

ХОЧУ ПОМОЧЬ
ПРАКТИКУ



practicum.yandex.ru

РЕКЛАМА · 18+ Я

Бесплатное занятие английским в Яндекс Практикуме

Тест на уровень языка • Разбор грамматики • Разговорная практика • За 30 минут

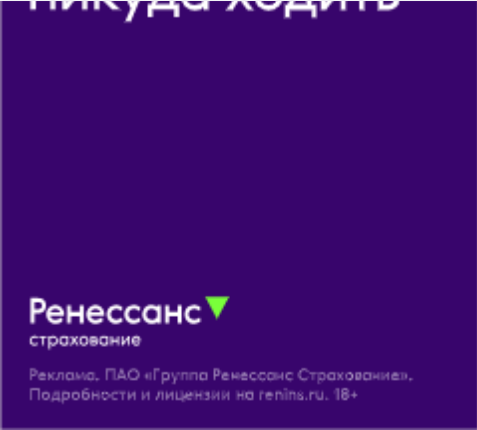
Узнать больше

Страница: / Класс TopologicalSorter(), модуль graphlib в Python

Топологическая сортировка узлов графа в Python

Модуль graphlib предоставляет функциональные возможности для топологической сортировки хэшируемых узлов графа (класс, реализующий функциональные возможности для топологической сортировки хэшируемых узлов графа).

Ссылка на документацию



```
import graphlib

# Новое в Python 3.9
graph = graphlib.TopologicalSorter(graph=None)
```

Параметры:

- graph=None - [словарь](#), представляющий ориентированный ациклический граф.

Возвращаемое значение:

- [Объект графа TopologicalSorter](#).

Описание:

[Класс graphlib.TopologicalSorter\(\)](#) предоставляет функциональные возможности для топологической сортировки хэшируемых узлов графа.

Топологический порядок - это такой линейный порядок вершин в графе, что для каждого направленного ребра $u \rightarrow v$ от вершины u к вершине v вершина u предшествует вершине v в этом порядке.

Например, вершины графа могут представлять задачи, которые должны быть выполнены, а ребра могут представлять ограничения, согласно которым одна задача должна быть выполнена раньше другой. В этом примере топологический порядок - это просто допустимая последовательность для задач. Полный топологический порядок возможен тогда и только тогда, когда граф не имеет ориентированных циклов, то есть если это ориентированный ациклический граф.

Если указан необязательный аргумент graph, это должен быть словарь, представляющий ориентированный ациклический граф, где ключи являются узлами, а значения являются итерациями всех предшественников этого узла в графе (узлы, которые имеют ребра, указывающие на значение в ключе). Дополнительные узлы могут быть добавлены к графу с помощью [метода .add\(\)](#).

Примеры использования класса graphlib.TopologicalSorter():

В общем случае шаги, необходимые для выполнения сортировки графа, следующие:

- Создайте экземпляр класса graphlib.TopologicalSorter() с необязательным начальным графом.
- Добавьте к графу дополнительные узлы [методом .add\(\)](#).
- На графе вызовите [метод .prepare\(\)](#).
- Пока значение [.is_active\(\)](#) имеет значение True, перебираем узлы, возвращаемые get_ready() и обрабатываем их.
- Вызываем [метод .done\(\)](#) на каждом узле по завершении обработки.

В случае, если требуется просто немедленная сортировка узлов в графе и не используется параллелизм, то можно на [странице](#) использовать удобный метод graphlib.TopologicalSorter.static_order():

Вверх

```
>>> import graphlib
>>> graph = {"D": {"B", "C"}, "C": {"A"}, "B": {"A"}}
>>> ts = graphlib.TopologicalSorter(graph)
>>> tuple(ts.static_order())
('РЕКЛАМА', 'B', 'D')
:
```

[Класс graphlib.TopologicalSorter\(\)](#) разработан для простой поддержки параллельной обработки узлов по мере их готовности. Например:

```
import graphlib

topological_sorter = graphlib.TopologicalSorter()

# Добавление узлов в 'topological_sorter'...
topological_sorter.prepare()
while topological_sorter.is_active():
    for node in topological_sorter.get_ready():
        # Рабочие потоки или процессы принимают узлы
        # для работы вне очереди "task_queue".
        task_queue.put(node)

    # Когда работа для узла завершена, воркеры помещают его в 'finalized_tasks_queue',
    # чтобы можно было еще получить узлы для работы.
    # Определение is_active() гарантирует, что на данном этапе в task_queue был
    # помещен хотя бы один узел, который еще не был передан в done(), поэтому
    # блокировка get() должна (в конечном итоге) быть успешной.
    # После вызова done(), снова возвращаемся к вызову get_ready(),
    # и как только станет возможно поместим только что освобожденные узлы в 'task_queue'.
    node = finalized_tasks_queue.get()
    topological_sorter.done(node)
```

Содержание раздела:
<ul style="list-style-type: none">КРАТКИЙ ОБЗОР МАТЕРИАЛА.Объект TopologicalSorter модуля graphlibИсключение CycleError модуля graphlib