アルゴリズムとデータ構造
第11回: グラフの表現方法

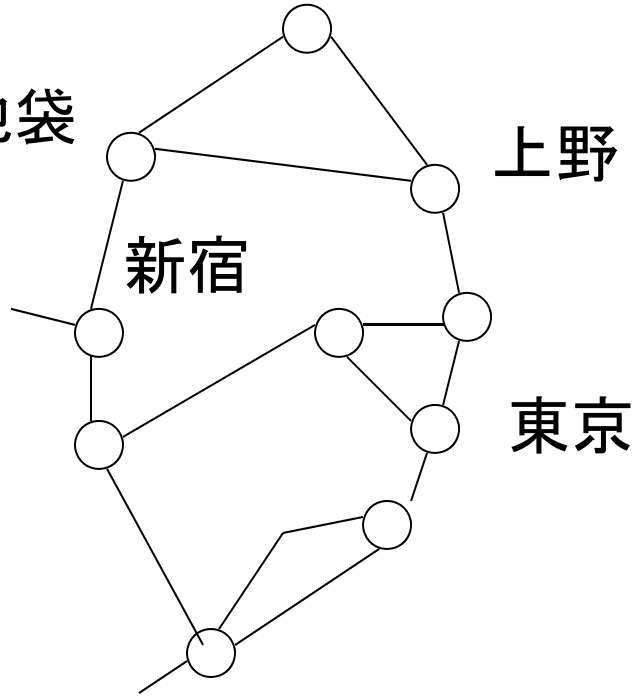
担当: 上原隆平 (uehara)

2014/05/22

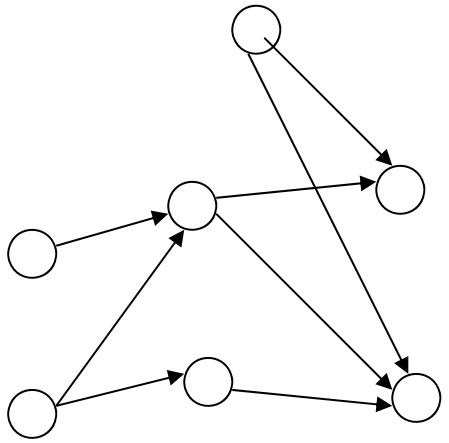
グラフ (graph)

- 頂点を辺で結んだもの
  - 有向グラフ: 辺に方向がある
  - 無向グラフ: 辺に方向がない

例: 路線図

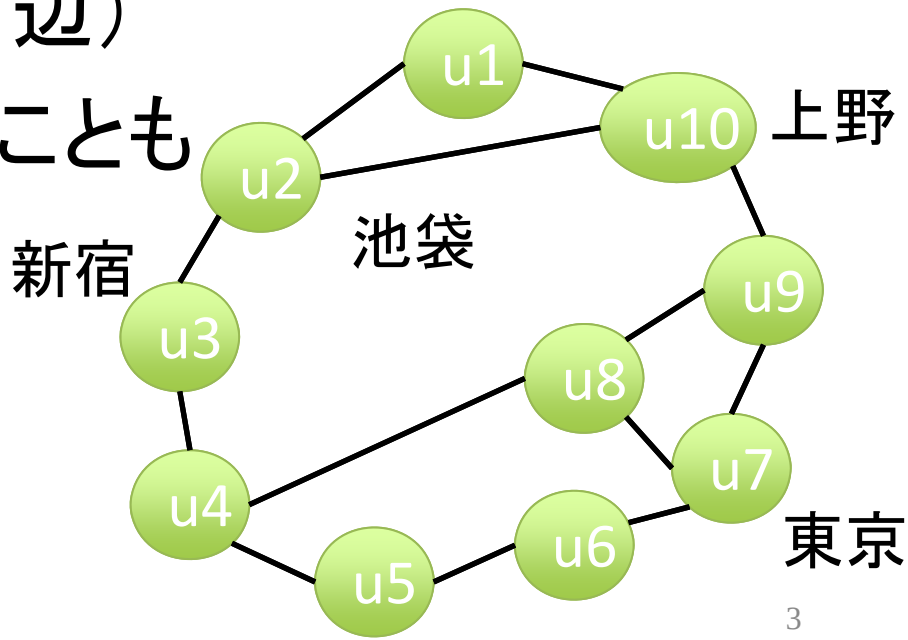


例: 講義の履修順序



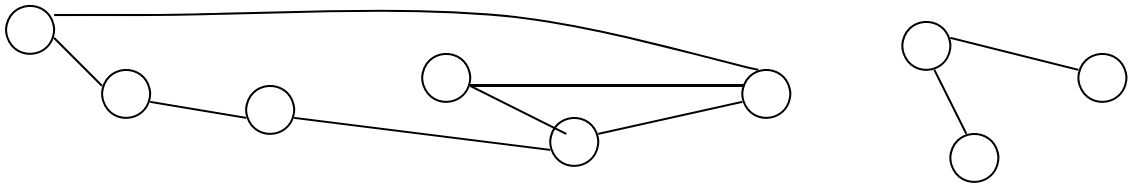
グラフ: 表記

- グラフ: $G = (V, E)$
  - V: 頂点集合，E: 辺集合
- 頂点: $u, v, \ldots \in V$
- 辺: $e = \{u, v\} \in E$（無向辺）
  $a = (u, v) \in E$（有向辺）
- 頂点や辺は重みを持つことも
  - $w(u), w(e)$
  - 距離，金額，時間など

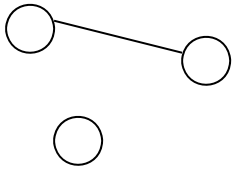


グラフ: 基本的な用語 (1/2)

- 路 (path): 辺で隣り合う頂点を繋いだもの
  - 単純路 (simple path): 同じ頂点を複数回通らない

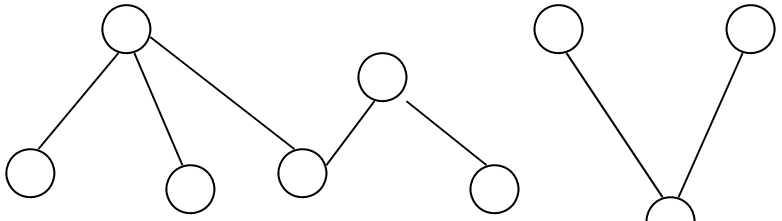


- 閉路 (cycle, closed path): v から v への路
- 連結グラフ (connected graph): どの2頂点間にも路が存在するグラフ

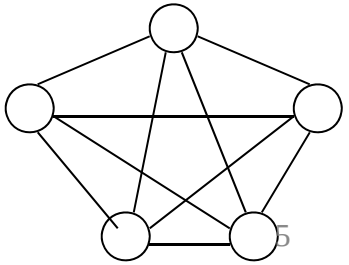


グラフ: 基本的な用語 (2/2)

- 森 (forest): 閉路を含まないグラフ
- 木 (tree): 連結で閉路を含まないグラフ



- 平面的グラフ (planar graph): 辺の交差なしで平面に描画できるグラフ
- 完全グラフ (complete graph): 全ての頂点対を辺で隣接させたグラフ
  - 完全グラフ K_5 は極小非平面的グラフ



グラフアルゴリズムの計算量

- 頂点数 n, 辺数 $m \in O(n^2)$
  - 無向グラフ: $m \leqq n(n\text{-}1)/2$
  - 有向グラフ: $m \leqq n(n\text{-}1)$

- 平面グラフでは $m \in O(n)$

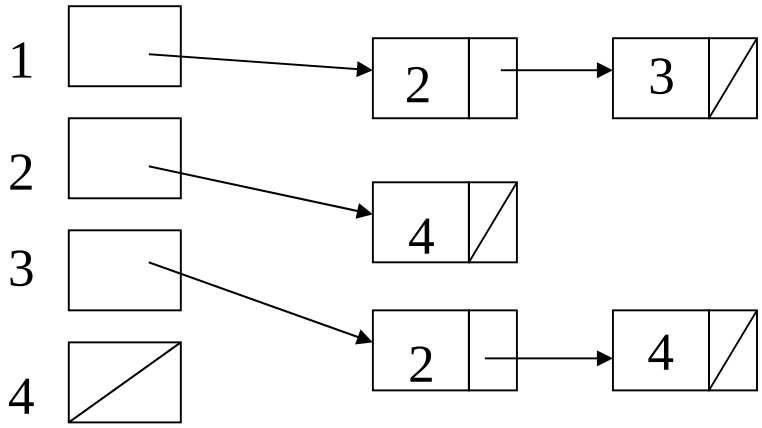
- グラフアルゴリズムの計算量は
  n や m の式で表す

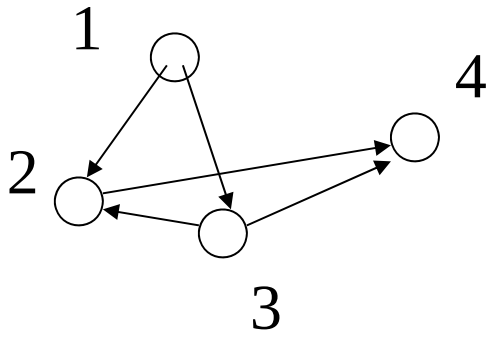
グラフの表現方法

- 隣接行列

$$M = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

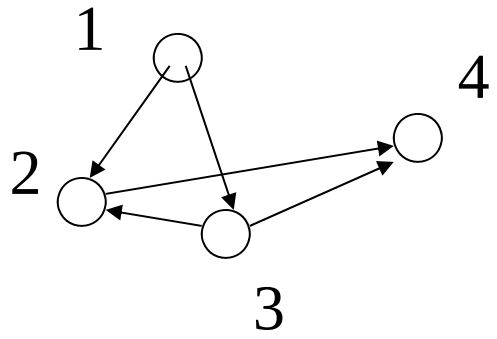
- 隣接リスト





グラフの表現方法:
行列表現(隣接行列)

- $(u, v) \in E \Rightarrow M[u, v] = 1$
- $(u, v) \notin E \Rightarrow M[u, v] = 0$

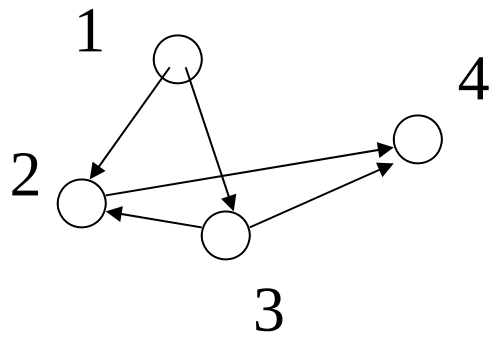


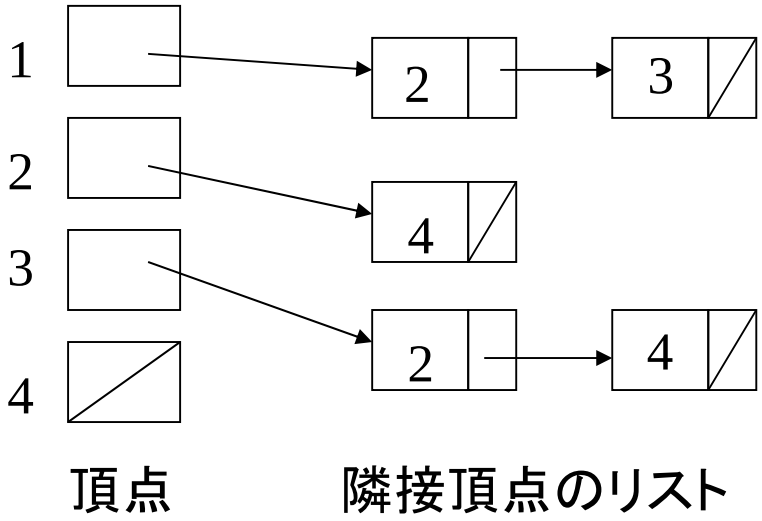
$$M = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

グラフの表現方法: リスト表現(隣接リスト)

- $(u, v) \in E \Leftrightarrow v \in T(u)$
  - $T(u)$ は u の隣接点のリスト

- 例:





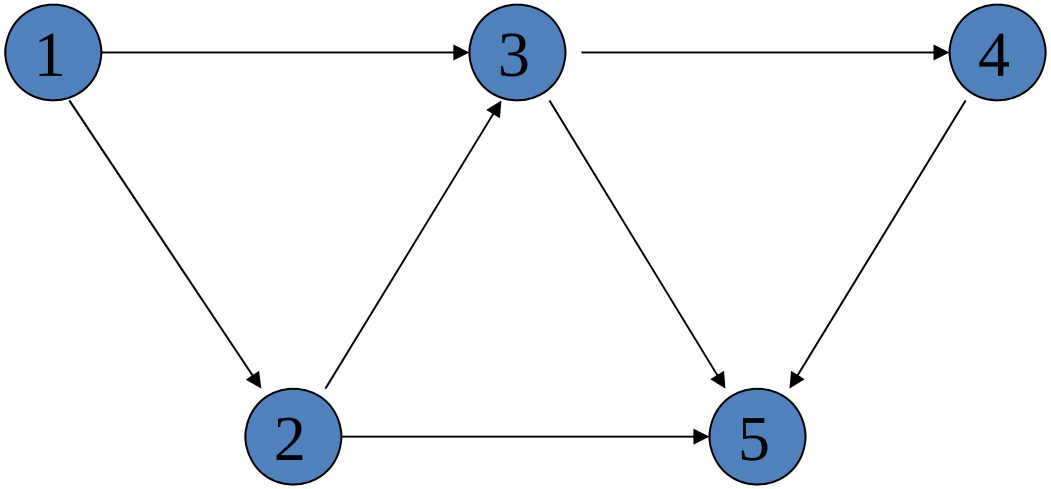
隣接行列 vs 隣接リスト

- メモリ量
  - 隣接行列: $\Theta(n^2)$
  - 隣接リスト: $\Theta(m \log n)$
- 隣接チェックにかかる時間: $(u, v) \in E$?
  - 隣接行列: $\Theta(1)$
  - 隣接リスト: $\Theta(n)$

Q. グラフの更新 (e.g., 頂点・辺の追加・削除) は？

トポロジカルソート

- $G = (V, E)$ を無閉路有向グラフとする
- 全ての有効辺 $(u, v) \in E$ について $n(u) < n(v)$ となる順序付け $n: V \to \mathbf{N}$ を作る問題



Q. 何故無閉路でないといけない?

トポロジカルソート: アルゴリズム

```
topological_sort(V,E,n[]){
  IN=/*長さ|V|のint配列*/;
  foreach v ∈ V do
    IN[v]=|{(u,v) in E}|;
  Q={v|IN[v]=0};
  count=1;
  while(|Q|>0){
    v=pop(Q);
    n[v]=count++;
    foreach (v,u) ∈ E do
      if(--IN[u]==0) push(Q,u);
  }
}
```

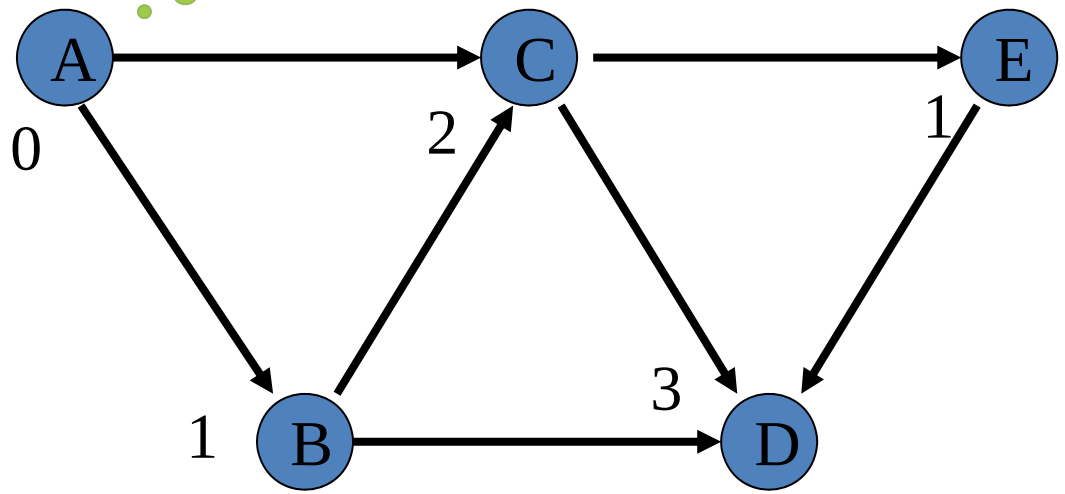
各頂点vの入次数
(vを終点とする辺の数)

入次数が0となる頂点
を集めたキュー

トポロ

キュー

A	B	C	D	E

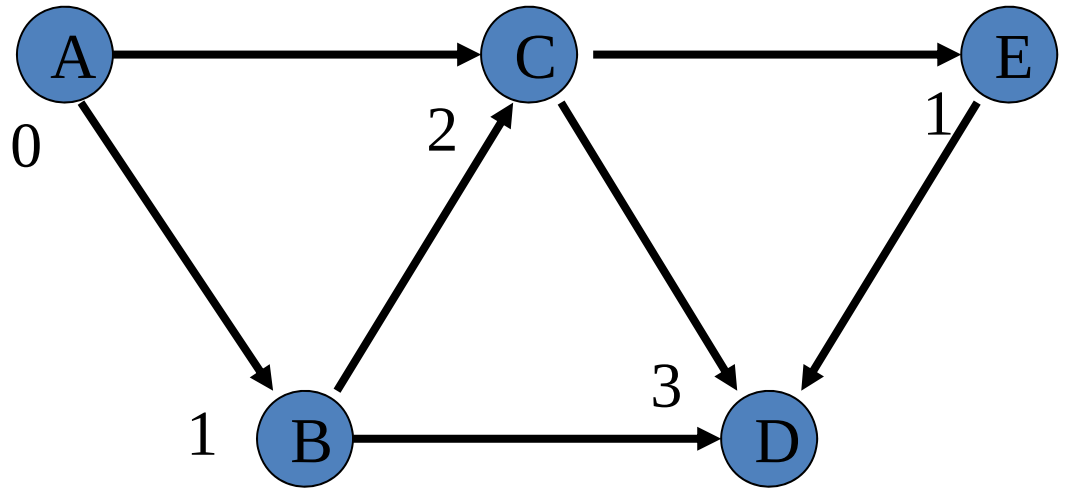


Q={A}

```
foreach v ∈ V do
  IN[v]=|{(u,v) in E}|;
Q={v|IN[v]=0};
```

トポロジカルソート

A	B	C	D	E
1				



Q={A}

Q={B}

```
count=1;
while(|Q|>0){
  v=pop(Q);
  n[v]=count++;
  foreach (v,u) ∈ E do
    if(--IN[u]==0) push(Q,u);
}
```

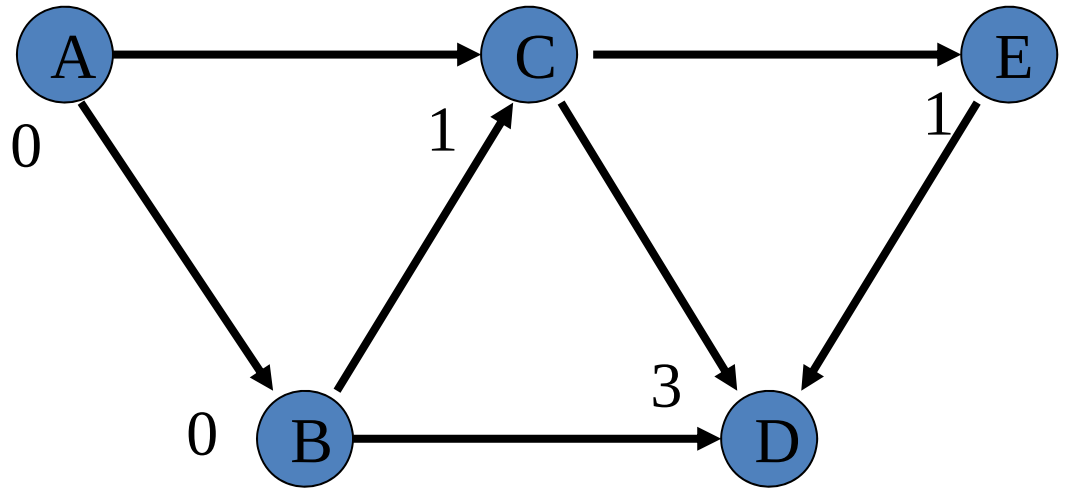
キューへ

(A,B): IN[B]→0

(A,C): IN[C]→1

トポロジカルソート

A	B	C	D	E
1	2			



Q={B}

Q={C}

```
count=1;
while(|Q|>0){
  v=pop(Q);
  n[v]=count++;
  foreach (v,u)  E do
    if(--IN[u]==0) push(Q,u);
}
```

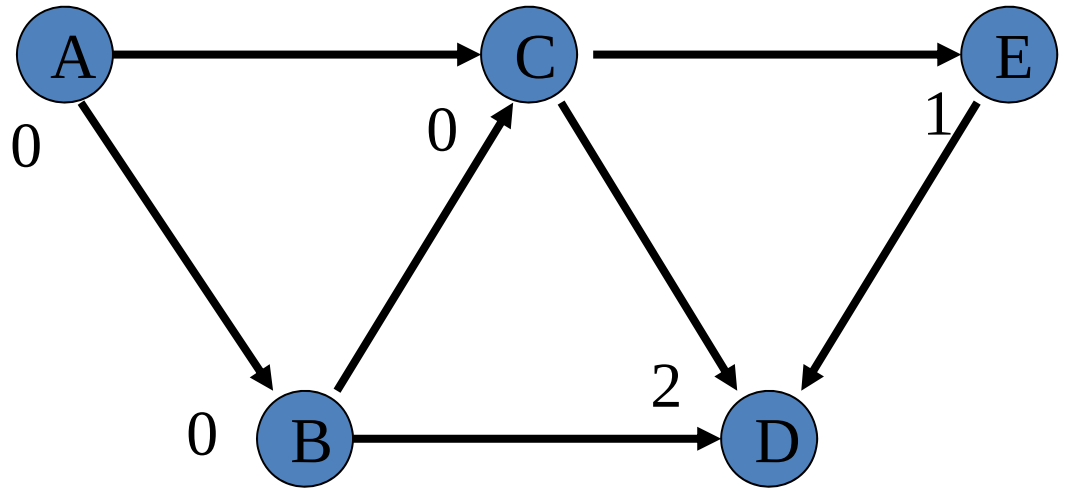
キューへ

(B,C): IN[C]→0

(B,D): IN[D]→2

トポロジカルソート

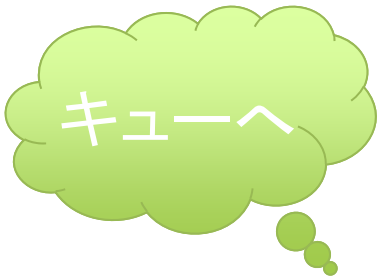
A	B	C	D	E
1	2	3		



Q={C}

Q={E}

```
count=1;
while(|Q|>0){
  v=pop(Q);
  n[v]=count++;
  foreach (v,u)  E do
    if(--IN[u]==0) push(Q,u);
}
```

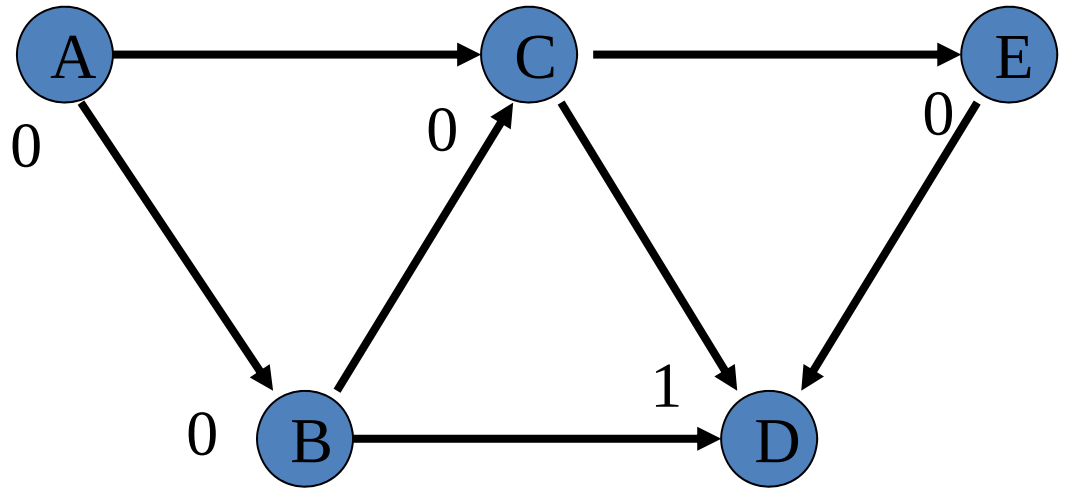


(C,E): IN[E]→0

(C,D): IN[D]→1

トポロジカルソート

A	B	C	D	E
1	2	3		4



Q={E}

Q={D}

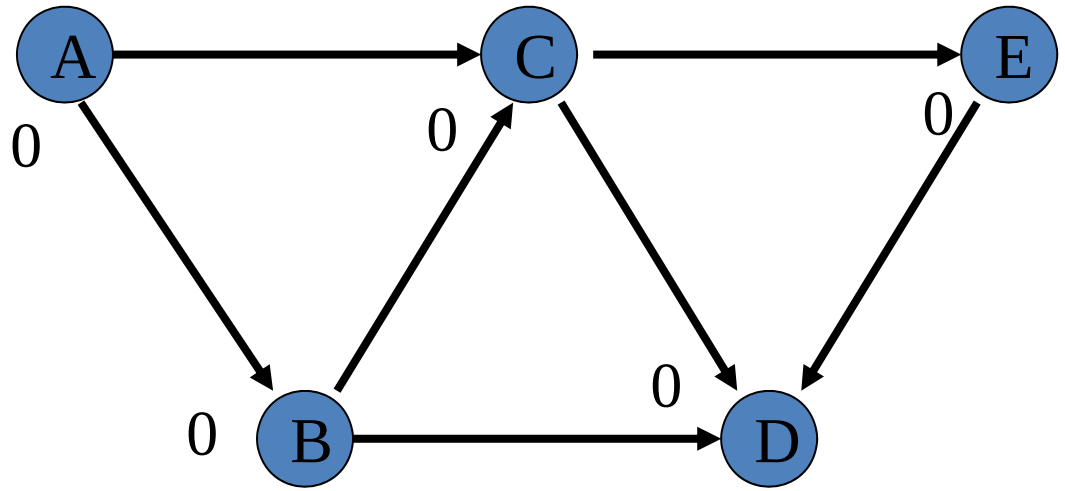
```
count=1;
while(|Q|>0){
  v=pop(Q);
  n[v]=count++;
  foreach (v,u)  E do
    if(--IN[u]==0) push(Q,u);
}
```

キューへ

(E,D): IN[E]→0

トポロジカルソート

A	B	C	D	E
1	2	3	5	4



Q={D}

Q={}

```
count=1;
while(|Q|>0){
  v=pop(Q);
  n[v]=count++;
  foreach (v,u)  E do
    if(--IN[u]==0) push(Q,u);
}
```

トポロジカルソート: 計算量
隣接リストを用いた場合

```
topological_sort(V,E,n[]){
  IN=/*長さ|V|のint配列*/;
  foreach v ∈ V do
    IN[v]=|{(u,v) in E}|;
  Q={v|IN[v]=0};
  count=1;
  while(|Q|>0){
    v=pop(Q);
    n[v]=count++;
    foreach (v,u) ∈ E do
      if(--IN[u]==0) push(Q,u);
  }
}
```

$O(|V|)$

$O(|E_v|)$

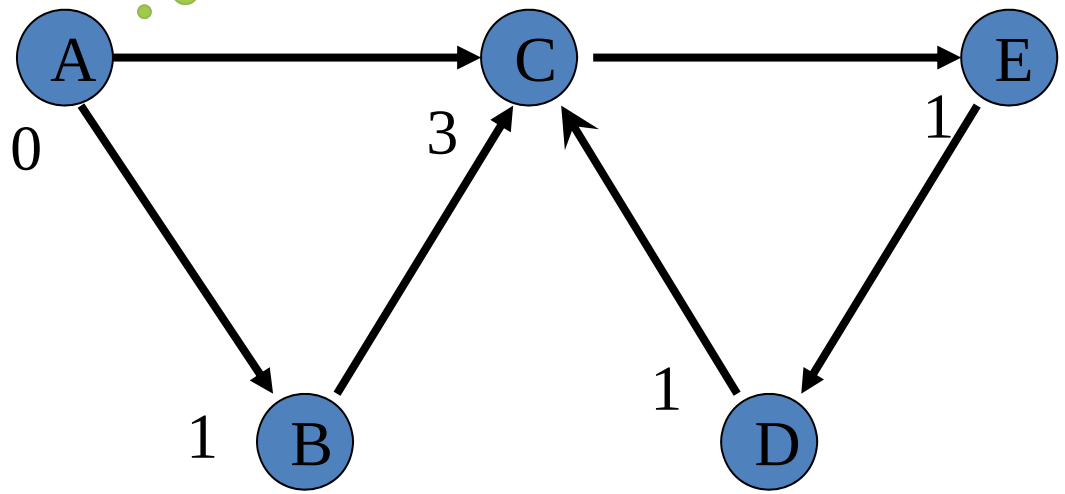
$O(\Sigma_v|E_v|) = O(|E|)$

∴計算量は$O(|V| + |E|)$

トポロジカルソート
閉路がない有向グラフ

キューへ

A	B	C	D	E

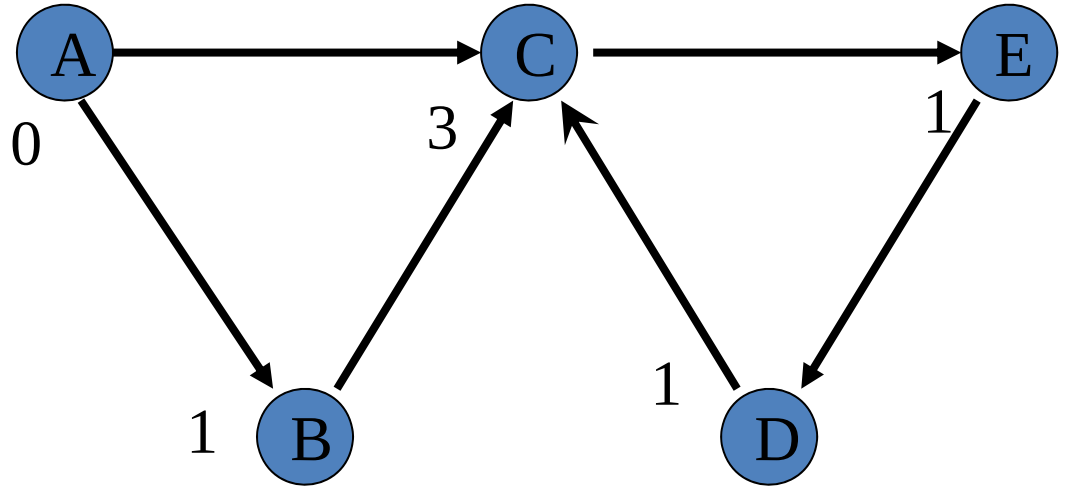


Q={A}

```
foreach v ∈ V do
  IN[v]=|{(u,v) in E}|;
Q={v|IN[v]=0};
```

トポロジカルソート: 閉路がある場合

A	B	C	D	E
1				



Q={A}

Q={B}

```
count=1;
while(|Q|>0){
  v=pop(Q);
  n[v]=count++;
  foreach (v,u) ∈ E do
    if(--IN[u]==0) push(Q,u);
}
```

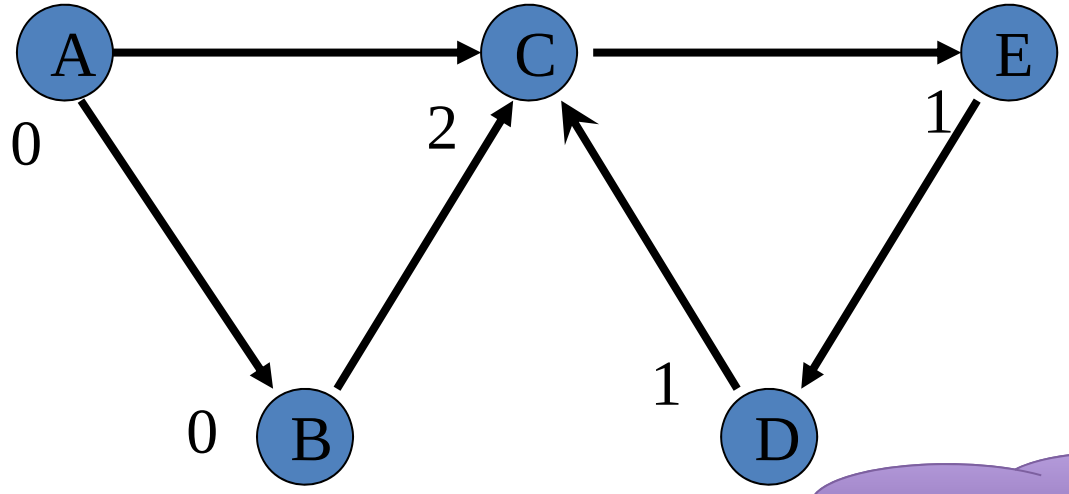
キューへ

(A,B): IN[B]→0

(A,C): IN[C]→2

トポロジカルソート: 閉路がある場合

A	B	C	D	E
1	2			



Q={B}

Q={}

終了してしまう!

```
count=1;
while(|Q|>0){
  v=pop(Q);
  n[v]=count++;
  foreach (v,u)  E do
    if(--IN[u]==0) push(Q,u);
}
```

(B,C): IN[C]→1

ミニ演習

- 頂点数4の無向グラフを全て描け
  - 頂点にラベルはないものとする
  - 描き方を変えれば同じになるものは省く
    - cf. グラフの同型性

- 頂点数 n の無向グラフは高々 2^{n^2} 個しかないことを証明せよ