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12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
http://www.dais.is.tohoku.ac.jp/~shioura/teaching

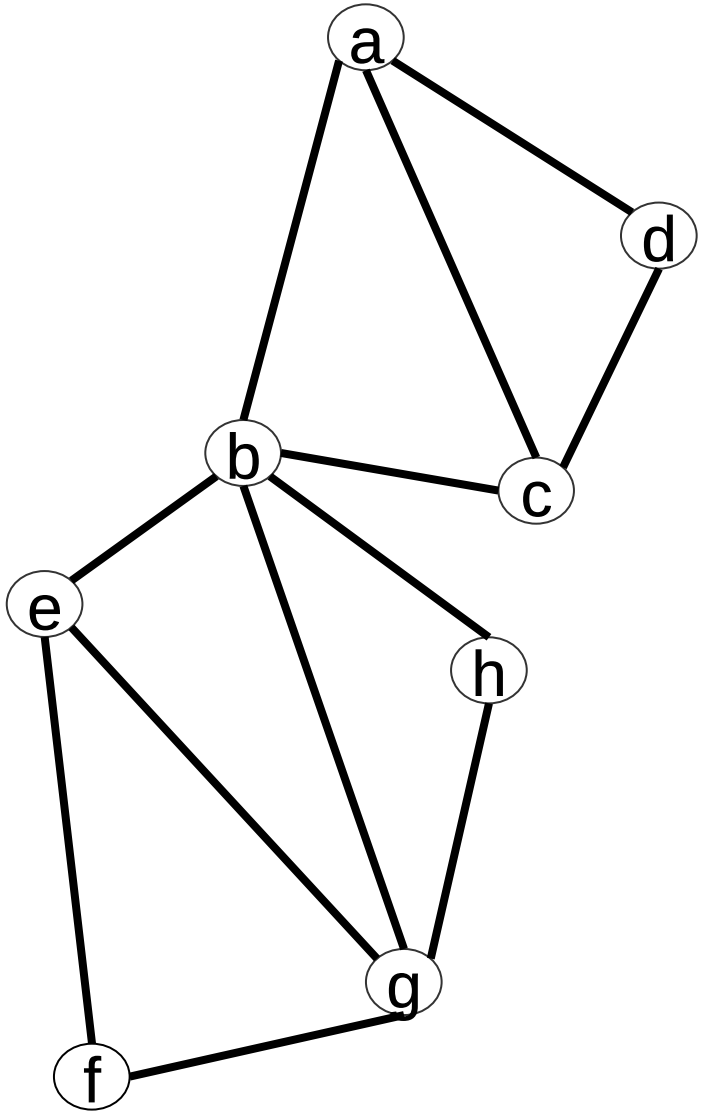
グラフの深さ優先探索

- 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∈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

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

DFS-VISIT(a)を実行

枝(a,b)を走査

DFS-VISIT(b)を実行

枝(b,g)を走査

DFS-VISIT(g)を実行

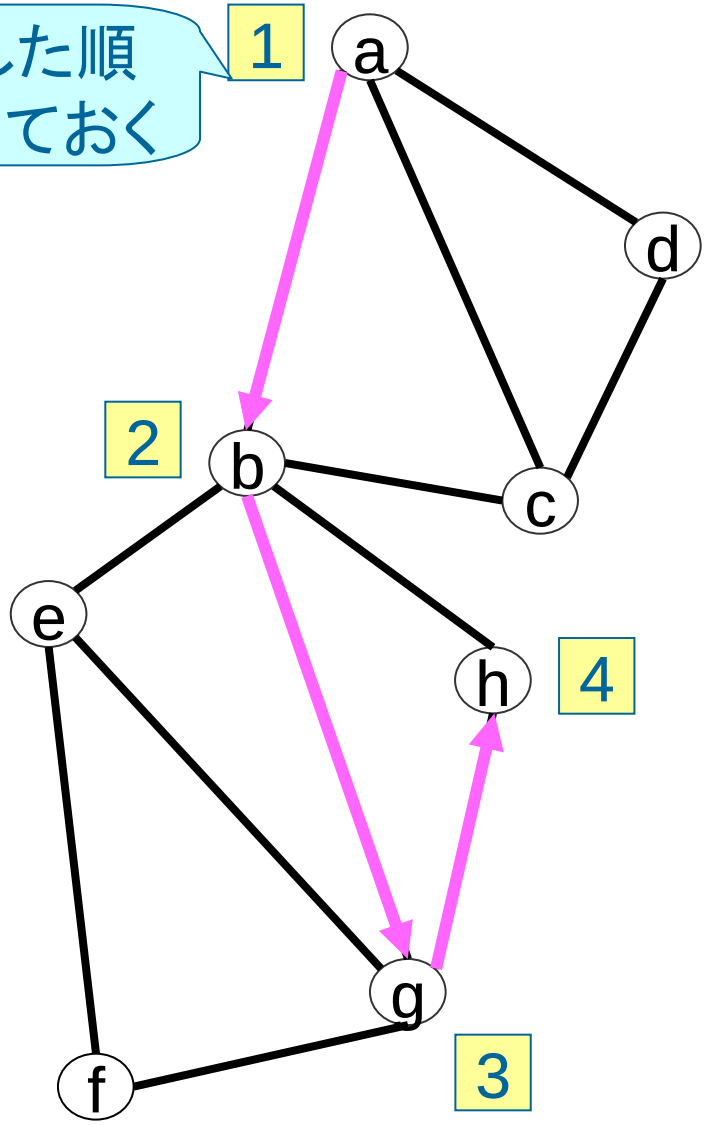
枝(g,h)を走査

DFS-VISIT(h)を実行

枝(h,b)を走査

枝(h,g)は走査済み

頂点を走査した順に番号を付けておく

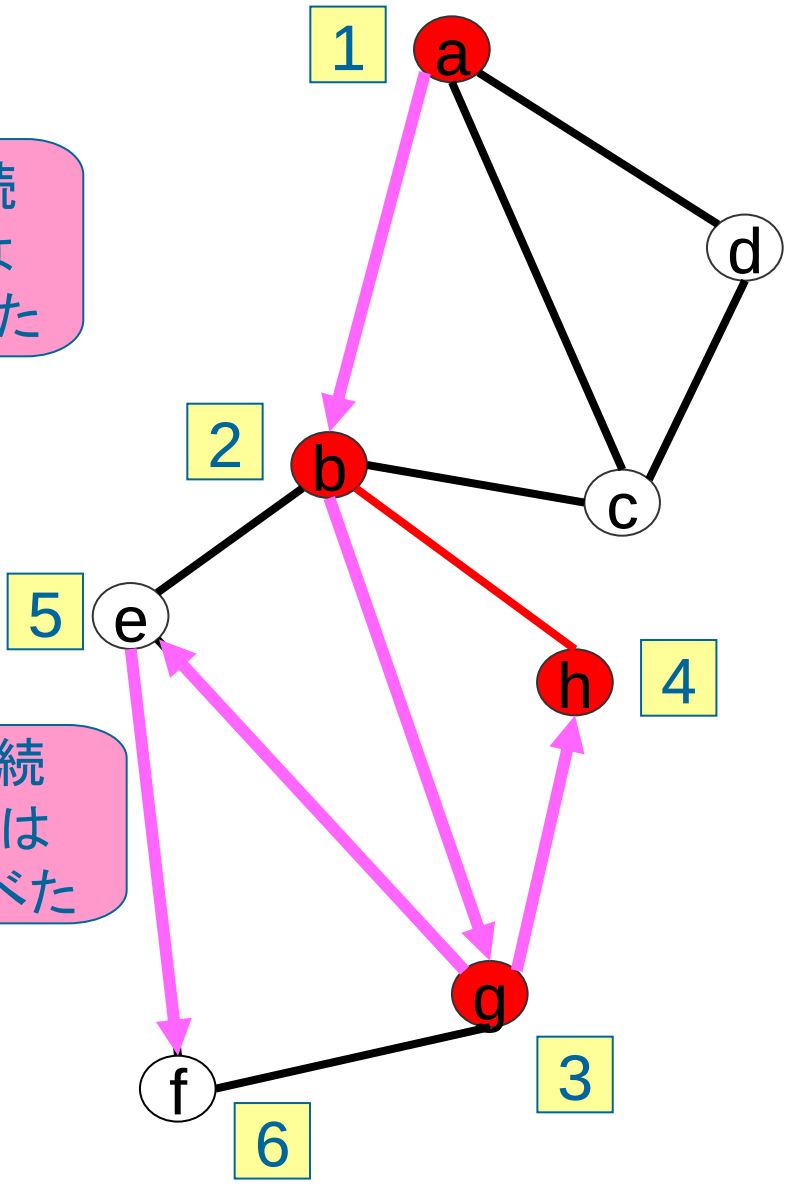


無向グラフの深さ優先探索

DFS-VISIT(a)を実行
枝(a,b)を走査
DFS-VISIT(b)を実行
枝(b,g)を走査
DFS-VISIT(g)を実行
枝(g,h)を走査
DFS-VISIT(h)を実行
枝(h,b)を走査
枝(h,g)は走査済み
DFS-VISIT(h)終了
枝(g,e)を走査
DFS-VISIT(e)を実行
枝(e,f)を走査
DFS-VISIT(f)を実行
枝(f,g)を走査
枝(f,e)は走査済み
DFS-VISIT(f)終了

h に接続する枝は全て調べた

f に接続する枝は全て調べた



無向グラフの深さ優先探索

⋮

DFS-VISIT(g)を実行

枝(g,h)を走査

⋮

枝(g,e)を走査

DFS-VISIT(e)を実行

枝(e,f)を走査

DFS-VISIT(f)を実行

枝(f,g)を走査

枝(f,e)は走査済み

DFS-VISIT(f)終了

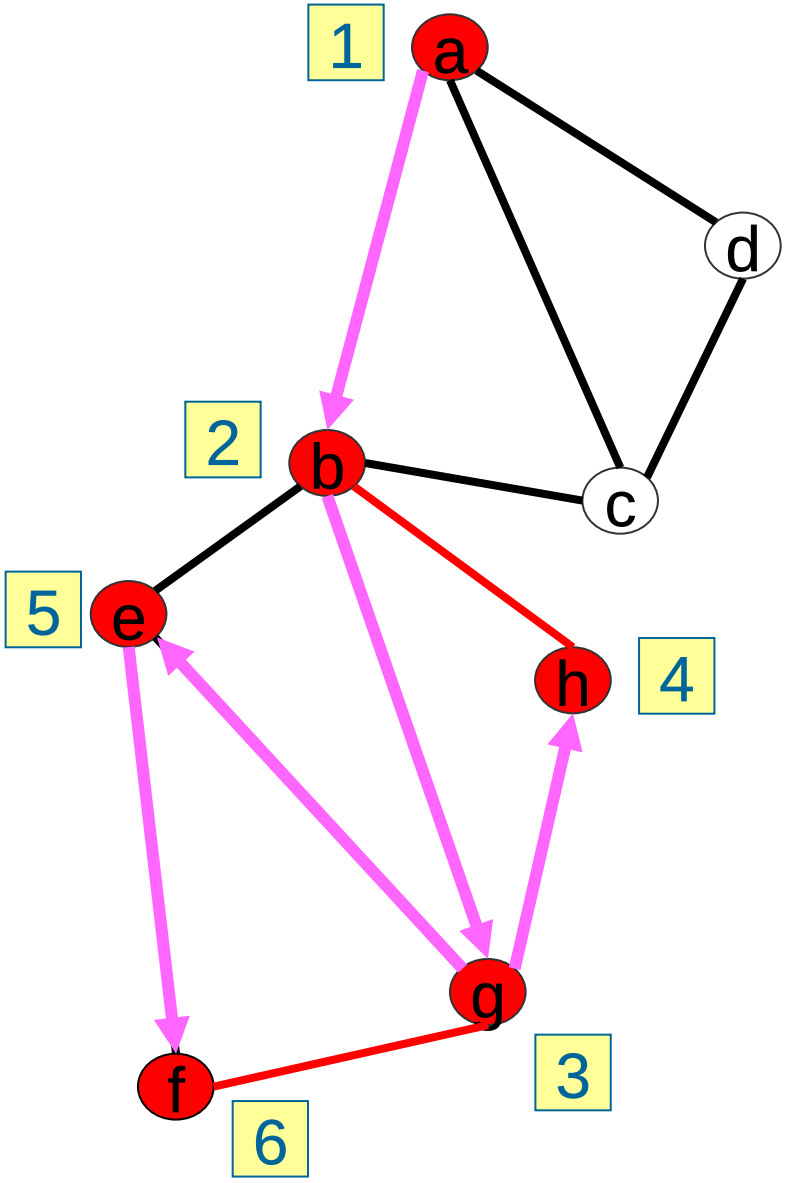
枝(e,g)は走査済み

枝(e,b)を走査

DFS-VISIT(e)終了

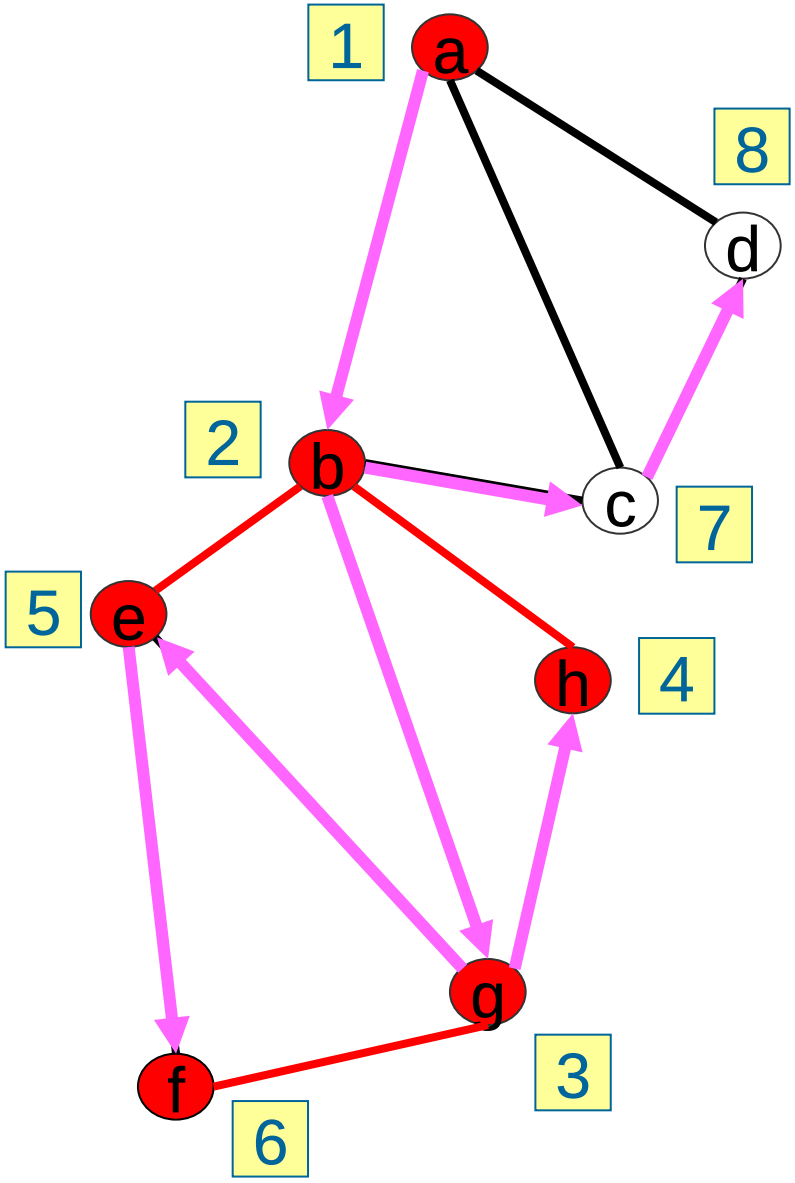
枝(g,b)は走査済み

DFS-VISIT(g)終了



無向グラフの深さ優先探索

DFS-VISIT(a)を実行
枝(a,b)を走査
DFS-VISIT(b)を実行
枝(b,g)を走査
DFS-VISIT(g)を実行
⋮
DFS-VISIT(g)終了
枝(b,e)は走査済み
枝(b,c)を走査
DFS-VISIT(c)を実行
枝(c,b)は走査済み
枝(c,a)を走査
枝(c,d)を走査
DFS-VISIT(d)を実行
枝(d,c)は走査済み
枝(d,a)を走査
DFS-VISIT(d)終了



無向グラフの深さ優先探索

DFS-VISIT(a)を実行

枝(a,b)を走査

DFS-VISIT(b)を実行

枝(b,g)を走査

⋮

枝(b,e)は走査済み

枝(b,c)を走査

DFS-VISIT(c)を実行

枝(c,b)は走査済み

枝(c,a)を走査

枝(c,d)を走査

⋮

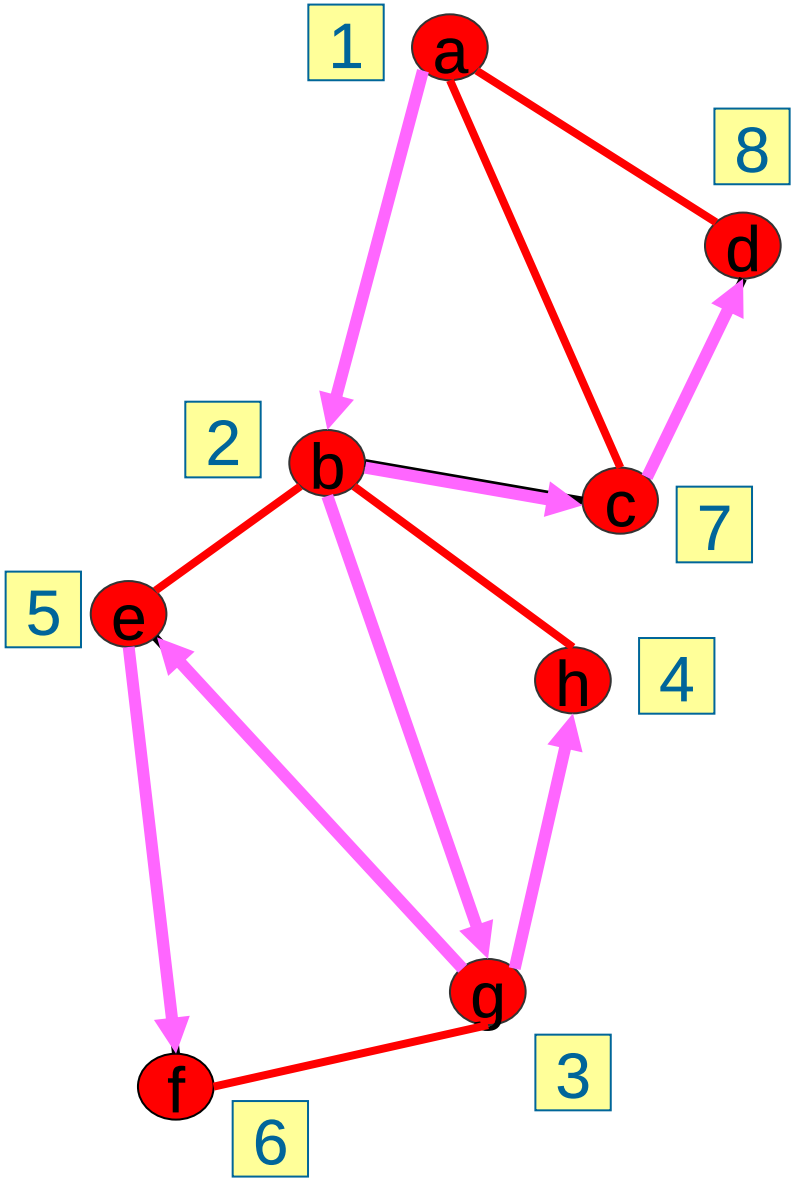
DFS-VISIT(c)終了

枝(b,a),(b,h)は走査済み

DFS-VISIT(b)終了

枝(a,c), (a,d)は走査済み

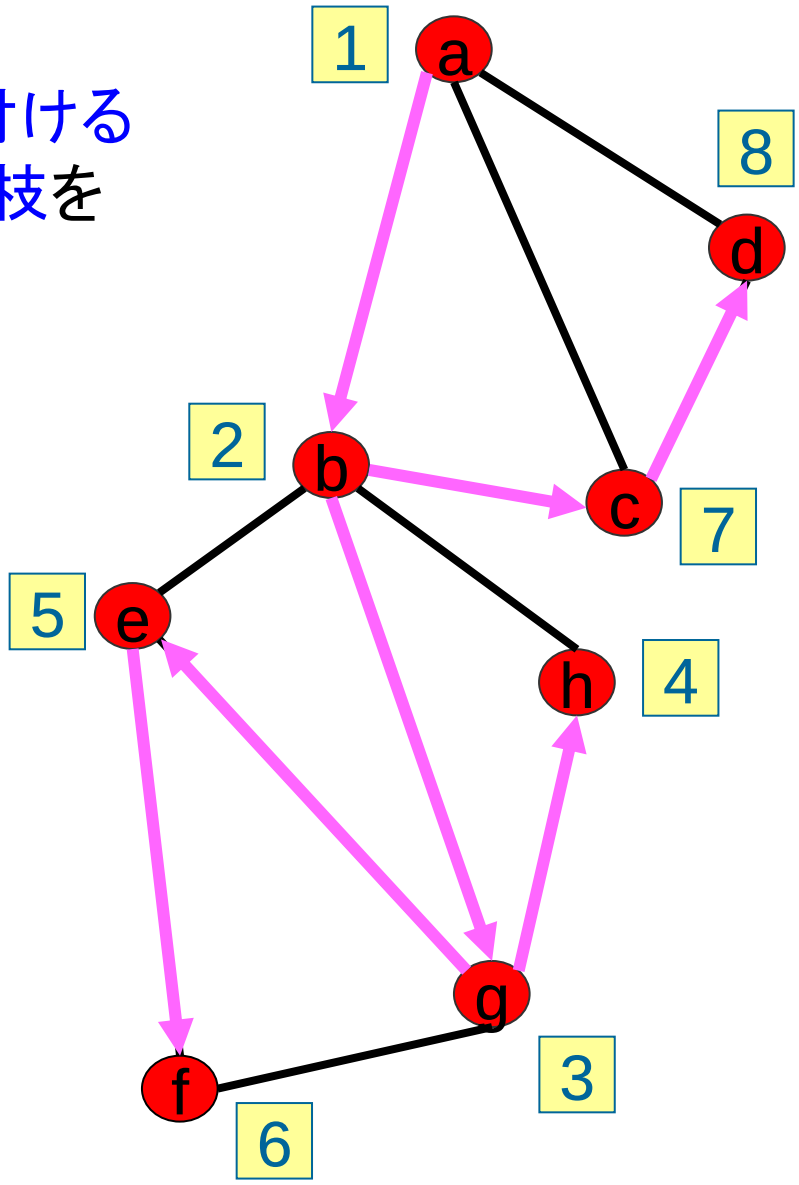
DFS-VISIT(a)終了



深さ優先探索における工夫

- 新たな頂点を走査する度に番号を付ける
- 新たな頂点を走査するときに使った枝を覚えておく

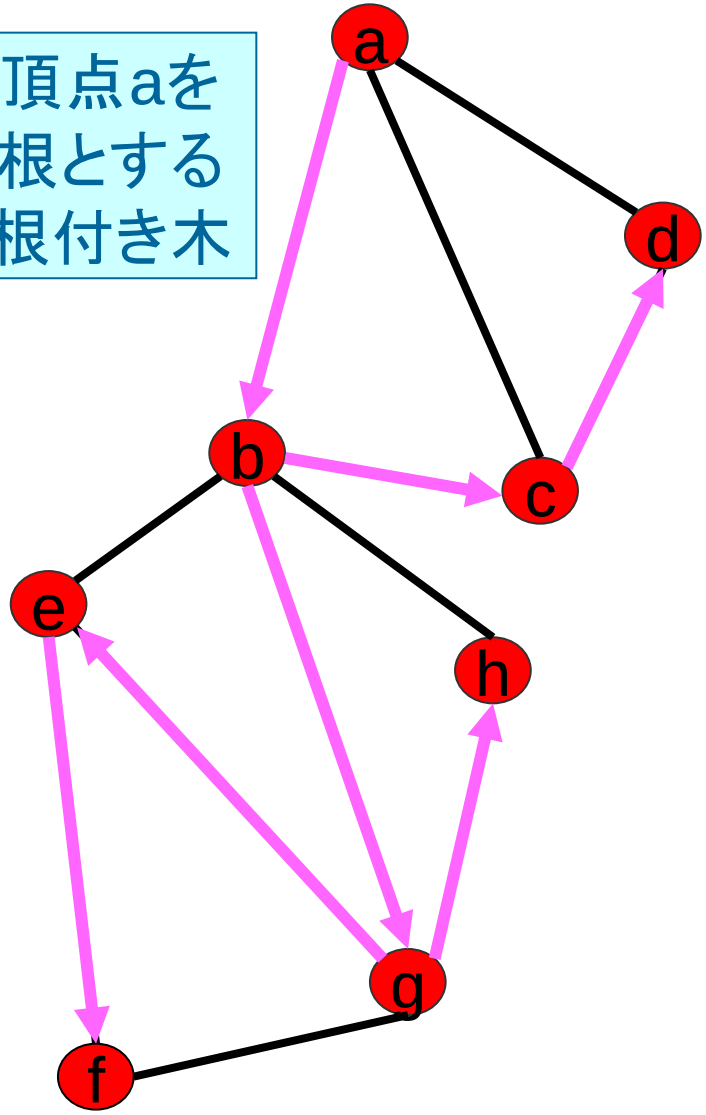
➔ 2連結成分等を計算するときに便利



深さ優先探索に関する性質

性質1:
無向グラフGが連結なとき,
新たな頂点を走査するときに使った
枝全体は全域木(最初に走査した頂点
を根とする根付き木) T になる

頂点aを
根とする
根付き木

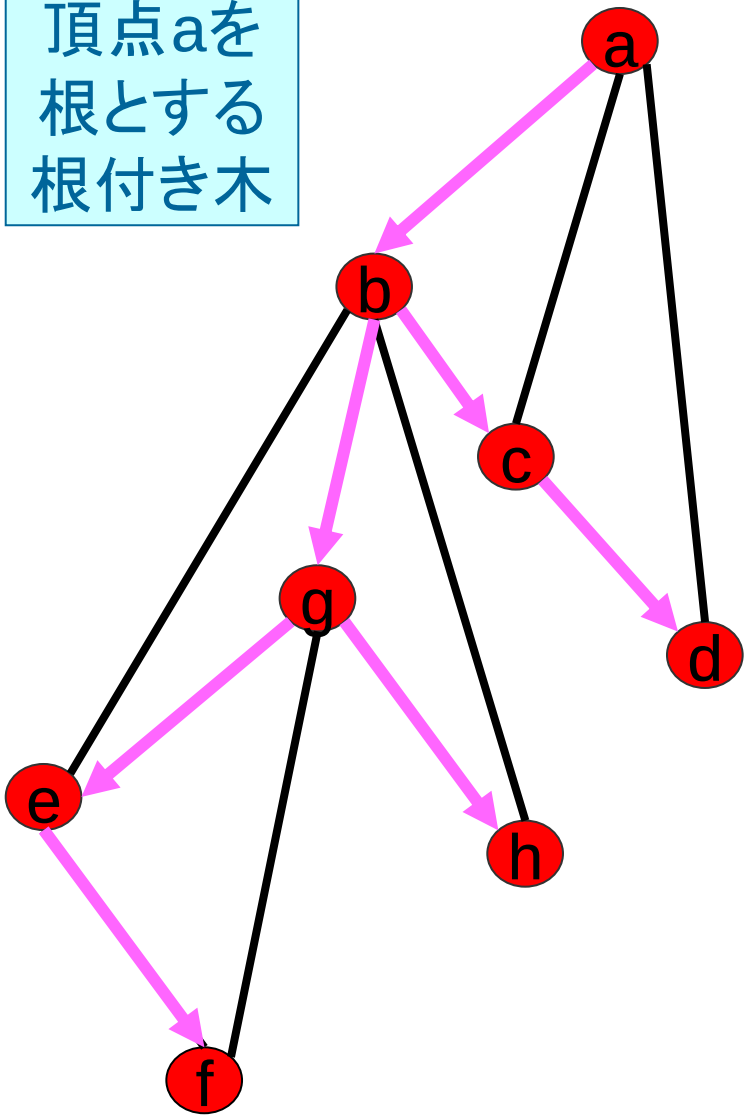


深さ優先探索に関する性質

性質2：根付き木Tにおいて，
頂点v は頂点u の子供
←→
(i) v より先に u が走査される
(ii) 枝 (u, v)が存在し，
DFS-VISIT(u) の実行時に，
DFS-VISIT(v) が再帰呼び出しされる

性質2'：根付き木Tにおいて，
頂点v は頂点u の子孫
←→ (i) v より先に u が走査される
(ii) DFS-VISIT(u) が終了するまでに v は走査される

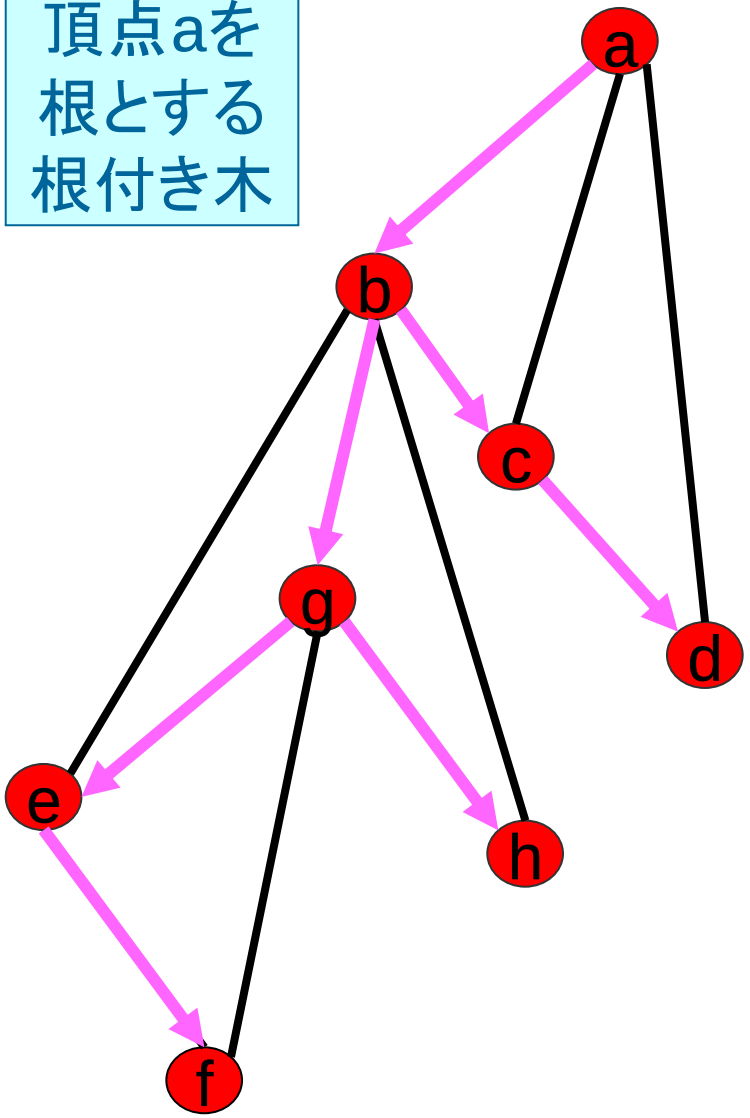
頂点aを根とする根付き木



深さ優先探索に関する性質

性質3:
根付き木Tに含まれない全ての枝は,
先祖と子孫を結ぶ枝である.

頂点aを
根とする
根付き木



証明: (u,v) はTに含まれない枝とする.
v より先に u が走査されたと仮定
➔アルゴリズムの動きより,
DFS-VISIT(u) が終了するまでに
v は必ず走査される
➔性質2' より, v は u の子孫

深さ優先探索の計算時間

(1) 各頂点，各枝を白く塗る
(2) 各頂点 u∈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

各頂点vに対し，
色が白➔DFS-VISIT(v)実行
　　v を黒く塗る
色が黒➔何もしない
∴ 各頂点vに対し
　　DFS-VISIT(v)は
　　ちょうど一回実行される

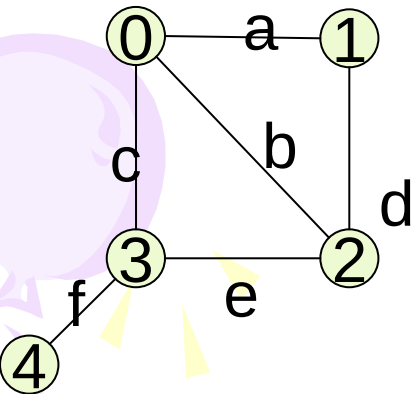
無向グラフの
データ構造に依存

DFS-VISIT(v)の実行時間：
再帰呼び出しを除くと，（v に接続する枝を求める時間）+O(d(v))

深さ優先探索の計算時間

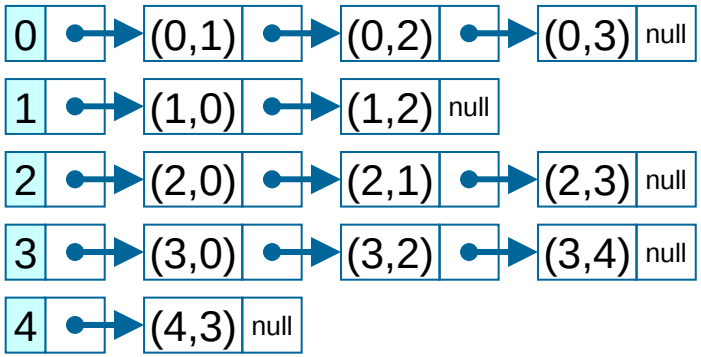
頂点 v に接続する枝を求める時間

- 接続行列を用いた場合：
  - 頂点v の行の要素を全て調べるので，O(m)時間
- 隣接行列を用いた場合：
  - 頂点v の行の要素を全て調べるので，O(n)時間
- 隣接リストを用いた場合：
  - 頂点vのリストの要素を全て調べるので，O(d(u))時間



	a	b	c	d	e	f
0	1	1	1	0	0	0
1	1	0	0	1	0	0
2	0	1	0	1	1	0
3	0	0	1	0	1	1
4	0	0	0	0	0	1

	0	1	2	3	4
0	0	1	1	1	0
1	1	0	1	0	0
2	1	1	0	1	0
3	1	0	1	0	1
4	0	0	0	1	0



深さ優先探索の計算時間

(1) 各頂点，各枝を白く塗る
(2) 各頂点 u∈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

各頂点vに対し，
色が白➔DFS-VISIT(v)実行
　　v を黒く塗る
色が黒➔何もしない
∴ 各頂点vに対し
　　DFS-VISIT(v)は
　　ちょうど一回実行される

データ構造として
隣接リストを利用

DFS-VISIT(v)の実行時間：
再帰呼び出しを除くと O(d(v))

深さ優先探索の実行時間：
$O(\Sigma_v\{d(v)+1\})=O(m+n)$

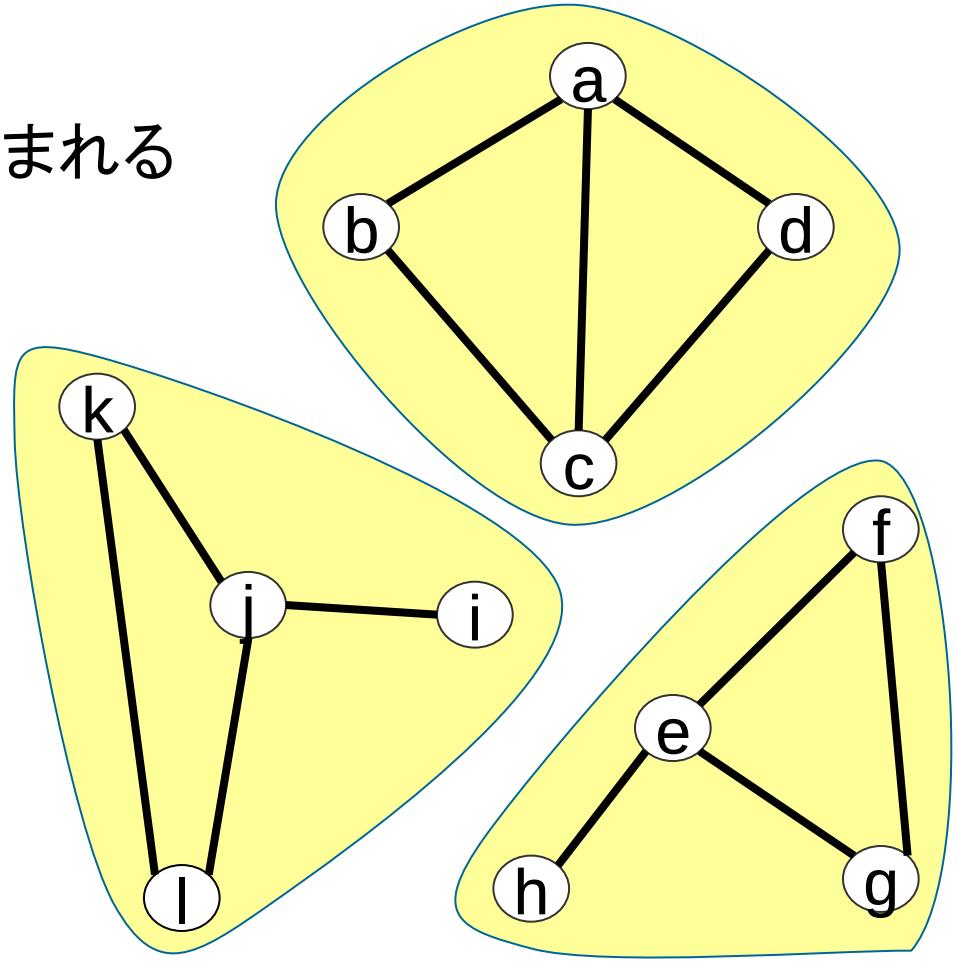
無向グラフの連結成分

- 無向グラフG＝(V，E)において，

頂点 u，v は同じ連結成分に含まれる⬅➡ uからvへの路が存在

- Gは連結⬅➡全ての頂点が
  同じ連結成分に含まれる
- Gの連結成分分解

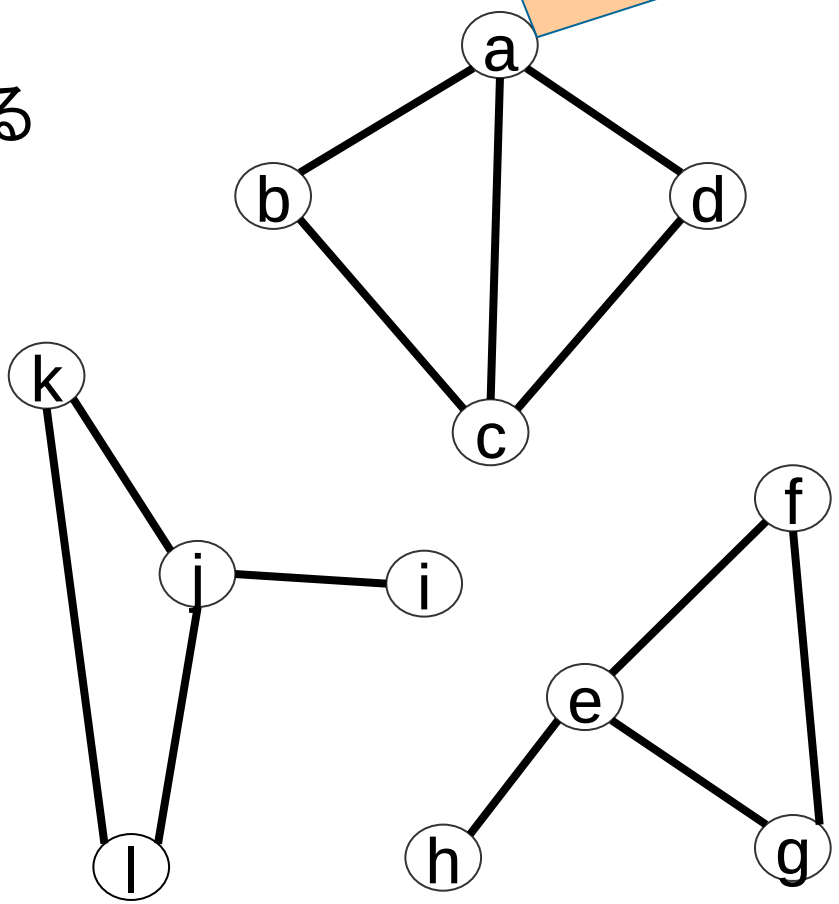
⬅➡極大な連結部分グラフに
よってグラフを分割したもの



連結成分を求める

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

DFS-VISIT(a)を実行
➔a,b,c,dのみが
走査される



補題：グラフのある頂点 u に対して
DFS-VISIT(u)を実行すると，
u と同じ連結成分に含まれる
頂点のみが走査され，
別の連結成分に含まれる
頂点は走査されない

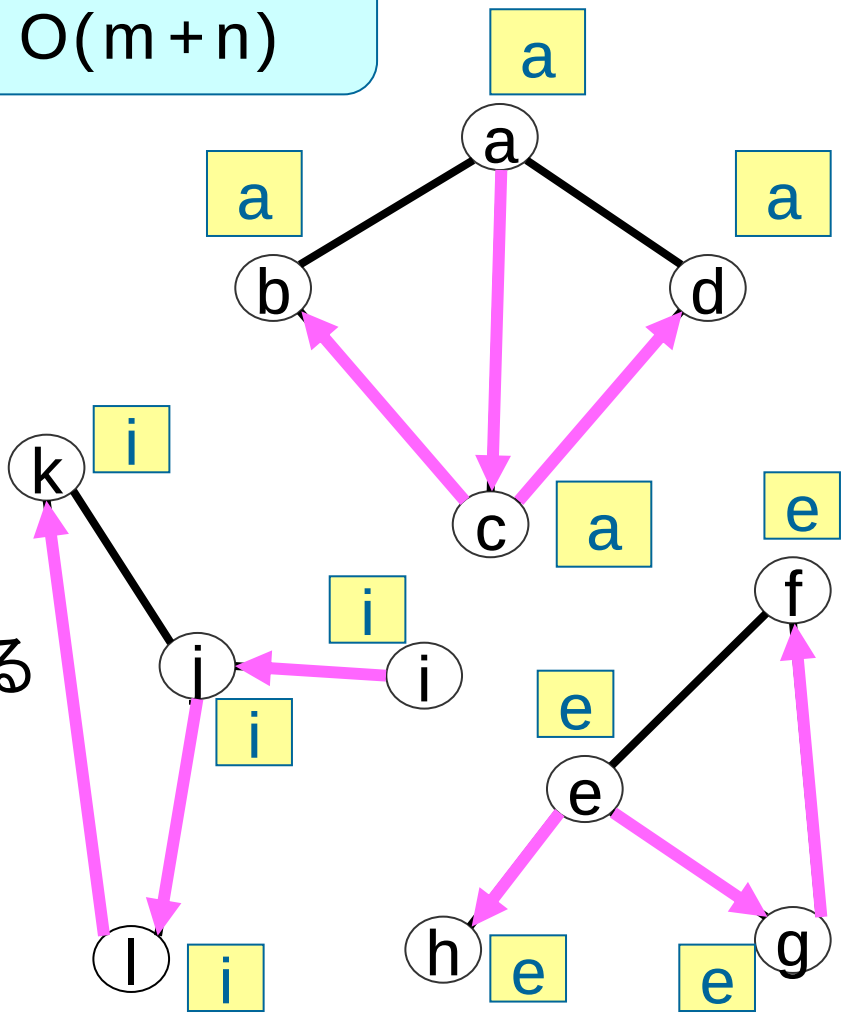
連結成分分解を求める

(1) 各頂点，各枝を白く塗る
(2) 各頂点 u∈V に対し，
　u が白色（未走査）ならば
　k = u とおき，
　手続きDFS-VISIT(u)を実行

手続き DFS-VISIT(u)
(a) u を黒く塗り，ラベルkを付ける
(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

同じラベルの頂点集合＝連結成分

計算時間は
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

  ←→u，v 以外の頂点 w を削除しても，

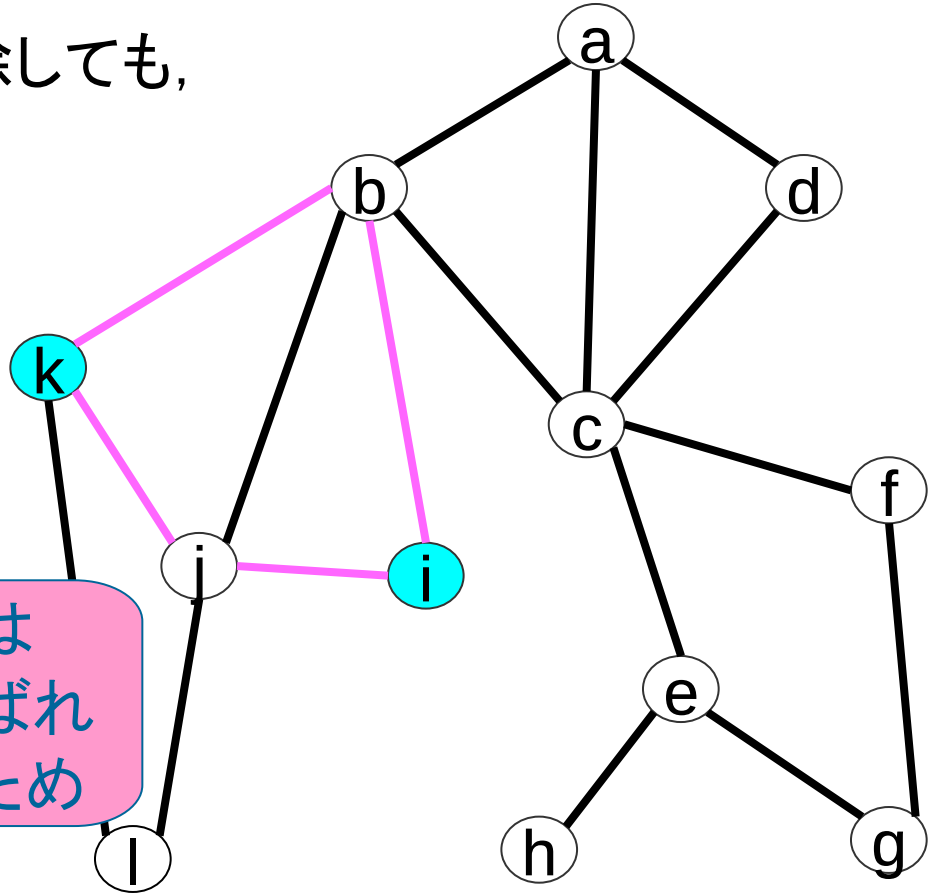
  uからvへの路が存在

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

  ←→u，v 以外の頂点 w を削除しても，

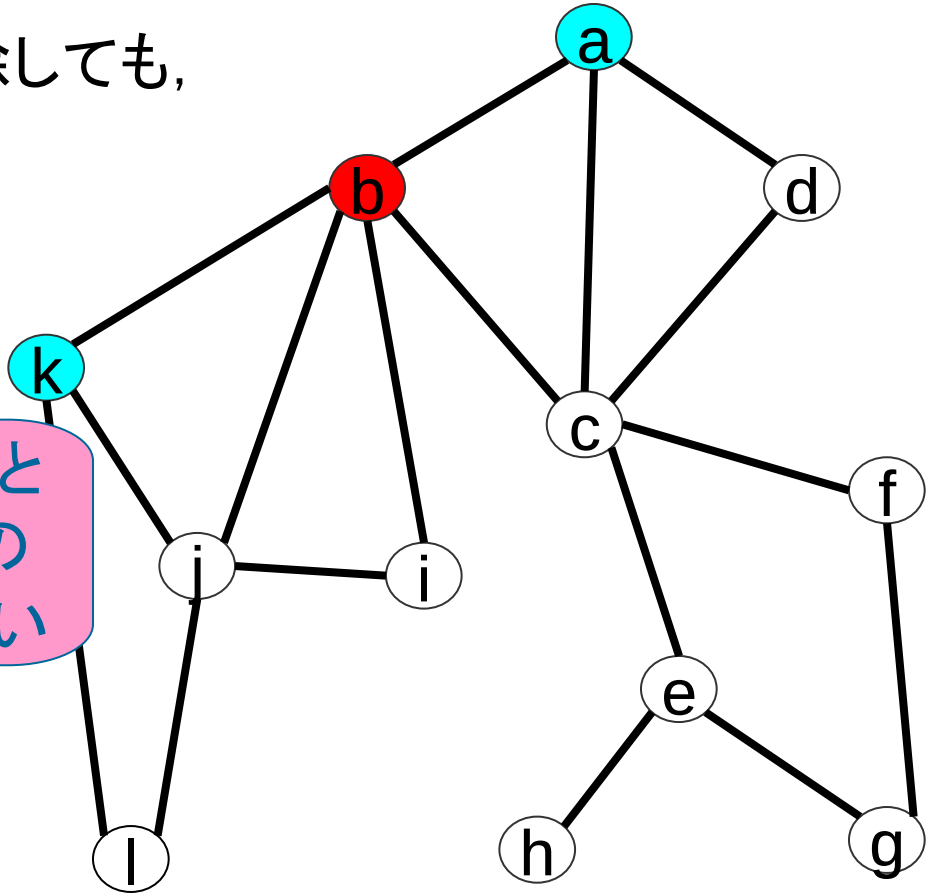
  uからvへの路が存在

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

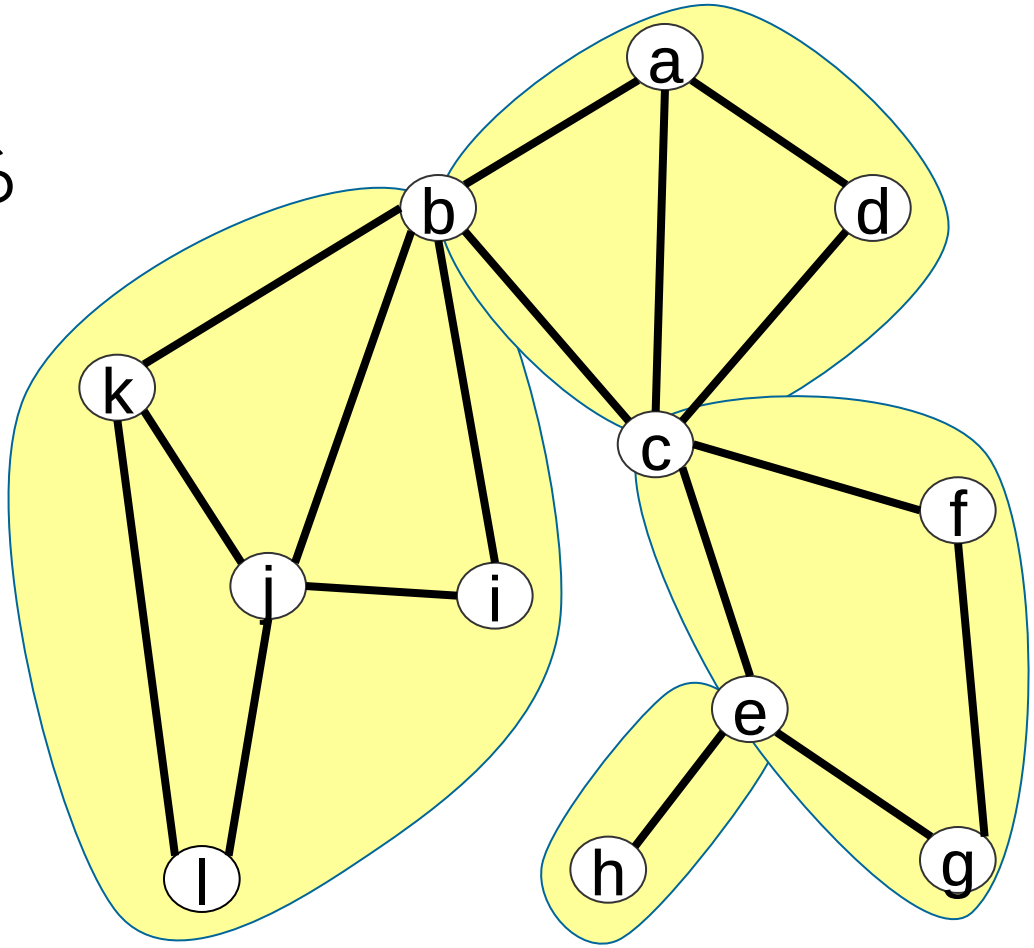


2連結成分分解と関節点

- 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

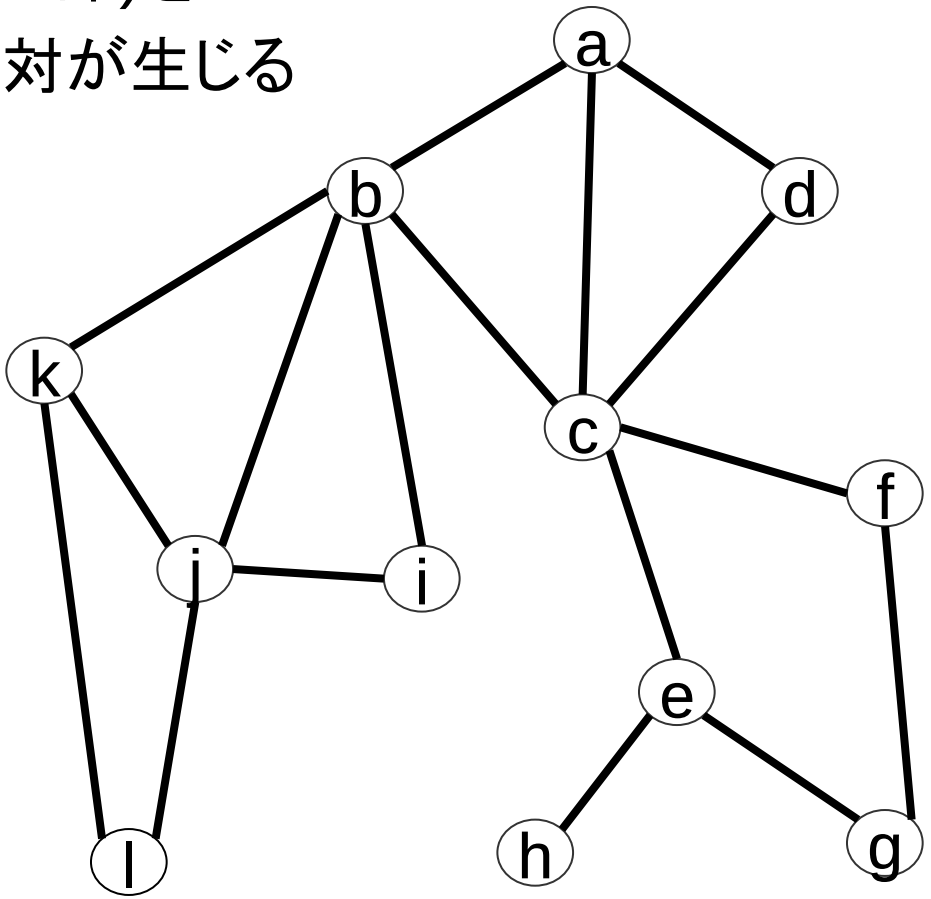
頂点 b, c, e は関節点



無向グラフの関節点

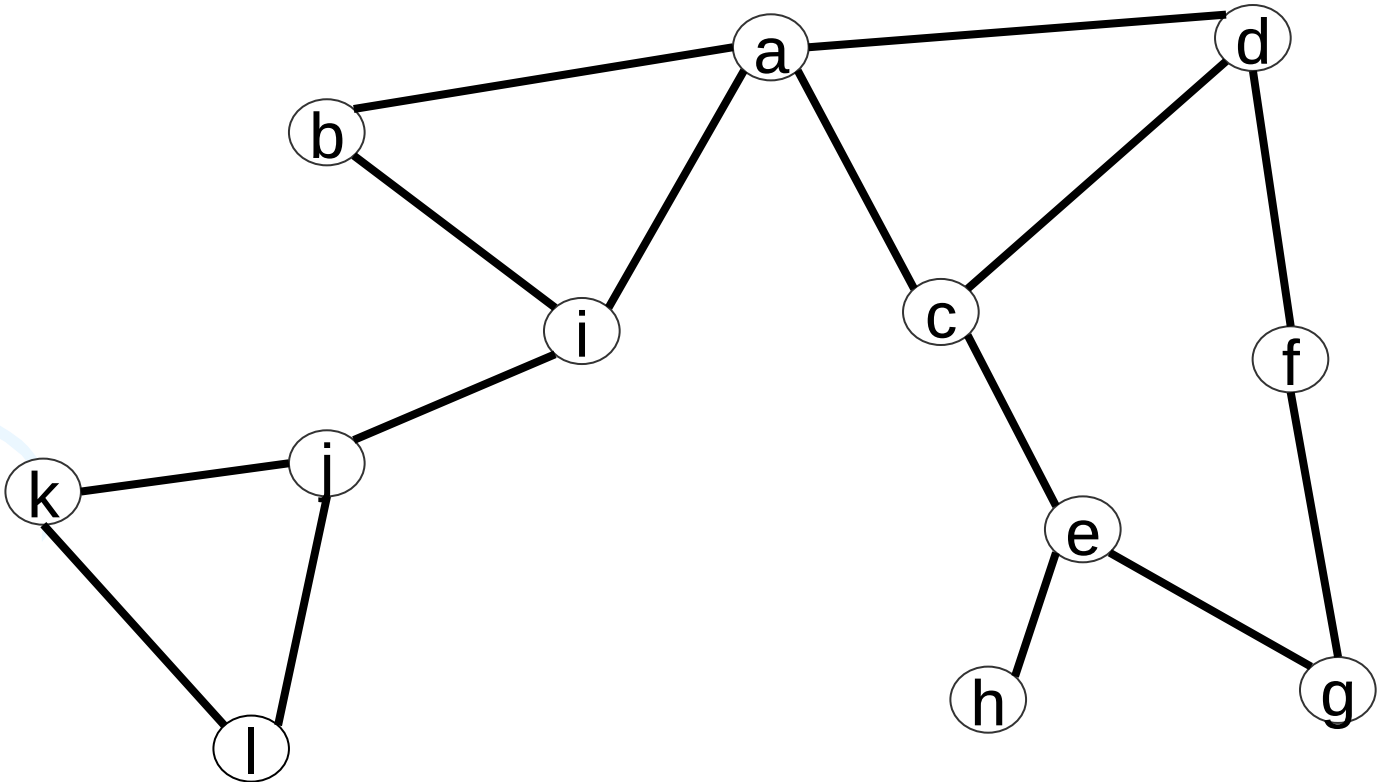
- **性質**：無向グラフG＝(V，E)において，頂点 u は関節点
  ←→u (および u に接続する枝全部)を
  削除すると，非連結になる頂点対が生じる

頂点 b，c，e は関節点
他の頂点は関節点ではない



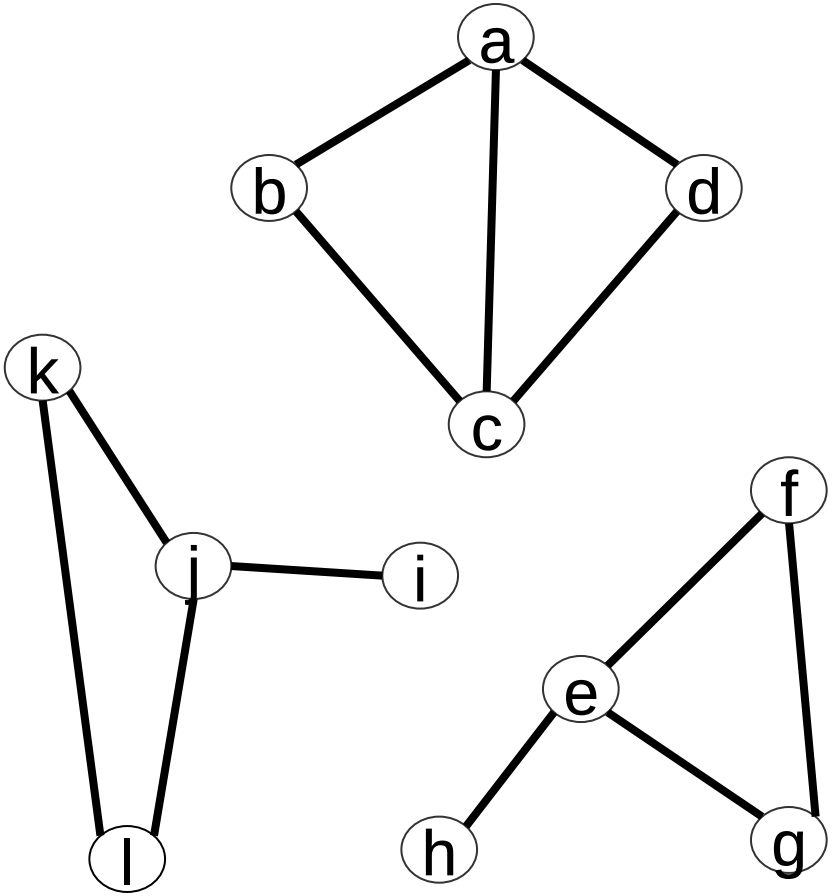
レポート問題その1
（締切：7／23授業開始10分後）

- 以下のグラフに対して，深さ優先探索を実行して
  各頂点が走査された順番の番号及び根付き木Tを計算しなさい.
  ただし頂点 a から深さ優先探索を開始するものとする.



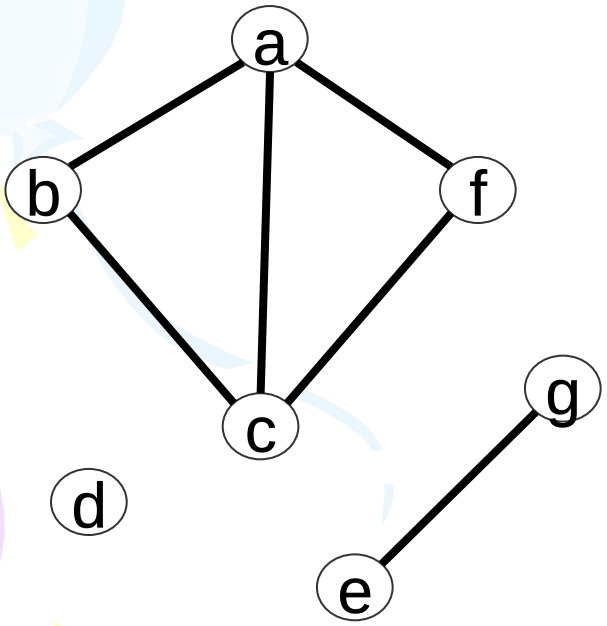
レポート問題その2
（締切：7／23授業開始10分後）

- 以下のグラフに対して，深さ優先探索を実行して
  各頂点が走査された順番の番号及び根付き木Tを計算しなさい.



レポート問題（締切：7／23）

- 深さ優先探索により連結成分分解を行なうプログラムを作成しなさい.
  使うグラフのデータ構造は，接続行列または隣接行列でOK.
  出力としては，各頂点のラベルを出力すればよい.



例：左のグラフの場合，
各頂点のラベルを label という
配列に保存し，その中身を
出力すればよい

a	b	c	d	e	f	g
a	a	a	d	e	a	e