グラフ

離散数学・オートマトン
2021 年後期

佐賀大学理工学部　只木進一

グラフとは

- 日常用語ではネットワーク (networks) ともいう
  - インターネット
  - ヒトの繋がり
  - 交通網
  - 作業手順
- 要素の繋がり方に注目

グラフの定義

- グラフ $G = (V, E)$
- 頂点 (node) の集合 V
- 辺 (edge) の集合 E
    - 頂点 u と v を結ぶ辺 $e = (u, v)$
    - 頂点 u と v を辺 e の端点という

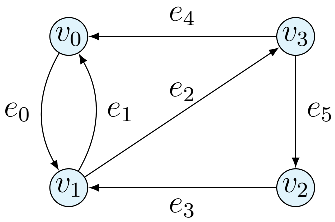
有向グラフと無向グラフ

- 無向グラフ: non-directed graphs
  - 辺に向きの無いグラフ
  - $\partial : E \to V \times V$: 辺から端点への写像
- 有向グラフ: directed graphs
  - 辺に向きの有るグラフ
  - $\partial^+ : E \to V$:始点
  - $\partial^- : E \to V$:終点

例 1

$$V = \{v_0, v_1, v_2, v_3\}$$
$$E = \{e_0, e_1, e_2, e_3, e_4, e_5\}$$

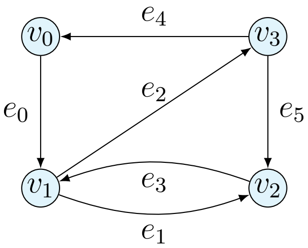
$$\partial^+ e_0 = v_0 \quad \partial^- e_0 = v_1 \quad \partial^+ e_1 = v_1 \quad \partial^- e_1 = v_0$$
$$\partial^+ e_2 = v_1 \quad \partial^- e_2 = v_3 \quad \partial^+ e_3 = v_2 \quad \partial^- e_3 = v_1$$
$$\partial^+ e_4 = v_3 \quad \partial^- e_4 = v_0 \quad \partial^+ e_5 = v_3 \quad \partial^- e_5 = v_2$$



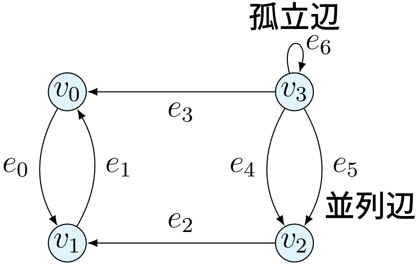
例 2

$$V = \{v_0, v_1, v_2, v_3\}$$
$$E = \{e_0, e_1, e_2, e_3, e_4, e_5\}$$

$$\partial^+ e_0 = v_0 \qquad \partial^- e_0 = v_1 \qquad \partial^+ e_1 = v_1 \qquad \partial^- e_1 = v_2$$
$$\partial^+ e_2 = v_1 \qquad \partial^- e_2 = v_3 \qquad \partial^+ e_3 = v_2 \qquad \partial^- e_3 = v_1$$
$$\partial^+ e_4 = v_3 \qquad \partial^- e_4 = v_0 \qquad \partial^+ e_5 = v_3 \qquad \partial^- e_5 = v_2$$



並列辺と孤立辺



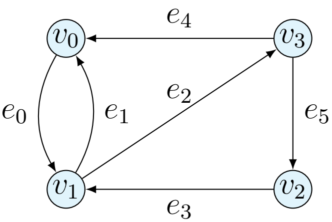
グラフの定義２

- 無向グラフに対して
  - $\delta : V \to 2^E$: 頂点から辺の集合
- 有向グラフに対して
  - $\delta^+ : V \to 2^E$: 頂点を始点とする辺の集合
  - $\delta^- : V \to 2^E$: 頂点を終点とする辺の集合

例 3

$$V = \{v_0, v_1, v_2, v_3\}$$
$$E = \{e_0, e_1, e_2, e_3, e_4, e_5\}$$

$$\delta^+ v_0 = \{e_0\} \quad \delta^- v_0 = \{e_1, e_4\} \quad \delta^+ v_1 = \{e_1, e_2\} \quad \delta^- v_1 = \{e_0, e_3\}$$
$$\delta^+ v_2 = \{e_3\} \quad \delta^- v_2 = \{e_5\} \quad \delta^+ v_3 = \{e_4, e_5\} \quad \delta^- v_3 = \{e_2\}$$

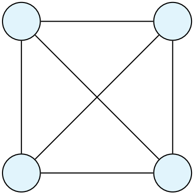


次数: degree

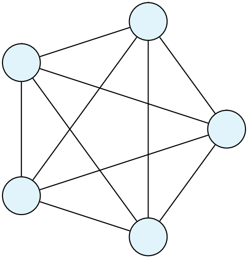
- 無向グラフに対して
  - 頂点 v を始点とする辺: $\delta : V \to 2^E$
  - 頂点 v の次数: $|\delta v|$
- 有向グラフに対して
  - $\delta^+ : V \to 2^E$: 頂点を始点とする辺の集合
  - $\delta^- : V \to 2^E$: 頂点を終点とする辺の集合
  - 頂点 v の正次数: $|\delta^+ v|$
  - 頂点 v の負次数: $|\delta^- v|$

完全グラフ: Complete Graphs

すべての頂点の組を結ぶ辺が存在



K_4

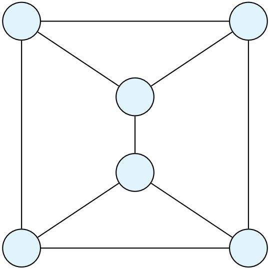


K_5

komplete: ドイツ語

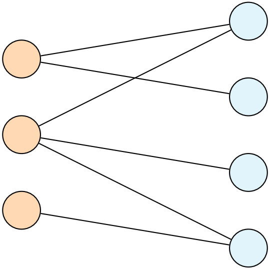
正則グラフ: Regular Graphs

すべての頂点の次数が等しい



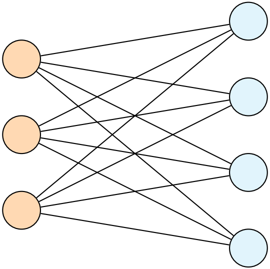
二部グラフ: Bipartites

頂点が 2 つの集合に別れ、集合内の辺が無い



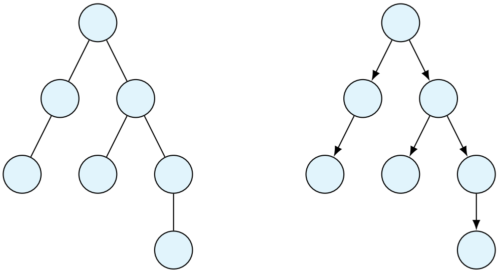
完全二部グラフ

左の各点が右の各点と結ばれている



木: Tree

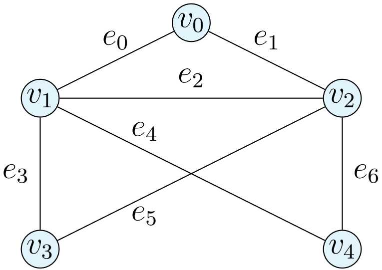
閉路の無いグラフ。有向と無向がある。



Euler 閉路: 一筆書き

- 「Königsberg の橋」: Graph 理論の端緒
  - https://www.britannica.com/science/Konigsberg-bridge-problem
  - Leonhard Euler (1707-1783)
- 無向グラフに対して、全ての辺を一度ずつ通り、元の頂点に戻る道を見つける
- 全ての頂点の次数が偶数の場合のみ、Euler 閉路が存在する

例 1



閉路の例

$$e_0 \to e_3 \to e_6 \to e_5 \to e_4 \to e_2 \to e_1$$

Algorithm 1 Euler 閉路列挙のアルゴリズム

```
//                                   ▷ E_Euler: 経由した辺のリスト、初期値 E_Euler = ∅
//                                   ▷ r: 始点
procedure ENUMERATEEULER(v, E_Euler)
    if (v == r) ∧ (|E_Euler| == |E|) then
        見つけた Euler 閉路 E_Euler を保存
    else
        for all e ∈ δv do                     ▷ //v に接続する全ての辺
            if e ∉ E_Euler then
                E'_Euler = E_Euler ∪ {e}
                w = ∂e \ {v}                  ▷ //辺 e の v と反対側の頂点
                enumerateEuler(w, E'_Euler)
            end if
        end for
    end if
end procedure
```

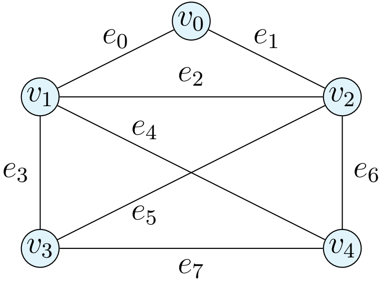
列挙の結果

```
['e0', 'e2', 'e5', 'e3', 'e4', 'e6', 'e1']
['e0', 'e2', 'e6', 'e4', 'e3', 'e5', 'e1']
['e0', 'e3', 'e5', 'e2', 'e4', 'e6', 'e1']
['e0', 'e3', 'e5', 'e6', 'e4', 'e2', 'e1']
['e0', 'e4', 'e6', 'e2', 'e3', 'e5', 'e1']
['e0', 'e4', 'e6', 'e5', 'e3', 'e2', 'e1']
['e1', 'e2', 'e3', 'e5', 'e6', 'e4', 'e0']
['e1', 'e2', 'e4', 'e6', 'e5', 'e3', 'e0']
['e1', 'e5', 'e3', 'e2', 'e6', 'e4', 'e0']
['e1', 'e5', 'e3', 'e4', 'e6', 'e2', 'e0']
['e1', 'e6', 'e4', 'e2', 'e5', 'e3', 'e0']
['e1', 'e6', 'e4', 'e3', 'e5', 'e2', 'e0']
```

Hamilton 閉路

- 無向グラフに対して、全ての頂点を一度ずつ経由して、始点に戻る閉路
- 巡回セールスマン問題等で必要となる

例 2



閉路の例

$$v_0 \to v_1 \to v_3 \to v_4 \to v_2 \to v_0$$

Algorithm 2 Hamilton 閉路列挙のアルゴリズム

```
//    ▷ V_Hamilton: 経由した頂点のリスト、初期値は V_Hamilton = {r}
//    ▷ r: 始点
procedure ENUMERATEHAMILTON(v, V_Hamilton)
    for all e ∈ δv do                          ▷ //v に接続する全ての辺
        w = ∂e \ {v}                            ▷ 辺 e の v と反対側の頂点
        if (w == r) ∧ (|V_Hamilton| == |V|) then
            見つけた Hamilton 閉路 V_Hamilton を保存
        else
            if w ∉ V_Hamilton then
                V'_Hamilton = V_Hamilton ∪ {w}
                enumerateHamilton(w, V'_Hamilton)
            end if
        end if
    end for
end procedure
```

列挙の結果

```
['v0', 'v1', 'v3', 'v4', 'v2']
['v0', 'v1', 'v4', 'v3', 'v2']
['v0', 'v2', 'v3', 'v4', 'v1']
['v0', 'v2', 'v4', 'v3', 'v1']
```