

# アルゴリズムと データ構造

コンピュータサイエンスコース  
知能コンピューティングコース

## 第13回 グラフの深さ優先探索

塩浦昭義

情報科学研究科 准教授

[shioura@dais.is.tohoku.ac.jp](mailto:shioura@dais.is.tohoku.ac.jp)

<http://www.dais.is.tohoku.ac.jp/~shioura/teaching>

# 期末試験について

- 日時: 8月6日(木)8:50~10:20
- 中間試験以降にレポートを一度も出していない場合、  
受験は不可
- 教科書, ノート等の持ち込みは一切不可
- 座席はこちらで指定
- 試験内容: 第7回~第13回(今回)の講義で教えたところ
  - アルゴリズムやデータ構造の挙動
  - 時間計算量の解析, および関連する証明問題
  - 用語の定義, など
- 50点満点, 24点以下は追試レポートもしくは単位不可
- 採点は8月7日(金)12時までには終える予定

# 授業の成績について

- 8月21日までには成績を確定させる予定
- **中間試験の得点＋期末試験の得点＋レポートの得点**が60点以上ならば単位は可，60点未満は不可
- 単位が心配な場合には，8月7日までに必ず問い合わせるように。
  - 問い合わせがない場合，救済措置はとりません
- レポート提出の頻度が低い学生には，一切救済措置はとりません。

# グラフの深さ優先探索

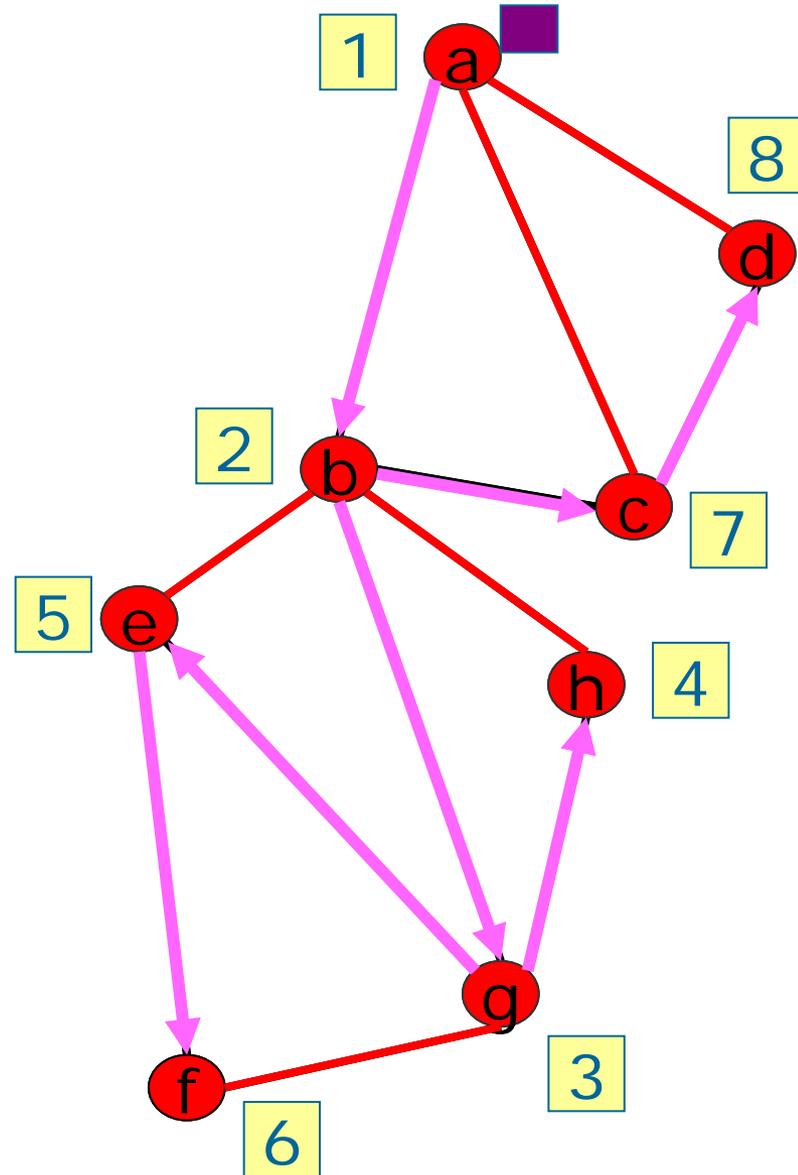
- 与えられたグラフを組織的に探索する方法のひとつ
- グラフの構造・性質を調べるときに有効な技法
  - 連結成分, 2連結成分, 強連結成分に分解
  - 閉路の検出
  - などなど

# 無向グラフの深さ優先探索

- (1) 各頂点, 各枝を白く塗る
- (2) 各頂点  $u \in V$  に対し,  
 $u$  が白色 (未走査) ならば  
手続き DFS-VISIT( $u$ ) を実行

## 手続き DFS-VISIT( $u$ )

- (a)  $u$  を黒く塗る
- (b)  $u$  に接続する各枝  $(u, v)$  に対し, 以下を実行:  
枝が白色 (未走査) ならば, 黒く塗る  
 $v$  が白色 (未走査) ならば  
DFS-VISIT( $v$ ) を再帰呼び出し



# 無向グラフの深さ優先探索

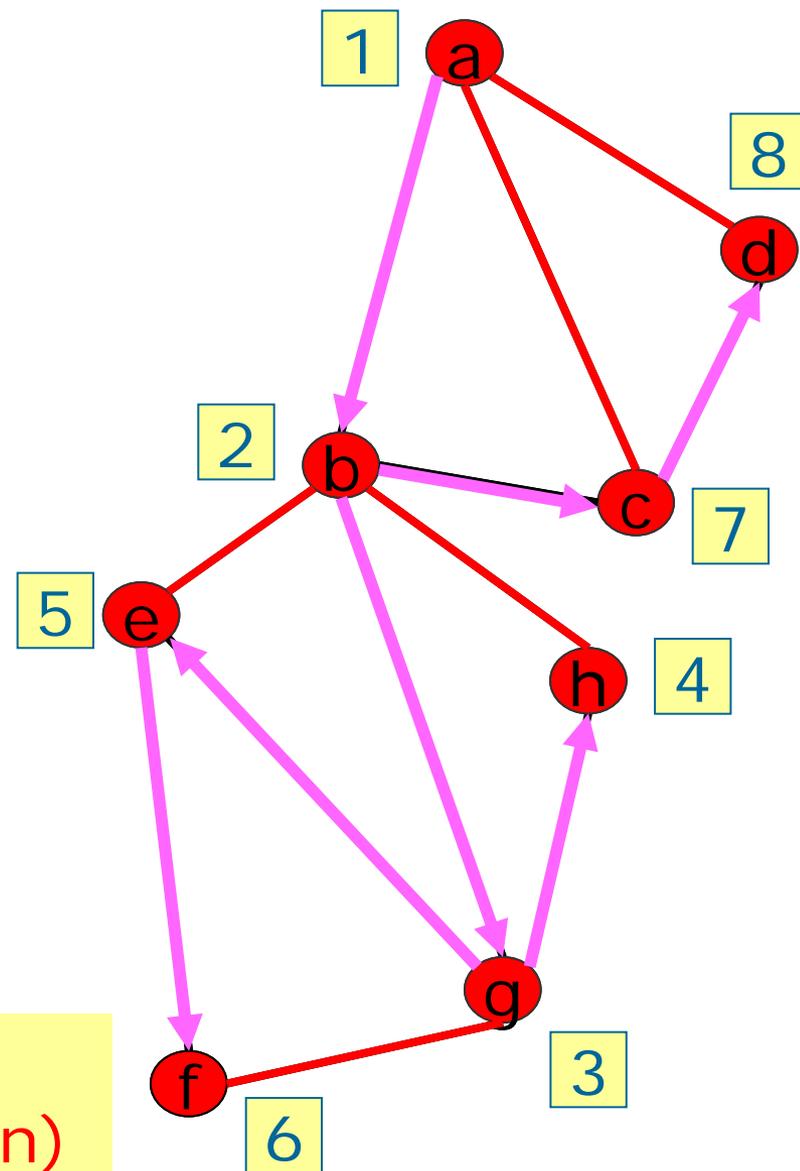
- (1) 各頂点, 各枝を白く塗る
- (2) 各頂点  $u \in V$  に対し,  
 $u$  が白色 (未走査) ならば  
手続き DFS-VISIT( $u$ ) を実行

## 手続き DFS-VISIT( $u$ )

- $u$  を黒く塗る
- $u$  に接続する各枝  $(u, v)$  に対し, 以下を実行:  
枝が白色 (未走査) ならば, 黒く塗る  
 $v$  が白色 (未走査) ならば  
DFS-VISIT( $v$ ) を再帰呼び出し

深さ優先探索の実行時間:

グラフを隣接リストで表現すると  $O(m+n)$



# 無向グラフの2連結成分

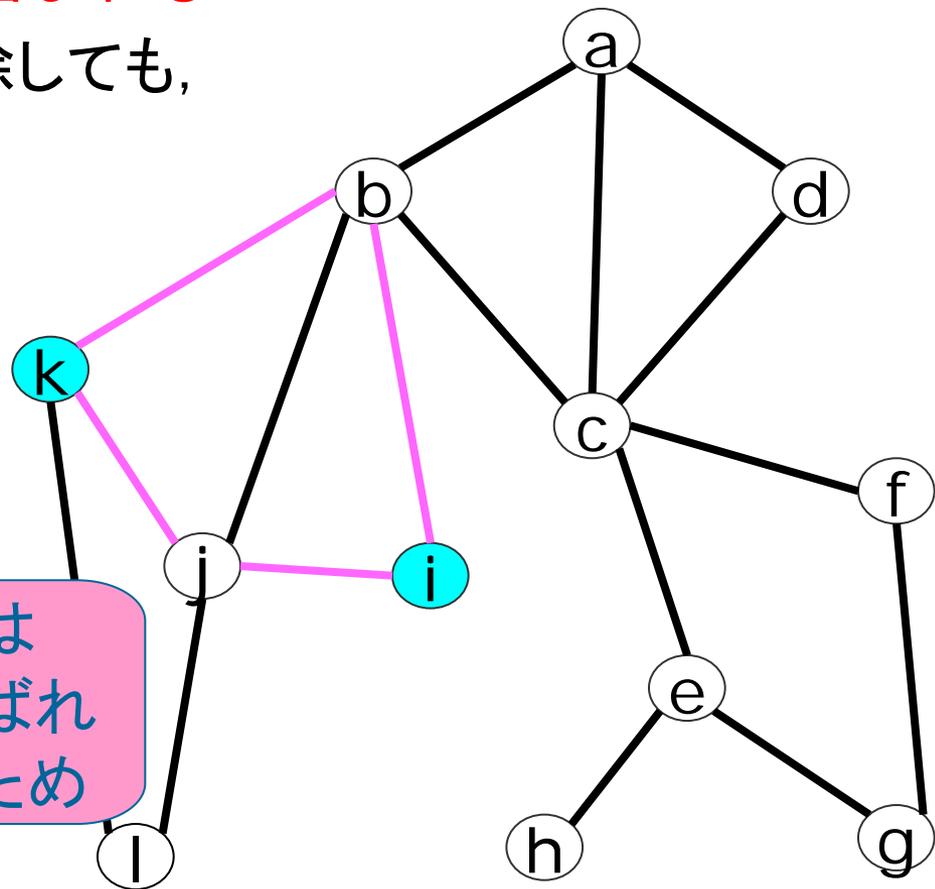
- 無向グラフ $G=(V, E)$ において,  
頂点  $u, v$  は同じ2連結成分に含まれる  
 $\leftrightarrow$   $u, v$  以外の頂点  $w$  を削除しても,  
 $u$  から  $v$  への路が存在

$k$  と  $i$  は同じ  
2連結成分に含まれる

$a$  と  $c$  は同じ  
2連結成分に含まれる

$e$  と  $h$  は同じ  
2連結成分に含まれる

$e$  と  $h$  は  
枝で結ばれ  
ているため



# 無向グラフの2連結成分

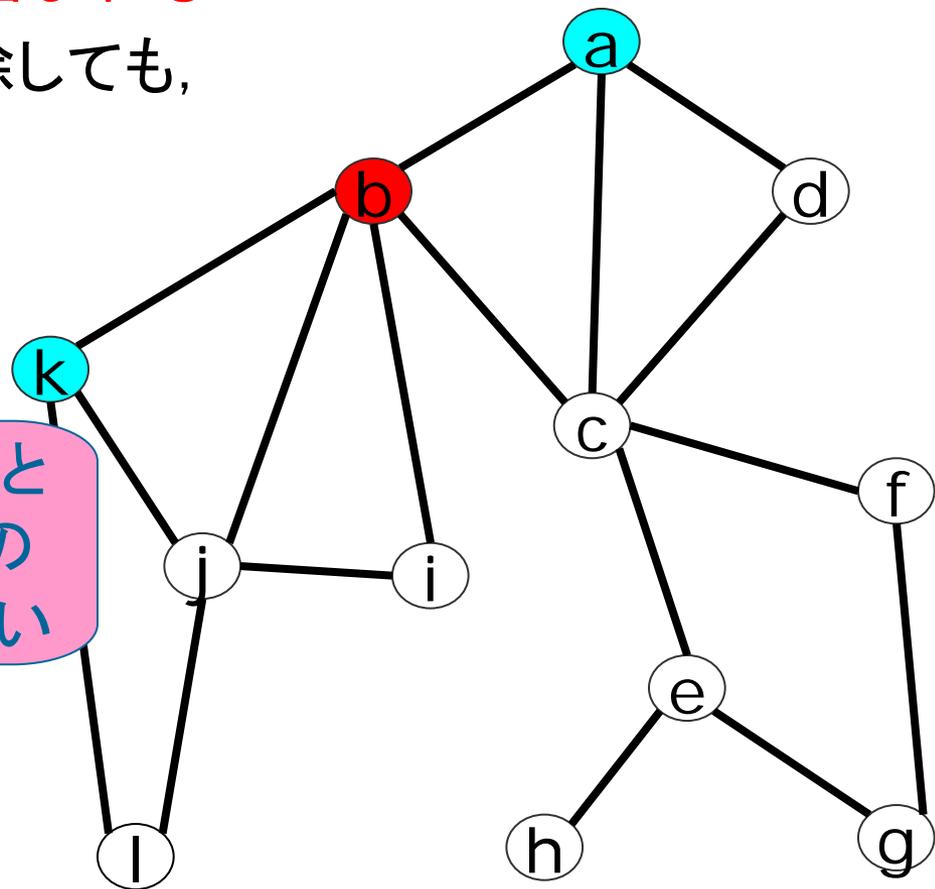
- 無向グラフ  $G=(V, E)$  において,  
頂点  $u, v$  は同じ2連結成分に含まれる  
 $\leftrightarrow$   $u, v$  以外の頂点  $w$  を削除しても,  
 $u$  から  $v$  への路が存在

a と k は同じ  
2連結成分に含まれない

c を削除すると  
b から g への  
路が存在しない

b を削除すると  
a から k への  
路が存在しない

b と g は同じ  
2連結成分に含まれない

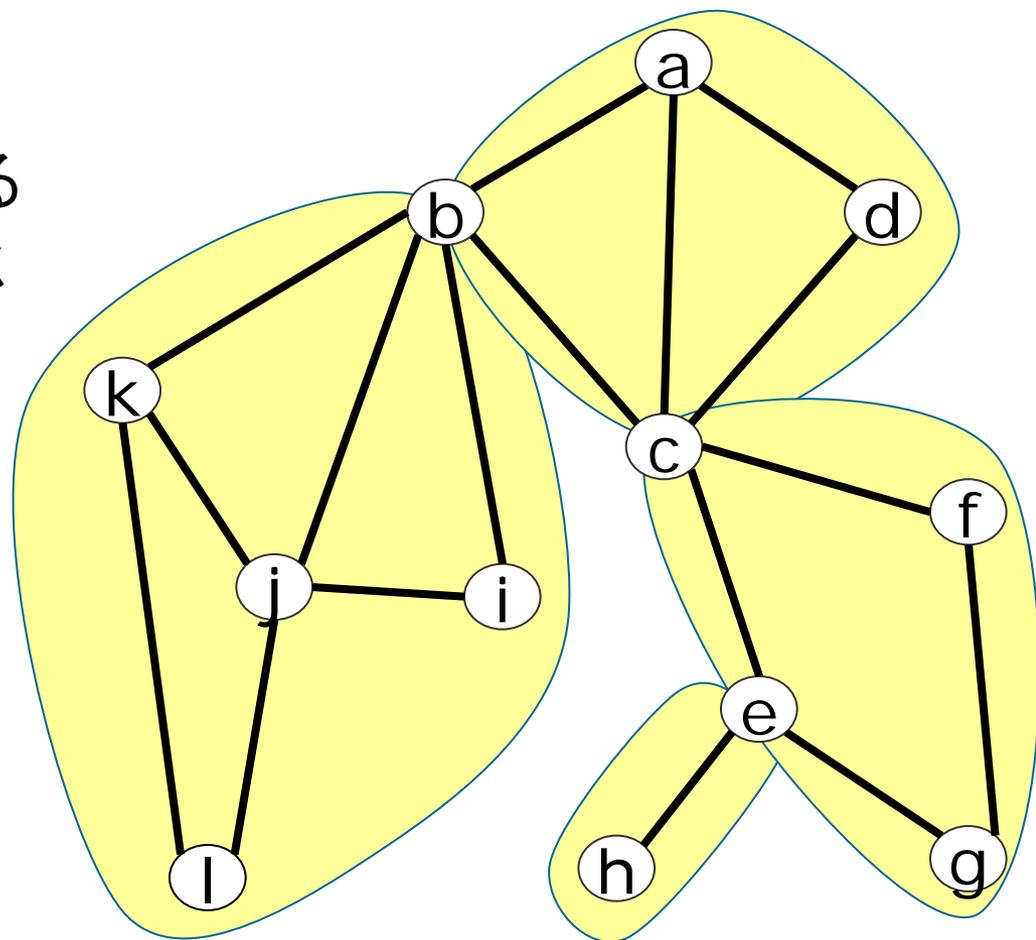


# 2連結成分分解と関節点

- 同じ連結成分に含まれる頂点をグループ分け  
→ **2連結成分分解**

- 複数の2連結成分に含まれる頂点が存在 → **関節点**と呼ぶ

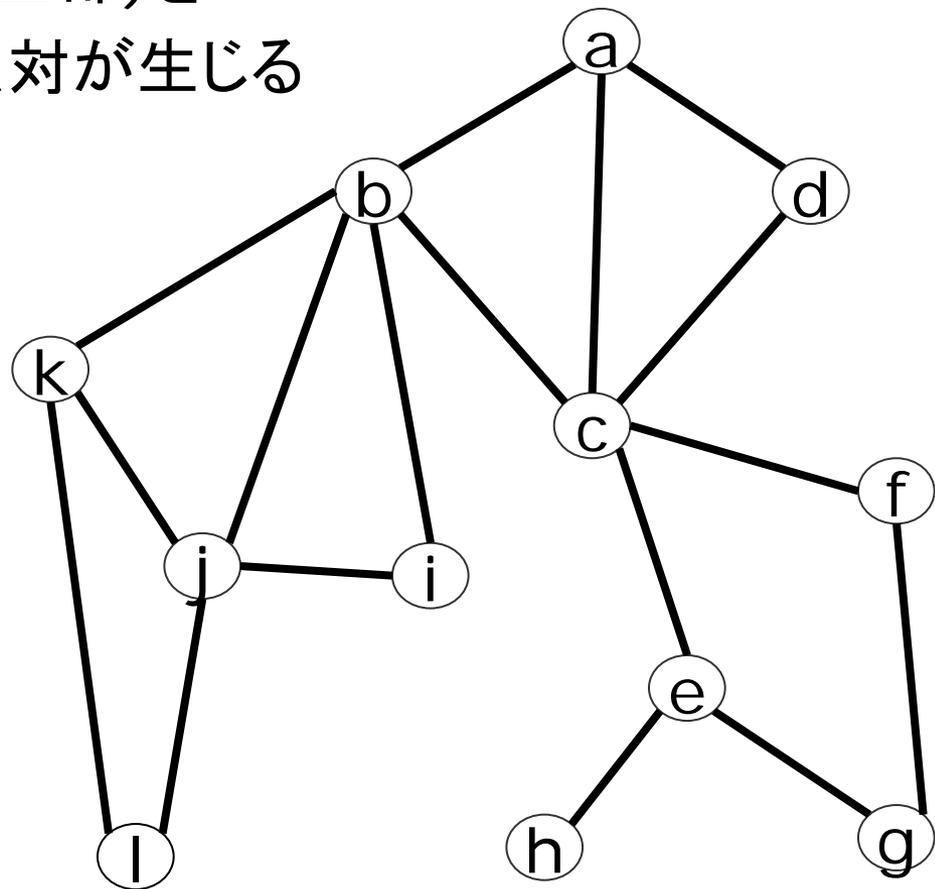
頂点 b, c, e は関節点



# 無向グラフの関節点

- 性質: 無向グラフ  $G=(V, E)$  において, 頂点  $u$  は関節点  $\iff u$  (および  $u$  に接続する枝全部) を削除すると, 非連結になる頂点对が生じる

頂点  $b, c, e$  は関節点  
他の頂点は関節点ではない





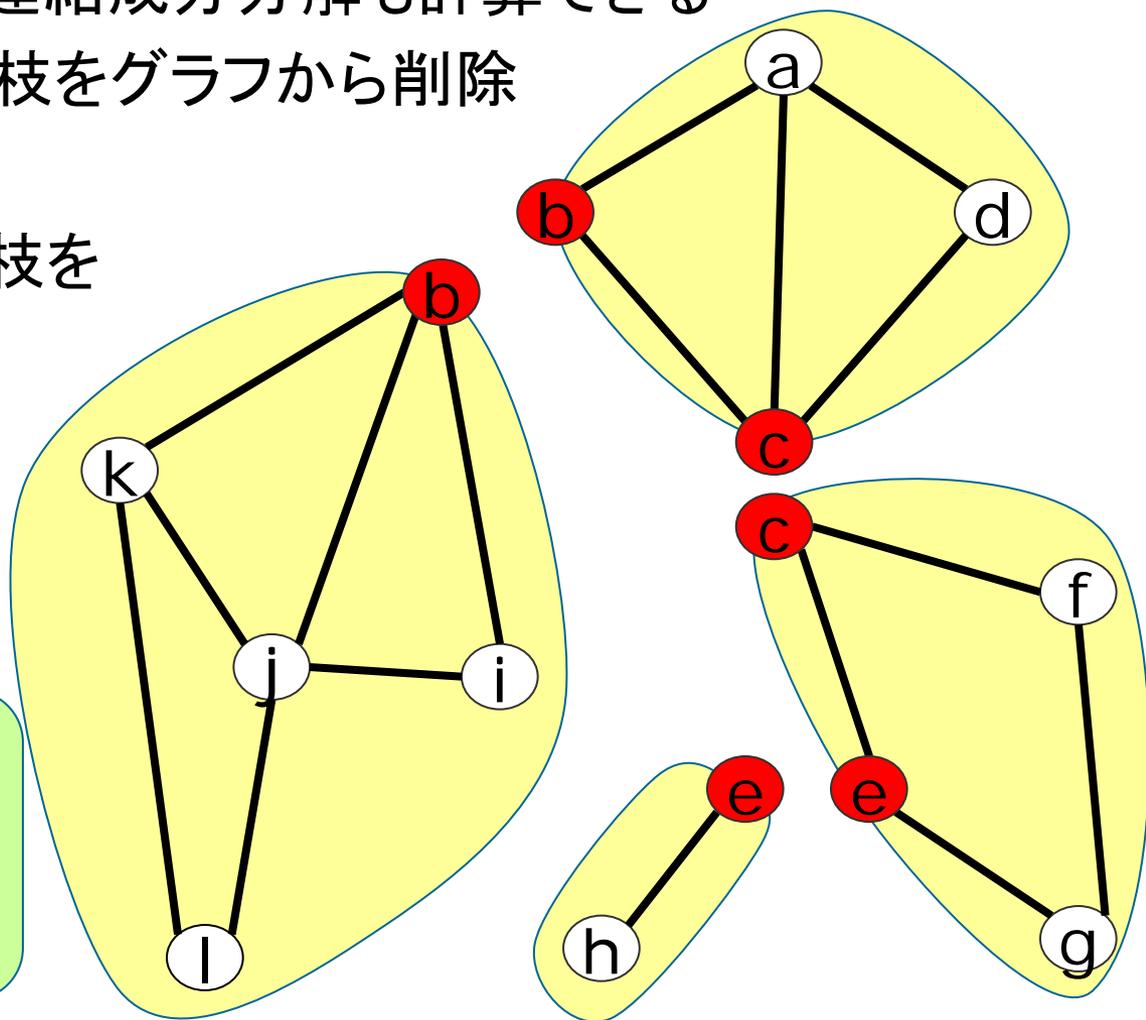
# 関節点から2連結成分分解を求める

関節点が計算できれば, 2連結成分分解も計算できる

- (1) 関節点および接続する枝をグラフから削除
- (2) 連結成分に分解
- (3) 関節点に接続していた枝を各連結成分に戻す

関節点が与えられれば,  
計算時間は  $O(m+n)$

深さ優先探索を使って  
全ての関節点を  
 $O(m+n)$ 時間で求める  
方法を説明する



# 関節点と lowpt

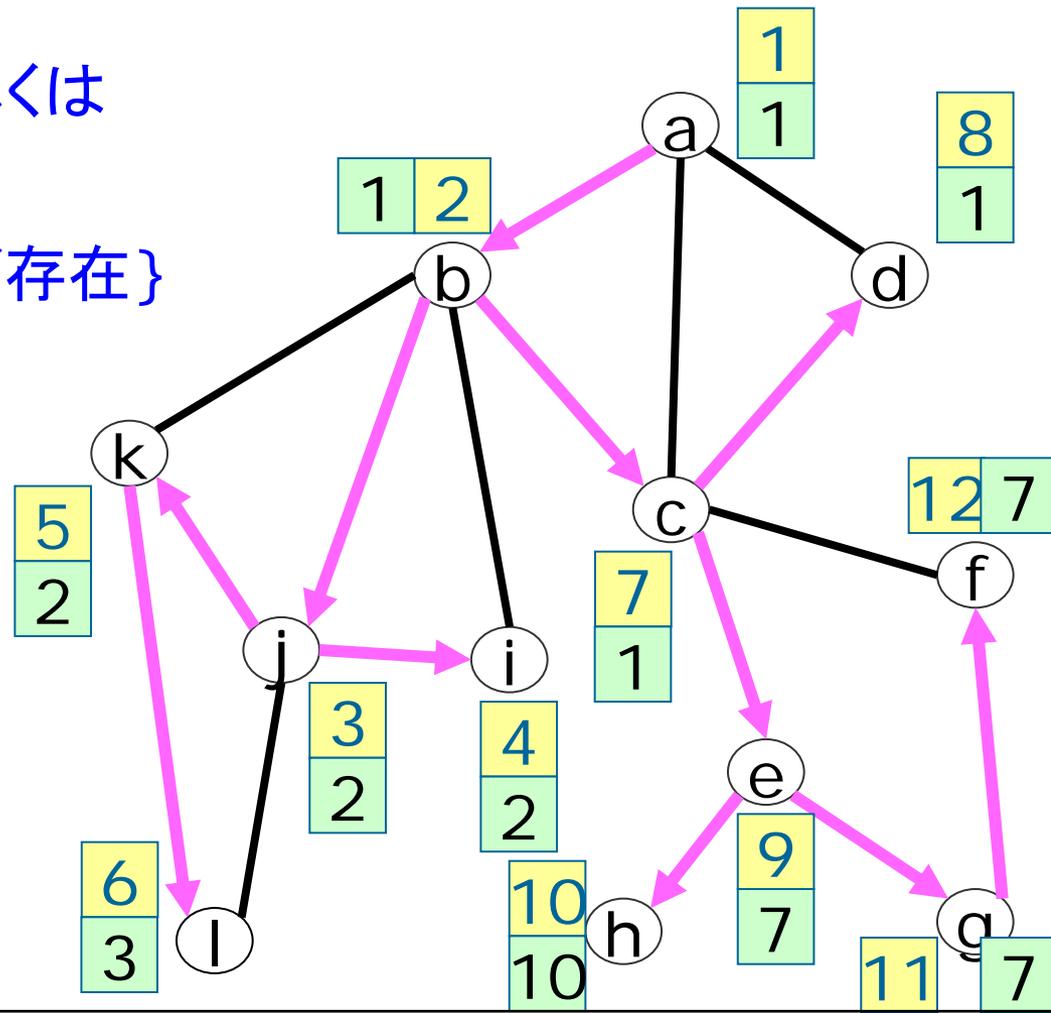
num[u]: 頂点 u が何番目に走査されたかを表す数字

lowpt[u]

=  $\min\{\text{num}[v] \mid v=u, \text{もしくは}$   
頂点uの子孫wに対し,  
Tに含まれない枝(v,w)が存在}

**例1:** 頂点 j の  
子孫 k に枝(b,k)が接続  
子孫 i に枝(b,i)が接続  
→  $\text{lowpt}[j] = \text{num}[b] = 2$

**例2:** 頂点 e の  
子孫 f に枝(c,f)が接続  
→  $\text{lowpt}[e] = \text{num}[c] = 7$



# 関節点と lowpt に関する性質

**定理:**  $u$  は関節点

$\leftrightarrow$   $u$  の子供  $v$  が存在して次の条件を満たす

- (i)  $\text{lowpt}[v] \geq \text{num}[u]$
- (ii)  $u$  が根のとき, 子供の数が2以上

**例1:** 頂点  $b$  と子供  $j$  に対し

(i)  $\text{lowpt}[j] = 2 = \text{num}[b]$

(ii) 頂点  $b$  は根ではない

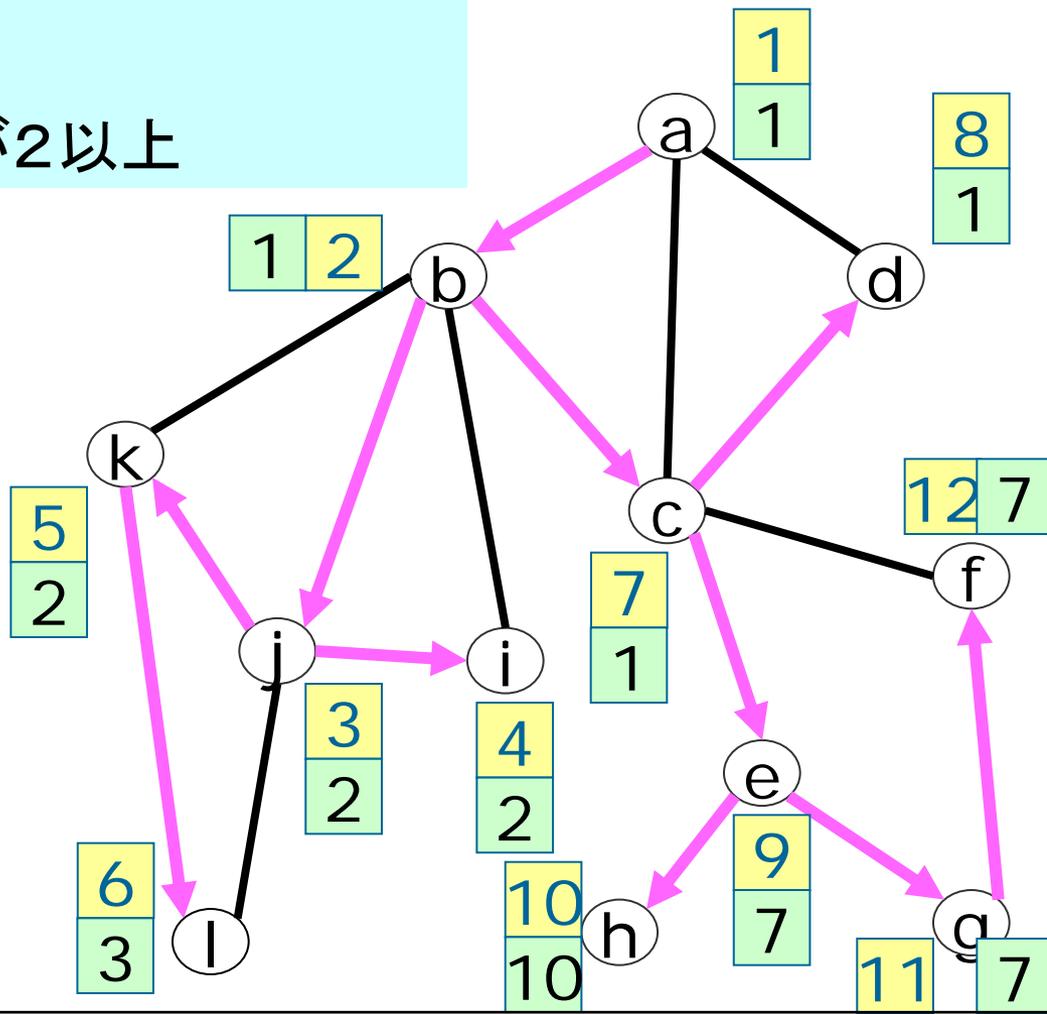
$\rightarrow b$  は関節点

**例2:** 頂点  $c$  と子供  $e$  に対し

(i)  $\text{lowpt}[e] = 7 = \text{num}[c]$

(ii) 頂点  $c$  は根ではない

$\rightarrow c$  は関節点



# 関節点と lowpt に関する性質

**定理:**  $u$  は関節点

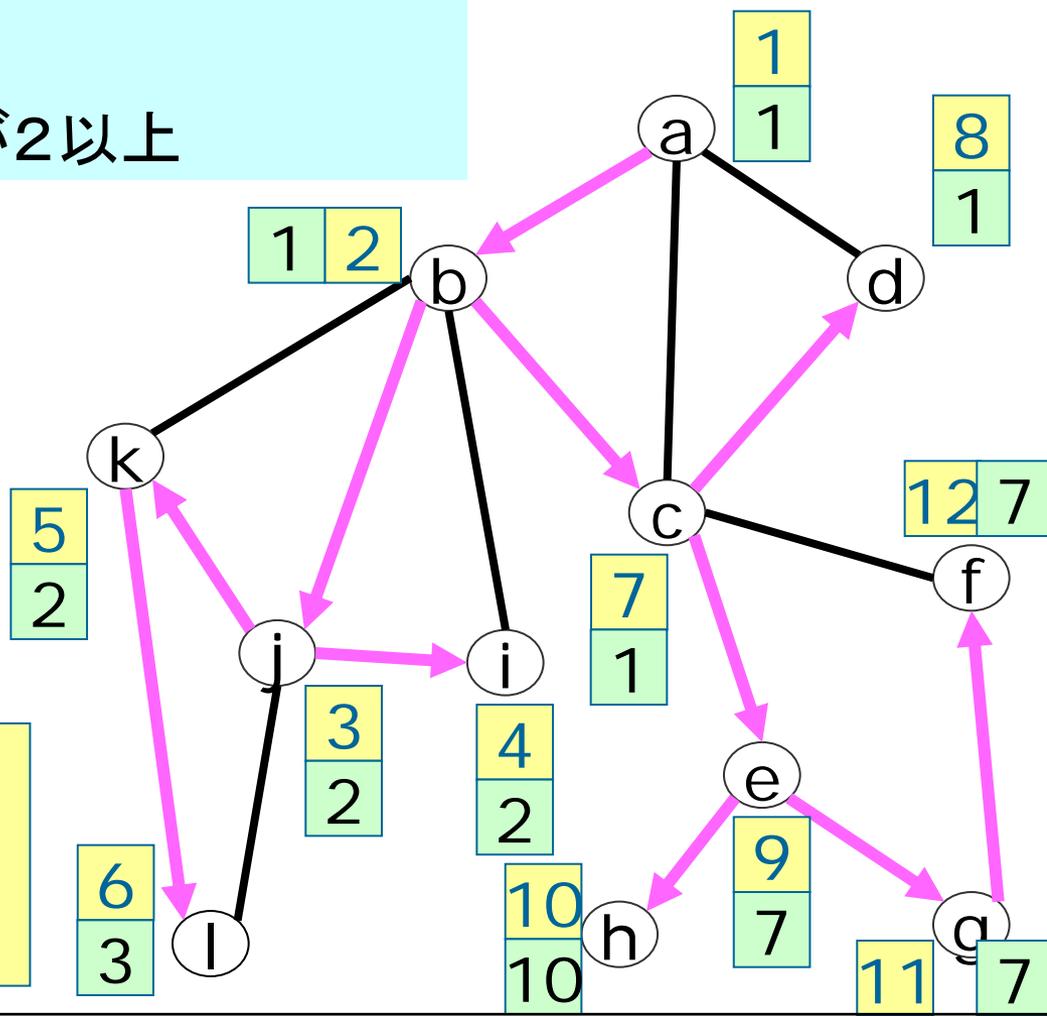
$\leftrightarrow$   $u$  の子供  $v$  が存在して次の条件を満たす

(i)  $\text{lowpt}[v] \geq \text{num}[u]$

(ii)  $u$  が根のとき, 子供の数が2以上

**例3:** 頂点  $j$  の子供は  $k, i$   
 $\text{lowpt}[k] = 2 < 3 = \text{num}[j]$   
 $\text{lowpt}[i] = 2 < 3 = \text{num}[j]$   
 $\rightarrow j$  は関節点ではない

**例4:** 頂点  $a$  の子供は  $b$   
(ii)  $b$  以外の子供は存在しない  
 $\rightarrow a$  は関節点ではない





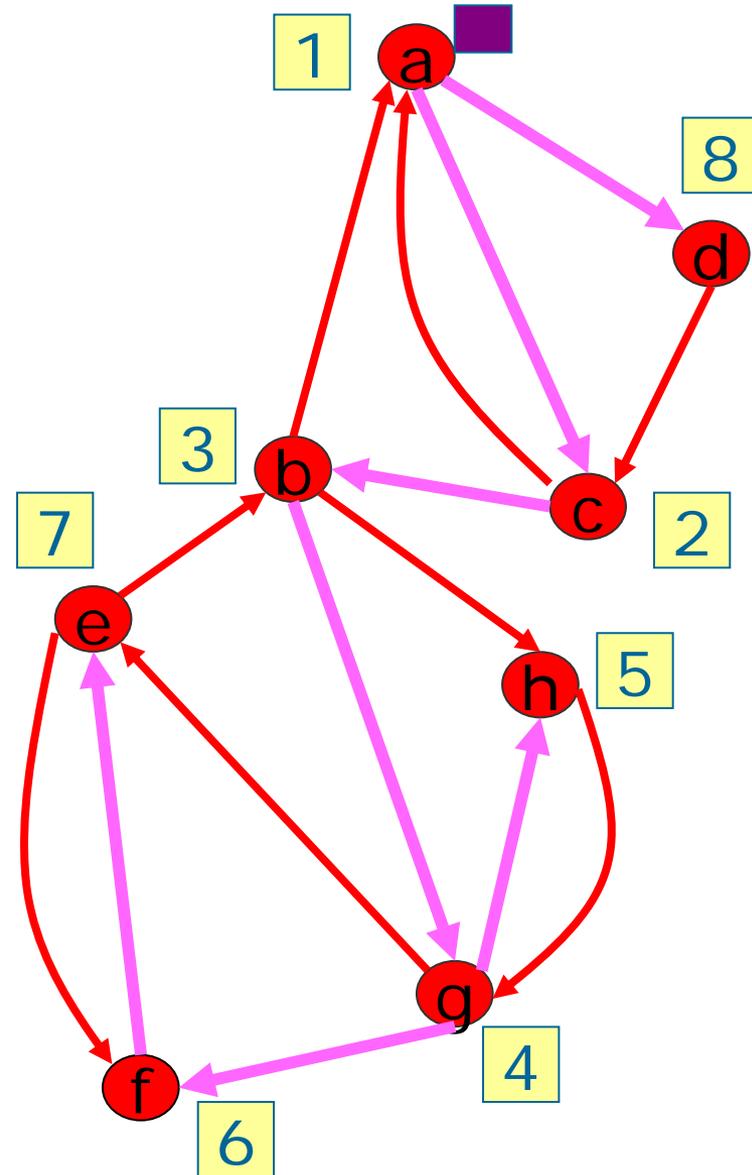
# 有向グラフの深さ優先探索

無向グラフの場合と基本的に同じ

- (1) 各頂点, 各枝を白く塗る
- (2) 各頂点  $u \in V$  に対し,  
 $u$  が白色(未走査)ならば  
手続きDFS-VISIT( $u$ )を実行

手続き DFS-VISIT( $u$ )

- (a)  $u$  を黒く塗る
- (b)  $u$  に接続する各枝  $(u, v)$  に対し, 以下を実行:  
枝が白色(未走査)ならば, 黒く塗る  
 $v$  が白色(未走査)ならば  
DFS-VISIT( $v$ ) を再帰呼び出し



# 有向グラフの深さ優先探索

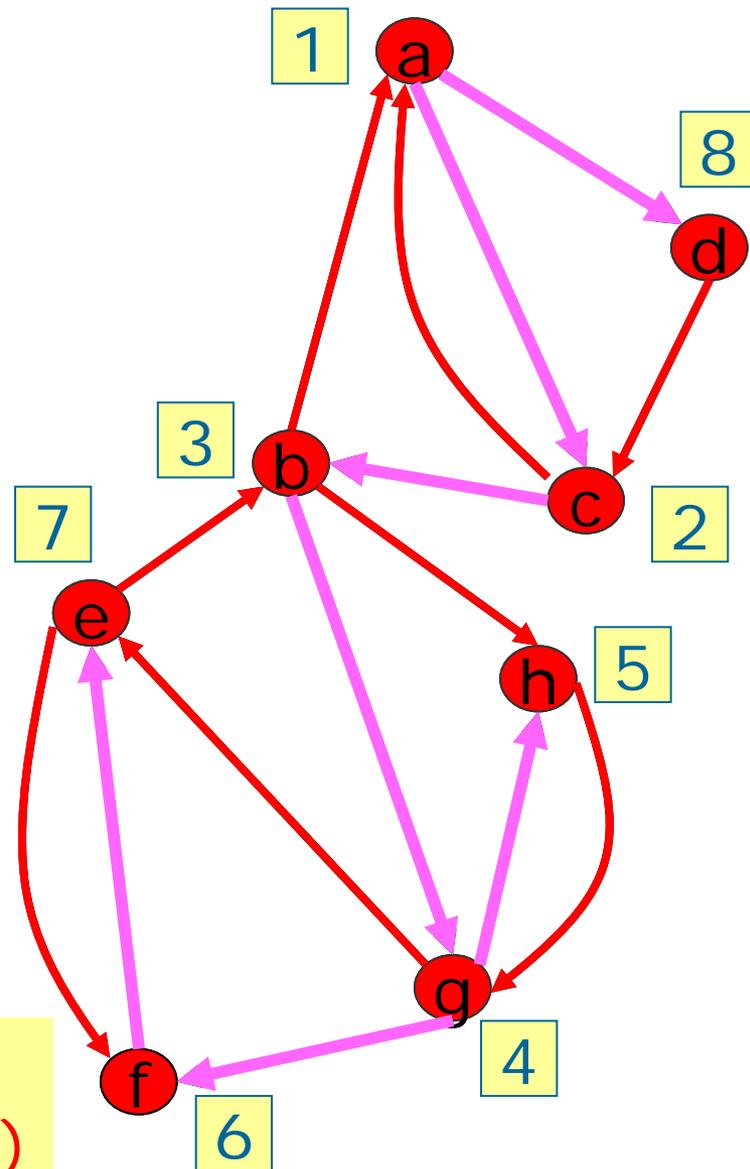
無向グラフの場合と基本的に同じ

- (1) 各頂点, 各枝を白く塗る
- (2) 各頂点  $u \in V$  に対し,  
 $u$  が白色 (未走査) ならば  
手続き DFS-VISIT( $u$ ) を実行

手続き DFS-VISIT( $u$ )

- (a)  $u$  を黒く塗る
- (b)  $u$  に接続する各枝  $(u, v)$  に対し, 以下を実行:  
枝が白色 (未走査) ならば, 黒く塗る  
枝が白色 (未走査) ならば

深さ優先探索の実行時間:  
グラフを隣接リストで表現すると  $O(m+n)$

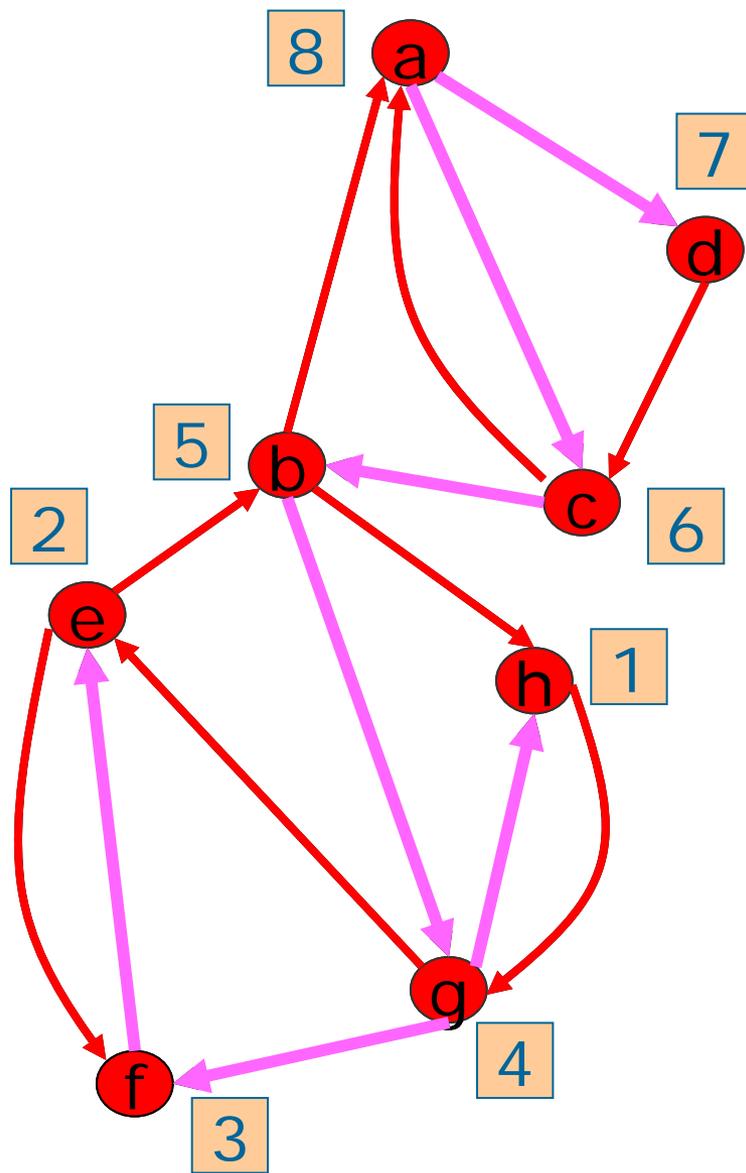


# 最後に訪問した時間による番号づけ

頂点を初めて訪問した時間の順に、  
各頂点に番号を付ける  
→ いろいろと便利

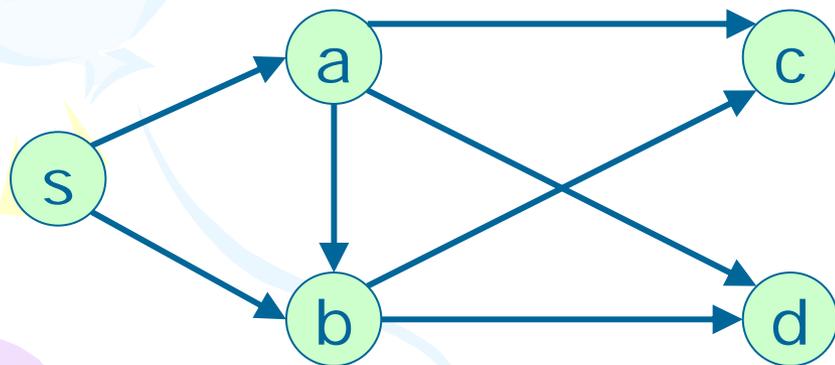
同様に、

頂点を最後に訪問した時間の順に、  
(頂点から出る枝を全て調べ終えた順に)  
各頂点に番号を付ける  
→ いろいろと便利

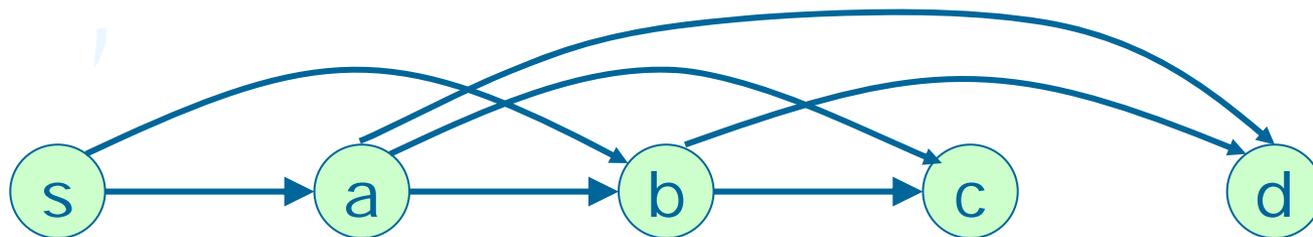
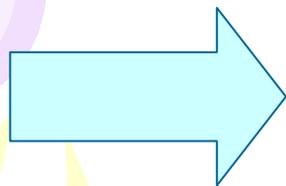


# 閉路のない有向グラフの トポロジカルソート

- 閉路のない有向グラフに対し, 以下の条件を満たすように頂点を並べることが可能
  - 各枝は, 必ず左から右に向かう
- このように頂点を並べること → **トポロジカルソート**



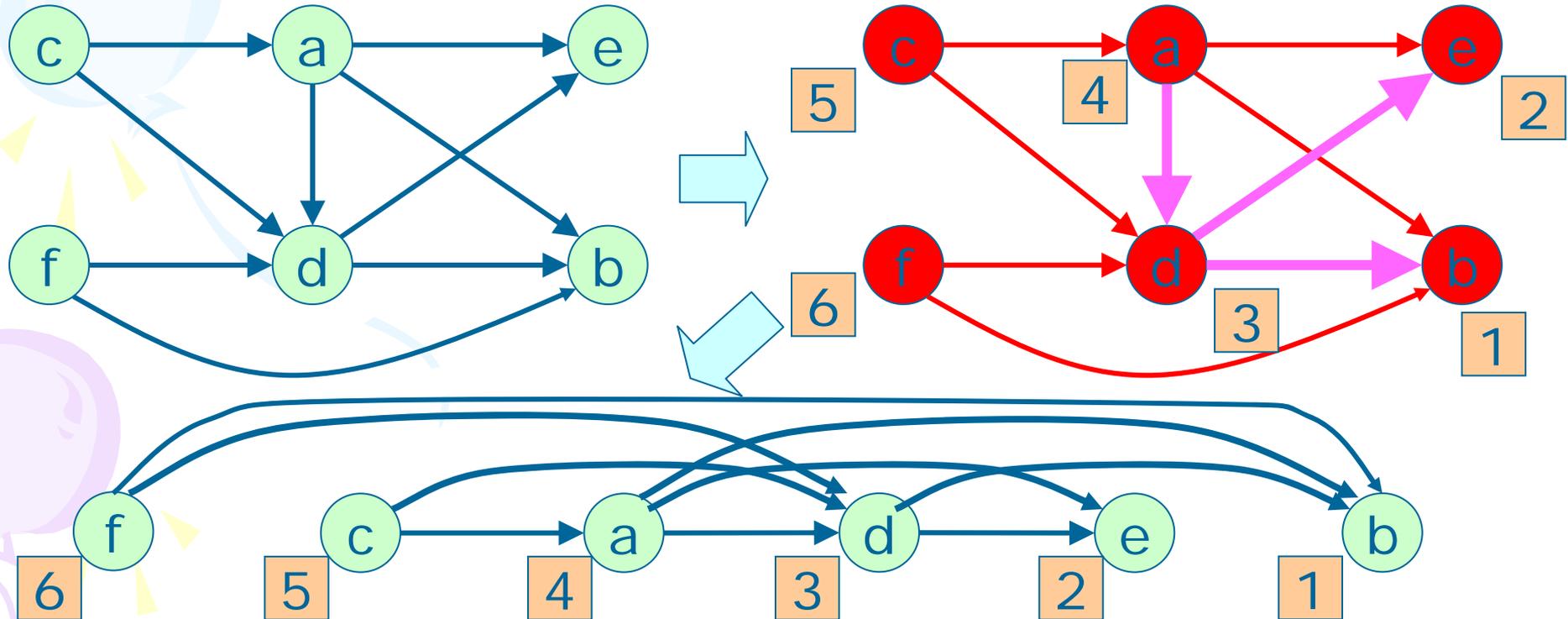
閉路のない  
有向グラフ



# トポロジカルソート を求めるアルゴリズム

深さ優先探索を利用→トポロジカルソートが計算できる

- (1) グラフに深さ優先探索を適用し、  
最後に訪問した時間により頂点を番号付けする
- (2) 番号の大きい方から小さい方に並べる



# トポロジカルソート を求めるアルゴリズムの正当性

グラフに深さ優先探索を適用し、最後に訪問した時間により  
頂点を番号付けする → 各頂点 $v$ の番号を  $f(v)$  と書く

全ての枝  $(u, v)$  に対して  $f(u) > f(v)$  ならばOK

∃  $(u, v)$ :  $f(u) < f(v)$  と仮定, 矛盾を導く

枝  $(u, v)$  を探索したとき,

(a)  $v$  は未訪問 →  $v$  は  $u$  の子孫 →  $f(u) > f(v)$  が成り立つ (矛盾)

(b)  $v$  は訪問終了後

→  $u$  の訪問は終了していないので  $f(u) > f(v)$  が成り立つ (矛盾)

(c)  $v$  は訪問中 →  $v$  は  $u$  の先祖 →  $u, v$  を含む閉路が存在 (矛盾)

