどれる
  - 注: 無向グラフは有向グラフで表現できる (枝数を倍にする)
- 枝には重み(長さ)がついていることもある
  - なければ全て長さ 1 とみなす

深さ優先探索，幅優先探索

- 木やグラフの探索法
- 深さ優先探索 (depth-first search, DFS)
  - 行き止まりになるまで先に進む
- 幅優先探索 (breadth-first search, BFS)
  - 全体を同時に探索する
- DFSはスタック(再帰)，BFSはキューで実現できる



DFS



BFS

DFS(G)

1. for each $u \in V[G]$
2. $color[u] \leftarrow$ 白
3. $\pi[u] \leftarrow$ NIL
4. $time \leftarrow 0$
5. for each $u \in V[G]$
6. if $color[u]$ = 白
7. DFS-Visit(u)

DFS-Visit(u)

1. $color[u] \leftarrow$ 灰
2. $time \leftarrow time + 1$
3. $d[u] \leftarrow time$
4. for each $v \in Adj[u]$
5. if $color[v]$ = 白
6. $\pi[v] \leftarrow u$
7. DFS-Visit(v)
8. $color[u] \leftarrow$ 黒
9. $time \leftarrow time + 1$
10. $f[u] \leftarrow time$

- 辺 (u, v) をたどったら，v の親を u にする $(\pi[v] = u)$
- グラフ $G_\pi = (V, E_\pi)$ を次のように定義する
  - $E_\pi = \{(\pi[v], v) \in E \mid v \in V$ かつ $\pi[v] \neq \mathrm{NIL}\}$
- G_π は深さ優先探索森 (depth-first forest)
  - 連結ならば深さ優先探索木 (depth-first tree)
- 各点は探索前は白,探索中は灰,探索終了後は黒
- 各点 v は2つのタイムスタンプを持つ
  - $d[v]$ は v を最初に発見した (灰色にした) 時刻
  - $f[v]$ は v の隣接リストを調べ終わった (黒にした) 時刻
- タイムスタンプは 1 から $2n$ の整数
- 全ての u に対し $d[u] < f[u]$
- DFSの実行時間は $\Theta(n+m)$

- 定理 (括弧付け定理):
  グラフ G に対する任意の深さ優先探索を考える.
  任意の2つの頂点 u, v に対し，以下の3つの条件の中の1つだけが成立する.
  - 区間 $[d[u], f[u]]$ と区間 $[d[v], f[v]]$ には共通部分が無く，深さ優先森において u と v はどちらも他方の子孫ではない.
  - 区間 $[d[u], f[u]]$ は区間 $[d[v], f[v]]$ に完全に含まれ，u が v の子孫となる深さ優先探索木が存在する
  - 区間 $[d[v], f[v]]$ は区間 $[d[u], f[u]]$ に完全に含まれ，v が u の子孫となる深さ優先探索木が存在する
- 注:2つの区間が部分的に重なることは無い

証明: まず $d[u] < d[v]$ の場合を考える

- $d[v] < f[u]$ のとき
  - u がまだ灰色のときに v が発見されている. つまり v は u の子孫.
  - v は u よりも後に発見されたので, u の探索が終わる前に v の探索は終わる. つまり $f[v] < f[u]$.
  - このとき区間 $[d[v], f[v]]$ は区間 $[d[u], f[u]]$に含まれる.
- $f[u] < d[v]$ のとき
  - $d[u] < f[u] < d[v] < f[v]$ なので区間に共通部分は無い.
  - つまり一方が探索中に他方が発見されない.
  - よってどちらも他方の子孫では無い
- $d[v] < d[u]$ の場合も同様に証明できる.

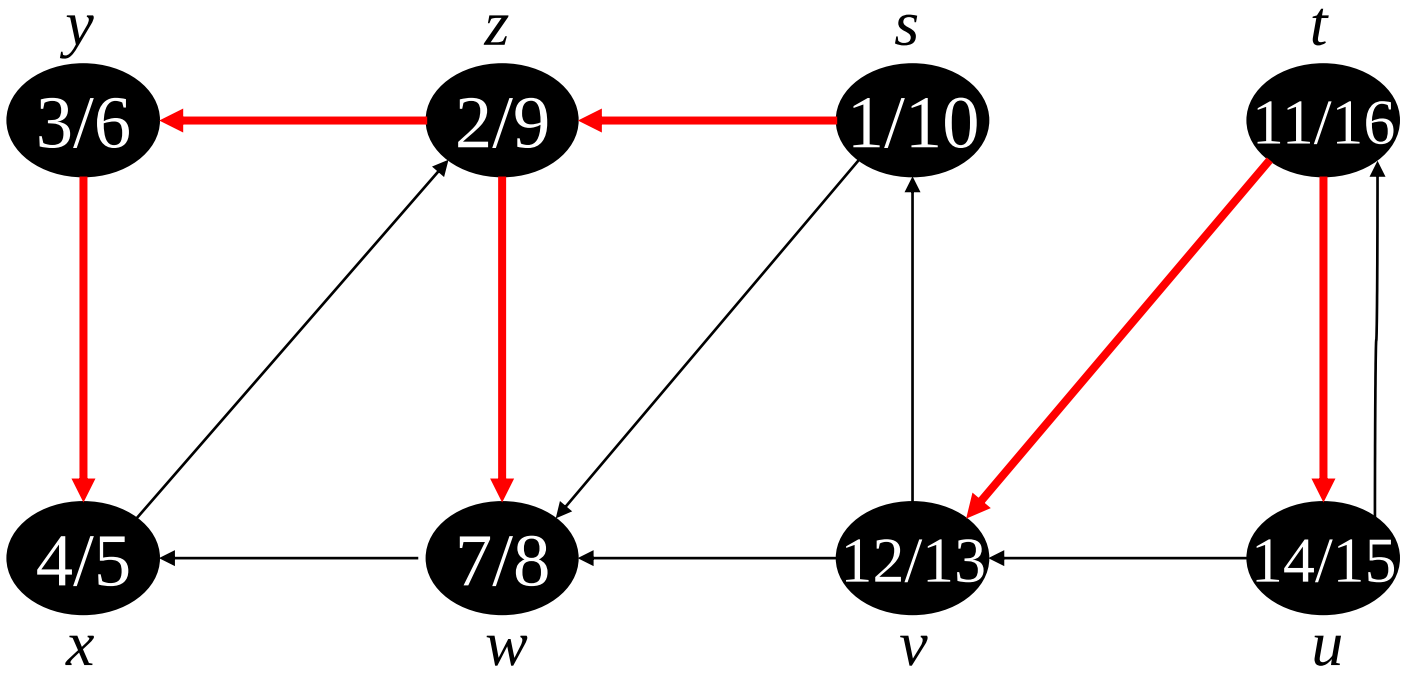
- 系: 有向または無向グラフに対する深さ優先探索森において頂点 v が 頂点 u $(\neq v)$ の子孫であるための必要十分条件は

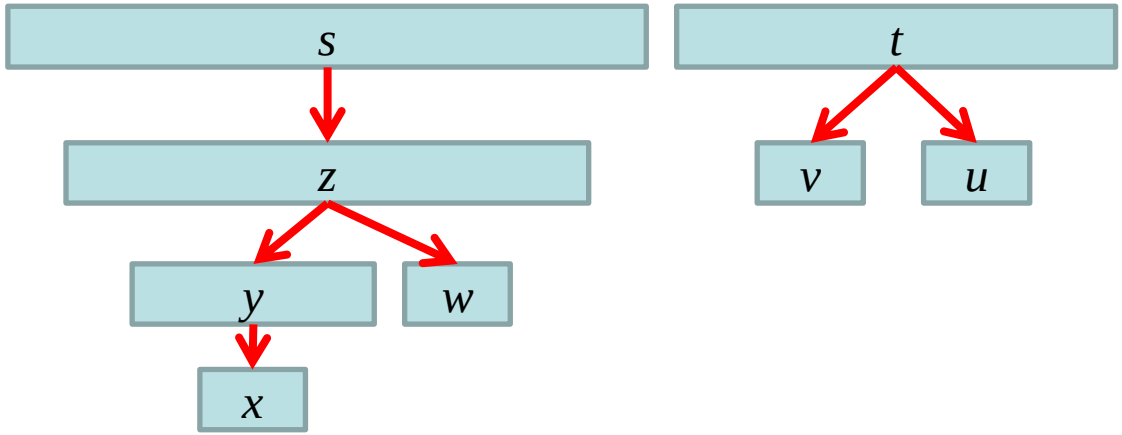
  $d[u] < d[v] < f[v] < f[u]$

- 定理 (白頂点経路定理): グラフ G の深さ優先探索森において, 頂点 v が 頂点 u の子孫であるための必要十分条件は, 探索が u を発見する時刻 $d[u]$ に u から v に至る白頂点だけからなるパスが存在することである.

- 証明:⇒) v が u の子孫であると仮定する. uv間のパス上の任意の頂点を w とする. w は u の子孫なので, $d[u] < d[w]$. つまり時刻 $d[u]$ では w は白.
- ⇐) 時刻 $d[u]$ に u から v に至る白頂点だけからなるパスが存在するが, 深さ優先探索木において v が u の子孫にならないと仮定する. 一般性を失うことなく, このパス上の v 以外の全ての頂点は u の子孫になると仮定できる (そのような点で u に一番近いものを考える).

- このパス上で v の直前の点を w とする.
- w は u の子孫 (uを含む) なので $f[w] \leq f[u]$.
- v の発見は u の後で, w の終了の前なので, $d[u] < d[v] < f[w] \leq f[u]$.
- 区間は部分的には重ならないので $f[v] < f[u]$. つまり v は u の子孫.





1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

(s (z (y (x x) y) (w w) z) s) (t (v v) (u u) t)

辺の分類

- グラフ G の探索森 G_π を1つ固定する.
  G の辺を4種類に分類する

1. tree edge (木辺)
   - G_πの辺
2. back edge (後退辺)
   - 辺 (u, v) で, u と v がある探索木の頂点とその先祖の場合. セルフループも後退辺とみなす.
3. forward edge (前進辺)
   - 辺 (u, v) で, 木辺でなく, u と v がある探索木の頂点とその子孫の場合.
4. cross edge (横断辺)
   - 上の3つ以外

定理: 無向グラフ G を深さ優先探索するとき，G の任意の辺は木辺または後退辺である.

証明: (u, v) を G の任意の辺とし，一般性を失うことなく $d[u] < d[v]$ と仮定する.

v は u の隣接リストに含まれるので，u に対する処理が終わる前に v に対する処理が終わる.

辺 (u, v) が最初に u から v に向かって探索されたなら，v はその時点では白である．すると (u, v) は木辺.

辺 (u, v) が最初に v から u に向かって探索されたなら，u はその時点では灰である．すると (u, v) は後退辺.

定理:

・辺 (u,v) が木辺または前進辺

$\Leftrightarrow d[u] < d[v] < f[v] < f[u]$

・辺 (u,v) が後退辺

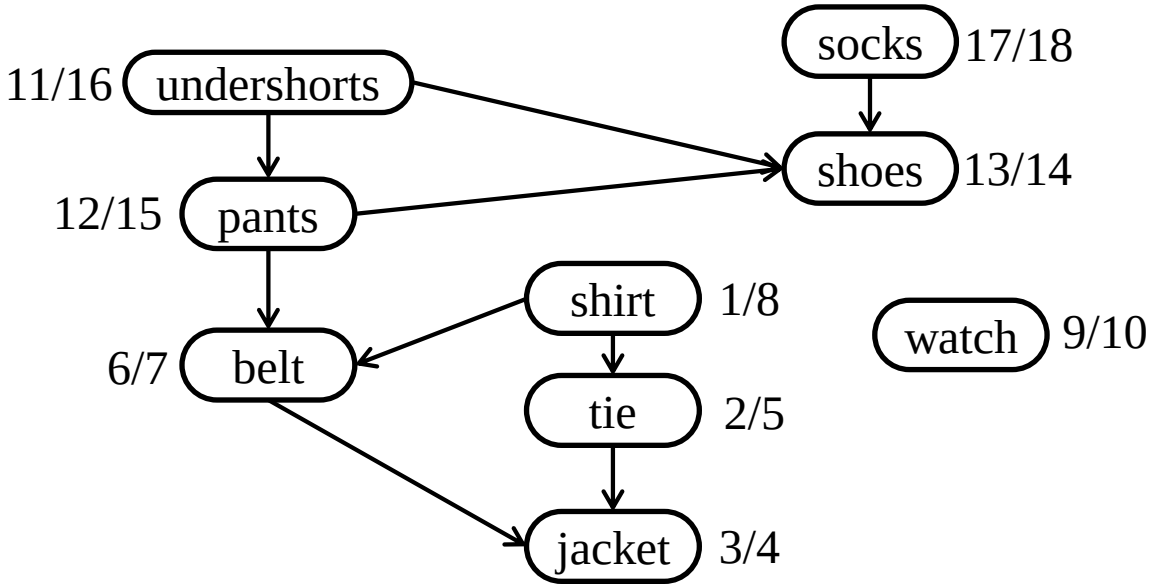
$\Leftrightarrow d[v] < d[u] < f[u] < f[v]$

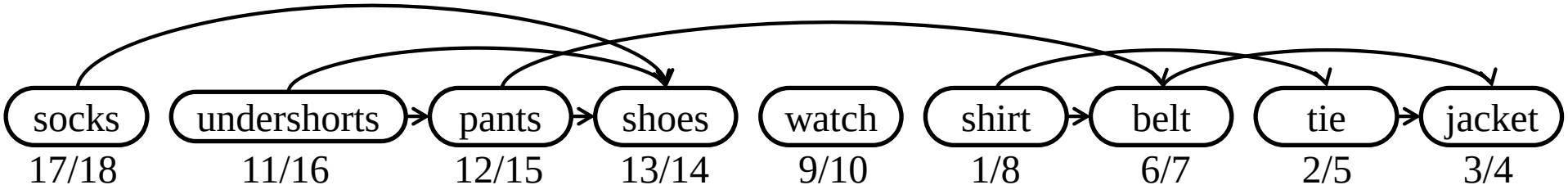
・辺 (u,v) が横断辺

$\Leftrightarrow d[u] < f[u] < d[v] < f[v]$

トポロジカルソート

- 入力: 閉路のない有向グラフ $G = (V, E)$
- 出力: Gの全頂点の線形順序で,
  G が辺 (u, v) を含む ⇔ この線形順序で u が v より先に現れる
- 注: グラフに閉路があればそのような線形順序は存在しない
- 半順序しか定義されていない集合に，それと矛盾しない全順序を1つ与えることに相当する.
- 線形 ($\Theta(n+m)$) 時間で求まる.
- 閉路を持たない有向グラフは DAG (directed acyclic graph) と呼ばれる





全ての有向辺が左から右の向きになっている
注:トポロジカルソートは1通りでは無い

TOPOLOGICAL-SORT(G)

1. $L \leftarrow \varnothing$ (空リスト)
2. for each $u \in V[G]$
3. $\quad color[u] \leftarrow$ 白
4. for each $u \in V[G]$
5. $\quad$ if $color[u]$ = 白
6. $\quad\quad$ Visit(u)

Visit(u)

1. if $color[u]$ = 灰
2. $\quad$ stop (DAGでは無い)
3. if $color[u]$ = 黒 return
4. $color[u] \leftarrow$ 灰
5. for each $v \in Adj[u]$
6. $\quad$ Visit(v)
7. $color[u] \leftarrow$ 黒
8. L の先頭に u を入れる

時間計算量: $\Theta(n+m)$

補題1: 有向グラフ G に閉路が存在しない
⇔ G を深さ優先探索した時に後退辺を生成しない

証明: →) 後退辺 (u, v) が存在すると仮定する.
つまり深さ優先探索木において v は u の祖先.
v から u のパスに後退辺(u, v)を加えると閉路になる.
←) G が閉路 c を含むと仮定する. c の中で最初に訪れられる点を v とし, c において v に入る辺を(u, v)とする. 時刻 $d[v]$ では, c 内の頂点は全て白である.
よって v から u まで白頂点のみからなるパスが存在するため, vは uの先祖であり, (u, v)は後退辺となる.

補題: トポロジカルソートのアルゴリズム実行時に辺 (u, v) が探索されたとする. グラフに閉路が無い $\Leftrightarrow$ v は灰色ではない.
(つまり, v が灰色なら解は存在しない)

証明: v が灰色なら, v は u の先祖であり, (u, v) が後退辺となる. 前の補題からグラフは閉路を持つ.
グラフに閉路があるなら, 後退辺 (u, v) が存在し, v は灰色である.

定理: TOPOLOGICAL-SORT(G)は有向グラフGが閉路を持たないなら G をトポロジカルソートし，閉路を持つならその1つを発見する.

証明: グラフが閉路を持つなら後退辺 (u, v) が存在し，v の色は灰となる．アルゴリズムは全ての辺を調べるので，閉路があるなら必ず発見する.
以下ではグラフが閉路を持たない場合を考える.

頂点はその探索の終了時にLの先頭に挿入される.
つまり L には頂点は終了時刻 $f[]$ の降順に入っている. よって, G の中に辺 (u, v) があるときに $f[u] > f[v]$ となっていることを示せばいい.

アルゴリズムが辺 (u, v) を探索する時点を考える.
v の色は白か黒である. v が白なら v は u の子孫になり, $f[u] > f[v]$ が成立する. v が黒なら v の処理は既に終了しているので, $f[v]$ は既に決定されている. u は探索中なので $f[u]$ は未決定である.
よって $f[u] > f[v]$ となる.

強連結成分分解

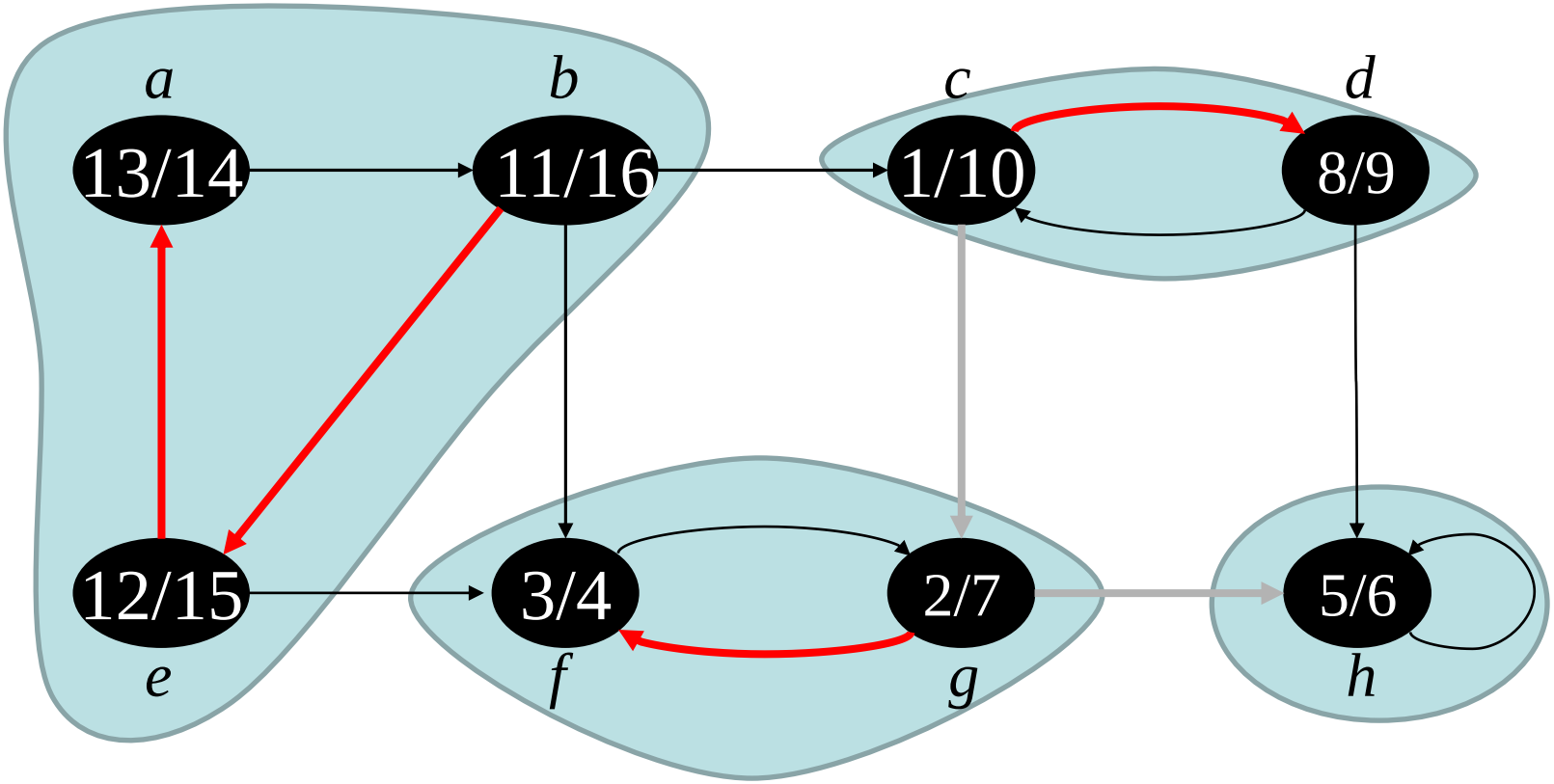
定義: 有向グラフ $G = (V, E)$ の強連結成分 (strongly connected component) とは，次の条件を満たす頂点の集合 $C \subseteq V$ で極大なものと定義する.

- C の全ての頂点対に対して $u \rightsquigarrow v$ かつ $v \rightsquigarrow u$

($u \rightsquigarrow v$ はuからvへ有向パスが存在することを表す.)

Cが極大とは，$C \subset C'$, $C \neq C'$ となる強連結成分 C' は無いという意味

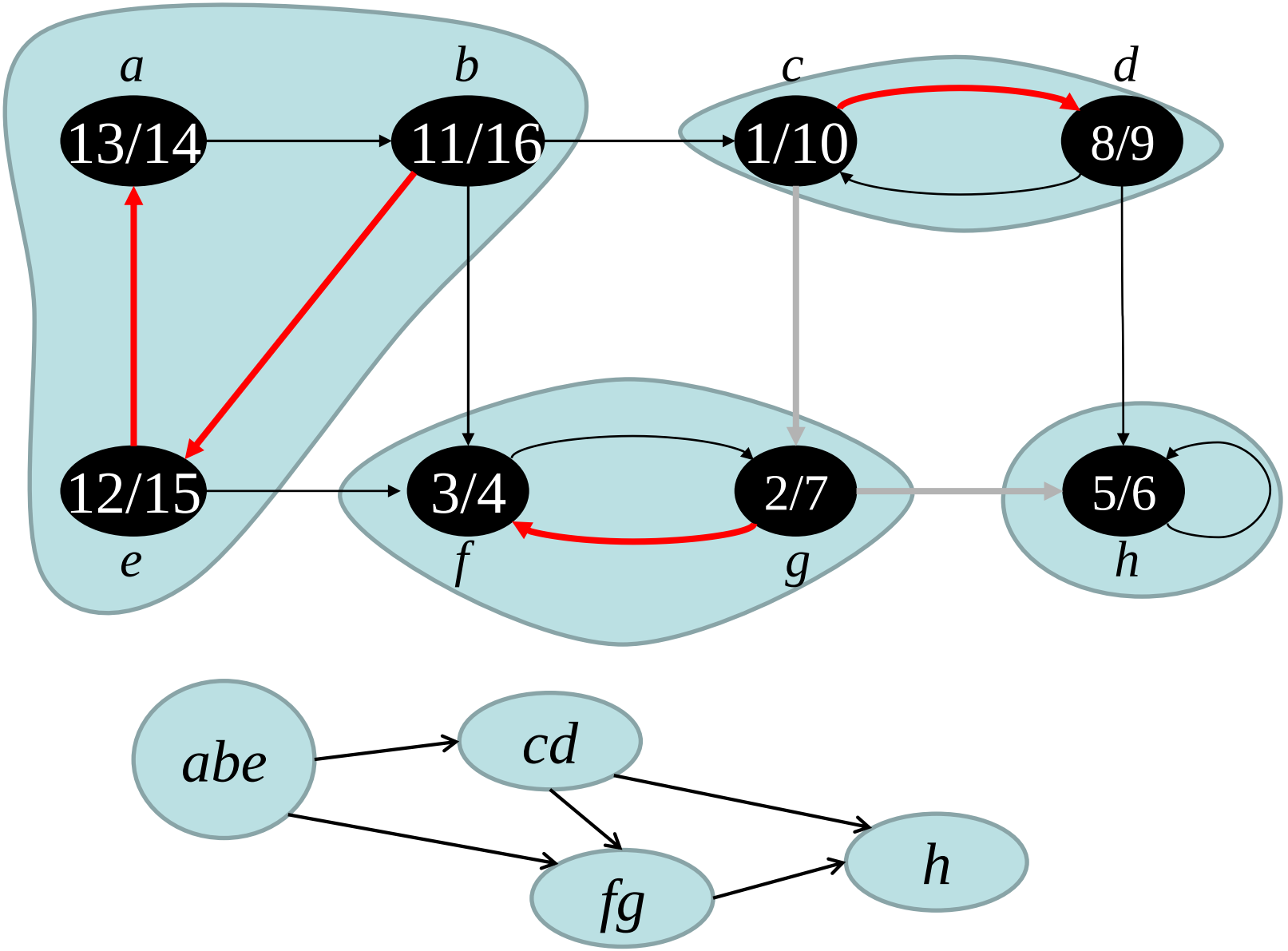
つまり，u と v は互いに到達可能である.



4つの強連結成分 (SCC) を持つ

注: 各SCCは1つの閉路とは限らない

1点でもSCCになる



各強連結成分を1点に縮約すると，DAGになる

定義: グラフ $G = (V, E)$ の転置とは，次のように定義されるグラフ $G^{\mathrm{T}} = (V, E^{\mathrm{T}})$ である.

$$E^{\mathrm{T}} = \{(v, u) \mid (u, v) \in E\}$$

G^{T} の隣接行列は，G の隣接行列の転置になる.

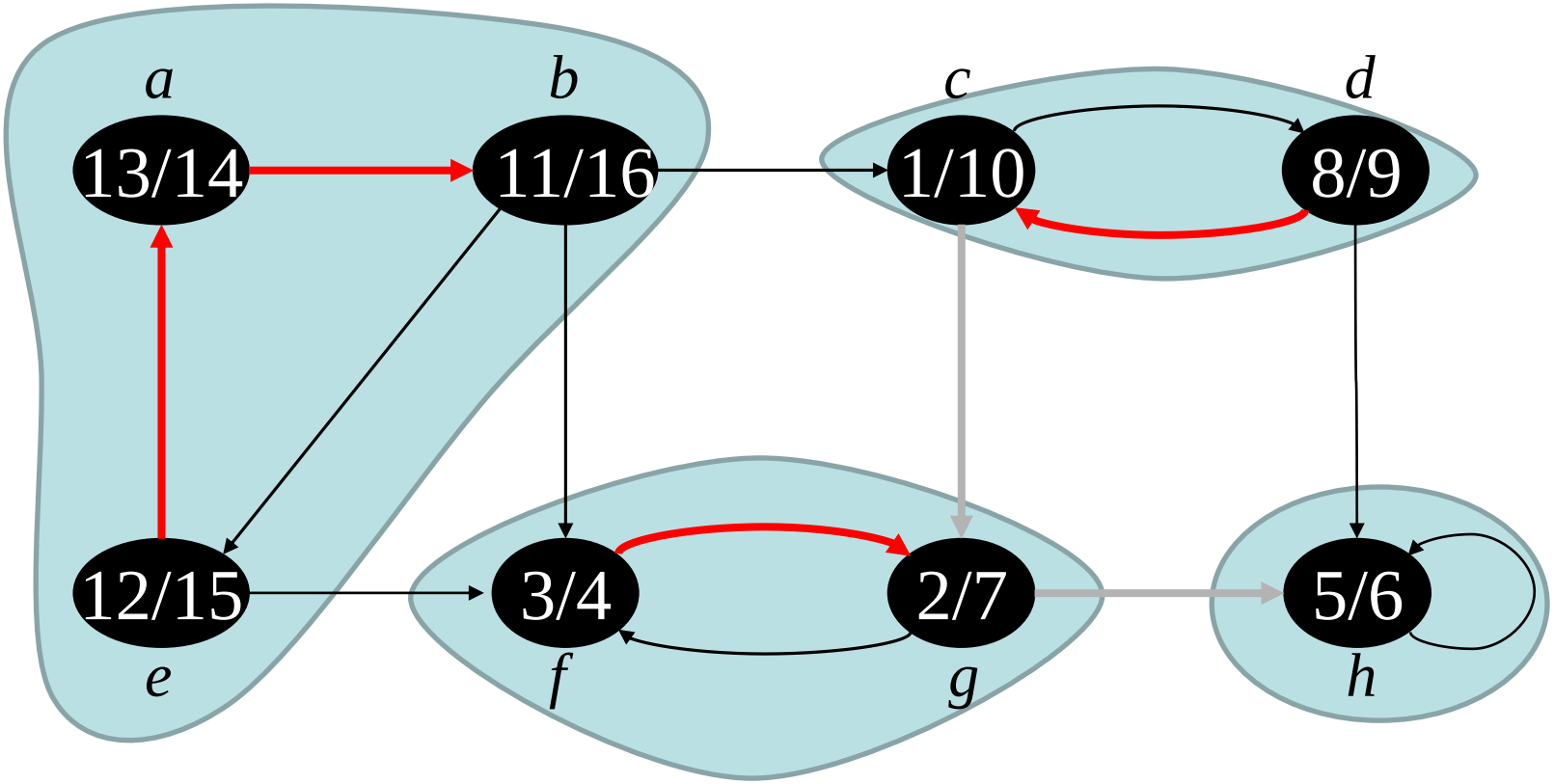
G^{T} は線形時間 $\Theta(n+m)$ で求まる.

G の各頂点 u に対し，その隣接点 $v \in Adj[u]$ を列挙し，辺 (v, u) を G^{T} の v の隣接リストに挿入

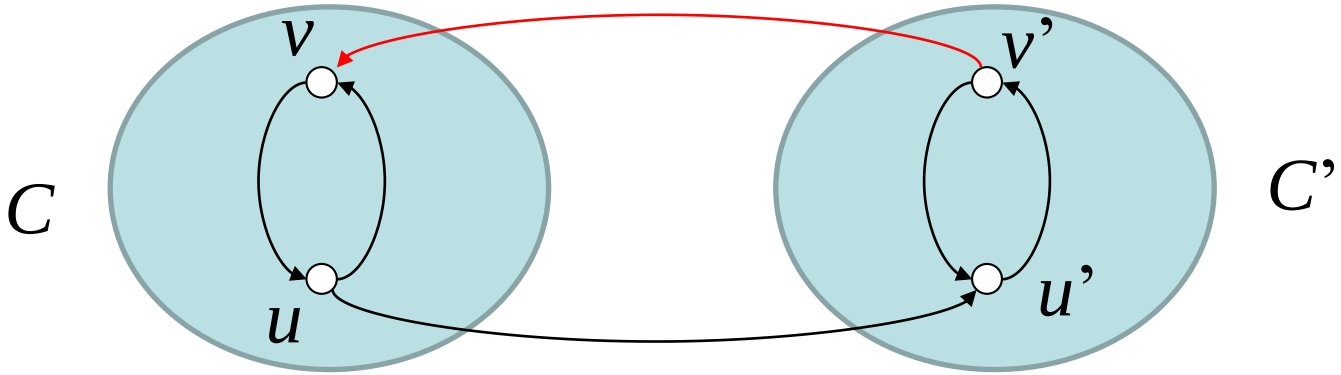
SCC(G) // Strongly-Connected-Components

1. DFS(G) を呼び出し，各頂点 u に対して終了時刻 $f[u]$ を計算する
2. G^{T} を計算する
3. DFS(G^{T}) を呼び出す．ただし 1. で計算した $f[u]$ の降順で頂点を探索する
4. 3. で生成した深さ優先探索森の各木の頂点集合を1つの強連結成分として出力する

$\Theta(n+m)$ 時間


a
b
c
d
13/14
11/16
1/10
8/9
12/15
3/4
2/7
5/6
e
f
g
h

補題: 有向グラフ $G=(V, E)$ の異なる2つの強連結成分を C と C', $u, v \in C$, $u', v' \in C'$ とする.
G に経路 $u \rightsquigarrow u'$ が存在するとき, G には経路 $v' \rightsquigarrow v$ は存在しない.
証明: G に経路 $v' \rightsquigarrow v$ が存在すると仮定すると, 経路 $u \rightsquigarrow u' \rightsquigarrow v'$ と $v' \rightsquigarrow v \rightsquigarrow u$ が存在する.
従って u と u' は互いに到達可能であり, C と C' が異なる強連結成分であることに矛盾する.

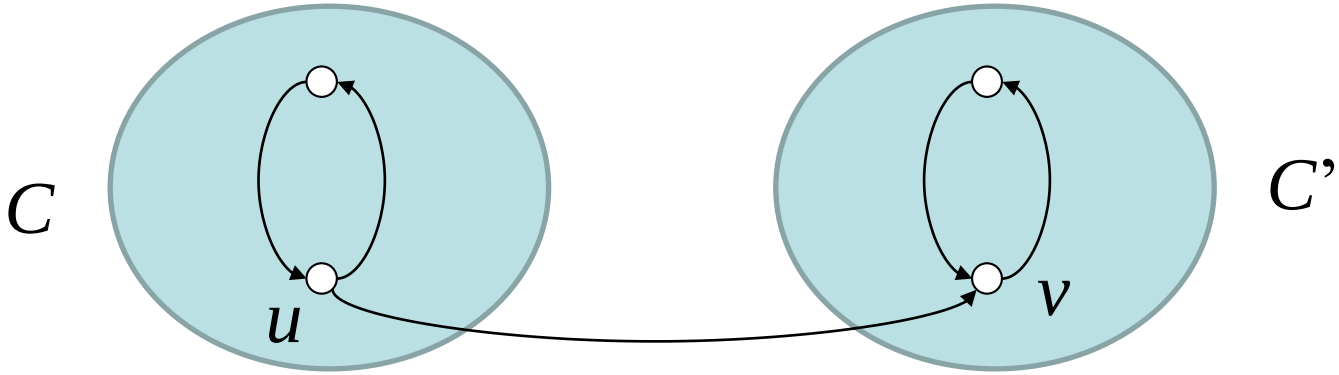


定義: $d(U) = \min_{u \in U} \{d[u]\}$

$f(U) = \max_{u \in U} \{f[u]\}$

つまり，$d(U)$ と $f(U)$ は U の頂点の発見時刻と終了時刻の中で最も早い時刻と最も遅い終了時刻.

補題: 有向グラフ $G=(V, E)$ の異なる2つの強連結成分を C と C' とする．$u \in C$, $v \in C'$ である辺 (u, v) が存在するなら，$f(C) > f(C')$ である.

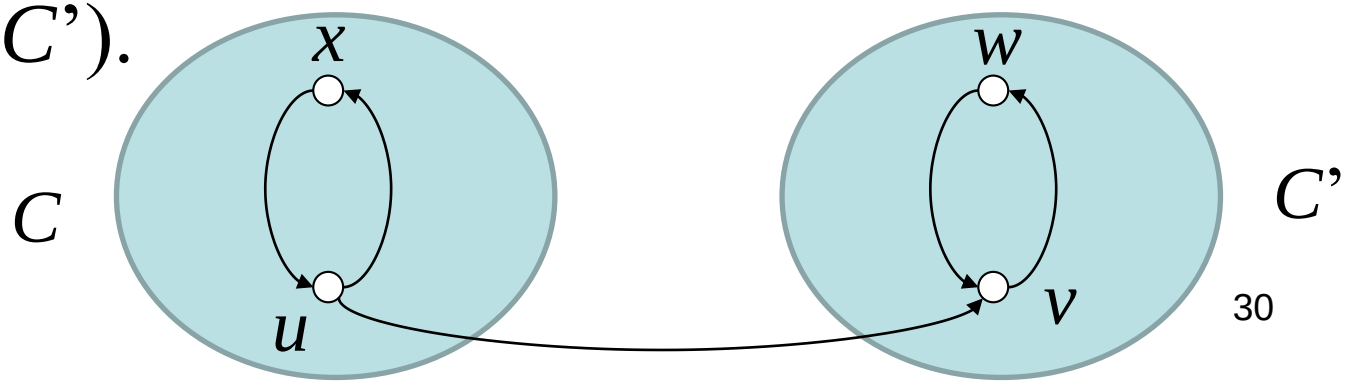


証明: $d(C) < d(C')$ の場合

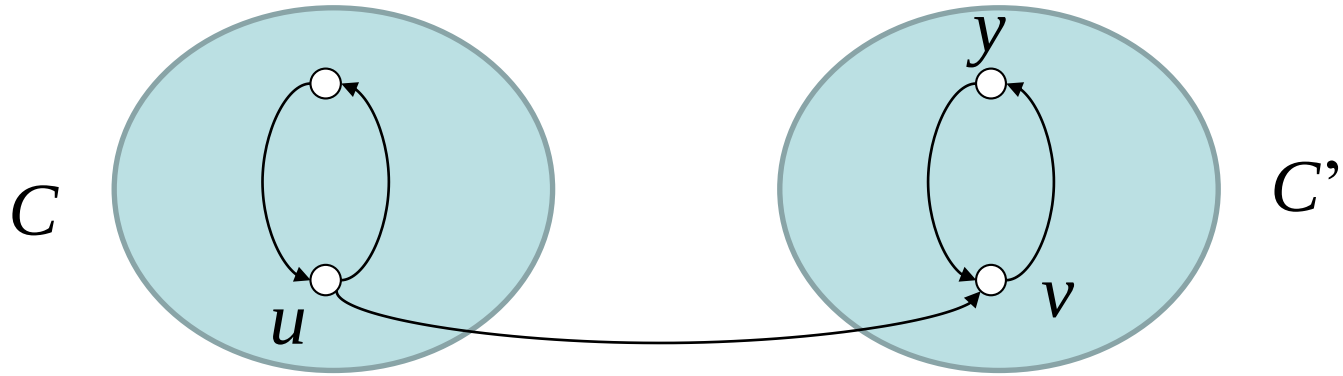
C の中で最初に発見される点を x とする.

時刻 $d[x]$ では C と C' の点は全て白で, x から u に至る白頂点だけからなるパスが G に存在する.

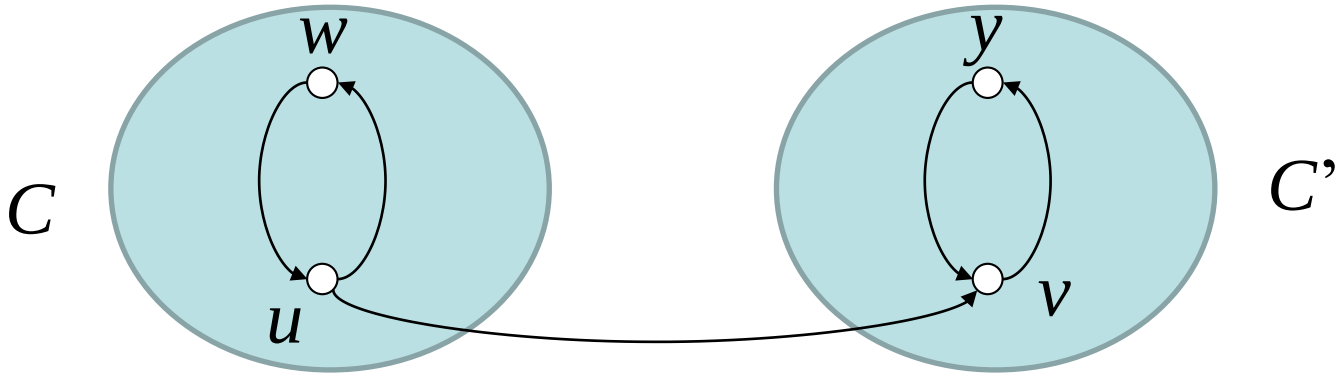
$(u, v) \in E$ なので, 時刻 $d[x]$ では x から任意の頂点 $w \in C'$ に至る白頂点だけからなるパスが G に存在する. つまり $x \rightsquigarrow u \rightarrow v \rightsquigarrow w$ である. 白頂点経路定理より C' の全頂点はDFS木で x の子孫となり, $f[x] = f(C) > f(C')$.



$d(C) > d(C')$ の場合．C' の中で最初に発見される点を y とする．時刻 $d[y]$ では C' の全頂点は白で，y から C' の各頂点へ至る白頂点からなるパスが存在する．白頂点経路定理より，C' の全頂点はDFS木において y の子孫となり，$f[y] = f(C')$ となる．時刻 $d[y]$ では C の全頂点は白である．C から C' に辺 (u, v) が存在するので，C' から C にはパスは存在しない．



C の任意の頂点は y から到達不可能なので,
時刻 $f[y]$ には C の全頂点はまだ白のままである.
従って, 任意の頂点 $w \in C$ に対して $f[w] > f[y]$,
すなわち $f(C) > f(C')$ が成立する.



系1: C と C' を有向グラフ $G=(V, E)$ の異なる2つの強連結成分とする. $u \in C$ と $v \in C'$ の間に辺 $(u, v) \in E^{\mathrm{T}}$ があるとき, $f(C) < f(C')$ である.

定理: アルゴリズム SCC は有向グラフ G の強連結成分を正しく求める.

証明: 帰納法で示す. アルゴリズムの3行目で求めた最初の k 個の木が強連結成分のときに, k+1 個目も強連結成分であることを示す.

最初の 0 個の木は強連結成分である.

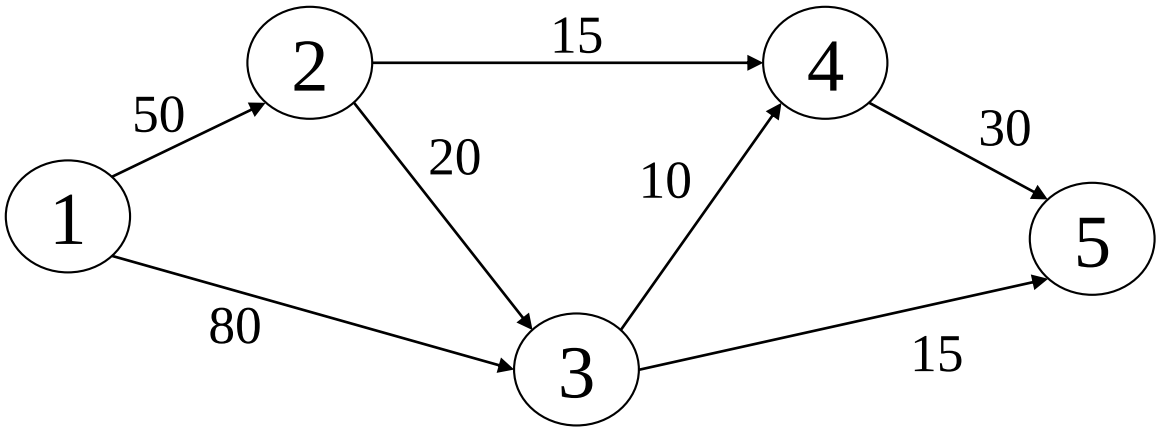
(k+1) 番目の木の根を u とし, u の属する強連結成分を C とする. この後で訪問する C 以外の任意の強連結成分 C' に対して $f[u] = f(C) > f(C')$ が成立する.

帰納法の仮定(最初の k 個の木が強連結成分)から, C の一部の頂点が既に訪れられていることはない.

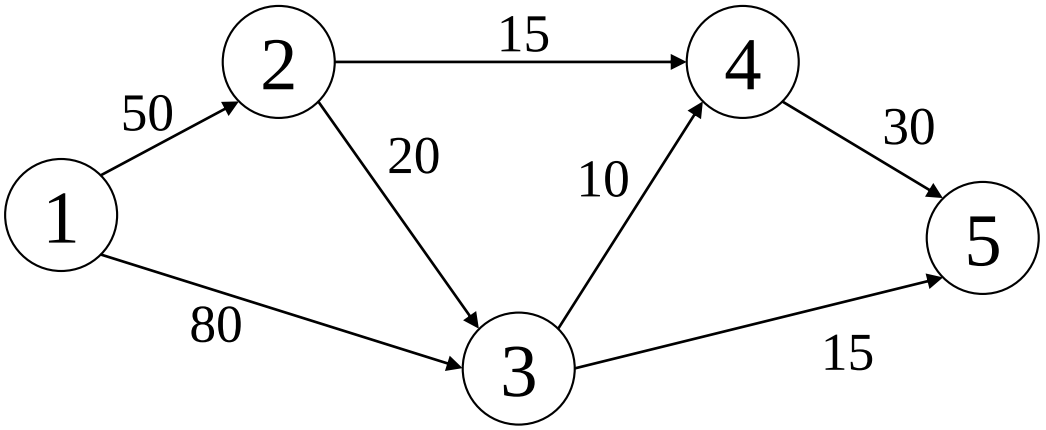
白頂点経路定理より，u からの深さ優先探索木において C の他のすべての頂点は u の子孫になる．さらに，帰納法の仮定と系1から，G^{T}の C から出る任意の辺は既に訪問された強連結成分への辺である．従って，C 以外の任意の強連結成分に属するどの頂点も，u からの深さ優先探索によって u の子孫になることはない．よって，u を根とする深さ優先探索木の全頂点は1つの強連結成分に対応する．

最短路問題

- 有向グラフ $G = (V, E)$ を考える
  - V: 節点集合, 節点数 n
  - E: 枝集合, 枝数 m
- グラフ G の各枝 $(i,j) \in E$ は長さ a_{ij} を持つ
- ある節点 $s \in V$ から別の節点 $t \in V$ への路の中で,
  最も長さの短いものを見つける問題を最短路問題という



- 下のネットワークで節点 1 を s, 節点 5 を t とすると, s から t へは $1\to2\to4\to5$, $1\to2\to3\to5$,$1\to3\to4\to5$, $1\to2\to3\to4\to5$,$1\to3\to5$ の5つの路がある
- 路の長さはそれぞれ 95, 85, 120, 110, 95 なので, 最短路は $1\to2\to3\to5$ である
- 大規模なネットワークでは全ての路を数え上げる方法は実用的ではない

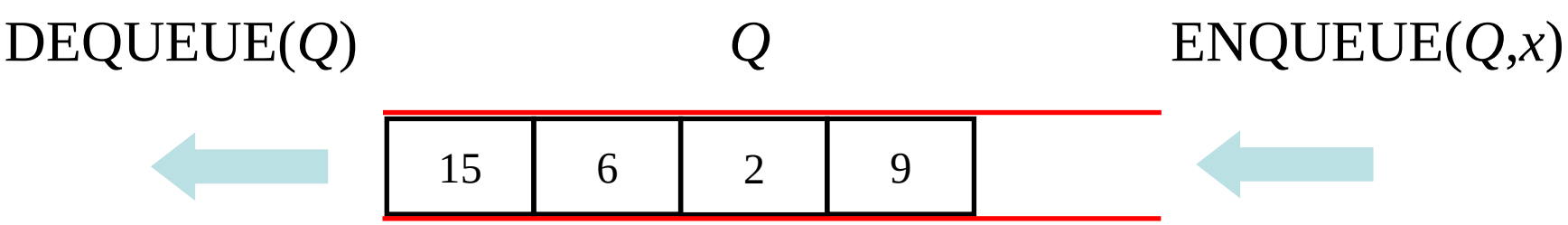


重み無しグラフの場合

- 重み無し ⇔ 全ての枝の重みが 1
- 幅優先探索で求まる
  - $O(n+m)$ 時間
- s からの距離が d の点の集合を V_d とする $(d \geq 0)$
  - $V_0 = \{s\}$
  - V_d = (V_{d-1}の点から枝1本で行ける点の集合)
    - ($V_0, V_1, \ldots, V_{d-1}$ に入っている点の集合)
- V_dを求めるには，V_{d-1} の各点 v に対し，v の隣接点でまだ訪れていない点を求めればいい
- V_dをキューで管理する

キュー (Queue)

- INSERT, DELETEの代わりにENQUEUE, DEQUEUEと呼ぶ
- 人の並んだ列と同じ
  - ENQUEUEでは列の最後に追加
  - DEQUEUEでは列の先頭を取り出す



- 各頂点に色を付ける
  - 白: 未発見
  - 灰: 発見済みだが隣接リストの探索が未完了
  - 黒: 発見済みで隣接リストの探索も完了
- 初期状態: s 以外の全ての点 u に対し
  - $color[u]$ = 白
  - $d[u] = \infty$ (s からの距離)
  - $\pi[u]$ = NIL (最短路木での親)
- 始点 s については
  - $color[s]$ = 灰
  - $d[s] = 0$
  - $\pi[s]$ = NIL (最短路木での根)

BFS(G, s)

1. $Q \leftarrow \emptyset$ (空集合)
2. ENQUEUE(Q, s)
3. while $Q \neq \emptyset$
4. $u \leftarrow$ DEQUEUE(Q)
5. for each $v \in Adj[u]$
6. if $color[v]$ = 白
7. $color[v]$ = 灰
8. $d[v] \leftarrow d[u] + 1$
9. $\pi[v] \leftarrow u$
10. ENQUEUE(Q, v)
11. $color[u] \leftarrow$ 黒

計算時間の解析

- 初期化以外で色が白になる点は無い
- ある点がキューに入れられるとき，その色は白で，キューに入ると灰になる

  ⇒各点は高々1回だけキューに挿入・削除される
- 各点 u の隣接リストが走査されるのは u がキューから削除されるときのみ

  ⇒各点の隣接リストは高々1回だけ走査される

  ⇒隣接リストの走査にかかる時間は全体で $O(m)$
- 初期化の時間は $O(n)$
- 全体で $O(n+m)$

正当性の証明

- 定義: s から v への最短路距離 (shortest-path distance) $\delta(s, v)$ を，s から v へのパスの辺数の最小値と定義する. パスが無ければ $\delta(s, v) = \infty$.
- 補題1: 有向または無向グラフ G において，
  任意の辺 $(u, v) \in E$ に対し，
  $$\delta(s, v) \leq \delta(s, u) + 1$$
- 証明: u が s から到達可能ならば v へも到達可能で，s から v へのパスは s から u への最短路に (u, v) を追加したものか，それより短いものとなる.
  u が s から到達不可能ならば $\delta(s, u) = \infty$ なので不等式は成立する.

- 補題2：BFSが停止した時，各点 $v \in V$ に対し $d[v] \geq \delta(s, v)$ が成立する．
- 証明：ENQUEUEの操作回数に関する帰納法で証明する．最初の s だけが挿入された段階では $d[s] = 0 = \delta(s, s)$，全ての点 $v \in V - \{s\}$ に対し $d[v] = \infty \geq \delta(s, v)$ なので成立．
- 点 u の隣接点 v を探索する場合を考える．
  帰納法の仮定により $d[u] \geq \delta(s, u)$．
  $d[v]$ を更新すると

$$\begin{aligned} d[v] &= d[u] + 1 \\ &\geq \delta(s, u) + 1 \\ &\geq \delta(s, v) \quad \text{(補題1より)} \end{aligned}$$

  となり，ENQUEUE後も成り立つ

- 補題3: BFSの実行中で，キュー Q が点 $\langle v_1, v_2, \ldots, v_r \rangle$ を含むとする (v_1が先頭，v_rが末尾).
  このとき $d[v_r] \leq d[v_1]+1$ かつ
  $d[v_i] \leq d[v_{i+1}]$ $(i=1,2,\ldots,r-1)$
- 証明: キューの操作回数に関する帰納法を用いる.
  初期段階はキューは s だけを含むので成り立つ.
- キューの先頭 v_1 を削除すると v_2 が新たな先頭になる. 帰納法の仮定より $d[v_1] \leq d[v_2]$.
  このとき $d[v_r] \leq d[v_1]+1 \leq d[v_2]+1$ となり成立.
- キューに頂点 v を挿入するとき ($v_{r+1} = v$ となる)
  現在頂点 u の隣接リストを走査しているとする.

- $d[v_{r+1}] = d[v] = d[u]+1$(アルゴリズムの動作)
- u を削除する前のキューは$\langle u = v_1, v_2, \ldots, v_r \rangle$なので帰納法の仮定から $d[v_r] \leq d[u]+1$.
- 従って $d[v_r] \leq d[u]+1 = d[v] = d[v_{r+1}]$.
  また, $d[v_{r+1}] = d[u]+1 = d[v_1]+1$.
- よって, u を削除後のキューでも $d[v_{r+1}] \leq d[v_1]+1$.
- 定理: アルゴリズムBFSは始点 s から到達可能な全ての頂点 $v \in V$ を発見し, 停止した時全ての v に対して $d[v] = \delta(s, v)$ が成立する. さらに, s から到達可能な全ての $v \neq s$ に対し, s から v への最短路の1つは, s から $\pi[v]$ への最短路の後に辺 $(\pi[v], v)$ を連接したものである.

- 証明: 背理法で示す. 最短路長に等しくない d 値を受け取る頂点が存在すると仮定する. v を, そのような頂点の中で $\delta(s, v)$ が最小の頂点とする.
- 補題2より $d[v] \geq \delta(s, v)$ なので, $d[v] > \delta(s, v)$.
- v が s から到達不可能ならば $\delta(s, v) = \infty$ なので $d[v]$ は正しい値 (∞). よって v は到達可能.
- u を s から v への最短路上の点で v の直前のものとすると $\delta(s, v) = \delta(s, u) + 1$ である.
- $\delta(s, u) < \delta(s, v)$ であり, 仮定より最短路長が $\delta(s, v)$ より短い場合は $d[u] = \delta(s, u)$ なので $d[v] > \delta(s, v) = \delta(s, u) + 1 = d[u] + 1$ …(1)

- キューから u を削除する場合を考える.
  v の色は白，灰，黒のいずれかである.
- v が白の時
  - $d[v] = d[u] + 1$ を実行するので式 (1) に矛盾.
- v が黒の時
  - v は既にキューから削除されている
  - $d[v] \leq d[u]$ なので式 (1) に矛盾.
- v が灰の時
  - ある点 w を削除した時に v は灰に彩色された
  - $d[v] = d[w]+1$
  - w は u よりも先にキューから削除されたので $d[w] \leq d[u]$
  - よって $d[v] \leq d[u]+1$ となり式 (1) に矛盾.

- 以上より，全ての点 v に対して $d[v] = \delta(s, v)$.
- $\pi[v] = u$ ならば $d[v] = d[u] + 1$ が成り立つ．よって辺 $(\pi[v], v)$ を連接すると最短路が求まる.
- グラフ $G_\pi = (V_\pi, E_\pi)$ を次のように定義する
  - $V_\pi = \{v \in V \mid \pi[v] \neq \mathrm{NIL}\} \cup \{s\}$
  - $E_\pi = \{(\pi[v], v) \in E \mid v \in V_\pi - \{s\}\}$
- V_π は s から到達可能な全頂点
- 全ての $v \in V_\pi$ に対し，s から v に至る唯一のパスが G_π に存在し，これが G における s から v への最短路になっている
- G_π は幅優先探索木(breadth-first tree)と呼ばれる