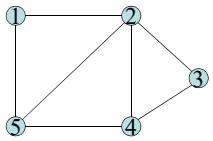
データ構造とアルゴリズムII

第7回
幅優先／深さ優先探索／トポロジカルソート

22. 基本的グラフアルゴリズム

無向グラフ

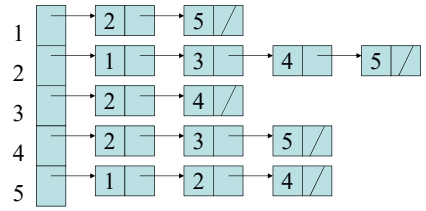
- 5個の頂点と7本の辺からなる無向グラフ



- 無向グラフ G=(V,E), Vは頂点集合, Eは辺集合. Eの要素は頂点のペア{u,v}によって表される. {u, v}と{v, u}は同じ辺を意味する.

隣接リスト

- 各頂点に関して, 隣接する(直接, 辺で結ばれた)頂点集合をリストで表現



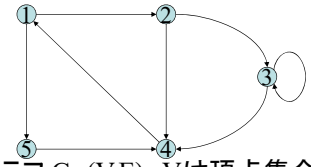
隣接行列

- 隣接関係を|V|×|V|の行列で表現. A[u][v]= 1/0により 辺{u, v} の存在／非存在を示す. 無向グラフなら行列は対称.

	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

有向グラフ

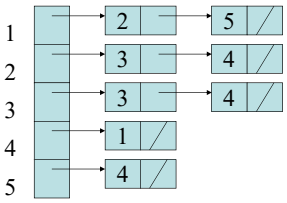
- 5個の頂点と8本の辺からなる有向グラフ



- 有向グラフ G=(V,E), Vは頂点集合, Eは辺集合. Eの要素は頂点の順序付きペア(u,v) によって表される. (u, v)と(v, u)は異なる. (u, u) なる辺(セルフループ)を許す.

隣接リスト

- 各頂点に関して，隣接する（自身からその頂点に向かう有向辺がある）頂点集合をリストで表現



338

隣接行列

- 隣接関係を|V|×|V|の行列で表現．A[u][v]= 1/0により 辺 (u, v) の存在／非存在を示す．対称とは限らない．

	1	2	3	4	5
1	0	1	0	0	1
2	0	0	1	1	0
3	0	0	1	1	0
4	1	0	0	0	0
5	0	0	0	1	0

339

幅優先探索（22.2）

- 幅優先探索 (Breadth-first search, BFS)は最も単純なグラフ探索アルゴリズムの一つ．
- 始点sが与えられたら，sから到達可能な節点を系統的に探索する．
- BFSはqueueを用いて実現可能．
- Prim の最小全域木アルゴリズムやDijkstraの単一始点最短路アルゴリズムはBFSの一種と考えられる．

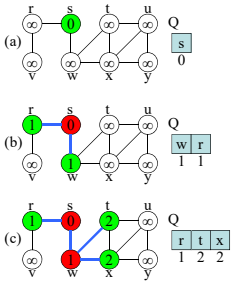
340

BFSによる探索

- まだ訪れていない節点の色は白．
- 初めて発見された節点は灰色（図中では緑）．
- 近傍がすべて発見されたら黒（図中では赤）．
- 灰色になる時に，始点sからの距離と，親となる節点を決定
- 有向でも無向でも動く（以下は無向連結グラフで説明）

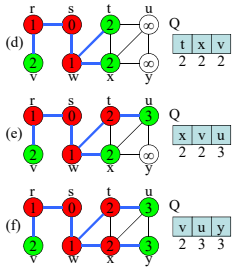
341

BFSによる探索例



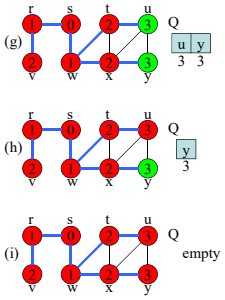
342

BFSによる探索例



343

BFSによる探索例

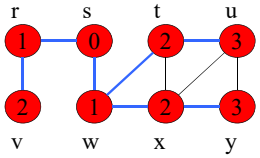



```
BFS(G,s)
1.  for each vertex u∈ G.V-{s}
2.        u.color = WHITE
3.        u.d = ∞
4.        u.π = NIL
5.  s.color = GRAY
6.  s.d = 0
7.  s.π = NIL
8.  Q = empty
9.  Enqueue(Q,s)
10. while Q≠empty
11.     u = Dequeue(Q)
12.     for each v∈ G.Adj[u]
13.       if v.color == WHITE
14.             v.color = GRAY
15.             v.d = u.d + 1
16.             v.π = u
17.             Enqueue(Q,v)
18.     u.color = BLACK
```

Time Complexity:
O(|E|+|V|)

BFSの性質

- 幅優先探索木が構成され，始点からの最短経路が与えられる．幅優先探索木は複数存在し得るが，始点からの最短経路は同じ．

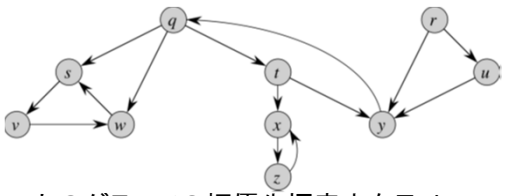


Breadth-first tree

有向／非連結グラフのBFS

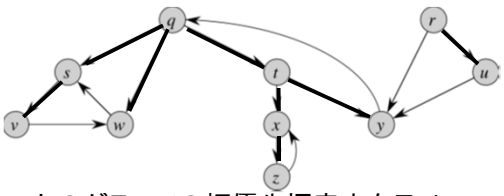
- 基本的な動作は同様
- queueが空になっても，まだ未発見の頂点が残っていれば，その一つを選んでqueueに加える

演習：幅優先探索



- 上のグラフでの幅優先探索木を示せ．ただし，頂点の選択順／隣接リストはアルファベット順とする．

演習：幅優先探索



- 上のグラフでの幅優先探索木を示せ．ただし，頂点の選択順／隣接リストはアルファベット順とする．

深さ優先探索（22.3）

- 可能ならばグラフの “より深い部分”を探索する.
- 未探索の外向辺が残る頂点中で，最後に発見した頂点vからの未探索の外向辺を探索
- vの辺をすべて探索し終わると，vを発見した時に通った辺を“バックトラック（逆戻り）”し，vの直前の頂点の未探索の外向辺を探索.
- この処理を元の始点から到達可能なすべての頂点を発見するまで続ける.
- 未発見の頂点が残っていれば，そのうち1つを新たな始点として選択し，そこから探索を続ける.
- 有向でも無向でも動く（以下は有向グラフで説明）

350

深さ優先探索アルゴリズム

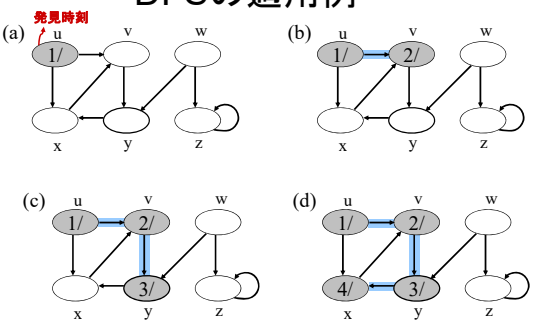
- 深さ優先探索は各頂点に時刻印をつける.
  どの頂点vも2つの時刻印をもつ.
- 最初の時刻印v.dはvを最初に発見したときにつける.
  このとき，頂点は灰色に塗られる.
  2番目の時刻印v.fは，vの隣接リストを調べ終えたときに記録する. このとき，頂点は黒色に塗られる.
- 時刻印をつけたら，時計の針が進む. $|V|$個の頂点それぞれについて発見する事象も終了する事象も一度しか起こらないので，時刻印は1から$2|V|$の範囲の整数となる.
- またすべての頂点について次式が成り立つ.

$$u.d < u.f$$

- 頂点uの色は時刻u.d以前は白，u.dとu.fの間は灰色，それ以降は黒である.

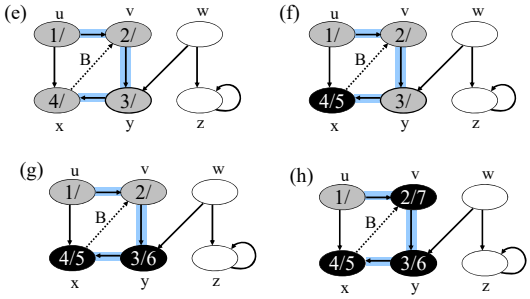
351

DFSの適用例



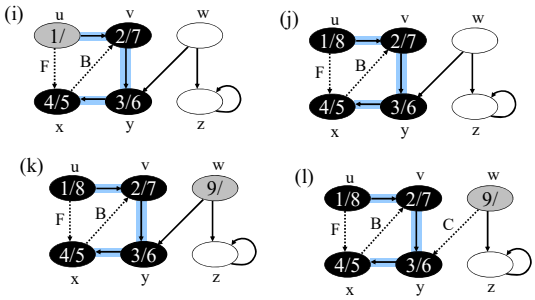
352

DFSの適用例



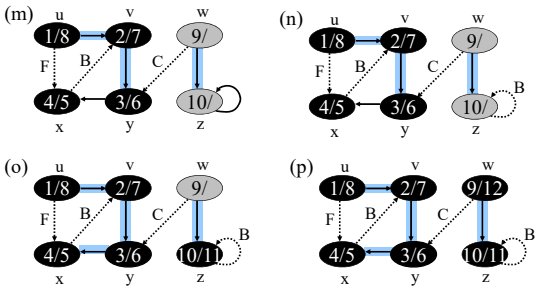
353

DFSの適用例



354

DFSの適用例



355

DFSアルゴリズム

```
DFS(G)
1.  for each vertex u∈ G.V
2.      do u.color = WHITE
3.         u.π = NIL
4.  time = 0
5.  for each vertex u ∈ G.V
6.       if u.color == WHITE
7.                DFS-Visit(G,u)
```

356

DFS-Visit

```
DFS-Visit(G, u)
1.  time = time+1     //u has just been discovered
2.   u.d = time
3.   u.color = GRAY
4.   for each v ∈ G.Adj[u]
5.       if v.color == WHITE
6.            v.π = u
7.            DFS-Visit(G, v)
8.   u.color = BLACK        //DFS-Visit(G, u) is done
9.   time = time + 1
10.  u.f = time
```

357

深さ優先探索の性質

- DFSの実行時間はO(|V|+|E|).
- “木辺”によって深さ優先森を形成．辺の選択順序が決まれば，これはユニークに決まる．

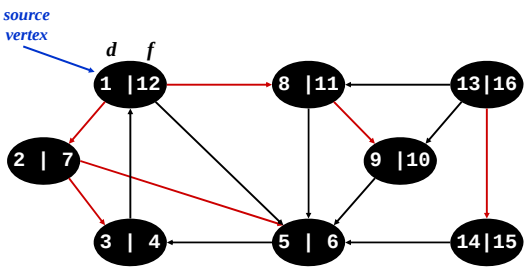
358

DFS: 辺の種類

- DFSによって，辺がいくつかの種類に分類される:
  - 木辺：深さ優先森中の木の辺，新しい（白）頂点に至る辺．
  - 後退辺：木辺以外で，祖先に向かう辺，灰色頂点から灰色頂点に至る辺．
  - 前進辺：木辺以外で，子孫に向かう辺，灰色から黒に至る辺．
  - 横断辺：その他，灰色から黒に至る辺．
- 多くのアルゴリズムでは木辺と後退辺が重要

359

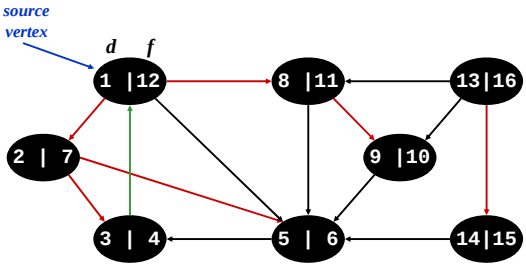
DFSの例



Tree edges

360

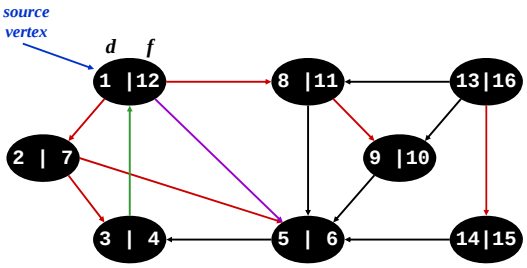
DFSの例



Tree edges *Back edges*

361

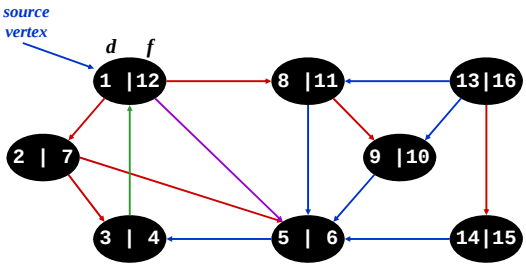
DFSの例



Tree edges *Back edges* *Forward edges*

362

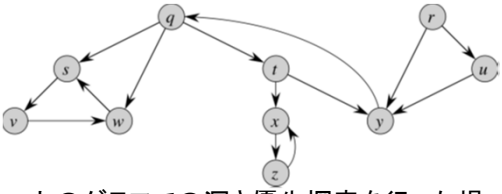
DFSの例



Tree edges *Back edges* *Forward edges* *Cross edges*

363

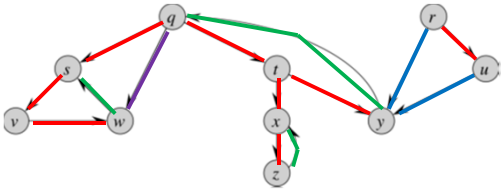
演習：深さ優先探索



- 上のグラフでの深さ優先探索を行った場合，各枝の種類（木辺，後退辺，前進辺，横断辺）を示せ．ただし，頂点の選択順／隣接リストはアルファベット順とする．

364

演習：深さ優先探索

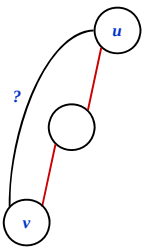


Tree edges *Back edges* *Forward edges* *Cross edges*

365

演習：DFS: 辺の種類

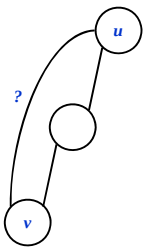
- Gが無向グラフの時，DFSは木辺か後退辺のみを生成することを示せ．



366

演習：DFS: 辺の種類

- Gが無向グラフの時，DFSは木辺か後退辺のみを生成することを示せ (定理22.10).
- 証明:
  - 辺(u, v)を考える
  - 一般性を失うことなしにu.d<v.dを仮定．
  - v.f<u.fのはず．
  - uからvに向けて，この辺を発見したなら，この時点vは白であるはず (vが既発見なら，vからuに向けて，すでにこの辺を発見していないといけない)
  - この場合，辺は木辺
  - vからuに向けて，この辺を発見したなら，この時点でvは灰色，uも灰色，よって辺は後退辺

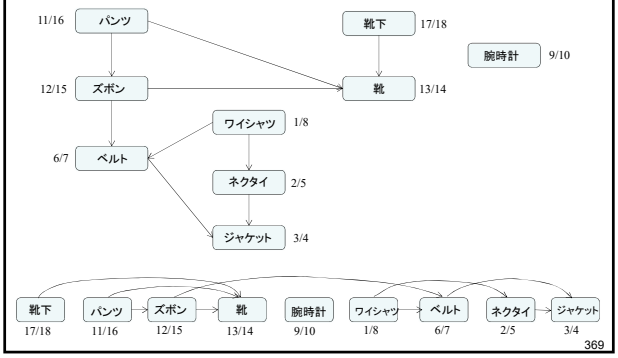


367

トポロジカルソート

- 閉路のない有向グラフ（Directed Acyclic Graph, DAG）G=(V,E)のトポロジカルソートとは，Gが辺(u,v)を含んでいればuがvより先に現れるような，全頂点への線形順序を求めること．
- 深さ優先探索を用いるとDAGのトポロジカルソートは簡単に得られる．
- グラフのトポロジカルソートは，水平な直線上に頂点を並べて，どの有向辺も左から右へ向かうように全頂点を並べることに対応．

トポロジカルソート



トポロジカルソートのアルゴリズム

TOPOLOGICAL-SORT(G)

1 DFS(G)を呼び出して，各頂点vの終了時刻 v.fを計算
2 終了した頂点を，連結リストの先頭に挿入
3 **return** 頂点の連結リスト

深さ優先探索がΘ(V+E)時間でできる．$|V|$個の頂点を連結リストの先頭に挿入するのは各々O(1)時間でできるから，トポロジカルソートはΘ(V+E)時間で実行できる．

演習：トポロジカルソート：正しさの証明

以下を証明せよ：

有向グラフGが閉路を持たないのは，Gを深さ優先探索したときに後退辺を生成しないときであり，かつそのときに限る．

演習：トポロジカルソート：正しさの証明

以下を証明せよ（補題22.11）：

有向グラフGが閉路を持たないのは，Gを深さ優先探索したときに後退辺を生成しないときであり，かつそのときに限る．

(証明)

⇨後退辺(u,v)が生じると仮定すると，深さ優先探索森において，vはuの先祖となるので，Gにvからuへの経路があることになるが，これに後退辺(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．vをcにおいて最初に発見される頂点とし，(u,v)をcにおいてvに入る辺とする．時刻 v.d では，vからuに至る白頂点の経路が存在する．よって，uは深さ優先探索森においてvの子孫になる．したがって，(u,v)は後退辺である．

トポロジカルソート：正しさの証明

定理22.12

TOPOLOGICAL-SORT(G)は，閉路のない有向グラフGをトポロジカルソートする．

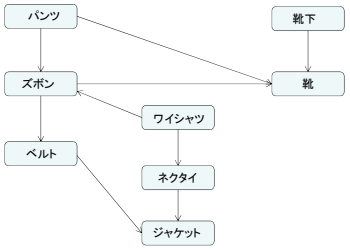
(証明)

与えられた有向グラフGに関してDFSを実行して各頂点の終了時刻を決定する．任意の異なる頂点$u,v \in V$の対に対して，Gの中にuからvへ至る辺があれば，v.f < u.fが成り立つことを示せばよい．

DFS(G)で調べる任意の辺(u,v)に関して，vは灰色ではない．なぜなら，そうするとvはuの先祖となり(u,v)は後退辺となるので，これは補題22.11と矛盾．vが白のとき，uの子孫になるのでv.f < u.f．vが黒のとき同様にv.f < u.f．したがって閉路のない有向グラフの任意の辺に対して v.f < u.f が成り立つので証明が完成．

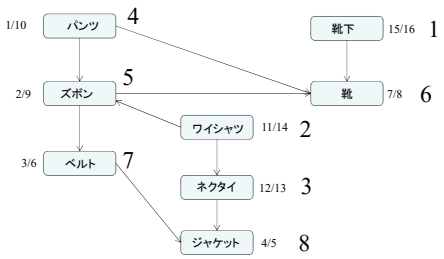
演習:トポロジカルソート

- 以下のグラフのトポロジカルソートを求めよ.



374

演習:トポロジカルソート



375