Лекция 15: Сети

Б.М.Верников, А.М.Шур

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

Понятие сети

Предположим, мы хотим кратчайшим образом добраться на автомобиле из пункта A в пункт B в своем родном городе. Мы открываем электронную карту города, отмечаем на ней пункт отправления A и пункт прибытия B, нажимаем на кнопку — и на карте появляется требуемый маршрут с указанием длины и/или ожидаемого времени в пути. Аналогичные задачи в режиме реального времени решают, например, спутниковые навигаторы, используемые сейчас едва ли не в каждом автомобиле. Попробуем взглянуть на эти задачи не со стороны пользователя, а со стороны «решателя». Для этого нам понадобится ориентированный аналог понятия взвешенного графа.

Определение

Сетью называется пара (G, λ), где G — орграф, а λ — функция, ставящая в соответствие каждой дуге орграфа G действительное число, называемое *длиной* этой дуги.

- В рамках этого курса мы рассматриваем только такие сети, в которых *длины всех дуг неотрицательны*.

Определение

Пусть (G, λ) — сеть. *Длиной цепи* в сети называется сумма длин дуг, входящих в эту цепь.
Расстоянием от вершины v до вершины w называется минимум из длин (v, w)-цепей в G, обозначаемый через $d(v, w)$. *Кратчайшей* называется любая (v, w)-цепь длины $d(v, w)$. Если вершина w не достижима из v, то полагают $d(v, w) = \infty$.

Отметим, что

- для любой вершины v существует тривиальная (v, v)-цепь, откуда $d(v, v) = 0$;
- если (v, w) — дуга, то $d(v, w) \leqslant \lambda(v, w)$, причем возможно и строгое неравенство (например, $\lambda(v, w) = 3$, а в орграфе есть дуги (v, u) и (u, w) длины 1).

Сформулированную в начале лекции «автомобильную» задачу можно перевести на язык теории графов следующим образом.

Задача о минимальном пути

Дана сеть (G, λ) и ее вершины v, w. Найти расстояние от v до w; если оно конечно, найти кратчайшую (v, w)-цепь.

Лемма о кратчайших цепях

Пусть кратчайшая (v, w)-цепь в сети (G, λ) имеет вид $v = v_0 \to v_1 \to \cdots \to v_k = w$. Тогда для любого $i = 1, \ldots, k$ начальный фрагмент $v_0 \to v_1 \to \cdots \to v_i$ данной цепи является кратчайшей (v, v_i)-цепью.

Доказательство очевидно: если бы существовала более короткая (v, v_i)-цепь, то, объединив ее с фрагментом $v_i \to \cdots \to w$ исходной цепи, мы получили бы и более короткую (v, w)-цепь. □

Данная лемма показывает, что в поиске кратчайшей (v, w)-цепи будут задействованы и другие кратчайшие цепи, т. е. поиск кратчайших цепей удобно производить «массово». Так, алгоритм Дейкстры, приводимый далее, одновременно ищет кратчайшие цепи от вершины v до всех вершин орграфа (есть и алгоритмы, которые одновременно ищут кратчайшие цепи между всеми парами вершин). При этом, чтобы восстановить все кратчайшие цепи из вершины v, достаточно для каждой вершины w орграфа хранить только ее «предшественника» — предыдущую вершину в кратчайшей (v, w)-цепи. (Из доказанной леммы следует, что если $v \to v_1 \to \cdots \to v_k$ — кратчайшая цепь, то v_{k-1} — предшественник v_k, v_{k-2} — предшественник v_{k-1}, и т. д., а предшественником v_1 является v.)

Алгоритм Дейкстры

Вход: сеть (G, λ) и ее вершина v.

Выход (для каждой вершины $w \neq v$ из G): расстояние $D(w) = d(v, w)$ и предшественник $P(w)$ в кратчайшей (v, w)-цепи в случае $d(v, w) < \infty$.

1. Положить $S = V(G) \setminus \{v\}$; $D(w) = \infty$, $P(w) = \varnothing$ для всех вершин $w \in S$; $D(v) = 0$ и $v^* = v$.
2. Для всех $w \in S$ таких, что в G существует дуга (v^*, w) и $D(v^*) + \lambda(v^*, w) < D(w)$, заменить $D(w)$ на $D(v^*) + \lambda(v^*, w)$ и положить $P(w) = v^*$.
3. Если $S \neq \varnothing$, найти вершину $w \in S$, для которой значение D минимально среди всех вершин из S и положить $v^* = w$; если при этом $D(v^*) < \infty$, исключить v^* из множества S и перейти на шаг 2.

- Алгоритм Дейкстры, как и другие алгоритмы поиска минимального пути, можно использовать, никак не модифицируя, и для работы с неориентированными взвешенными графами.

Алгоритм Дейкстры: доказательство корректности

Очевидно, алгоритм закончит работу, поскольку после каждой итерации размер множества S уменьшается на единицу. Сделаем три замечания:

1. если величина $D(w)$ на какой-то итерации становится конечной, то вершина w будет выбрана в качестве v^* на одной из последующих итераций;
2. на любой итерации, на которой величина $D(w)$ конечна, она равна длине какой-то (v, w)-цепи;
3. если w достижима из v, то величина $D(w)$ на какой-то итерации станет конечной. (Это условие получается многократным применением следующего наблюдения: если $D(u)$ конечна для некоторой вершины u, то на той итерации, когда $v^* = u$, значение D станет конечным для всех вершин, являющихся концами дуг с началом u.)

Докажем, что по окончании работы алгоритма $D(w) = d(v, w)$ для любой вершины w. Для случая, когда $D(w) = \infty$, это следует из замечания 3. Пусть $D(w)$ конечно; проверим, что $D(w) = d(v, w)$ в момент, когда w выбрана в качестве v^* (после этого момента значение $D(w)$ уже не изменится). По замечанию 2, $D(w) \geqslant d(v, w)$ в любой момент.

Доказывать будем индукцией по числу итераций алгоритма. *База индукции* (вершина v^*, выбранная *до* первой итерации) выполнена, см. шаг 1 алгоритма.

Алгоритм Дейкстры: доказательство корректности (окончание)

Шаг индукции. Пусть в ходе очередной итерации выполнено присвоение $v^* = w$. Рассмотрим кратчайшую (v, w)-цепь C. Среди вершин цепи C, принадлежащих множеству S, выберем ближайшую к v и обозначим ее через y, а предшествующую ей вершину цепи — через x. (Вершины y и x с такими свойствами обязательно найдутся, поскольку $w \in S$, $v \notin S$.)

Поскольку цепь C — кратчайшая, ее начальный отрезок от v до y через x является кратчайшей (v, y)-цепью по лемме о кратчайших цепях. Следовательно, $d(v, y) = d(v, x) + \lambda(x, y)$. Вершине x значение v^* присваивалось на какой-то из предыдущих итераций. По предположению индукции, на той итерации выполнялось равенство $D(x) = d(v, x)$, а значит, после шага 2 алгоритма имеем для нового значения $D(y)$

$$D(y) \leqslant D(x) + \lambda(x, y) = d(v, x) + \lambda(x, y) = d(v, y).$$

Следовательно, $D(y) = d(v, y)$. Далее, поскольку в ходе рассматриваемой итерации выполнено присвоение $v^* = w$, в этот момент $D(w) \leqslant D(y)$. Но мы доказали, что $D(y) = d(v, y)$, а $d(v, y) \leqslant d(v, w)$ по выбору y, т. е. $D(w) \leqslant d(v, w)$, откуда $D(w) = d(v, w)$, что и требовалось.

Осталось заметить, что в ходе работы алгоритма значение $D(w)$ уменьшалось в последний раз (т. е. до $d(v, w)$) при обработке вершины $P(w)$, а значит, именно $P(w)$ — предшественник w в кратчайшей (v, w)-цепи.

Алгоритм Дейкстры: пример

Рассмотрим сеть, изображенную на рис. 1.

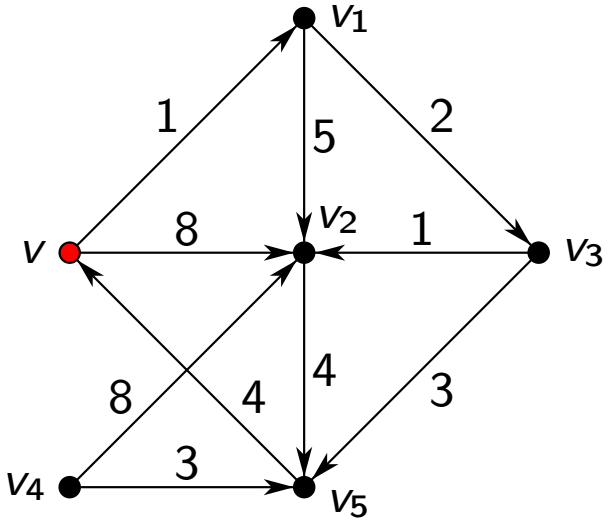

Рис. 1

Выполнение алгоритма Дейкстры описано на следующем слайде.

Алгоритм Дейкстры: пример (2)

Выполним алгоритм Дейкстры пошагово. Вначале положим $S = \{v_1, v_2, v_3, v_4, v_5\}$, $D(w) = \infty$ и $P(w) = \varnothing$ для всех $w \in S$, а также $v^* = v$ и $D(v^*) = 0$. Далее будем выполнять в цикле шаги 2 и 3.

- Итерация 1: заменим $D(v_1)$ на $D(v^*) + \lambda(v^*, v_1) = 1$, $D(v_2)$ на $D(v^*) + \lambda(v^*, v_2) = 8$, положим $P(v_1) = P(v_2) = v$, $v^* = v_1$ и $S = \{v_2, v_3, v_4, v_5\}$.
- Итерация 2: заменим $D(v_2)$ на $D(v_1) + \lambda(v_1, v_2) = 6$, $D(v_3)$ на $D(v_1) + \lambda(v_1, v_3) = 3$, положим $P(v_2) = P(v_3) = v_1$, $v^* = v_3$ и $S = \{v_2, v_4, v_5\}$.
- Итерация 3: заменим $D(v_2)$ на $D(v_3) + \lambda(v_3, v_2) = 4$, $D(v_5)$ на $D(v_3) + \lambda(v_3, v_5) = 6$, положим $P(v_2) = P(v_5) = v_3$, $v^* = v_2$ и $S = \{v_4, v_5\}$.
- Итерация 4: положим $v^* = v_5$, $S = \{v_4\}$.
- Итерация 5: положим $v^* = v_4$ и закончим работу.

В результате, мы нашли все расстояния от v и (через массив предшественников P) все цепи минимальной длины:

- $d(v, v_1) = 1$, $v \to v_1$;
- $d(v, v_2) = 4$, $v \to v_1 \to v_3 \to v_2$;
- $d(v, v_3) = 3$, $v \to v_1 \to v_3$;
- $d(v, v_4) = \infty$, цепи не существует;
- $d(v, v_5) = 6$, $v \to v_1 \to v_3 \to v_5$.

Алгоритм Дейкстры: пример (3)

Ход выполнения алгоритма Дейкстры удобно записывать в виде таблицы. Разобранному выше примеру соответствует следующая таблица:

v_1	v_2	v_3	v_4	v_5
1, v	8, v	∞, $\varnothing$	∞, $\varnothing$	∞, $\varnothing$
$*$	6, v_1	**3**, v_1	∞, $\varnothing$	∞, $\varnothing$
$*$	**4**, v_3	$*$	∞, $\varnothing$	6, v_3
$*$	$*$	$*$	∞, $\varnothing$	**6**, v_3
$*$	$*$	$*$	**∞**, $\varnothing$	$*$

В каждой строке этой таблицы для вершины w записываются значения $D(w)$ и $P(w)$ после выполнения шага 2 на очередной итерации. Если $w \notin S$, то вместо значений $D(w)$ и $P(w)$ пишется $*$. Для вершины, которая при ближайшем выполнении шага 3 будет выбрана в качестве v^*, значение $D(w)$ набрано жирным шрифтом.

После заполнения этой таблицы расстояние от v до данной вершины w выписывается автоматически — это последнее из чисел, стоящих в столбце, соответствующем вершине w, а кратчайшая (v, w)-цепь выписывается «задом наперед» по значениям P.

Задача сетевого планирования

В некоторых приложениях теории графов требуется найти цепь не минимальной, а максимальной длины между данными вершинами. Приведем одну из задач такого рода.

Задача сетевого планирования

Проект есть множество *работ* $v_1, v_2, \ldots, v_n$. Для каждой работы v_i известно время $t(v_i)$, необходимое для ее выполнения. Кроме того, дан список условий, каждое из которых имеет вид «началу работы v_i должно предшествовать выполнение работы v_j». *Выполнение проекта* состоит в выполнении всех работ с соблюдением поставленных условий. Требуется найти минимально возможное время выполнения проекта.

Представим условия задачи сетевого планирования в виде сети (G, λ). Вершинами орграфа G являются работы, а каждая дуга (v_j, v_i) означает условие выполнения работы v_j до начала v_i. Естественно считать, что этот орграф не содержит циклов (иначе выполнение проекта невозможно), а тогда в нем есть хотя бы один источник и хотя бы один сток. Добавим к G две фиктивные вершины: работу v_0 «начало проекта» и работу v_{n+1} «завершение проекта», с нулевым временем выполнения. Из вершины v_0 проведем дуги во все источники орграфа, а из всех стоков — дуги в v_{n+1}. Полученный даг с единственным источником v_0 и единственным стоком v_{n+1} и есть G. Наконец, положим $\lambda(v_j, v_i) = t(v_i)$ для всех дуг из G.

Задача сетевого планирования (2)

Определение

Сеть (G, λ), построенная по проекту по указанным правилам, называется *сетевым графиком проекта*.

Чтобы завершить моделирование задачи сетевого планирования на языке орграфов, осталось сказать на этом языке, что требуется найти в сетевом графике. Формулировка этого требования основана на следующей совсем не очевидной теореме.

Теорема о времени выполнении проекта

Минимальное время выполнения проекта равно максимальной длине цепи из источника в сток сетевого графика этого проекта.

Доказательство теоремы приведено на следующих слайдах.

Задача сетевого планирования на языке орграфов

Дана сеть без циклов с одним источником и одним стоком. Требуется найти максимальную длину цепи из источника в сток.

- Алгоритм поиска цепи максимальной длины похож (в случае дага, общий случай сложнее!) на алгоритм Дейкстры, поэтому мы не будем на нем останавливаться.

Раннее время начала и окончания

Для доказательства теоремы о времени выполнении проекта нужна подготовительная работа. Определим для каждой из работ $v_0, v_1, \ldots, v_{n+1}$ два числовых параметра.

Определение

Для произвольной работы v_i ее *раннее время начала* $\mathrm{PBH}(v_i)$ и *раннее время окончания* $\mathrm{PBO}(v_i)$ определяются следующими тремя условиями:

1. $\mathrm{PBH}(v_0) = 0$;
2. $\mathrm{PBO}(v_i) = \mathrm{PBH}(v_i) + t(v_i)$ для всех $i = 0, 1, \ldots, n+1$;
3. $\mathrm{PBH}(v_i) = \max\{\mathrm{PBO}(v_j) \mid v_j$ предшествует $v_i\}$ для всех $i = 1, \ldots, n+1$.

Из определения можно сделать два заключения:

- в любой момент времени, меньший чем $\mathrm{PBH}(v_i)$, выполнение хотя бы одной из предшествующих v_i работ не завершено;
- если выполнение каждой работы v_i начато в момент $\mathrm{PBH}(v_i)$, то к моменту $\mathrm{PBO}(v_{n+1})$ проект будет выполнен.

Таким образом, справедлива

Лемма о раннем времени

Минимальное время выполнения проекта равно $PBO(v_{n+1})$. □

Лемма о сортировке

Вершины любого дага можно занумеровать так, что любая дуга будет вести из вершины с меньшим номером в вершину с большим номером.

Доказательство. Проведем индукцию по n — числу вершин в даге G. *База индукции* ($n = 1$) очевидна.

Шаг индукции. Выберем произвольный сток v дага G (ввиду отсутствия циклов, в G есть хотя бы один сток). Орграф $G - v$ является дагом с $n - 1$ вершиной; по предположению индукции, его вершины можно занумеровать от 1 до $n - 1$ требуемым образом. Приписав вершине v номер n, получим требуемую нумерацию дага G. □

- С учетом данной леммы мы будем считать, что работы в проекте уже занумерованы так, что $j < i$ для любой дуги (v_j, v_i).

Доказательство. Так как по лемме о раннем времени минимальное время выполнения проекта равно $\mathrm{PBO}(v_{n+1})$, достаточно доказать, что для любого $i = 0, \ldots, n+1$ число $\mathrm{PBO}(v_i)$ равно максимальной длине (v_0, v_i)-цепи в сетевом графике. Проведем индукцию по i. *База индукции* ($i = 0$) очевидна.

Шаг индукции. По определению $\mathrm{PBH}(v_i)$ равно максимальному из значений $\mathrm{PBO}(v_k)$ по всем предшествующим работам; пусть этот максимум достигается на вершине v_j. Тогда $\mathrm{PBH}(v_i) = \mathrm{PBO}(v_j)$ и $\mathrm{PBO}(v_i) = \mathrm{PBO}(v_j) + t(v_i)$. По предположению индукции, максимальная длина (v_0, v_j)-цепи равна $\mathrm{PBO}(v_j)$. Добавив к этой цепи дугу (v_j, v_i), получим (v_0, v_i)-цепь длины $\mathrm{PBO}(v_j) + t(v_i) = \mathrm{PBO}(v_i)$. Эта цепь максимальна по выбору v_j, поскольку во всех (v_0, v_i)-цепях последняя дуга имеет одну и ту же длину, а значит, длины цепей сравниваются по значениям $\mathrm{PBO}(v_k)$. □

Критические работы

Распространенный вариант задачи сетевого планирования требует, помимо вычисления минимального времени выполнения проекта, получения списка «критических» работ.

Определение

Работа v_i, где $1 \leqslant i \leqslant n$, называется *критической*, если проект не может быть выполнен за минимальное время при условии что v_i начата позже своего раннего времени начала.

Критические работы можно описать при помощи следующих понятий.

Определение

Для произвольной работы v_i ее *позднее время начала* $\text{ПВН}(v_i)$ и *позднее время окончания* $\text{ПВО}(v_i)$ определяются следующими тремя условиями:

1. $\text{ПВО}(v_{n+1}) = \text{РВО}(v_{n+1})$;
2. $\text{ПВН}(v_i) = \text{ПВО}(v_i) - t(v_i)$ для всех $i = 0, 1, \ldots, n+1$;
3. $\text{ПВО}(v_i) = \min\{\text{ПВН}(v_j) \mid v_i \text{ предшествует } v_j\}$ для всех $i = 0, \ldots, n$.

Замечание

Работа v_i является критической, если и только если $РВО(v_i) = ПВО(v_i)$. □

Чтобы переформулировать задачу нахождения критических работ в терминах орграфов, нужна

Теорема о критических работах

Критическими работами являются те и только те работы проекта, через которые проходит хотя бы одна цепь максимальной длины из источника в сток сетевого графика этого проекта.

Доказательство. Выше мы доказали, что для любой вершины v_i максимальная длина (v_0, v_i)-цепи равна $\text{РВО}(v_i)$. Совершенно аналогично доказывается, что максимальная длина (v_i, v_{n+1})-цепи равна $T - \text{ПВО}(v_i)$, где $T = \text{РВО}(v_{n+1})$ — минимальное время выполнения проекта. Тогда

- если (v_0, v_{n+1})-цепь максимальной длины (эта длина равна T по теореме о времени выполнения проекта) проходит через вершину v_i, то эта цепь получена соединением (v_0, v_i)-цепи максимальной длины и (v_i, v_{n+1})-цепи максимальной длины; складывая длины этих цепей и приравнивая сумму к T, получаем $\text{РВО}(v_i) = \text{ПВО}(v_i)$, т. е. работа v_i — критическая;
- если работа v_i — критическая, то, соединяя (v_0, v_i)-цепь максимальной длины и (v_i, v_{n+1})-цепь максимальной длины, получим проходящую через v_i (v_0, v_{n+1})-цепь длины T, т. е. максимальной длины.

С учетом теоремы о критических работах, задача сетевого планирования с критическими работами на языке орграфов формулируется так:

Задача сетевого планирования с критическими работами

Дана сеть без циклов с одним источником и одним стоком. Требуется найти все цепи максимальной длины из источника в сток.

На рис. 2 приведен пример сети (G, λ), построенной по проекту, состоящему из восьми работ. Максимальные цепи из источника в сток выделены жирными линиями, критические работы — красным цветом. Минимальное время выполнения проекта равно 8.

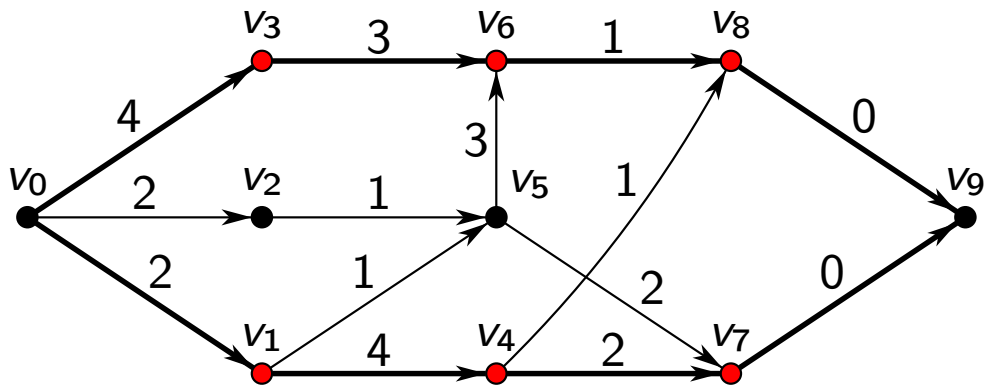

Рис. 2. Сетевой график и критические работы проекта