2

Лекция 16–17 февраля

Обходы и деревья

Константин Чухарев

Дискретная математика — Весна 2027

Обходы графа

> *Не все, кто блуждает, потеряны.*
>
> ***— Дж. Р. Р. Толкин***

Зачем обходить граф?

Граф задан, но из пары множеств мало что видно.

Обход превращает граф в структуру с порядком: что за чем посещено.

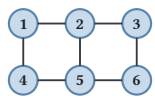
- Найти компоненты связности.
- Найти кратчайшие пути от старта.
- Найти циклы и топологический порядок.
- Построить остов.

Любой обход честен к графу: каждая вершина и каждое ребро просматриваются $O(1)$ раз, итого $O(V + E)$.

Поиск в ширину

1. Помещаем стартовую вершину в очередь.
2. Пока очередь не пуста:
   - извлекаем вершину u из головы;
   - всех непосещённых соседей v помечаем и ставим в хвост.

Очередь честная: кто пришёл раньше, тот выходит раньше.



BFS идёт слоями: соседи старта, затем соседи соседей.

Слой вершины — её расстояние от старта.

Слои и кратчайшие пути

Родительские ссылки BFS («кто меня обнаружил») образуют дерево.

Путь от старта до любой вершины по этим ссылкам — кратчайший.

Пример

В социальной сети слои BFS — круги знакомых: друзья, друзья друзей.

Эксперимент Милгрэма (1967) показал: письма между незнакомцами доходили по цепочке длиной около шести.

BFS даёт кратчайшие пути, пока рёбра одинаковые.

Как только у рёбер появляются веса, слои врут — вернёмся к этому в лекции о кратчайших путях.

Компоненты за один обход

Компоненты связности считаются обходом.

Запуск из непосещённой вершины собирает ровно её компоненту.

Повторяем запуски, пока остаются непосещённые вершины.

Число запусков — число компонент, а общее время остаётся $O(V + E)$: каждая вершина по-прежнему посещена один раз.

Двудольность за один обход

BFS красит слои попеременно: старт в цвет A, его соседи в цвет B, и так далее.

Ребро внутри одного цвета вместе с путями к старту образует цикл нечётной длины.

> Слои без внутренних рёбер — двудольность.
>
> Для большего числа цветов простого критерия нет — к раскраске вернёмся в конце модуля.

Поиск в глубину

1. Для каждой непосещённой вершины u запускаем $\mathrm{DFS}(u)$.
2. $\mathrm{DFS}(u)$:
   - помечает u и записывает время входа $\mathrm{pre}(u)$;
   - для каждого непосещённого соседа v вызывает $\mathrm{DFS}(v)$;
   - записывает время выхода $\mathrm{post}(u)$.

Структура данных — стек вызовов: уходим вглубь, пока можно.

> DFS не ищет ближайшее — он исследует надолго и тщательно.
>
> Времена pre и post превращают обход в часы графа.

Времена pre и post

Для каждой вершины интервал $[\mathrm{pre}(u), \mathrm{post}(u)]$ — время её жизни в обходе.

Теорема 1 (Вложенность интервалов)

Для любых вершин u, v интервалы $[\mathrm{pre}(u), \mathrm{post}(u)]$ и $[\mathrm{pre}(v), \mathrm{post}(v)]$ либо не пересекаются, либо один вложен в другой.

Доказательство

Интервалы не могут пересекаться по краю: если $\mathrm{pre}(u) < \mathrm{pre}(v) \le \mathrm{post}(u)$, то v открыта при живой u и будет закрыта раньше — все потомки закрываются до закрытия предка.

Вложенность — это карта рекурсии: кто в чьей ветке стека находился.

На ней держатся поиск циклов, мостов и топологическая сортировка.

Глубина рекурсии

DFS живёт в стеке вызовов, и стек не бесконечен.

Пример

На графе-пути в 10^5 вершин наивная рекурсия падает раньше, чем посетит все вершины.

Лечение — явный стек: тот же порядок обхода без ограничения глубины.

Очереди BFS глубина не грозит: память $O(V)$ при любом графе.

Выбор обхода — ещё и инженерное решение.

BFS против DFS

Свойство	BFS	DFS
Структура	очередь	стек
Порядок	по слоям	вглубь
Находит	кратчайшие пути	циклы, топологию
Сложность	$O(V + E)$	$O(V + E)$

Сложность одинакова, информация разная: BFS отвечает «как далеко», DFS — «как устроено».

Цикл найдёт DFS

Как понять, есть ли в графе цикл?

Пример

В неориентированном графе DFS находит цикл ровно тогда, когда выходит на уже посещённую вершину, которая не его родитель.

Ребро к «чужому» предку замыкает круг.

Ациклический граф — это дерево или лес, о которых дальше.

Проверка на циклы и построение остова — один и тот же обход.

Топологическая сортировка

Ориентированный граф без циклов называется **DAG** (*directed acyclic graph*).

- Зависимости сборки: модуль a собирается раньше модуля b.
- Пререквизиты курсов.
- Порядок одевания: носки раньше ботинок.

Определение 1 (Топологический порядок)

Нумерация вершин DAG, при которой каждое ребро $u \to v$ идёт от меньшего номера к большему.

В лекциях семестра 1 про порядок топологическая сортировка была определением: линейное продолжение частичного порядка.

Теперь это алгоритм, и работает он за $O(V + E)$.

Алгоритм Кана

1. Посчитать входящие степени всех вершин.
2. Поставить в очередь вершины с нулевой входящей степенью — истоки.
3. Пока очередь не пуста: извлечь исток, вывести его, уменьшить степени его соседей.
4. Вершина с обнулённой степенью встаёт в очередь.

Пример

Зависимости: core → utils, utils → app, utils → tests.

Порядок core, utils, app, tests — топологический.

Если выведено меньше вершин, чем n, оставшиеся образуют цикл: сборка невозможна.

Цикл в зависимостях — фатальная ошибка проекта.

Обходы и код

- Очередь BFS — `deque` в Python или `VecDeque` в Rust.
- Рекурсия DFS — стек вызовов, при нехватке глубины переписывается на явный стек.
- Сборщики `make` и `ninja` строят граф зависимостей и запускают цели в топологическом порядке.
- Сборщик мусора обходит граф живых объектов — пометкой (mark-and-sweep).

> «Пометить вершину» в коде — это флаг `visited`: без него обход зациклится на первом же цикле.

Деревья

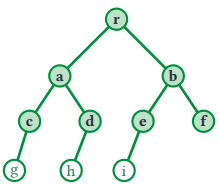
> *Родственные связи всех существ одного класса иногда изображали в виде великого дерева. Я верю, что эта метафора во многом соответствует истине.*
>
> ***— Чарльз Дарвин***

Дерево и лес

Определение 2

Дерево — связный граф без циклов.

Лес — граф без циклов.



Пример

Цепочка $1 — 2 — 3 — 4 — 5$ — дерево: 5 вершин, 4 ребра.

Цепочки $1 — 2 — 3$ и $4 — 5$ — лес, а два треугольника — не лес: в каждом цикл.

Сколько рёбер у дерева?

Теорема 2 (Число рёбер)

Дерево с n вершинами имеет ровно $n - 1$ ребро.

В любом дереве с $n \geq 2$ есть хотя бы два листа — вершины степени 1.

Доказательство

Набросок: от каждой вершины, кроме корня, ведёт ровно одно родительское ребро — итого $n - 1$.

Самый длинный путь заканчивается в двух вершинах степени 1, и меньше двух листьев не бывает.

Дерево — минимальный связный каркас: убрать любое ребро — связь потеряна, добавить любое — появился цикл.

Пять характеризаций

Теорема 3 (Эквивалентные определения дерева)

Для связного графа с n вершинами эквивалентны:

1. ациклический.
2. имеет $n - 1$ ребро.
3. между любой парой вершин — единственный путь.
4. каждое ребро — мост.

Доказательство

Набросок: связность и $n - 1$ ребро запрещают цикл — убрав ребро цикла, сохраняем связность и получаем $n - 2$ ребра.

Единственность пути равносильна ацикличности, а немостовое ребро всегда можно обойти, так что каждое ребро дерева — мост.

Деревья вокруг

- Файловая система: каталоги и файлы, корень — `/`.
- DOM страницы: теги и вложенность.
- Наследование классов в языках с одиночным наследованием.
- Родословная одного рода.

> Каждый раз вопрос один: у каждого объекта ровно один «родитель», корень — начало.
>
> Именно это дерево и фиксирует.

Остовное дерево

Определение 3 (Остовное дерево)

Остовное дерево (*spanning tree*) графа — подграф, содержащий все вершины и являющийся деревом.

Теорема 4 (Существование)

Любой связный граф имеет остовное дерево.

Доказательство

Обойдём граф и оставим рёбра, ведущие к новым вершинам: результат связен (каждая вершина достигнута) и ацикличен (каждое ребро добавляло вершину).

По характеризациям это дерево.

Сколько остовных деревьев?

Теорема 5 (Формула Кэли)

Полный граф K_n имеет ровно n^{n-2} остовных деревьев.

Пример

K_4: $4^2 = 16$ остовных деревьев.

K_{10}: уже 10^8 — перебор безнадёжен, нужен алгоритм.

Кэли вывел формулу в 1889 году.

Считать такие числа научит комбинаторика во второй половине семестра.

Крускал и Прим

Рёбрам приписаны веса — ищем остов минимального суммарного веса.

- **Крускал**: сортируем рёбра, добавляем легчайшее, не создающее цикла.
- **Прим**: растим дерево из одной вершины, добавляя легчайшее ребро наружу.

Оба алгоритма жадные, и оба оптимальны.

Жадность оптимальна не везде: здесь работает обменный аргумент — выбранное ребро вписывается в минимальный остов заменой ребра не меньшего веса, и стоимость остова не растёт.

Проверка «не создаёт ли ребро цикла» в Крускале — это система непересекающихся множеств: компоненты леса сливаются по мере добавления рёбер.

Лекция 2: что мы поняли?

- Обходы идут за $O(V + E)$: BFS — очередью и слоями, DFS — стеком и временами pre и post.
- BFS даёт кратчайшие пути в невзвешенном графе, DFS — циклы и топологический порядок.
- Топологическая сортировка из лекций о порядке превратилась в алгоритм Кана за линейное время.
- Дерево — связный ациклический граф: $n - 1$ ребро, единственный путь, все рёбра — мосты.

Вопросы для самопроверки

1. Почему BFS находит кратчайшие пути, а DFS — нет?
2. Что означает вложенность интервалов pre и post? Что она говорит о рекурсии?
3. Докажите, что в дереве с $n \geq 2$ есть хотя бы два листа.
4. Объясните, почему граф связен тогда и только тогда, когда в нём есть остовное дерево.
5. В каком порядке сборщик обработает цели, если зависимости образуют цикл?

Чтение к следующей паре

- m09 — «Обходы графа», «Топологическая сортировка», «Деревья», «Минимальные остовные деревья».
- Флеш-опросник — в начале лекции 3.