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

期末試験について

- 日時：8月5日（木）8:50～10:20
- 受験資格：
  - 中間試験に合格（合格49名，不合格9名）
  - 中間試験以降にレポートを一回以上提出
- 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（動的計画法）～第13回（最終回）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6日（金）までに終える予定

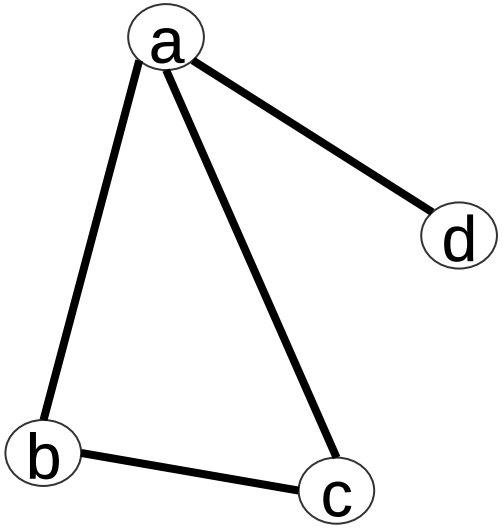
グラフの深さ優先探索

- 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

アルゴリズム

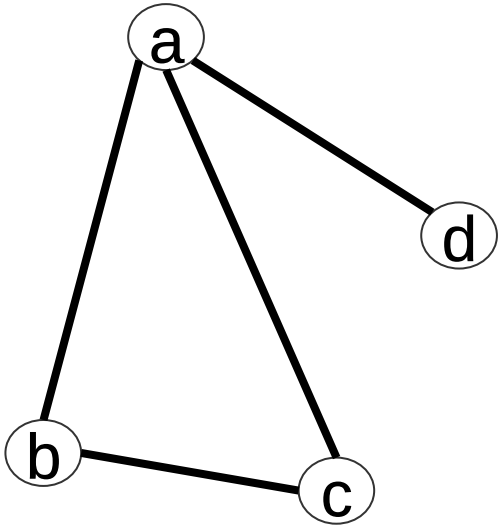
(1) 各頂点，各枝を白く塗る
(2) 各頂点 u∈V に対し，
　　u が白色（未走査）ならば
　　手続きDFS-VISIT(u)を実行

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



実行例1

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（再帰呼び出し）
- 枝(a,c)を走査
  - DFS-VISIT(b)を実行（再帰呼び出し）
- 枝(a,d)を走査
  - DFS-VISIT(d)を実行（再帰呼び出し）

実行例1

DFS-VISIT(a)を実行

a は白➔黒く塗る

枝(a,b)を走査

(a,b)は白➔黒く塗る

b は白➔DFS-VISIT(b)を実行

b は白➔黒く塗る

枝(b,a)を走査

(b,a)は黒➔何もしない

枝(b,c)を走査

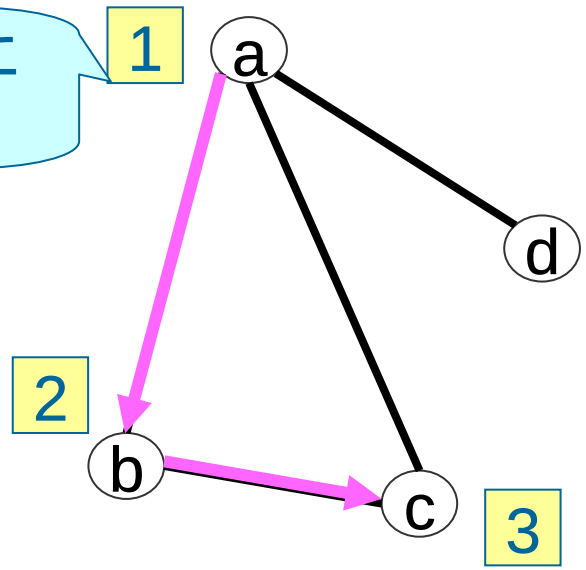
(b,c)は白➔黒く塗る

c は白➔DFS-VISIT(c)を実行

c は白➔黒く塗る

枝(c,a)を走査

頂点を走査した順に
番号を付けておく

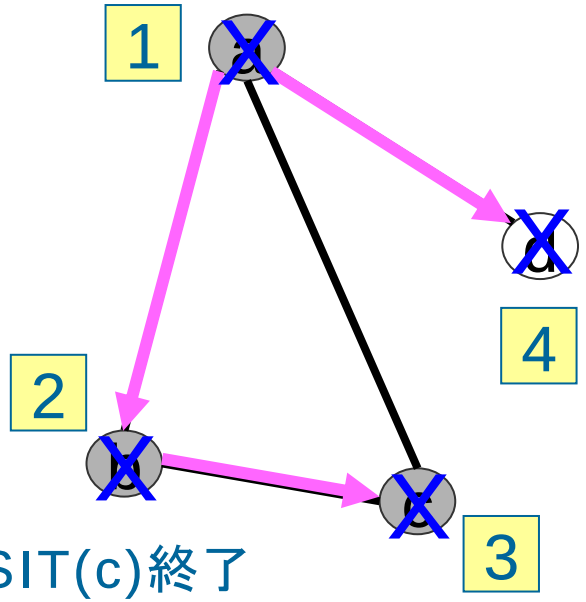


手続き DFS-VISIT(u)

(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

実行例1



枝(c,a)を走査

(c,a)は白➔黒く塗る

a は黒➔何もしない

枝(c,b)を走査

(c,b)は黒➔何もしない

cに接続する枝の走査終了➔DFS-VISIT(c)終了

bに接続する枝の走査終了➔DFS-VISIT(b)終了

枝(a,c)を走査

(a,c)は黒➔何もしない

枝(a,d)を走査

(a,d)は白➔黒く塗る

d は白➔DFS-VISIT(d)を実行

d は白➔黒く塗る

枝(d,a)を走査

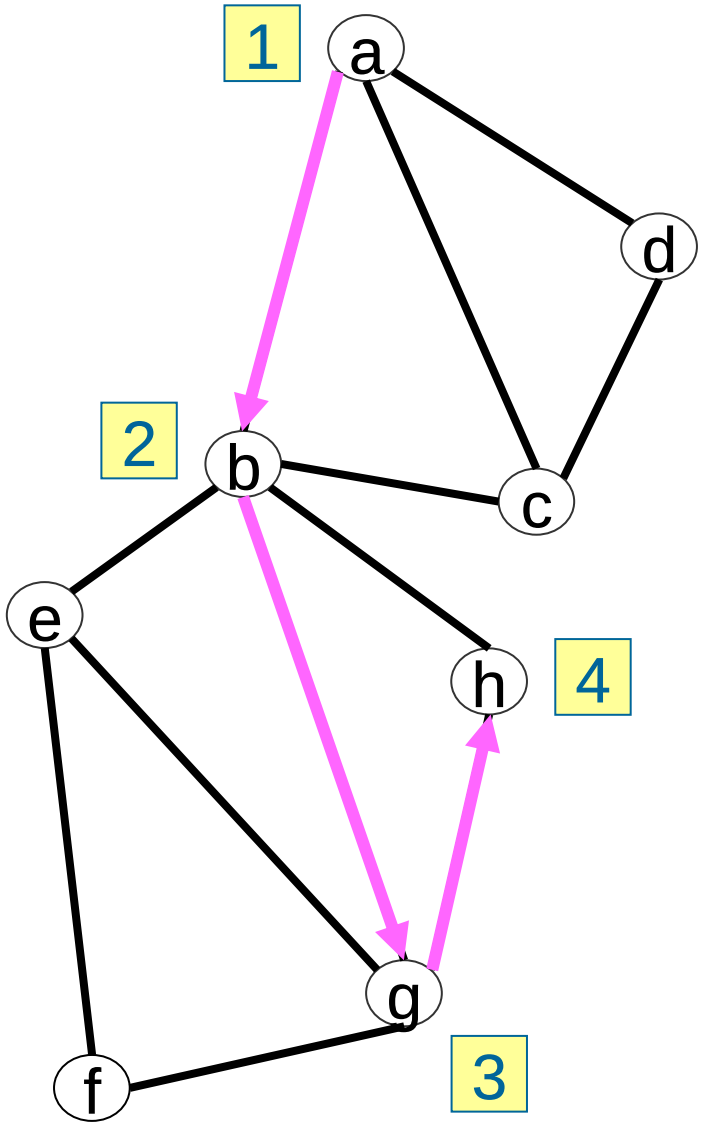
(d,a)は黒➔何もしない

dに接続する枝の走査終了➔DFS-VISIT(d)終了

aに接続する枝の走査終了➔DFS-VISIT(a)終了

実行例2

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



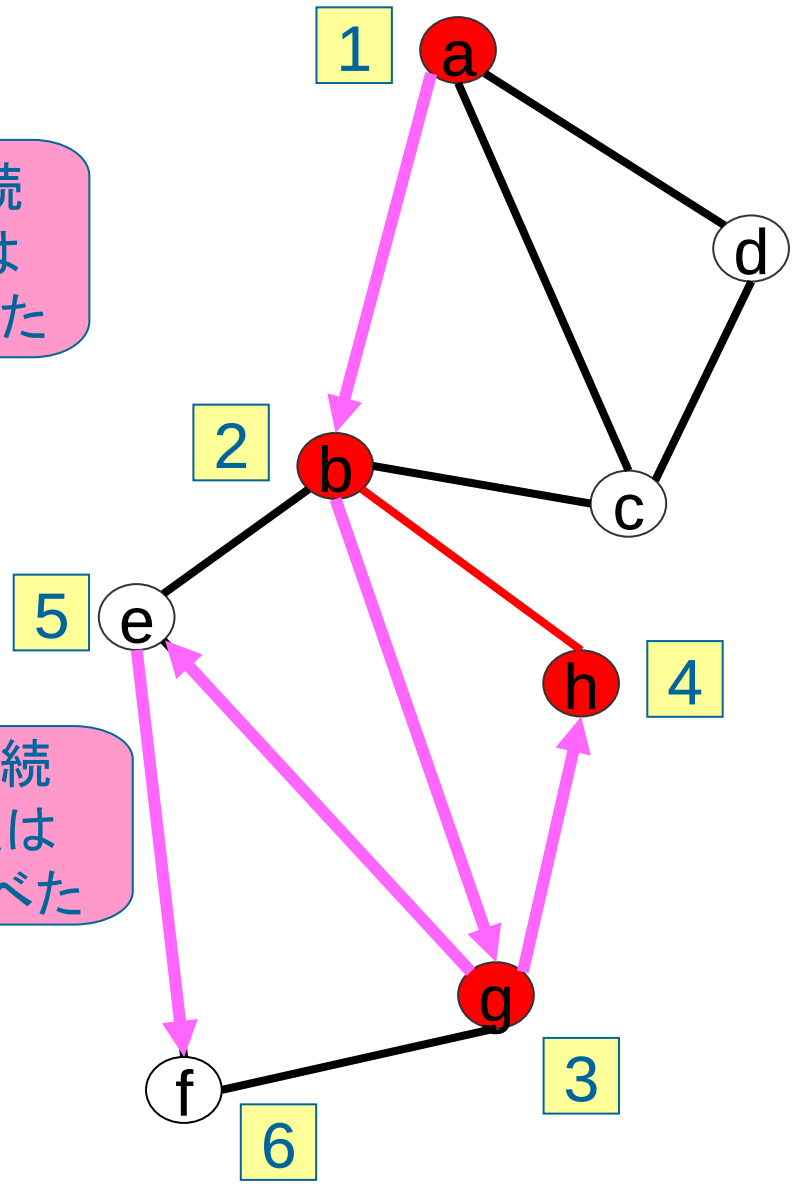
DFS-VISIT(a)を実行
枝(a,b)を走査
DFS-VISIT(b)を実行
枝(b,g)を走査
DFS-VISIT(g)を実行
枝(g,h)を走査
DFS-VISIT(h)を実行
枝(h,b)を走査
枝(h,g)は走査済み

実行例2

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



実行例2

⋮

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

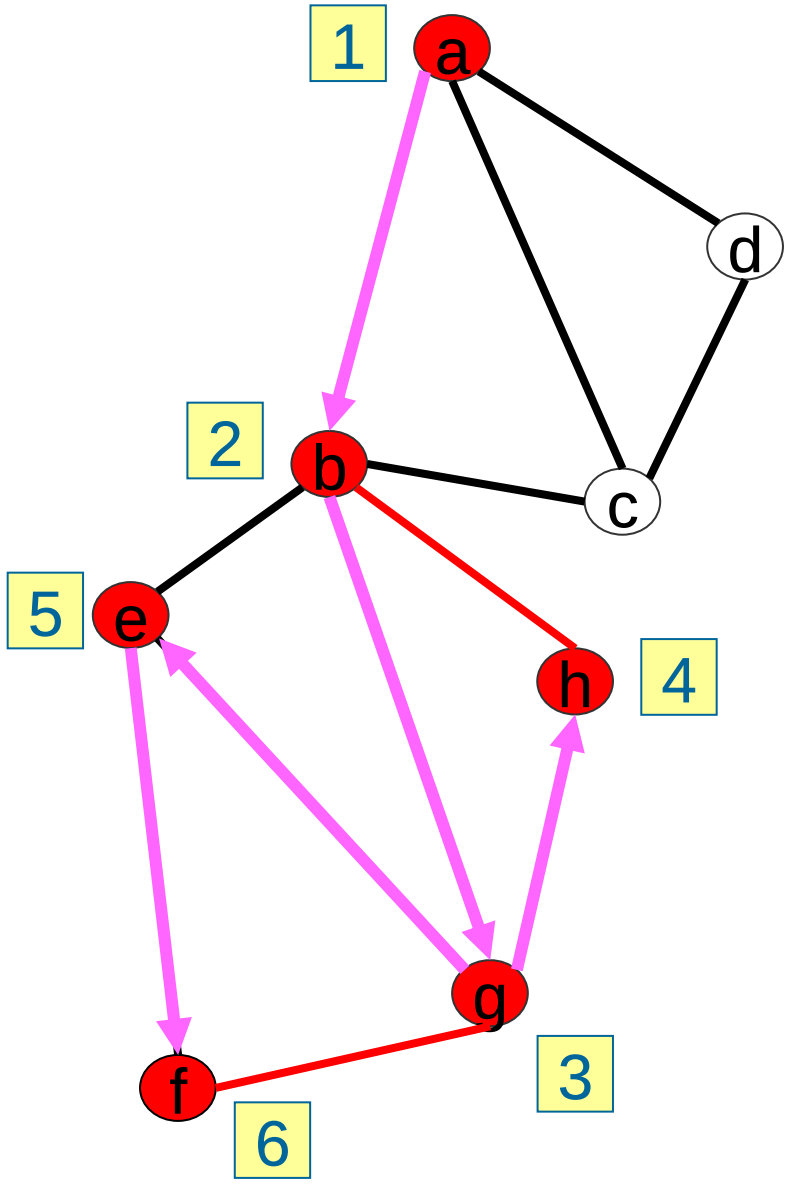
枝(e,g)は走査済み

枝(e,b)を走査

DFS-VISIT(e)終了

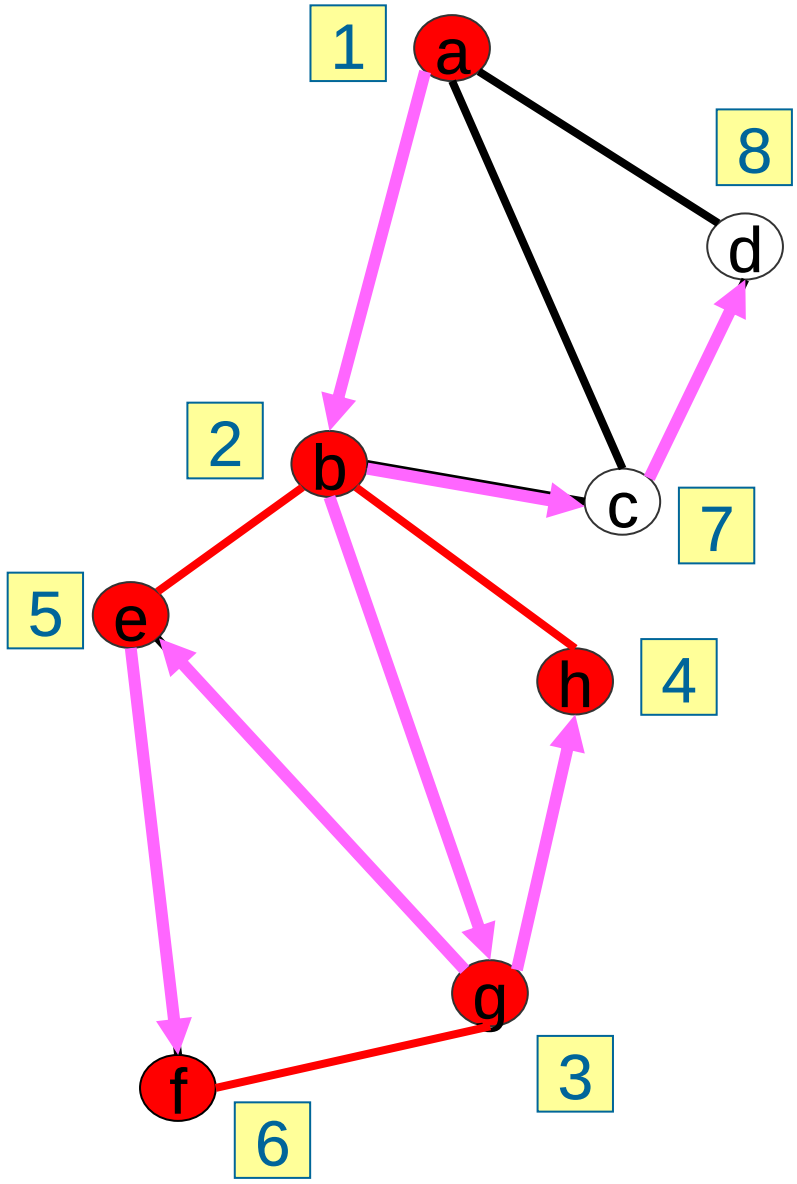
枝(g,b)は走査済み

DFS-VISIT(g)終了



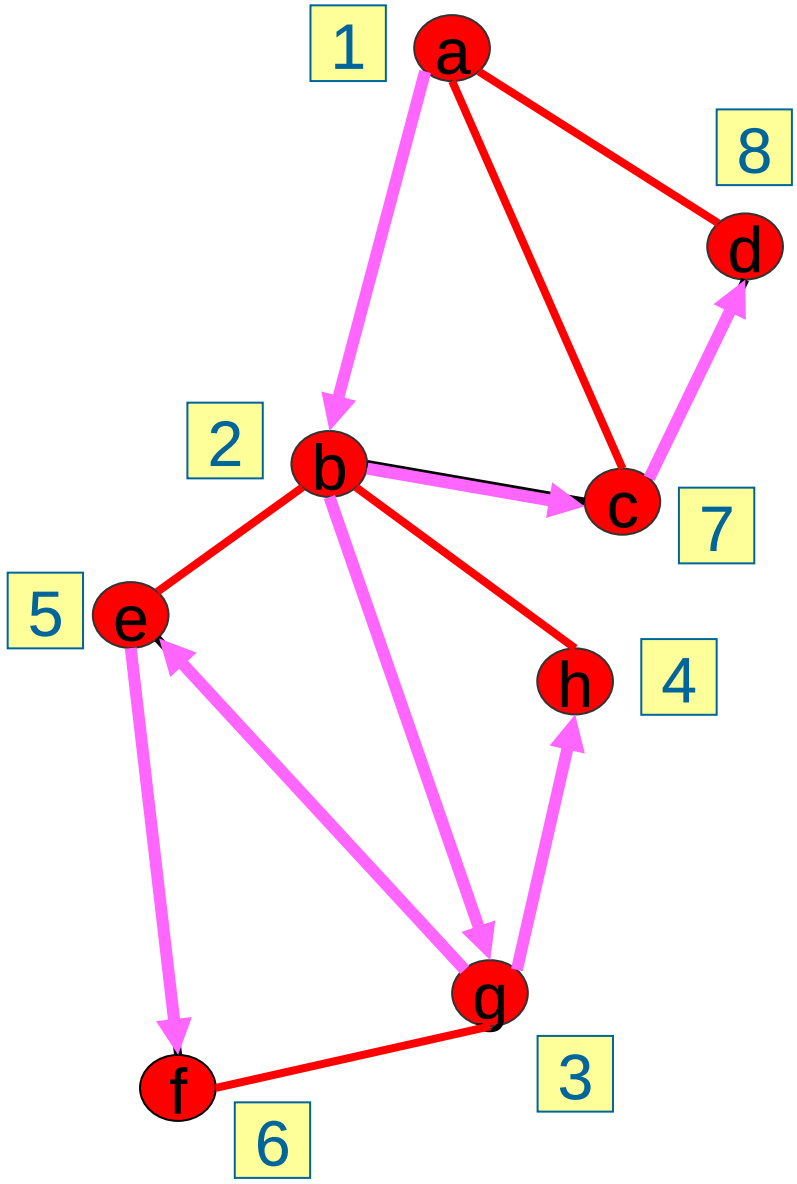
実行例2

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



実行例2

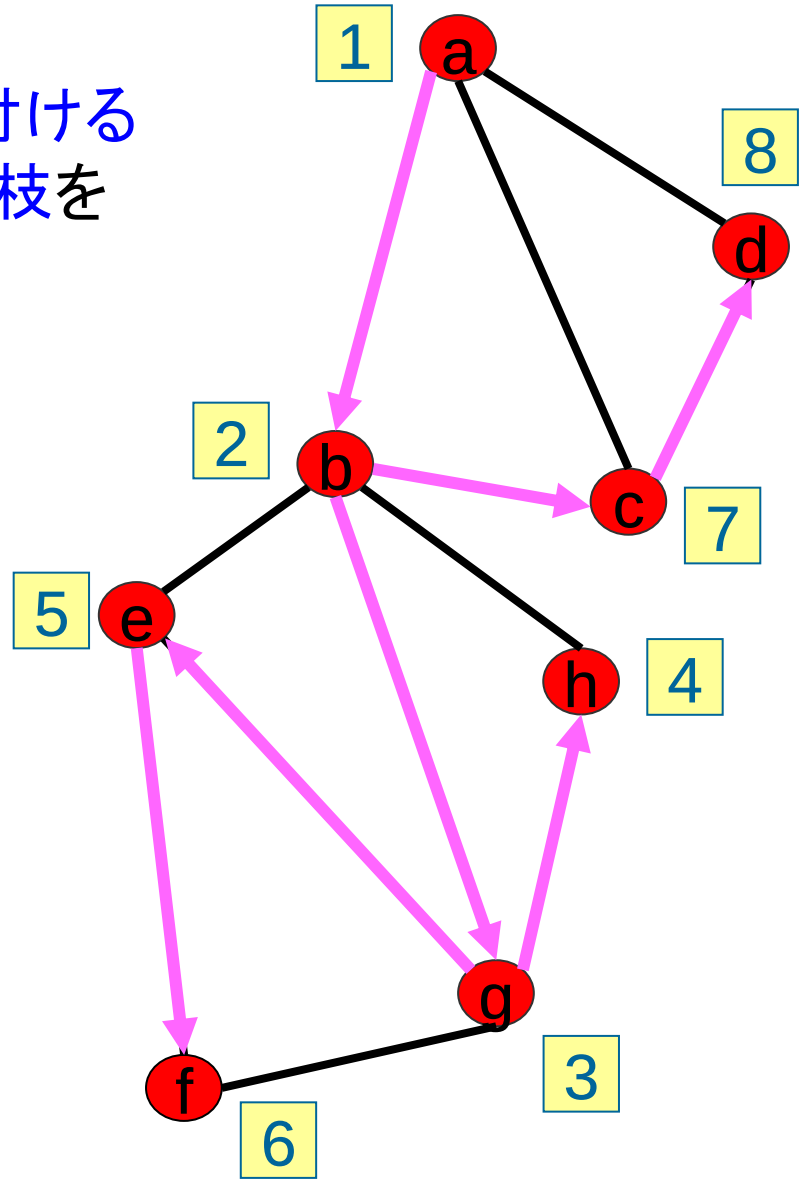
DFS-VISIT(a)を実行
枝(a,b)を走査
DFS-VISIT(b)を実行
枝(b,g)を走査
⋮
枝(b,e)は走査済み
枝(b,c)を走査
DFS-VISIT(c)を実行
枝(c,b)は走査済み
枝(c,a)を走査
枝(c,d)を走査
⋮
DFS-VISIT(c)終了
枝(b,a)は走査済み
DFS-VISIT(b)終了
枝(a,b)は走査済み
DFS-VISIT(a)終了



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

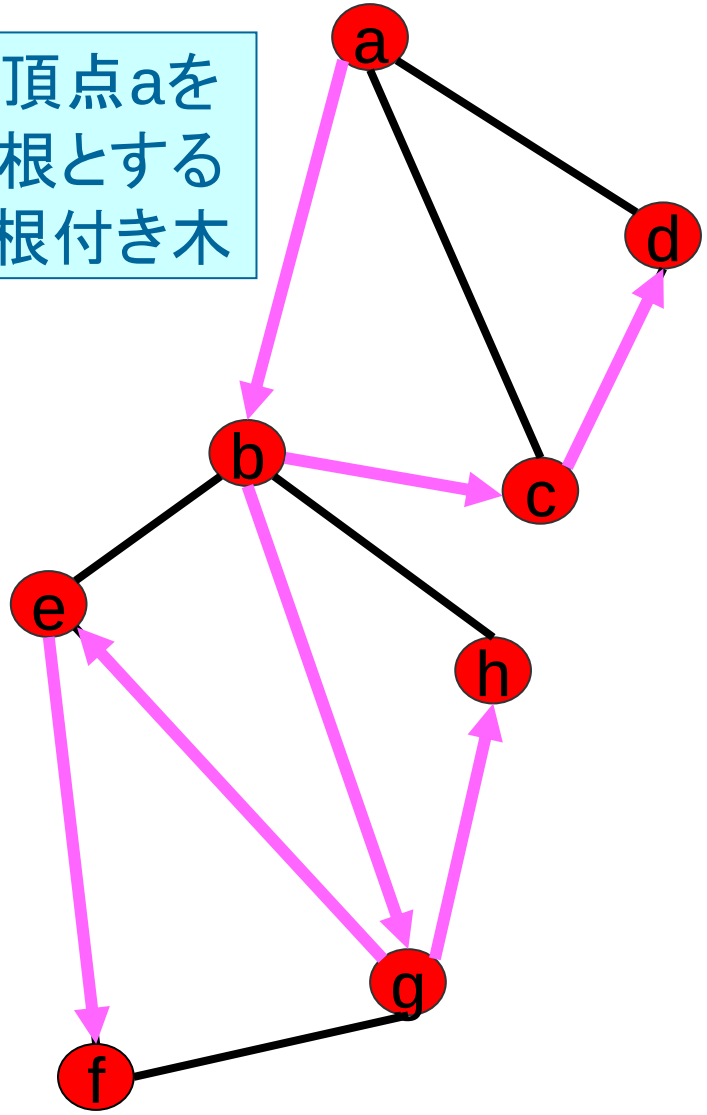


深さ優先探索木

性質1：
無向グラフGが連結な場合，
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）になる

深さ優先探索木
と呼ばれる

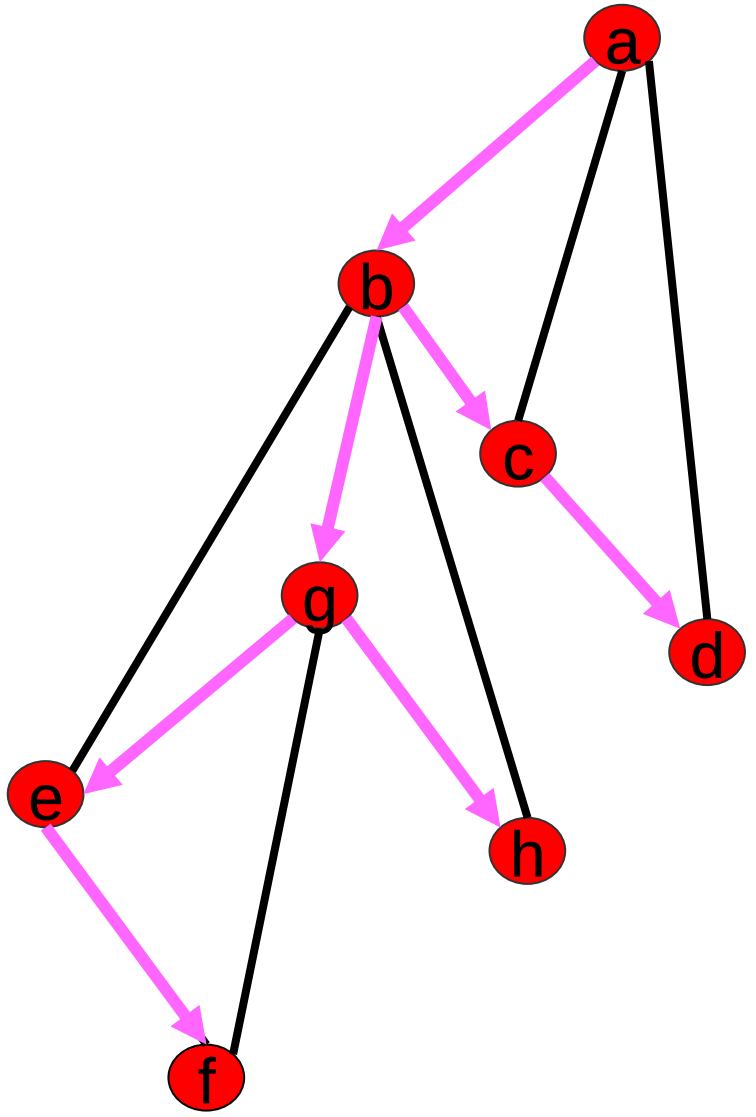
頂点aを
根とする
根付き木



深さ優先探索木に関する性質

性質2：深さ優先探索木 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

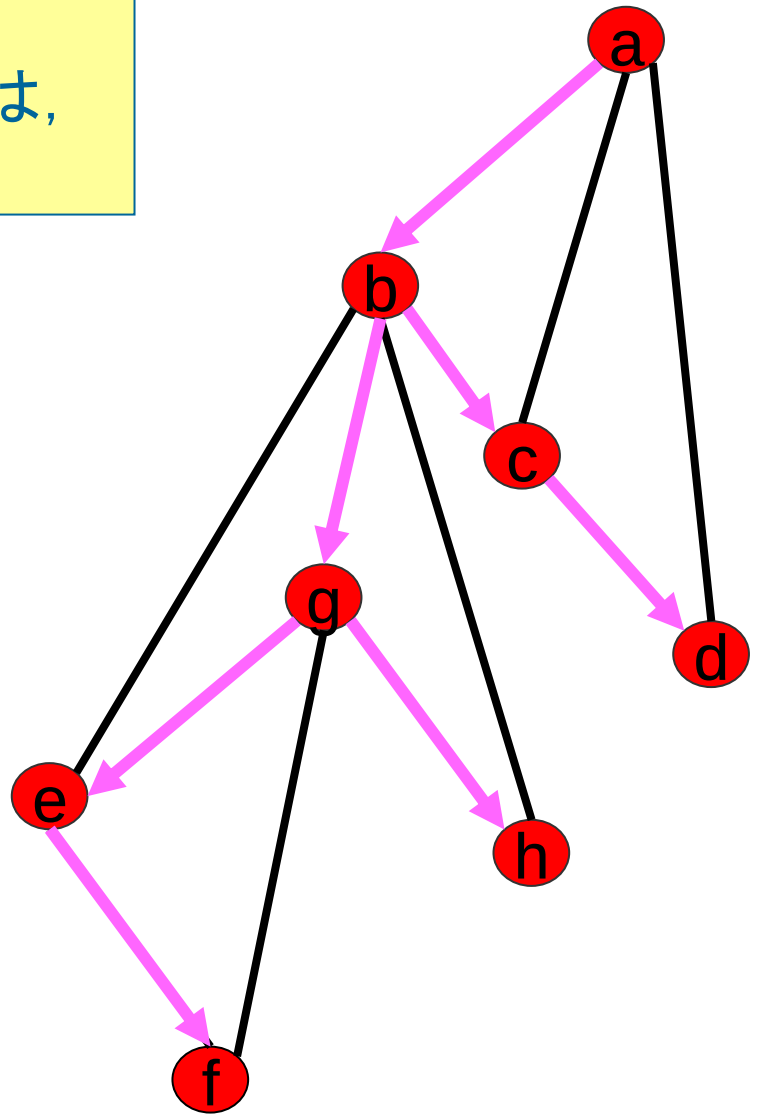
性質2'：深さ優先探索木 T において，
頂点v は頂点u の子孫
←→ (i) v より先に u が走査される
(ii) DFS-VISIT(u) が終了する
までに v は走査される



深さ優先探索木に関する性質

性質3:
深さ優先探索木Tに含まれない全ての枝は,
先祖と子孫を結ぶ枝である.

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

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

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

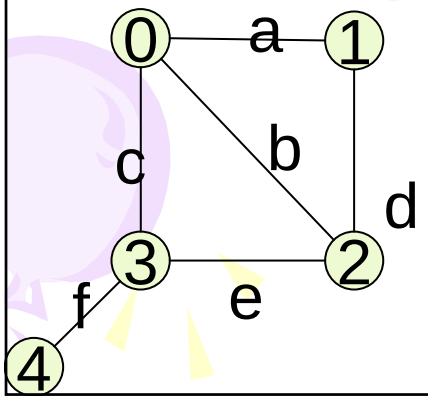
無向グラフの
データ構造に依存

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

深さ優先探索の計算時間

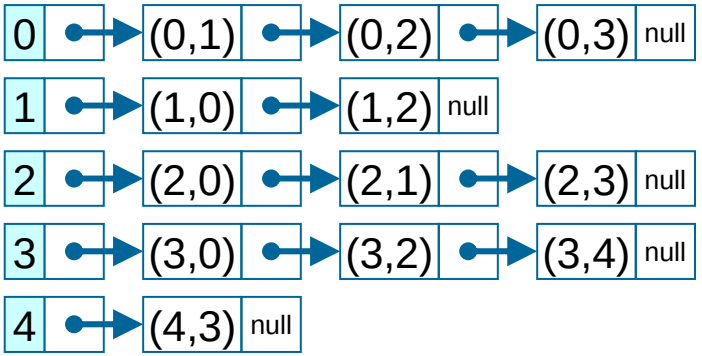
頂点 u に接続する枝を求める時間

- 接続行列を用いた場合：
  - 頂点u 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u 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u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 $u \in V$ 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

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(u)の実行時間：
再帰呼び出しを除くと O(d(u))

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

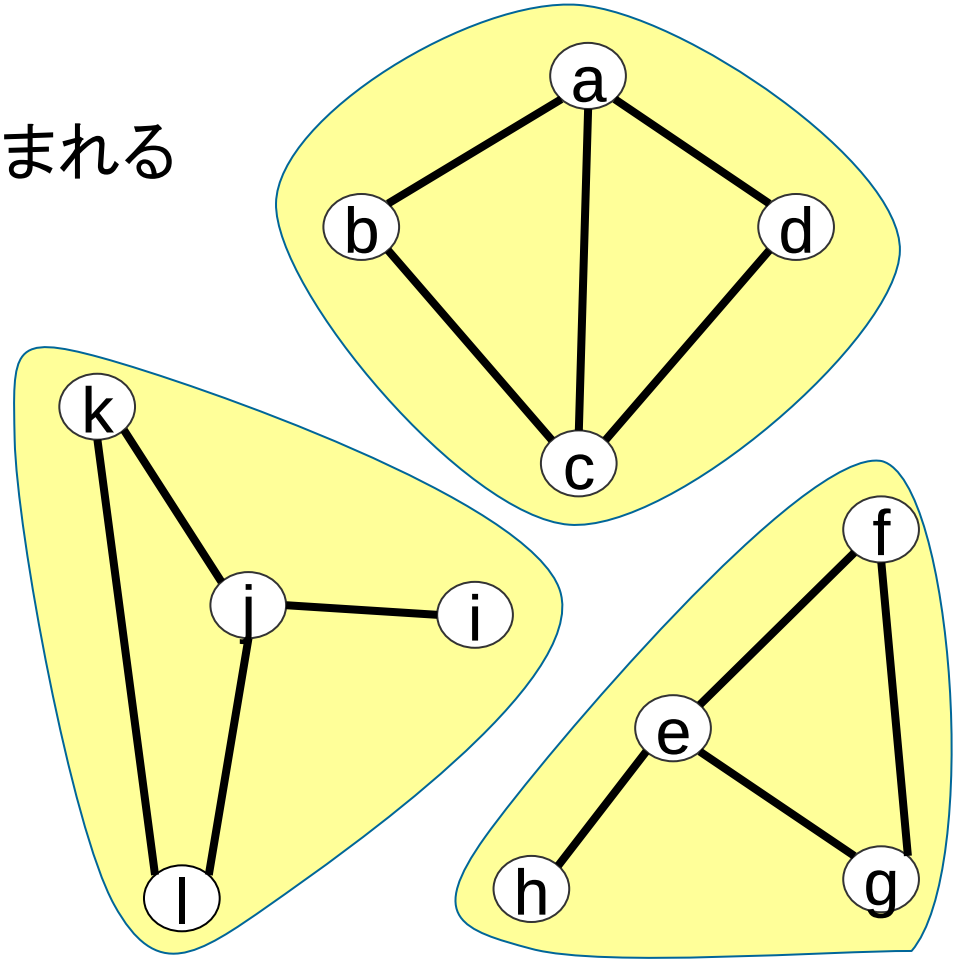
無向グラフの連結成分

- 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

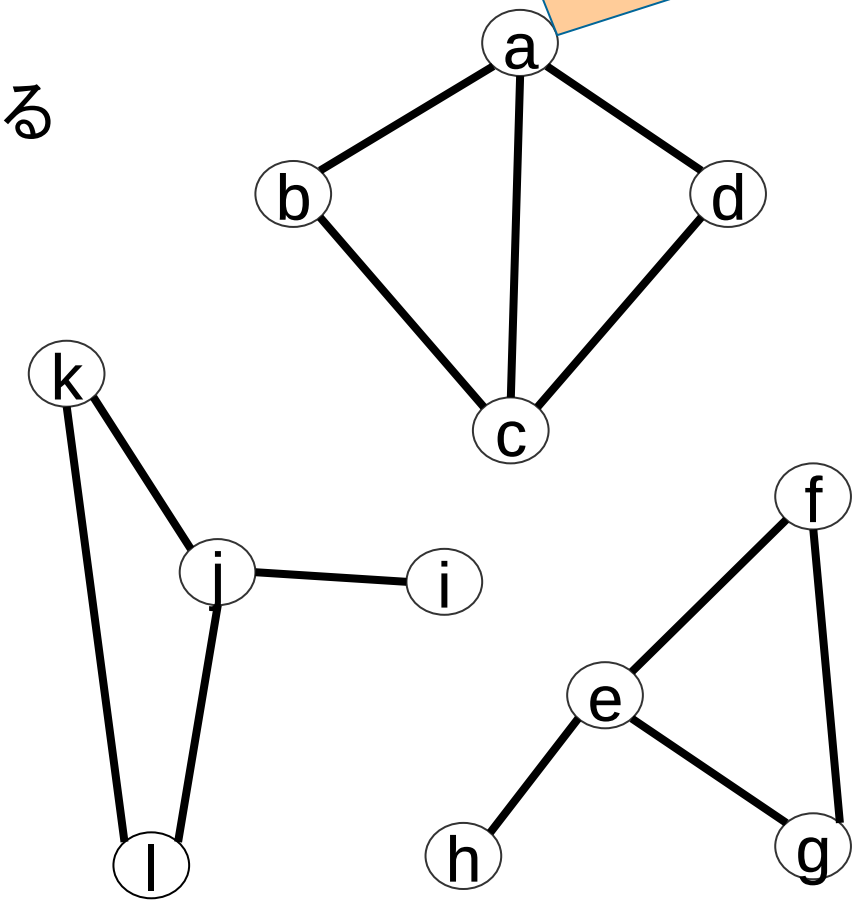


連結成分を求める

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

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

DFS-VISIT(a)を実行
➔a,b,c,dのみが
走査される



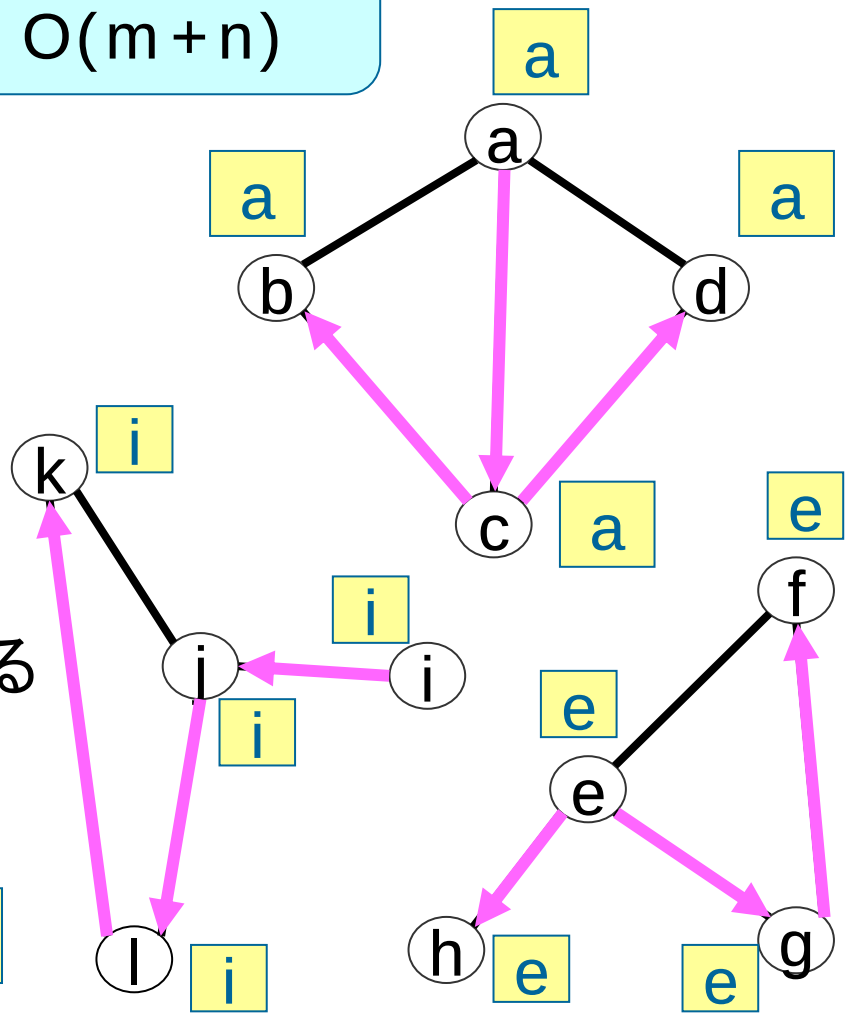
連結成分分解を求める

(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)



レポート問題(締切:7／22)

- 以下のグラフに対して，深さ優先探索を実行して
  num[v]の値及び根付き木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.