Курс “Алгоритмы на python”

Занятие #9
Графы (продолжение)

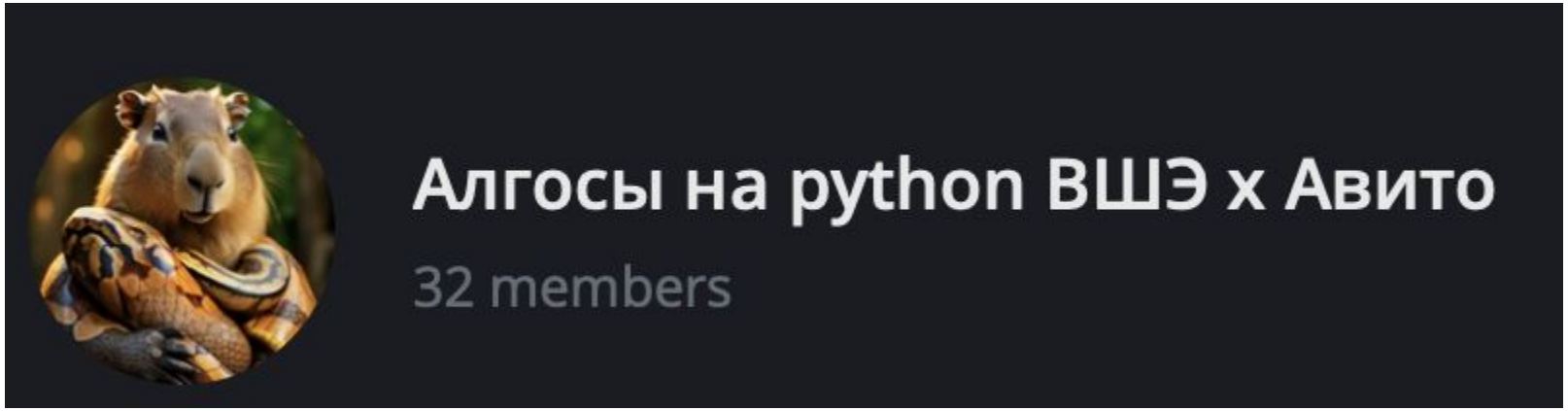
Сентябрь 2025

Avito ВШЭ ФКН

Единая точка входа/выхода – степик

- https://stepik.org/course/251189/

Вопросы и обсуждения – чат



Посещаемость

Орг моменты

Дедлайны

5 дз
* без штрафов 29 октября включительно
* минус балл — 12 ноября включительно

6 дз
* без штрафов 5 ноября включительно
* минус балл — 19 ноября включительно

7 дз
* без штрафов 14 ноября включительно
* минус балл — 28 ноября включительно

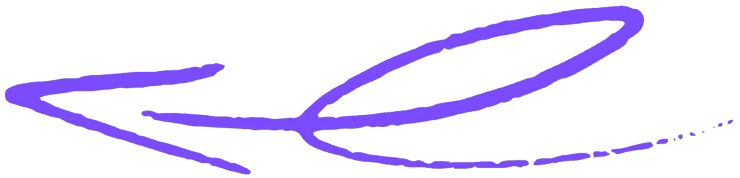
Структура курса «Алгоритмы на питоне»

1 модуль

- Введение в алгоритмы
- Базовые структуры данных
- Хеш-таблицы
- Бинарные деревья поиска
- Рекурсия
- Сортировки
- Кучи

2 модуль

- Графы
- Графы (продолжение)
- Алгоритмы в строках
- Алгоритмы в ML
- Алгоритмы в LLM



План занятия

- Часть I. Ориентированные графы
- Часть II. DFS
- Часть III. Поиск циклов
- Часть IV. DAG, топологическая сортировка

9 Графы (продолжение)

Ориентированный граф

Более сложная структура связности.
Рисуем на доске.

Ориентированный граф

Напоминание о связности в неориентированном графе

Граф

Что важно в графе:
связанное – ”можно соединить путем”

Ориентированный граф

Слабая связность – представим, что граф неориентированный

Ориентированный граф

Сильная связность – не представим, что граф неориентированный

Ориентированный граф. DFS

Граф не разваливается, но если запустим DFS – он обойдет не все. Рисуем на доске.

Ориентированный граф. DFS

Запускаем DFS, пока все не обойдем

Ориентированный граф. DFS

Посмотрим на те ребра, которые обошел DFS

Ориентированный граф. DFS

Каждый запуск DFS – дерево

Ориентированный граф. DFS

Что с остальными ребрами?

Ориентированный граф. DFS

3 типа
1 – из предка в потомка (прямое)

Ориентированный граф. DFS

3 типа
1 – из предка в потомка (прямое)
2 – из потомка в предка (обратное)

Ориентированный граф. DFS

3 типа
1 – из предка в потомка (прямое)
2 – из потомка в предка (обратное)
3 - ни 1, ни 2 (перекрестные)

Ориентированный граф. DFS

Итог: DFS на ориентированном графе – лес деревьев и ребра трех типов

Ориентированный граф. DFS

Как по ребру определить какого оно типа?

Ориентированный граф. DFS

Нужно уметь, что одна вершина является предком другой через время входа и время выхода.

Время – глобальная переменная, которая увеличивается на 1

Ориентированный граф. DFS

Чтобы проверить, что одна вершина является предком другой, нужно проверить, что отрезок времени вложен в другой отрезок.
Тут тоже рисуем.

Ориентированный граф. Цикл

Есть граф, ориентированный.
Найти цикл, если он есть

Ориентированный граф. Цикл

Какой тип ребра порождает цикл?

Ориентированный граф. Цикл

Обратное ребро – порождает цикл

Ориентированный граф. Цикл

Пойдем путем dfs. Найдем ребро второго типа. Такое ребро породит цикл

Ориентированный граф. Цикл

Допустим, нашли. Как вывести сам цикл?

Ориентированный граф. Цикл

Допустим, нашли. Как вывести сам цикл? Идти от ребенка к родителям (так как родитель один).

Ориентированный граф. Цикл

Как искать ребра 2 типа? Хотим их отличать от всего остального. Считать время входа и выхода не хочется.

Ориентированный граф. Цикл

Предок – если сейчас в dfs. То есть начали dfs, но еще не вышли из рекурсии. Рисуем на доске.

Ориентированный граф. Цикл

Когда входим в dfs – помечаем, что вошли, а когда выходим, будем снимать эту метку.

Ориентированный граф. Цикл

Итог. У вершины 3 состояния

- еще не заходили
- сейчас вошли
- уже вышли

DAG. Топологическая сортировка.

Неориентированный граф без циклов — дерево.
Ориентированный граф без циклов — весело. Структура взаимосвязей.

DAG. Топологическая сортировка.

Как проверить, что граф ацикличный?
Найти циклы.

DAG. Топологическая сортировка.

Естественное желание – топологическая сортировка.

DAG. Топологическая сортировка.

Хотим взять и упорядочить вершины графа, в таком порядке, чтобы все ребра шли в одну сторону – слева направо.

Зачем?
Есть граф зависимостей, хотим понять, в каком порядке грузить так, чтобы все компоненты, от которых зависимость зависит, были уже загружены.

DAG. Топологическая сортировка.

Способ 1.
Возьмем любую вершину, в которую не входят ребра, поставим ее на первое место. Удалим эту вершину из графа.

Найдем вершину, в которую не входят ребра, поставим ее на след место. Удалим ее из графа.

DAG. Топологическая сортировка.

Топологическая сортировка неоднозначна

DAG. Топологическая сортировка.

Способ 2.
DFS

Запускаем DFS
Для каждой вершины записываем время выхода из нее

Сортируем по убыванию времени выхода

DAG. Топологическая сортировка.

Apache Airflow

Community Meetups Documentation Use Cases Announcements Blog Ecosystem

Version: 3.1.2

Search docs

CONTENT

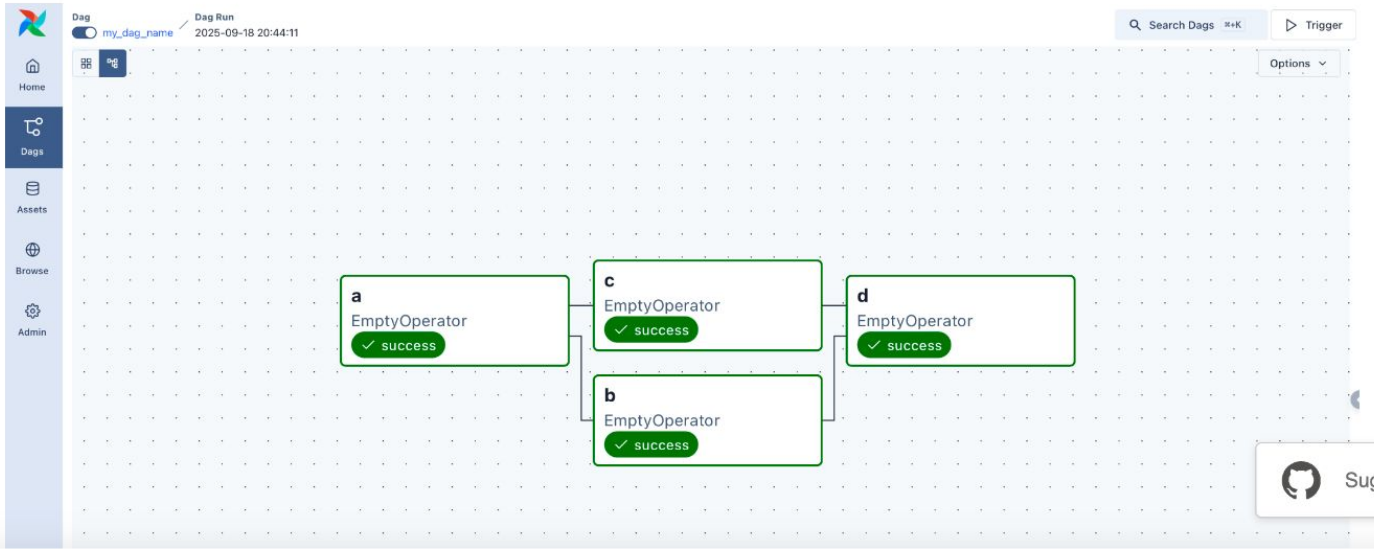
- Overview
- Quick Start
- Installation of Airflow®
- Security
- Tutorials
- How-to Guides
- UI Overview
- Core Concepts
  - Architecture Overview
  - Dags
    - Declaring a Dag
      - Loading Dags
      - Running Dags
      - Dag Assignment
      - Default Arguments
      - The Dag decorator
    - Control Flow
      - Dynamic Dags
    - Dag Visualization
      - Dag & Task
      - Documentation

Dags [link]

A Dag is a model that encapsulates everything needed to execute a workflow. Some Dag attributes include the following:

- **Schedule**: When the workflow should run.
- **Tasks**: tasks are discrete units of work that are run on workers.
- **Task Dependencies**: The order and conditions under which tasks execute.
- **Callbacks**: Actions to take when the entire workflow completes.
- **Additional Parameters**: And many other operational details.

Here's a basic example Dag:



Dags
- Declaring a Dag
  - Task Dependencies
- Loading Dags
- Running Dags
- Dag Assignment
- Default Arguments
- The Dag decorator
- Control Flow
  - Branching
  - Latest Only
  - Depends On Past
  - Trigger Rules
  - Setup and teardown
- Dynamic Dags
- Dag Visualization
  - TaskGroups

Suggest a change on this page

Packaging Dags