

明日から graphlib、みんなで使おう

Hayao Suzuki

PyCon JP 2025 at International Conference Center Hiroshima

September 26, 2025

Share it

GitHub

- <https://github.com/HayaoSuzuki/pyconjp2025>

Hashtag

- #pyconjp #PyConJP2025

Who am I ?

お前誰よ

Name Hayao Suzuki (鈴木 駿)

~~Twitter~~ X @CardinalXaro

Work ソフトウェアエンジニア at 東京ガス株式会社

東京ガス株式会社について

- 一都六県に都市ガス・電気などのエネルギーを供給する会社
- 東京ガスは PyCon JP 2025 の Gold スポンサーです
- ソフトウェアエンジニアを絶賛募集中
<https://www.tokyo-gas-recruit.com/career/>

Who am I ?

翻訳

- Effective Python 第3版 (O'Reilly Japan) New!
- ハイパーモダン Python (O'Reilly Japan)
- Python Distilled (O'Reilly Japan)

監訳・監修

- ロバスト Python (O'Reilly Japan)
- 入門 Python 3 第2版 (O'Reilly Japan)
- Python クイックリファレンス 第4版 (O'Reilly Japan)

Who am I ?

過去の発表（抜粋）

- Let's implement useless Python objects(PyCon APAC 2023)
- 組み込み関数 pow の知られざる進化 (PyCon JP 2021)
- インメモリーストリーム活用術 (PyCon JP 2020)
- 君は cmath を知っているか (PyCon mini Shizuoka 2020)
- Python と楽しむ初等整数論 (PyCon mini Hiroshima 2019)
- SymPy による数式処理 (PyCon JP 2018)

一覧は <https://xaro.hatenablog.jp/> を参照してください

明日から `graphlib`、みんなで使おう

忙しい人向けの要約

- トポロジカルソートとは、有向非巡回グラフの頂点集合の線型順序である
- graphlib は、トポロジカルソートが実装された標準ライブラリである
- graphlib は、簡易的なタスクランナーとして使える

グラフって、何？

定義（無向グラフ）

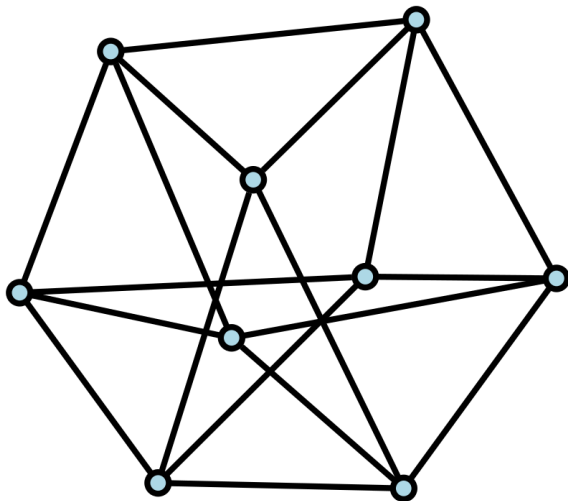
有限集合 V および $V \times V$ の非順序対からなる集合の部分集合 E の組 $G = (V, E)$ を無向グラフと呼ぶ。

定義（有向グラフ）

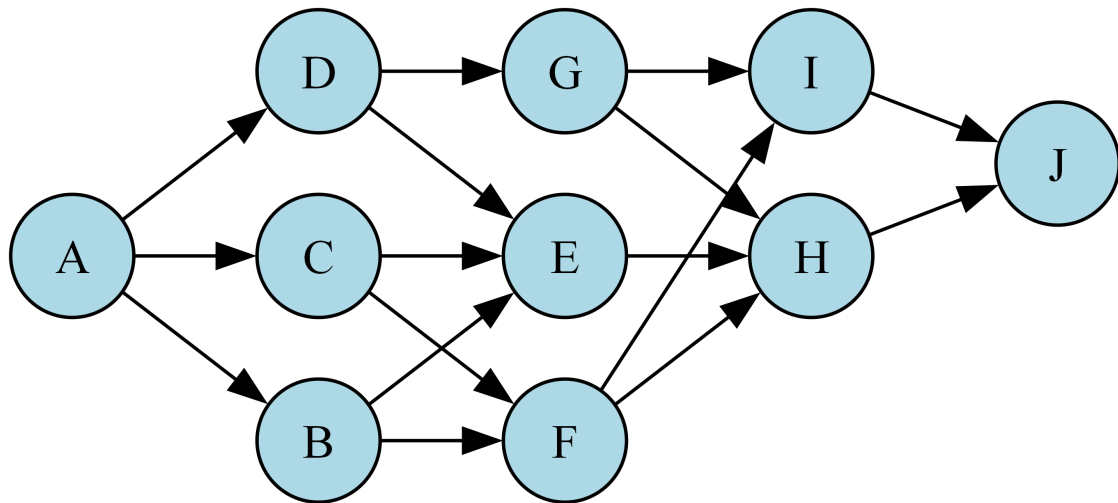
有限集合 V および $V \times V$ の順序対からなる集合の部分集合 E の組 $G = (V, E)$ を有向グラフと呼ぶ。

- V を頂点集合、 E を辺集合と呼ぶ。
- V の要素を G の頂点、 E の要素を G の辺と呼ぶ。

無向グラフの例



有向グラフの例



つまり…どういうことだってばよ？

…で、そのグラフは当社で働くうえで
何のメリットがあるとお考えですか？

来いよベネット！ 定義なんか捨ててかかって来い！

忙しい人向けのグラフ

- グラフとは、「もの」とその「関係」を数学的にモデル化したものである
- 「もの」は人間、駅、タスク、サーバなど
- 「関係」は人間関係、線路、依存関係、ネットワークなど

トポロジカルソートって、何？

定義（二項関係）

集合 A の直積 $A \times A$ の部分集合 R を二項関係と呼ぶ。
また、 $(a, b) \in R$ を aRb と表す。

定義（半順序関係）

以下の性質を満たす関係 R を半順序関係と呼ぶ。

反射律 $\forall a \in A$ に対して、 aRa である

反対称律 $a, b \in A$ に対して、 aRb かつ bRa ならば $a = b$ である

推移律 $a, b, c \in A$ に対して、 aRb かつ bRc ならば aRc である

半順序関係は、大小関係や比較の概念を抽象したもの。

トポロジカルソートって、何？

定義（線型順序）

集合 A の半順序関係 R において、 $\forall a, b \in A$ に対して aRb または bRa が成り立つならば、 R は線型順序であると呼ぶ。

定義（トポロジカルソート）

有向グラフ $G = (V, E)$ のトポロジカルソートとは、 V の線型順序 (V, \leq) で、 $(u, v) \in E$ ならば $u \leq v$ を満たすものである。

トポロジカルソートって、何？

定義（歩道）

グラフ $G = (V, E)$ の頂点 u_1 から u_k への長さ k の歩道 W とは、頂点の列 (u_1, u_2, \dots, u_k) で、 $(u_i, u_j) \in E (1 \leq i < j \leq k)$ を満たすものである。特に、 $u_1 = u_k$ の場合、 W 閉じていると呼ぶ。

定義（道）

グラフ $G = (V, E)$ の歩道 W において、頂点および辺がすべて異なるものは道と呼ぶ。特に、閉じた道を閉路と言う。

定義（有向非巡回グラフ）

有向グラフ $G = (V, E)$ において、閉路を含まないものを有向非巡回グラフと呼ぶ。

トポロジカルソートって、何？

命題（トポロジカルソート可能）

有向グラフ G がトポロジカルソート可能であるための必要十分条件は、有向グラフ G が有向非巡回グラフであることである。

トポロジカルソート可能 \Rightarrow 有向非巡回グラフ.

トポロジカルソート可能だがグラフ G に閉路が存在すると仮定する。
その閉路を $v_i, v_j, v_k (i < j < k)$ として、かつ $v_i \leq v_j \leq v_k$ とする。
 $v_i, v_j, v_k (i < j < k)$ は閉路なので、 v_k から v_i への辺が存在する。
つまり、 $v_k \leq v_i$ となるが、それは仮定 $v_i \leq v_j \leq v_k$ に矛盾する。



補題

有向非巡回グラフには入次数 0 の頂点が必ず存在する。

証明.

有向非巡回グラフの最長の道 v_1, \dots, v_k を 1 つ選ぶ。 v_1 に入る辺が存在するならば、 u, v_1, \dots, v_k かつ $(u, v_1) \in E$ のような頂点 u が存在することになるが、 v_1, \dots, v_k が最長の道であるという仮定に反する。よって、 v_1 の入次数は 0 となり、有向非巡回グラフには入次数 0 の頂点が必ず存在する。 □

え？ まだあるんですか？ 楽しい証明コーナー

補題

有向非巡回グラフ G から入次数 0 の頂点およびそこから出る辺を取り除いた部分グラフ H もまた有向非巡回グラフである。

証明.

部分グラフ H が有向非巡回グラフではないと仮定する。 H の閉路を C とした場合、 H の構成方法から C には取り除いた入次数 0 の頂点およびそこから出る辺は含まれない。つまり、 H は G の部分グラフなので、閉路 C は G にも存在することになる。しかし、それは G が有向非巡回グラフであるという仮定に反する。□

有向非巡回グラフ \Rightarrow トポロジカルソート可能.

有向非巡回グラフ G には必ず入次数 0 の頂点が必ず存在するので、その頂点およびそこから出る辺を取り除く。取り除いたグラフもまた有向非巡回グラフなので、頂点が無くなるまでそれを繰り返す。取り除いた順番に頂点を並べたものがトポロジカルソートである。□

重要：証明がアルゴリズムになっていることに注意。

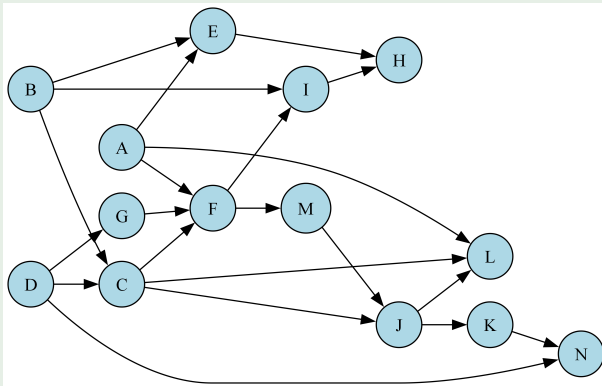
離散数学の講義ではなく、
PyCon JP ですよ

忙しい人向けのトポロジカルソート

- 半順序関係は、大小関係や包含関係を抽象化したもの
- 線型順序は、どれでも比較できるやつ、ぐらゐの理解で
- トポロジカルソートは、有向グラフの頂点をいい感じに順序付けしたもの
- トポロジカルソートと有向非巡回グラフは表裏一体
- トポロジカルソートに関する証明がアルゴリズムになっている

トポロジカルソートの使い道とは

Q: 頂点をタスク、辺をタスクの依存関係とする。
どの順番でタスクを実行すればよいか？



トポロジカルソートの使い道とは

Ans: グラフをトポロジカルソートして、その順番にやればよい。

$A \rightarrow B \rightarrow D \rightarrow E \rightarrow C \rightarrow G \rightarrow F \rightarrow I \rightarrow M \rightarrow H \rightarrow J \rightarrow K \rightarrow L \rightarrow N$

どうやったの？

すでに graphlib、僕は使ったよ

graphlib: グラフ構造を操作する機能

- 現在は TopologicalSorter のみ、単純な標準ライブラリ
- Python 3.9 で追加
- <https://github.com/python/cpython/issues/61207> で議論されていた。

graphlib の使い方

トポロジカルソートの実行

```
from graphlib import TopologicalSorter

graph = {
    "D": {"B", "C"},
    "C": {"A"},
    "B": {"A"},
}

order = TopologicalSorter(graph).static_order()
print(f"Topological order: {" → ".join(order)}")
# A → C → B → D
```

$D \leftarrow B$, $D \leftarrow C$ のようなイメージで定義する。

graphlib の使い方

有向非巡回グラフではないケース

```
from graphlib import TopologicalSorter
```

```
graph = {  
    "A": {"C"},  
    "B": {"A"},  
    "C": {"B"}  
}
```

```
order = TopologicalSorter(graph).static_order()  
print(f"Topological order: {" → ".join(order)}")  
# graphlib.CycleErrorが発生する
```

graphlib：依存関係のあるタスクを実行するタスクランナー

- 並列実行しやすいアルゴリズム
- ドキュメントにそれとなく触れられているが、不十分...
- 生成 AI の力を借りてそれとなく作ってみよう！

タスクの定義

dataclass によるタスク定義

```
@dataclass(frozen=True, slots=True)
class Task:
    name: str
    action: Action
    deps: set[str] = field(default_factory=set)
```

頂点はハッシュ可能である必要があるので、`frozen=True` を使う。

グラフの定義

```
def run(tasks, max_workers=None):  
    by_name = {t.name: t for t in tasks}  
    ts = TopologicalSorter({t.name: set(t.deps) for t in tasks})  
    ts.prepare()  
    results: dict[str, object] = {}  
    ...
```

並行処理：スレッドに投入

```
def run(tasks, max_workers=None):
    ...
    with ThreadPoolExecutor(max_workers) as pool:
        while ts.is_active():
            ready = ts.get_ready()
            futs = {}
            for name in ready:
                print(f"run {name}")
                futs[pool.submit(by_name[name].action)]
            ...
```


タスクランナー

並行処理：完了待ち

```
def run(tasks, max_workers=None):  
    ...  
    with ThreadPoolExecutor(max_workers) as pool:  
        while ts.is_active():  
            ...  
            for fut in as_completed(futs):  
                name = futs[fut]  
                results[name] = fut.result()  
                print(f"done {name}")  
                ts.done(name)  
    return results
```

まとめ

- トポロジカルソートとは、有向非巡回グラフの頂点集合の線型順序である。
- 有向グラフ G がトポロジカルソート可能であるための必要十分条件は、有向グラフ G が有向非巡回グラフであることである。
- graphlib は、トポロジカルソートが実装された標準ライブラリである。
- graphlib は、簡易的なタスクランナーを実装する際に、実行順序を決定する仕組みとして使える。