Лекция 6: Деревья

Б.М.Верников, А.М.Шур

Уральский федеральный университет,
Институт математики и компьютерных наук,
кафедра алгебры и дискретной математики

Определение

Деревом называется связный граф без циклов.

Примеры деревьев изображены на рис. 1 (см. также рис. 3 и 4 в лекции 1).

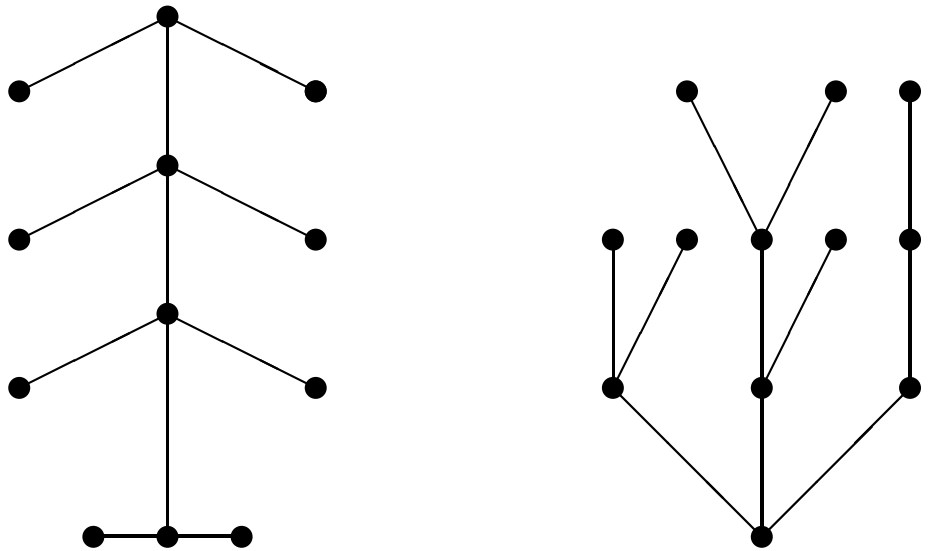

Рис. 1. Два дерева

Замечание

Всякое дерево является обыкновенным графом. В самом деле, петля — это цикл длины 1, а кратное ребро любой кратности содержит цикл длины 2. Значит, граф без циклов не содержит кратных ребер и петель.

Корневое изображение дерева

Для всякого дерева существует его так называемое *корневое изображение*, играющее важную роль в приложениях теории графов. Его построение основано на следующей лемме.

Лемма о разбиении дерева

Множество вершин произвольного дерева G можно разбить в объединение непустых попарно непересекающихся подмножеств $V_0, V_1, \ldots, V_k$ так, что будут выполнены следующие условия:

(i) *множество V_0 состоит из одной вершины;*

(ii) *для всякого $i = 1, \ldots, k$ никакие вершины из V_i не смежны;*

(iii) *для всякого $i = 1, \ldots, k$ любая вершина из V_i смежна ровно с одной вершиной из V_{i-1} и не смежна ни с одной вершиной из $V_{i-2}, \ldots, V_0$.*

Доказательство. Обозначим через v_0 произвольную вершину дерева G; пусть k — максимум из расстояний от v_0 до других вершин дерева. Для любого $i = 0, \ldots, k$ положим $V_i = \{v \in V(G) \mid \rho(v_0, v) = i\}$. Очевидно, что никакие из множеств V_i не пересекаются, и что $V_0 = \{v_0\}$. Далее, каждая вершина из множества V_i имеет смежную вершину во множестве V_{i-1} (если расстояние между v_0 и v равно i, то расстояние между v_0 и предпоследней вершиной кратчайшей (v_0, v)-цепи равно $i - 1$), т. е., в частности, все множества V_i непусты.

Лемма о разбиении дерева (окончание доказательства)

Пусть $v \in V_i$. Тогда все смежные с v вершины принадлежат одному из множеств V_{i-1}, V_i или V_{i+1}, поскольку расстояния от одной вершины (v_0) до двух смежных вершин не могут отличаться более, чем на единицу. Осталось показать, что у v_i нет смежной вершины во множестве V_i и нет двух смежных вершин во множестве V_{i-1}. Как видно из рис. 2, на котором изображены кратчайшие цепи из рассматриваемых вершин до вершины v_0, в каждом из этих случаев в графе G обнаруживается цикл. □

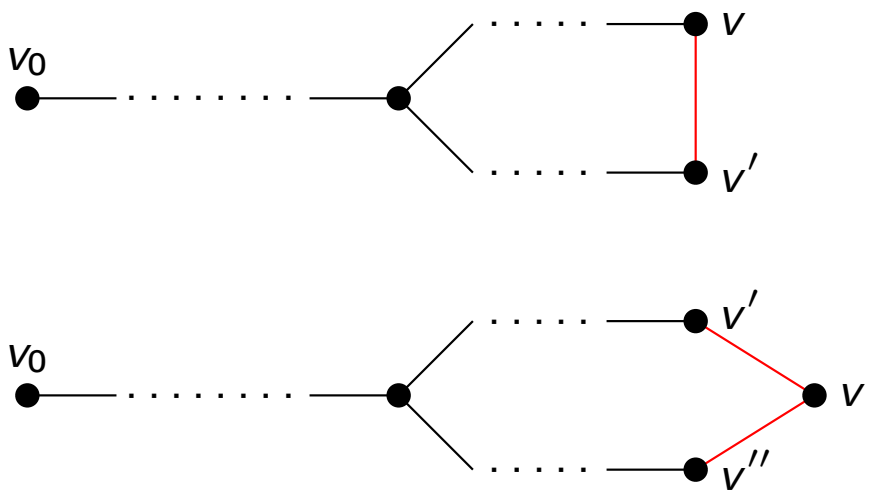

Рис. 2

Построение корневого изображения дерева

Пусть G — дерево, а $V_0, V_1, \ldots, V_k$ — наборы вершин, построенные в лемме о разбиении дерева, причем $V_0 = \{v_0\}$. Изобразим вершину v_0 вверху, вершины из множества V_1 — на одном уровне и ниже, чем v_0, вершины из множества V_2 — на одном уровне и ниже, чем вершины из V_1, и т. д. При этом, если на i-м уровне слева направо изображены вершины $v_1, v_2, \ldots$, то на $(i+1)$-м уровне слева направо изображаются сначала вершины, смежные v_1, затем смежные v_2, и т. д. При таком изображении графа ребра не будут пересекаться. Полученный граф, изоморфный G, и будет *корневым изображением* дерева G. Вершина v_0 называется *корнем* дерева. Ниже на рис. 3 приведены дерево и одно из его корневых изображений (одно и то же дерево имеет много корневых изображений, в зависимости от выбора корня).

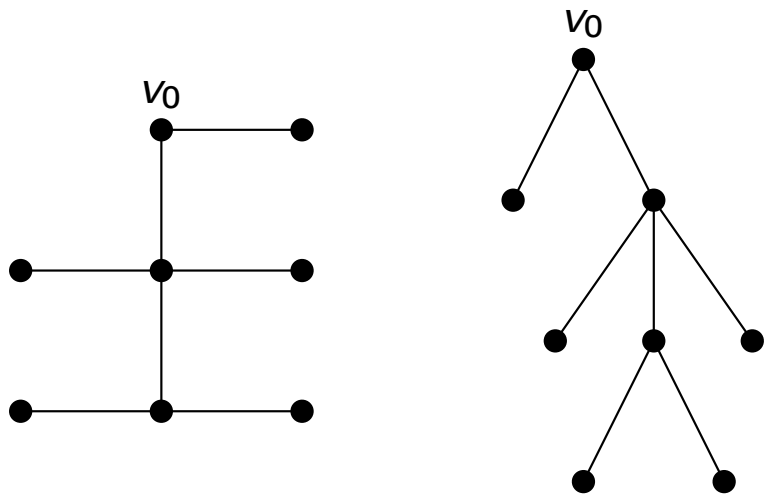

Рис. 3. Дерево и его корневое изображение

Корневые деревья

Корневое изображение — это «геометрическое» представление важнейшей в информатике структуры данных — *корневого дерева*. От обычного дерева, т. е. связного графа без циклов, корневое дерево отличается следующими дополнительными условиями:

- есть выделенная вершина — *корень дерева*;
- у любой вершины v, кроме корня, есть *отец* в дереве — это единственная смежная с v вершина, расстояние от которой до корня меньше, чем от v до корня;
- все вершины, смежные данной, кроме ее отца, называются ее *сыновьями*; вершины без сыновей называются *листьями* дерева;
- на множестве вершин определены бинарные отношения «быть предком» (u — предок v, если u принадлежит цепи от корня до v), «быть потомком» (u — потомок v, если v — предок u) и «быть братом» (u — брат v, если у них совпадают отцы).

Применение корневых деревьев очень разнообразно. Мы ограничимся несколькими примерами на следующем слайде.

Применение корневых деревьев

- *Дерево перебора вариантов.* Когда компьютерная программа выбирает лучшее из возможных действий (например, ход при игре в шахматы), она строит дерево, корнем которого является текущая позиция в игре, сыновьями корня — позиции, возникающие после различных ее ходов, сыновьями сыновей — позиции после ответов противника на эти ходы, и т. д. На основе заложенной в программу функции оценки позиции «отсекаются» определенные части дерева и выбирается лучший ход.
- *Дерево поиска.* Предположим, в графе G нужно найти расстояние между вершинами u и v и кратчайшую (u, v)-цепь. Построим дерево с корнем u: сыновьями корня сделаем все вершины, смежные с u, потом обработаем всех сыновей по очереди, добавляя к каждому из них в сыновья все смежные вершины, ранее в дерево не включенные, потом проделаем это с сыновьями сыновей, и т. д. Номер итерации, на которой в дереве появится вершина v, будет искомым расстоянием, а единственная (u, v)-цепь в дереве — кратчайшей (u, v)-цепью в G.
- *Синтаксическое дерево.* Чтобы перевести фразу с одного языка на другой (да и просто, чтобы понять ее смысл), нужно выяснить ее структуру, найти главные члены предложения, зависимые от них, зависимые от зависимых и т. д. Структурированная таким образом фраза также будет иметь вид корневого дерева.

Условия, эквивалентные условию «быть деревом»

Далее до конца лекции рассматриваются обычные (не корневые) деревья.

Теорема о деревьях

Для обыкновенного графа G следующие условия эквивалентны:

(i) *G — дерево;*

(ii) *любые две различные вершины в G соединены ровно одной цепью;*

(iii) *G не содержит циклов, но если любую пару несмежных в G вершин соединить ребром, то в полученном графе будет ровно один цикл;*

(iv) *G — связный граф и $n(G) = m(G) + 1$;*

(v) *граф G не содержит циклов и $n(G) = m(G) + 1$;*

(vi) *G — связный граф, и всякое ребро в G является мостом.*

Доказательство. Вначале предположим, что G — дерево и докажем условия (ii)–(vi).

Теорема о деревьях: доказательство (1)

(ii). Дерево — связный граф, а значит, любые две вершины соединены цепью. Предположим, что вершины u и v соединены в G не менее чем двумя цепями и получим противоречие. Пусть $u = v_0 \to v_1 \to \cdots \to v_k = v$ и $u = v'_0 \to v'_1 \to \cdots \to v'_\ell = v$ — различные (u, v)-цепи. Поскольку первые вершины в этих цепях совпадают, существует число i такое, что $v_0 = v'_0$, $\ldots$, $v_i = v'_i$, но $v_{i+1} \neq v'_{i+1}$. Далее, пусть j — наименьшее из чисел, больших i, такое, что вершина v_j принадлежит второй цепи (такое j существует, поскольку в рассматриваемых цепях совпадают и последние вершины). Тогда маршрут $v_i \to \cdots \to v_j = v'_r \to \cdots \to v'_i = v_i$ не содержит повторяющихся ребер, а значит, является циклом в G, противоречие.

(iii). При добавлении к (u, v)-цепи ребра (u, v) очевидно возникает цикл. Таким образом, из связности G следует, что цикл возникает при добавлении любого ребра. Если при добавлении ребра (u, v) возникло более одного цикла, значит, вершины u и v соединены более чем одной цепью, что невозможно, так как условие (ii) мы уже доказали.

(iv), (v). Чтобы сравнить число вершин и ребер в G, возьмем корневое изображение G с корнем v_0. Каждому ребру можно взаимно однозначно сопоставить вершину, являющуюся нижним концом этого ребра в корневом изображении. Таким образом, ребер в G столько же, сколько вершин, не совпадающих с корнем, т. е. $n(G) - 1$.

Теорема о деревьях: доказательство (2)

(vi). Поскольку ни одно ребро дерева не лежит в цикле, это условие прямо следует из теоремы о мостах (лекция 5).

Теперь покажем, что каждое из условий (ii)–(vi) влечет что G — дерево.

(ii). Поскольку две любые вершины соединены цепью, G связен, а так как цепь единственна — в G нет циклов (две вершины, находящиеся в цикле, соединены по крайней мере двумя цепями — фрагментами этого цикла).

(iii). Циклов в G нет по условию. Предположим, что в G более одной компоненты связности. Соединим ребром две вершины из разных компонент. В полученном графе новое ребро будет мостом по определению. По теореме о мостах оно не лежит ни в каком цикле, т. е. при его добавлении цикл не образовался, противоречие.

Теорема о деревьях: доказательство (3)

(iv). Вначале докажем, что в связном графе $m(G) \geqslant n(G) - 1$. Возьмем граф без ребер с n вершинами и будем добавлять ребра по одному. Что происходит при добавлении в граф одного ребра? Если добавленное ребро в новом графе оказалось мостом, то по лемме об удалении моста (лекция 5) новый граф содержит ровно на одну компоненту связности меньше, чем старый. Если же добавленное ребро — не мост, то число компонент связности не изменилось. Поскольку в исходном графе n компонент связности, необходимо как минимум $n - 1$ ребер, чтобы сделать его связным.

Граф G связен по условию. Если в нем есть цикл, удалим из него ребро и получим, по теореме о разрыве цикла (лекция 3), связный граф, у которого ребер на 2 меньше, чем вершин, что невозможно, как мы только что доказали.

(v). Так как G не содержит циклов, каждая из его компонент связности является деревом, а значит, по доказанному ранее, число ребер в ней на единицу меньше числа вершин. Поскольку это же условие выполняется и для всего графа, компонента связности может быть только одна.

(vi). Связность G дана по условию, а отсутствие циклов прямо следует из теоремы о мостах. Доказательство теоремы завершено. □

Каркас связного графа

Определение

Каркасом связного графа называется суграф этого графа, являющийся деревом.

Утверждение

Любой связный граф содержит каркас.

Доказательство. Если связный граф G не содержит циклов, то он сам является своим каркасом. В противном случае выберем произвольное ребро e графа G, входящее в цикл, и удалим его из G. По теореме о разрыве цикла (лекция 3) граф $G - e$ будет связным. Будем повторять процедуру удаления ребра из цикла, пока не получим связный граф без циклов. (В соответствии с теоремой о деревьях, это произойдет в тот момент, когда число оставшихся ребер станет на единицу меньше числа вершин.) Поскольку мы удаляли только ребра, полученный граф содержит все вершины исходного. Значит, он является каркасом в G по определению. □

С понятием каркаса графа связана следующая задача теории графов.

Задача о минимальном соединении

Имеется несколько городов, никакие два из которых не соединены дорогой. Для любых двух городов известна стоимость прокладки дороги между ними (либо известно, что дорогу между ними проложить невозможно). Требуется построить дороги так, чтобы можно было проехать из любого города в любой другой (возможно, не напрямую, а проездом через другие города), и чтобы суммарная стоимость прокладки всех дорог была минимальной.

Если заменить города и дороги узлами и линиями связи, задача становится более практической: в случае дорог обычно принципиально наличие достаточно коротких дорог между городами, а в случае линий связи и сигнала, идущего со скоростью света, длина линии значения не имеет, принципиально именно наличие связи.

Взвешенные графы

Для моделирования задачи о минимальном соединении нам потребуется следующие понятия.

Определение

Взвешенным графом называется пара (G, ϕ), где G — граф, а ϕ — функция, ставящая в соответствие каждому ребру графа действительное число. Число $\phi(e)$ называется *весом ребра* e. *Весом подграфа* называется сумма весов входящих в него ребер.

Заметим, что

- в зависимости от задачи, смоделированной взвешенным графом, вес иногда называют *длиной*, *стоимостью* или *пропускной способностью*;
- во всех изучаемых нами задачах веса ребер неотрицательны.

Задача о минимальном соединении на языке теории графов

Представим задачу о минимальном соединении графом: вершины соответствуют городам, ребра — дорогам, которые можно проложить, а их веса — стоимости прокладки этих дорог. Полученный граф должен быть связным (по построенным дорогам надо иметь возможность проехать из любого города в любой другой). В этом графе нужно выбрать некоторые ребра так, чтобы:

(i) любые вершины были связаны цепью, состоящей только из выбранных ребер;

(ii) суммарный вес выбранных ребер был минимальным.

Все города и построенные дороги образуют суграф исходного графа. Условие (i) означает, что суграф должен быть связным. А условие (ii) показывает, что этот суграф не может содержать циклов: в противном случае после удаления из нашего суграфа любого ребра, входящего в цикл, вес суграфа уменьшится, а связность останется в силу теоремы о разрыве цикла (лекция 3). Таким образом, выбранный нами суграф должен быть деревом, т. е. каркасом исходного графа, что дает нам следующую формулировку.

Задача о минимальном соединении на языке теории графов

Дан взвешенный связный граф, в котором веса всех ребер положительны. Требуется найти каркас этого графа с минимально возможным весом.

Алгоритм Краскала

Приведем следующий алгоритм решения задачи о минимальном соединении. Заметим, что он работает и в случае, когда некоторые ребра имеют нулевой вес (т. е. некоторые дороги уже проложены).

Алгоритм Краскала

Вход: связный взвешенный граф (G, ϕ) с неотрицательными весами.
Выход: список T ребер каркаса минимального веса в G.

1. Положить $T = \varnothing$; выписать все ребра из G в список S по возрастанию веса.
2. Выбрать в S ребро e минимального веса и удалить из S.
3. Если e не образует цикла с ребрами из T, добавить e в T.
4. Если список S не пуст, вернуться на шаг 2.

Замечание

Работу алгоритма Краскала можно останавливать раньше: по теореме о деревьях каркас должен содержать $n(G) - 1$ ребер, так что по достижении этого количества ни одно из оставшихся в S ребер в T попасть не может. Следовательно, можно поставить «счетчик» добавленных ребер и на шаге 4 проверять, достигнуто ли требуемое значение.

Алгоритм Краскала: доказательство корректности

Из формулировки алгоритма Краскала совсем не очевидно, что он находит именно то, что заявлено. Чтобы доказать корректность алгоритма, нужно установить три вещи.

- Алгоритм заканчивает работу на любых корректных данных (это очевидно, так как список S рано или поздно будет исчерпан).
- Алгоритм строит каркас (очевидно, алгоритм строит суграф без циклов; если бы этот суграф имел более одной компоненты связности, то в силу связности исходного графа G, в G было бы ребро между вершинами этих компонент; тогда алгоритм добавил бы это ребро в T, поскольку оно не образует цикла с вершинами из T).
- Алгоритм строит каркас минимального веса — приведем подробное доказательство.

Доказательство. Пусть $K = \langle V(G), T\rangle$ — каркас, построенный алгоритмом Краскала, $\phi(K)$ — его вес. Предположим, что в G есть каркас меньшего веса; среди всех таких каркасов выберем каркас K', который отличается от K в минимальном числе ребер (т. е. число ребер, которые есть в K' и отсутствуют в K, минимально по всем каркасам веса меньшего чем $\phi(K)$).

Алгоритм Краскала: доказательство корректности (2)

Пусть e' — ребро минимального веса среди тех, которые есть в K' и отсутствуют в K. Алгоритм Краскала не добавил ребро e' в T, т. е. оно образует цикл с ребрами, выбранными в T ранее (см. рис. 4), а значит, вес каждого из этих ребер не больше $\phi(e')$. Какое-то из этих ребер, например, e, не принадлежит K', иначе бы в K' образовался цикл.

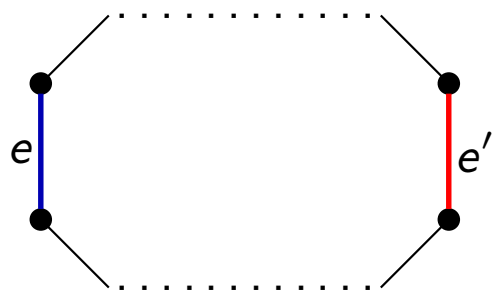

Рис. 4

Добавим e к каркасу K'; по теореме о деревьях, при этом образуется один цикл. Теперь удалим ребро e', разрывая этот цикл и снова получая каркас в силу теоремы о разрыве цикла (лекция 3). Обозначим этот новый каркас за K''. Имеем

$$\phi(K'') = \phi(K') + \phi(e) - \phi(e') \leqslant \phi(K') < \phi(K),$$

причем K'' отличается от K меньшим числом ребер, чем K'. Полученное противоречие с выбором K' доказывает, что построенный алгоритмом Краскала каркас минимален.

Алгоритм Краскала: пример (1)

Применим алгоритм Краскала для нахождения каркаса графа, изображенного на рис. 5.

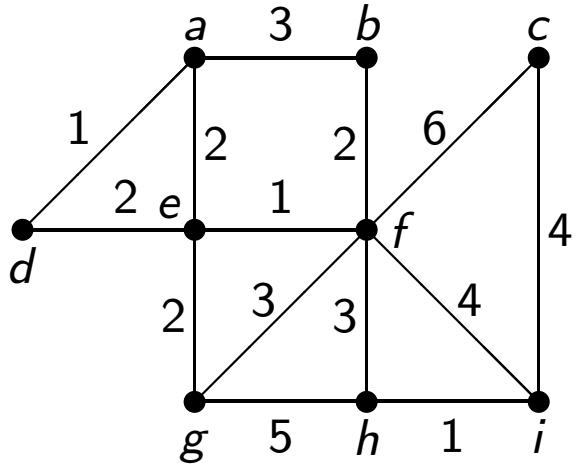

Рис. 5

Упорядоченный список ребер выглядит так:

$$(a,d),(e,f),(h,i),(a,e),(b,f),(d,e),(e,g),$$
$$(a,b),(f,g),(f,h),(c,i),(f,i),(g,h),(c,f).$$

Алгоритм Краскала: пример (2)

Действуя по алгоритму, последовательно поместим в T ребра (a, d), (e, f), (h, i), (a, e) и (b, f). На следующем шаге из списка будет взято ребро (d, e); оно образует с ранее выбранными ребрами цикл $a \to e \to d \to a$, и не добавляется в T. Далее, в T будет добавлено ребро (e, g) и не будут добавлены ввиду появления циклов ребра (a, b) и (f, g). Наконец, на двух следующих шагах в T добавятся ребра (f, h) и (c, i), после чего в T окажется требуемое для каркаса число ребер (8) и алгоритм можно будет остановить. В результате получим каркас исходного графа, изображенный на рис. 6, который и является каркасом наименьшего веса (16) в исходном графе.

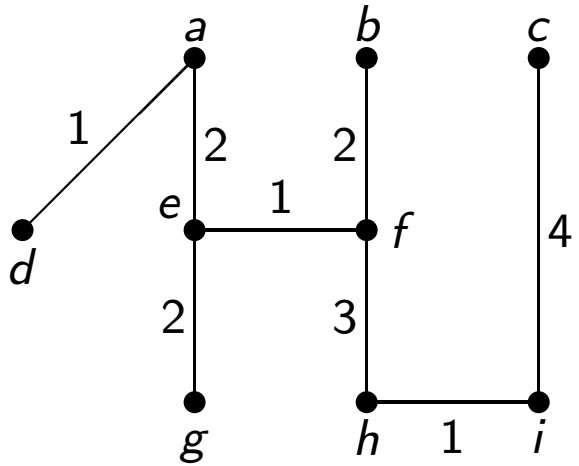

Рис. 6