Матмех СПбГУ,
Кафедра исследования операций

Дискретный анализ	Дополнение 1

И. В. Романовский, Представление графа в программе: начальный ввод данных и кластер операций

1. Введение

В моем курсе дискретного анализа [1] много рассказывается о графах и их использовании при решении различных прикладных задач, но мало говорится о практической реализации алгоритмов. Здесь рассказывается о том, как графы — такой чисто геометрический объект — можно изучать в компьютерах.

Напомним, что графом называется тройка $\langle M, N, T\rangle$, где M — конечное множество, элементы которого называются *вершинами графа*, N — конечное множество, элементы которого называются *дугами графа*, а T — отображение из N в $M \times M$, сопоставляющее каждой дуге $j \in M$ упорядоченную пару вершин (i_1, i_2). Вершина i_1 называется *началом* дуги j, а i_2 — ее *концом*. Дальше параметр T будет опускаться.

Поначалу кажется странным, что такое определение может для чего-нибудь понадобиться, — какое-то несерьезное «судоку». Действительно, вспомним о первом появлении графов — в знаменитой задаче о семи кенигсбергских мостах.

Эти мосты избражены на Рис. 1[1]*. Два берега реки Прегель и два острова соединены между собой семью мостами, как показано на гравюре. Спрашивалось: можно ли пройтись по всем этим мостам, побывав на каждом ровно по одному разу. Местные интеллектуалы сами ответить на этот вопрос не смогли и спросили великого Эйлера. Он нарисовал картинку, на которой два берега были изображены четырьмя точками (помеченными буквами A, B, C, D) и соединил эти точки линиями, изображающими переходы через мосты.*

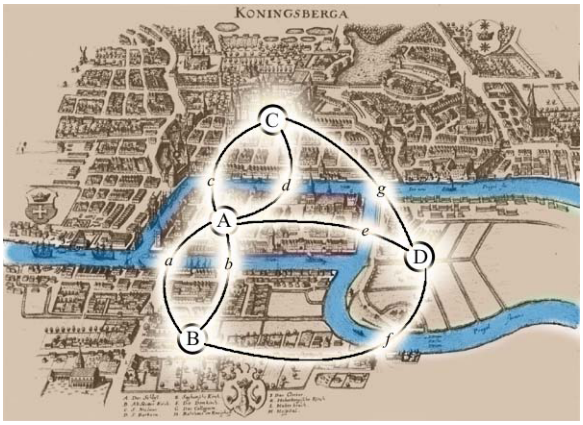

Рис. 1: Мосты в Кенигсберге и эйлеровское решение задачи

[1] Рисунок взят из `http://yak15.narod.ru/KenigsG.jpg`

Сразу стало ясно, что такая прогулка невозможна: в требуемом обходе каждая точка участвует один раз, и этим она связана с двумя переходами. А на рисунке каждая точка связывается с нечетным числом переходов. Получились чисто математические условия, которые невыполнимы.

Очень часто граф изображается примерно так, как на рис. 1.

Вершины изображаются точками, или кружками, или другими специальными значками, а дугами — линиями произвольной формы, на которых стрелка указывает направление от начала дуги к концу (на рис. 1 стрелок нет, так как изображен *неориентированный граф*, — это другая разновидность графов).

2. Задание графа в исходных данных

Чтобы задать граф нужно указать число вершин или диапазон их номеров и список дуг. Дуги удобно располагать по одной в строке в фиксированном формате. Например, можно разделять данные внутри строки пробелами, выравнивая их в столбцы. Полезно принять соглашение, что строка, начинающаяся со знака "%", является комментарием и пропускается.

Такая договоренность позволяет включить в файл дополнительную информацию. В частности можно

а) поместить в начале файла информацию о файле, его назначении, времени создания и месте хранения, имени автора и размере файла;

б) написать над столбцами имена или характеристики полей типовой строки;

в) временно удалять некоторые строки информации, "закомментировав" их.

Например, файл

```
% Пример файла, описывающего граф с 10 дугами,
%   с вершинами, нумеруемыми от 1 до 5,
%   и с дугами, у которых задано имя, начало, конец и длина. 15.04.14 И.Романовский
%  имя   начало конец  длина
   a12   1      2      7
   a14   1      4      18
   a23   2      3      16
   a25   2      5      9
   a31   3      1      -7
   a34   3      4      -4
   a41   4      1      6
   a42   4      2      9
   a54   5      4      6
   a53   5      3      1
%END    с такой строкой очень удобно проверять завершение чтения
```

Этому файлу соответствует граф, показанный на рис. 2.

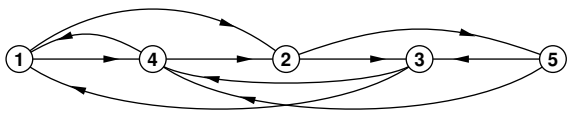

Рис. 2: Граф с 5 вершинами и 10 дугами, заданный списком дуг

В качестве упражнения, напишите около дуг их длины, взяв их из файла.

Нам пригодится еще один пример.

```
% Пример файла, описывающего граф с 10 дугами,
%   с вершинами, нумеруемыми от 1 до 6,
%   в котором не из всех вершин выходят дуги
%      составил 18.04.14   И.Романовский
%  имя   начало конец  длина
   a12   1      2      7
   a32   3      2      -7
   a13   1      3      9
   a14   1      4      12
   a34   3      4      -4
   a64   6      4      4
   a35   3      5      5
   a65   6      5      9
   a16   1      6      6
   a36   3      6      9
%END
```

Ему соответствует граф, показанный на рис. 3.

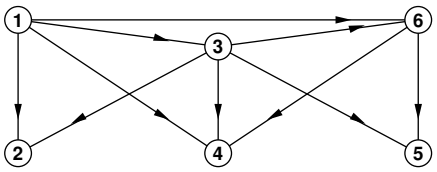

Рис. 3: Граф, заданный вторым списком дуг

3. Задание графа в программе

При выборе представления графа в программе важно иметь в виду *кластер основных используемых в данной задаче операций над графом.*[2]

Мы начнем с самого простого случая, когда для вычислений достаточно доступа к *звездам выходящих из вершин дуг.* Выходящие звезды — на рис. 2 это $N_1 = \{a12, a13, a14\}$, $N_2 = \{a23, a25\}$, $N_3 = \{a31, a34\}$, $N_4 = \{a41, a43\}$, $N_5 = \{a53\}$.

Если сгруппировать дуги в выходящие звезды, получатся тоже кластеры; их будет легко просматривать, и такое представление будет удобно при некоторых использованиях динамического программирования. Но в тех случаях, когда нужно рассматривать входящие звезды, представление полезно заменить.

[2]Слово cluster имеет латинское происхождение (средневековое) и сейчас широко используется. В математике: кластер — это класс родственных элементов статистической совокупности. В астрономии: звездный кластер (звездное скопление) — группа звезд, связанных друг с другом силами гравитации. В объектно-ориентированном программировании понятие кластер было введено также для характеризации объектов, обладающих одними и теми же программными свойствами.

Звезды выходящих дуг можно задать цепными списками, указав для каждой вершины голову ее выходящий звезды, а для каждой дуги — следующую дугу в звезде

Вершина	1	2	3	4	5
Голова списка	$a12$	$a23$	$a34$	$a41$	$a54$

Дуга	$a12$	$a14$	$a23$	$a25$	$a31$	$a34$	$a41$	$a42$	$a54$	$a53$
$next$	$a13$	nil	$a25$	nil	$a34$	nil	$a42$	nil	$a53$	nil

Можно, конечно, изобразить и граф с двумя звездами вершин — выходящей (F) и входящей (T)

Вершина	1	2	3	4	5
Голова списка F	$a12$	$a23$	$a34$	$a41$	$a54$
Голова списка T	$a31$	$a12$	$a23$	$a14$	$a25$

Дуга	$a12$	$a14$	$a23$	$a25$	$a31$	$a34$	$a41$	$a42$	$a54$	$a53$
$nextF$	$a14$	nil	$a25$	nil	$a34$	nil	$a42$	nil	$a53$	nil
$nextT$	$a42$	$a54$	$a53$	nil	$a41$	nil	nil	nil	$a34$	nil

Подготовить такие данные не трудно, даже легче, чем те данные, о которых речь пойдет сейчас. Но выполнять с ними массовые операции относительно сложно. Мы сейчас посмотрим вариант, который заметно экономнее по памяти и обеспечивает очень простой просмотр выходящей звезды любой вершины.

В этом методе составляется массив дуг, в котором слитно записываются выходящие звезды всех вершин. Выпишем их сначала так

Дуга	$a12$	$a14$	$a23$	$a25$	$a31$	$a34$	$a41$	$a42$	$a54$	$a53$
Начало	1	1	2	2	3	3	4	4	5	5
Конец	2	4	3	5	1	4	1	2	4	3

Видно, что информация в строке начал какая-то однообразная. Ее можно заменить информацией о том, с какого по какой элемент находятся дуги звезды N_i. Например, звезда N_4 находится в элементах с 7 по 8, а N_3 — с 5 по 6. У двух соседних звезд верхний предел предшествующей звезды на единицу меньше начального элемента следующей. Можно сэкономить и запоминать только один предел, например, верхний, а нижний вычислять по предыдущей звезде. Если звезда пуста, это не беда, нужно взять ее верхний предел равным верхнему пределу предшествующей звезды. Получится, например, диапазон от 7 до 6 — пустой, таких индексов нет. Что-то нужно сделать только с начальной звездой, у которой нет предшественника. Это просто — нужно сделать этого предшественника искусственно, с верхним значением, предшествующем начальному элементу массива. В примере рис. 2 мы получим

Номер звезды	0	1	2	3	4	5
Верхний предел	0	2	4	6	8	10

а в примере на рис. 3

Номер звезды	0	1	2	3	4	5	6
Верхний предел	0	4	4	8	8	8	10

4. Переход от внешнего представления графа к внутреннему

Для того, чтобы подготовить задание графа в виде массива, в котором хранятся последовательно записанные выходящие звезды дуг, проще всего воспользоваться очень экономной возможностью *многократного чтения* файла, содержащего исходные данные. Когда программа читает последовательный файл, она связывает внешнее имя файла с внутренним идентификатором, а затем открывает этот файл для чтения с начала (в Паскале это делает процедура `reset`). В любой момент можно файл открыть заново, и он будет читаться опять с начала.

Мы предлагаем читать исходный файл три раза. *При первом прочтении* определяется диапазон номеров вершин графа $nVert$ и число дуг $nArc$. Знание этих чисел позволяет разместить в оперативной памяти массивы: массив дуг $arc[1 : nArc]$ и массив разметки $lst[0 : nArc]$. Первоначально все элементы массива lst обнуляются.

При втором прочтении для каждой вершины i вычисляется число выходящих из нее дуг: просто $lst[i]$ превращается в счетчик числа дуг, выходящих из i, при прочтении каждой дуги к нужному счетчику добавляется единица.

Вычисленные так количества суммируются с накоплением и записываются в том же массив со сдвигом на одну позицию. Это будут указатели позиций выходящих звезд: при каждом использовании счетчика его нужно увеличить на 1.

Номер звезды	0	1	2	3	4	5	6
Массив счетчиков	0	4	0	4	0	0	2
Счетчики со сдвигом	0	0	4	4	8	8	8

При третьем прочтении идет заполнение массива дуг: когда обрабатывается очередная дуга из входного файла, по индексу начала находится значение его счетчика, прибавить к нему, как говорилось, единицу и по получившемуся индексу записать дугу (в нашем случае конец дугу и стоимость).

счетчики	0	1	2	3	4	5	6	дуга	1	2	3	4	5	6	7	8	9	10
0	0	0	4	4	8	8	8	no	0	0	0	0	0	0	0	0	0	0
1	0	1	4	4	8	8	8	$a12$	2	0	0	0	0	0	0	0	0	0
2	0	1	4	5	8	8	8	$a32$	2	0	0	0	2	0	0	0	0	0
3	0	2	4	5	8	8	8	$a13$	2	3	0	0	2	0	0	0	0	0
4	0	3	4	5	8	8	8	$a14$	2	3	4	0	2	0	0	0	0	0
5	0	3	4	6	8	8	8	$a34$	2	3	4	0	2	4	0	0	0	0
6	0	3	4	6	8	8	9	$a64$	2	3	4	0	2	4	0	0	4	0
7	0	3	4	7	8	8	9	$a35$	2	3	4	0	2	4	5	0	4	0
8	0	3	4	7	8	8	10	$a65$	2	3	4	0	2	4	5	0	4	5
9	0	4	4	7	8	8	10	$a16$	2	3	4	6	2	4	5	0	4	5
10	0	4	4	8	8	8	10	$a36$	2	3	4	6	2	4	5	3	4	5

(Утешьтесь тем, что вы можете смотреть эту таблицу выборочно, а мне пришлось писать ее всю — это было труднее!)

5. Использование этого представления графа в программе

Важно, что для просмотра выходящей из вершины i звезды дуг используется очень простой цикл

```
for iArc := lst[i-1]+1 to lst[i] do ...
```

Если потребуется, то дуги, выходящие из вершины i, можно переставлять внутри "их" диапазона `lst[i-1]+1..lst[i]`, например, сортируя их по какому-либо признаку или отбирая те дуги, которые еще могут быть использованы в вычислениях. Конечно, такие действия нужно делать очень осторожно.

6. Еще одно представление графа

Среди других представлений графа можно отметить представление, которое используется в алгоритмах поиска максимального потока через сеть, начиная с ранних работ Б. В. Черкасского. Сейчас в одном из самых продвинутых алгоритмов Черкасского и Голдберга [2] это представление выглядит следующим образом.

Граф $\langle M, N\rangle$ в соответствии с алгоритмом требуется "симметризовать": каждом дуге $j \in N$ сопоставляется встречная дуга $\mathrm{contr}\, j$, и условные потоки на этих дугах в сумме составляют исходную пропускную способность дуги j. Если в исходном графе уже существовала встречная дуга, то сумма условных потоков этих дуг принимается, конечно, равной сумме их исходных пропускных способностей.

Главное преимущество этого представления — быстрое выполнение пересчета потоков вдоль пути. При этом возникает и полезнное ускорение в нахождении начала произвольной дуги.

Список литературы

[1] Романовский, И. В., *Дискретный анализ*, Изд. 4-е, — СПб.: Невский диалект, БХВ-Петербург, 2008, 335 с.

[2] Cherkassky B. V., Goldberg A. V., On implementing the push-relabel method for the maximum flow problem, *Algorithmica*, 1997, 19(4): 390–410.