Акимушкин Василий Александрович,
Поздняков Сергей Николаевич

ЗАДАЧА О МАКСИМАЛЬНОМ ПАРОСОЧЕТАНИИ И ЗАДАЧА О НАЗНАЧЕНИЯХ

В этом сюжете мы познакомимся с двумя алгоритмами: один из них решает задачу оптимального разбиения на пары элементов двух различных множеств, некоторым образом связанных между собой, например, распределение имеющихся подарков так, чтобы они удовлетворили пожеланиям максимального числа одариваемых ими. Второй алгоритм решает аналогичную задачу в условиях, когда каждый подарок имеет разное значение для разных людей и эта значимость оценивается числом. Алгоритм максимизирует суммарное значение подарков для одариваемых ими людей при условии, что каждый получает не более одного подарка.

Для решения этих задач нам понадобится понятие *двудольного графа*.

Вершины такого графа делятся на две части: с одной стороны находятся элементы одного множества (например, люди), с другой стороны – элементы второго множества (например, подарки, рис. 1).

Далее от каждой вершины левой доли (от каждого человека) проводятся ребра к нескольким вершинам второй доли (конкретно к тем подаркам, которые можно подарить этому человеку, чтобы он остался доволен, рис. 2).

Теперь выбираем множество ребер так, чтобы каждое из них соединяло ровно одного человека с одним подарком (это и оз-

Рис. 1

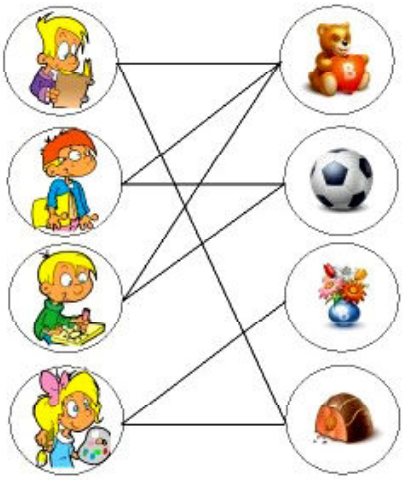


Рис. 2

Листинг 1

```
АЛГОРИТМ
ПОВТОРЯТЬ-ПОКА (есть путь от источника к стоку)
    ПОСТРОИТЬ путь
    ИЗМЕНИТЬ НАПРАВЛЕНИЯ РЕБЕР вдоль этого пути
    УДАЛИТЬ фиктивные ребра в этом пути (начальное и конечное ребра
пути)
КОНЕЦ ЦИКЛА
```

начает распределение имеющихся подарков между членами группы).

Такое множество ребер называется *паросочетанием двудольного графа*.

Первая задача состоит в том, чтобы выбрать паросочетание с максимальным количеством ребер. Это означает, что таким распределением подарков будет удовлетворено максимальное число человек.

Такое паросочетание называется *максимальным паросочетанием двудольного графа*.

Алгоритм построения максимального паросочетания

Добавим две вершины к нашему двудольному графу: одну слева от всех вершин, другую – справа. Левая вершина называется началом, или *источником*. Правая вершина называется концом, или *стоком*.

Добавим ребра, ведущие от источника ко всем вершинам левой части, и ребра, идущие от всех вершин правой части к стоку (эти две вершины и ребра носят вспомогательный характер, и их иногда называют *фиктивными*). Всем ребрам дадим направление слева направо (рис. 3).

Далее алгоритм будет последовательно перебирать пути из левой вершины в правую и делать некоторые преобразования над ребрами этих путей. При этом по ходу работы алгоритма фиктивные ребра будут удаляться (рис. 4).

По окончании работы алгоритма ребра, ориентированные справа налево, и дадут искомое паросочетание.

Особенностью алгоритма является то, что в начале работы алгоритма, как правило, находятся короткие пути – пути из трех ребер. Построение такого пути означает назначение очередному человеку подарка.

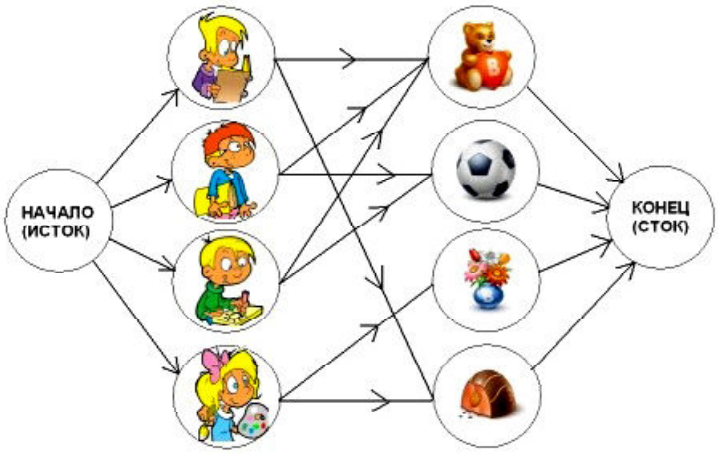


Рис. 3

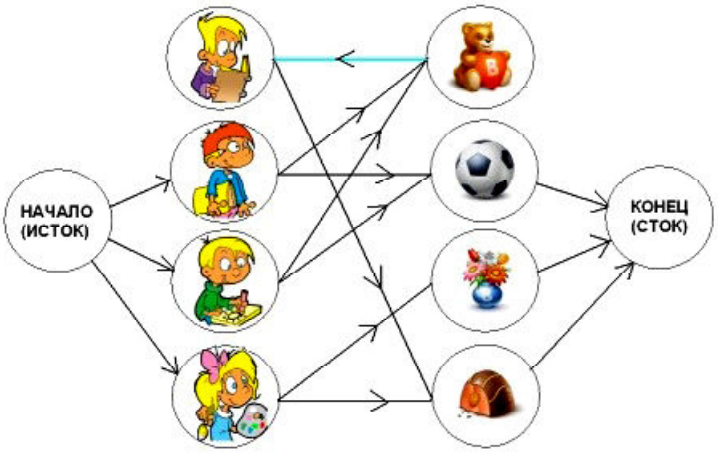


Рис. 4

Однако может оказаться, что короткого пути нет. Тогда будет строиться путь, включающий ребра, ориентированные справа налево (что означает, что они уже один раз были пройдены слева направо, рис. 5).

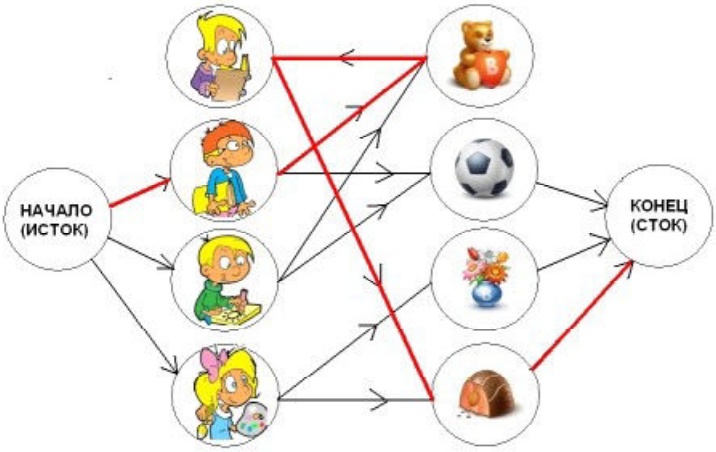


Рис. 5

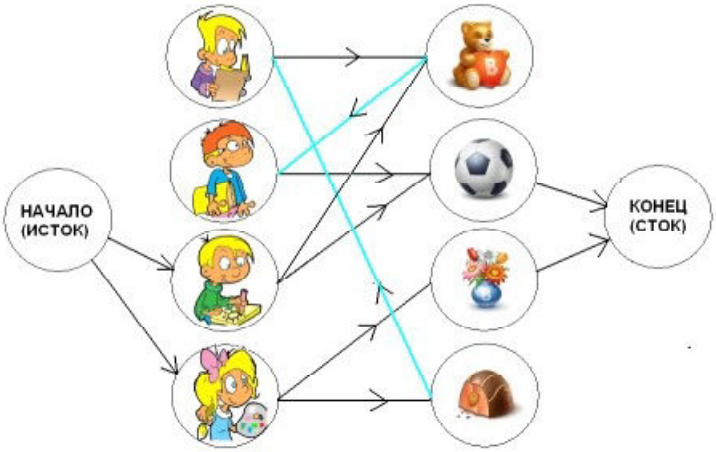


Рис. 6

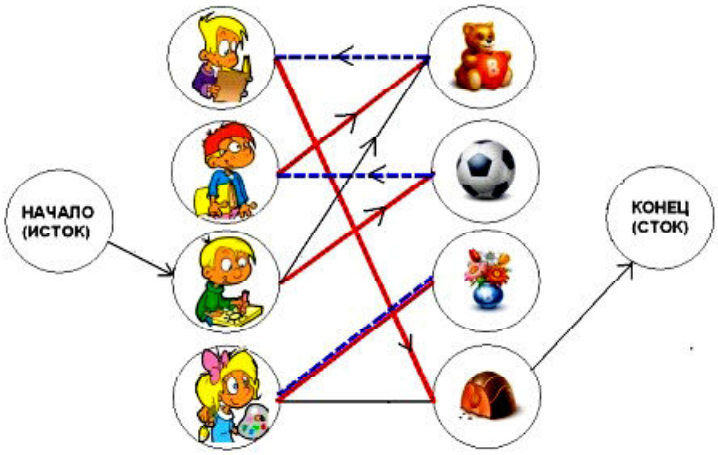


Рис. 7

После того как мы поменяем ориентацию всех ребер такого «длинного» пути, окажется, что в паросочетание входит на одно ребро больше. При этом одному новому человеку подарок будет назначен, а у остальных людей, вошедших в этот путь, подарки изменятся. Таким образом, мы частично перераспределили подарки так, чтобы большее число людей было удовлетворено подарками (рис. 6).

Докажем, *что приведенный алгоритм строит максимальное паросочетание.*

Заметим, что нахождение каждого нового пути дает нам одно новое ребро в паросочетании. Действительно, поскольку путь начинается в левой части, а заканчивается в правой, то ребер, ориентированных слева направо в нем будет на одно больше, чем ребер, ориентированных справа налево. Это и доказывает сделанное замечание.

Может ли оказаться так, что существует большее паросочетание, а наш алгоритм работу закончил, его не построив?

Докажем невозможность этого от противного. Возьмем лучшее паросочетание (ребра его отметим красным (сплошная линия)) и наложим его на паросочетание, построенное алгоритмом (его ребра отметим синим (пунктирная линия)). Часть ребер в обоих паросочетаниях совпадут, и их можно не рассматривать (на рис. 7 это нижнее отмеченное ребро).

Остальные распадутся на цепочки чередующихся красных и синих ребер (на рисунке такая цепочка одна). Так как в красном паросочетании ребер по предположению больше, найдется цепочка, в которой красных ребер больше, то есть она начинается и заканчивается красным ребром.

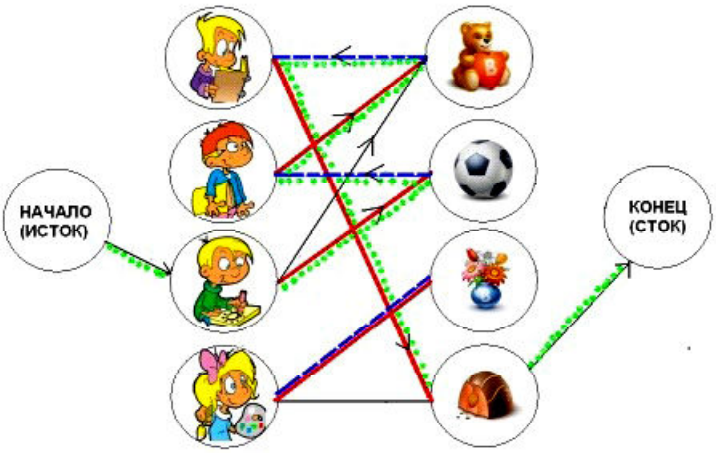


Рис. 8

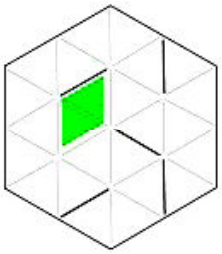


Рис. 9

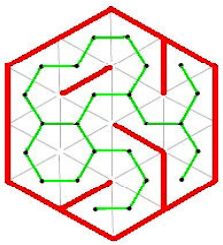


Рис. 10

Так как ни одно из синих (пунктирная линия) ребер не выходит из концов построенной цепочки, то фиктивные ребра, ведущие в начальную вершину цепочки и из конечной вершины цепочки не могли быть удалены нашим алгоритмом, а значит, найдется улучшающий путь (рис. 8, ребра красного (сплошная линия) паросочетания цепочки ориентированы слева направо, так как не вошли в найденное алгоритмом паросочетание), что противоречит тому, что наш алгоритм заканчивает работу, только когда такого пути нет! Теорема доказана.

Манипулятор задачи о паросочетании

В предлагаемом вам задании нужно применить алгоритм построения паросочетаний к построению паркета из ромбовидных плиток (с углом 60 градусов), который как можно лучше заполняет шестиугольное помещение с несколькими стенками.

Заметим, что пол этого помещения можно разделить на правильные треугольники так, что каждая плитка будет покрывать ровно два соседних треугольника (рис. 9).

Построим граф комнаты, соединив вершины соседних треугольников (если треугольники разделяет стенка, соединяющее ребро не проводится, рис. 10).

Обратим внимание на то, что граф получился двудольный. Это не так очевидно, однако легко представить шахматную раскраску треугольников, на которые мы разделили пол помещения. Тогда середины треугольников одного цвета будут вершинами одной доли графа, а вершины другого цвета образуют другую долю.

В манипуляторе предлагается «растащить» вершины разных долей в разные части полуплоскости и убедиться в том, что граф действительно двудольный (см. правую часть рис. 11).

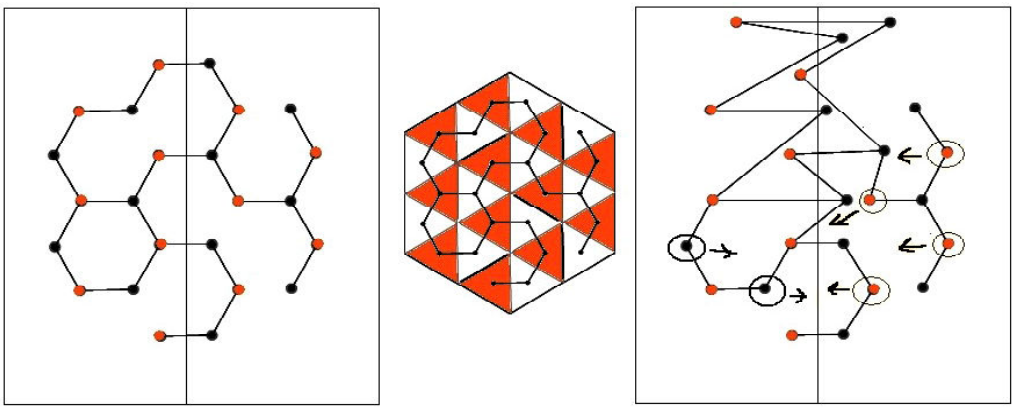


Рис. 11

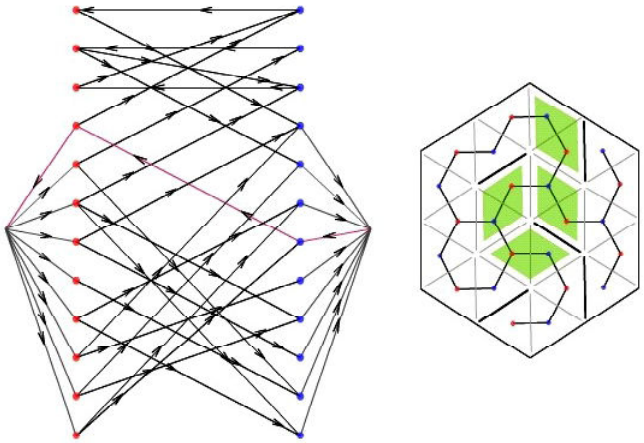


Рис. 12

После этого начинает работать уже описанный выше алгоритм. На картинке вы можете наблюдать за тем, какое расположение паркетных плиток соответствует каждому построенному паросочетанию (рис. 12).

При неправильном построении пути плитки могут наложиться друг на друга, что запрещается при конструировании паркета.

Задача о назначениях

Выше был разобран алгоритм построения максимального паросочетания на примере распределения подарков так, чтобы подходящие подарки получили как можно больше детей. На этом примере мы познакомились с понятием двудольного графа и паросочетанием – подмножеством ребер, никакие два из которых не имеют общей вершины.

Эту задачу можно усложнить, если учесть цену подарков. Так продавец желает продать подарков на максимальную сумму (а оптовый покупатель наоборот). Если добавить к условию цену, то задача о максимальном паросочетании превращается в задачу о назначениях (в исходной формулировке задачи вместо подарков были должности, на которые назначались сотрудники, а вместо цены – эффективность работы сотрудников на разных должностях; такая формулировка точнее отражает специфику задачи, так как на одну должность можно назначить только одного человека, с подарками условие «каждому ребенку не более одного подарка» является искусственным).

На рис. 13 показан двудольный граф, в котором рёбра в виде облачков соответствуют предложениям о покупке бриллиантов, а стрелки – паросочетанию, которое описывает возможное распределение бриллиантов между покупателями (рис. 13 *а*; каждому покупателю полагается не более одного бриллианта!).

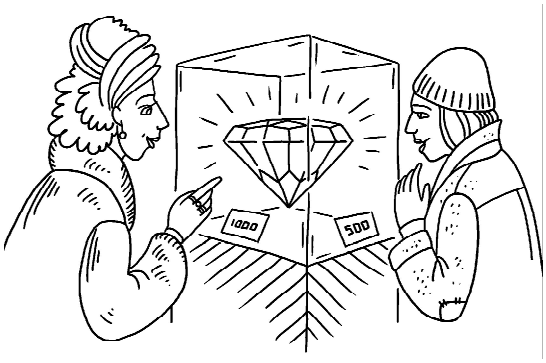
Является ли это назначение лучшим с точки зрения продавца бриллиантов?

Для ответа на этот вопрос добавим к ценам, которые предложены другими показателями знак минус (рис. 13 *б*). После этого попробуем найти замкнутую цепочку из чередующихся стрелочек и облачков, сумма чисел вдоль которой отрицательна (такая цепочка называется *отрицательным циклом*). Если нашли, то можно переназначить покупки так, чтобы общая сумма покупок возросла. Например, в левом верхнем углу рис. 13 *б* можно найти такую цепочку (рис. 14 *а*).

На рис. 14 *б* показаны веса рёбер после переназначения: если раньше сумма для этих двух покупок была равна 550, то после переназначения она стала равной 900.

Кроме операции переназначения может создаться ситуация с назначением одного и отменой другого числа покупок. Например, на рис. 15 *а* показана ситуация, когда целесообразна отмена двух покупок, а назначение только одной, которая по сумме будет больше двух предыдущих.

Мы можем свести эту ситуацию к предыдущей, если добавим недостающую «стрелку с пузырьками» с ценой 0.



а) *б)*

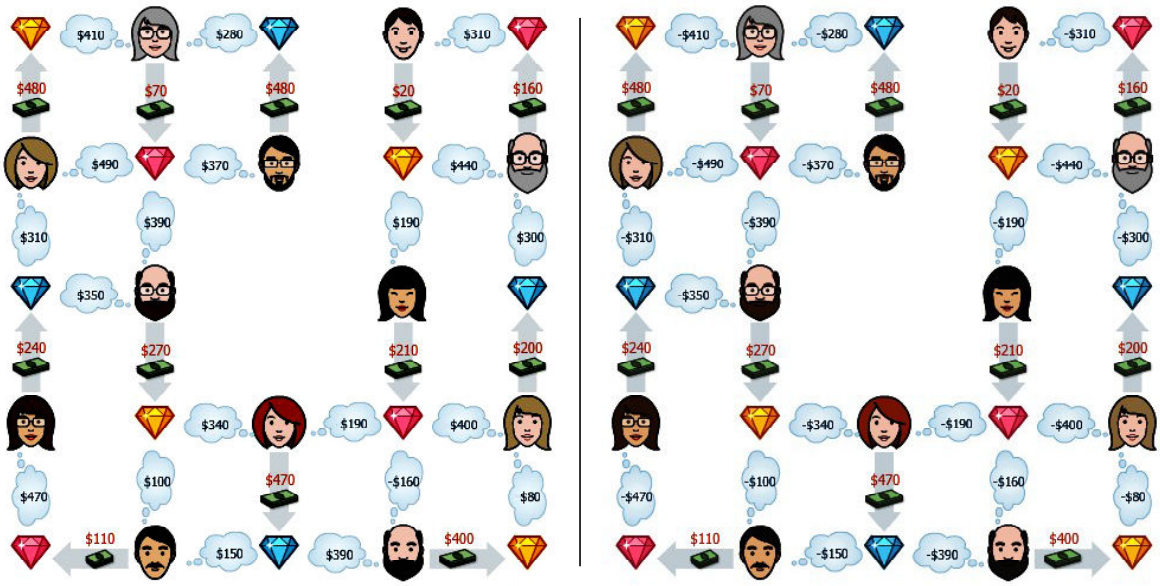


Рис. 13

Таким образом, если соединить всех покупателей со всеми бриллиантами и поставить цену покупки 0, то если покупатель не планирует покупать этот бриллиант, увеличение стоимости продаж можно вычислить алгоритмом поиска циклов отрицательной длины.

Алгоритм поиска циклов отрицательной длины мы обсудим в конце занятия. Сейчас же заметим, что добавление большого числа фиктивных стрелок (с ценой равной нулю) затруднит демонстрацию работы алгоритма на графе покупок. Поэтому улучшение назначений покупок будем вести поиском улучшающих цепочек, в которых стрелки и облачка чередуются. Какими могут быть

а) *б)*

Рис. 14

такие цепочки? Переберем все возможные варианты:

1) циклы отрицательной длины;
2) цепочки отрицательной длины, кото-

а) *б)* *в)*

Рис. 15

рые начинаются и кончаются стрелками (проходы по стрелкам означают отмену покупок);

3) цепочки отрицательной длины, которые начинаются облачком, а кончаются стрелкой, если в начало цепочки не ведёт ни одна стрелка (это означает, что бриллиант, соответствующий первой вершине, не куплен никем);

4) цепочки отрицательной длины, которые начинаются стрелкой, а кончаются облачком, если из конца цепочки не выходит ни одна стрелка (это означает, что покупатель, соответствующий последней вершине, не имел ранее покупок);

5) цепочки отрицательной длины, которые начинаются и кончаются облачком, если концы цепочки не являются началом или концом стрелок (это означает, что соответствующий началу цепочки бриллиант не был ранее куплен и что покупатель, соответствующий концу цепочки, ранее ничего не купил).

Нетрудно видеть, что этим исчерпываются все варианты улучшающих цепочек.

(*Это утверждение требует доказательства. Мы предлагаем читателям доказать это самостоятельно по аналогии с задачей о максимальном паросочетании*).

На рис. 16 и 17 показано последовательное применение правил 2, 3 и 5 для увеличения суммарной стоимости продаж. Заметим, что эти цепочки можно было объединить в одну цепочку типа 1 и сделать улучшение за один шаг.

Алгоритм нахождения цикла отрицательной длины

Взвешенный граф удобно представлять таблицей – матрицей графа. Каждой строке

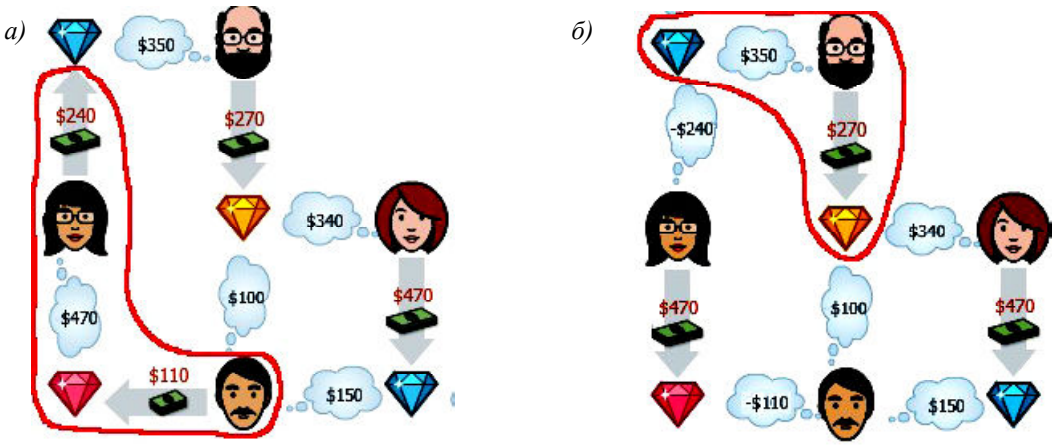


Рис. 16

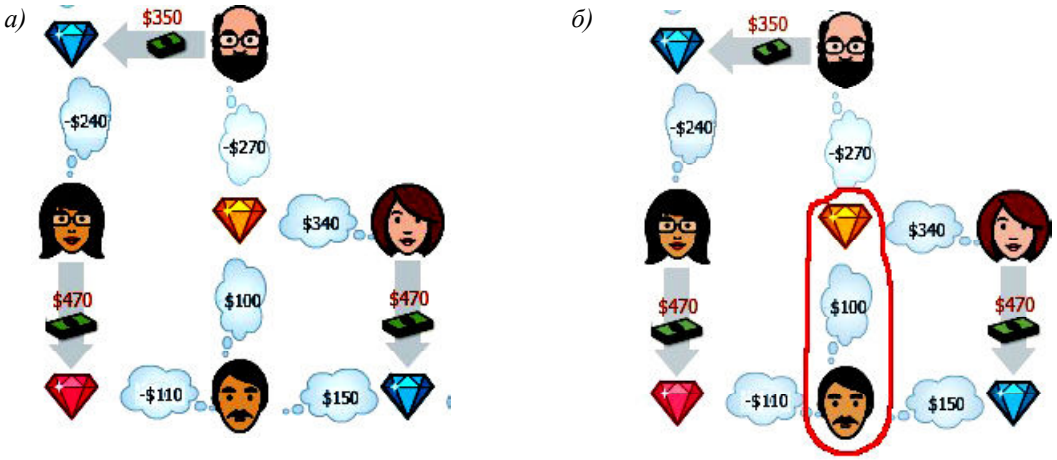


Рис. 17

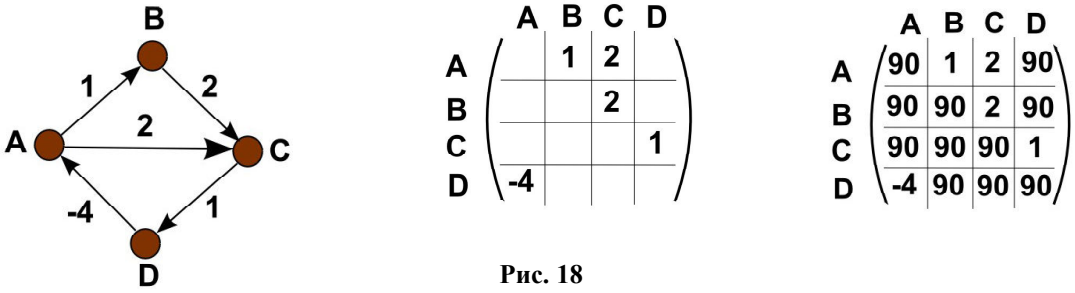


Рис. 18

и каждому столбцу соответствует вершина графа. Далее на пересечении строки A и столбца B стоит вес ребра AB и т. д. (рис. 18).

Что поставить в остальных клетках? Поскольку рёбер между остальными вершинами нет, добраться из одной такой вершины в другую невозможно, иными словами, расстояние между такими вершинами и вес соответствующего ребра как бы бесконечны. На практике же вместо «бесконечного числа» достаточно взять какое-либо большое число. Например, для нашей задачи достаточно будет числа 90.

Заметим, что числа в получившейся матрице можно трактовать как длины кратчайших путей между вершинами, состоящих из одного ребра. Теперь попробуем придумать способ, как по матрице графа узнать длины кратчайших путей, в которых может быть не только одно, но и два ребра.

Например, из вершины B в вершину D ребра нет (из-за этого длина пути из одного ребра из B в D положена «бесконечности», для которой в нашем случае взято расстояние 90). Однако из B в D можно попасть через вершину C, и тогда расстояние уменьшится до 2 + 1 = 3. В общем случае, кроме C, промежуточными могли выступить и другие вершины (в нашем случае A). Таким образом, для нахождения кратчайшего расстояния между B и D по путям из двух рёбер нужно найти минимальную из двух сумм |BC| + |CD| и |BA| + |AD|. Чтобы окончательно обобщить это правило, добавим в качестве промежуточных вершин B и D, тогда расстояние d(B;D) между B и D по путям из двух рёбер находится по формуле:

$$d(B;D) = \min\{|BA| + |AD|;\ |BB| + |BD|;\ |BC| + |CD|;\ |BD| + |DD|\} \quad (1)$$

Смысл расстояний |BB| и |DD| станет ясным, если обратить внимание на то, что было добавлено в матрицу графа. Числа на её диагонали можно интерпретировать как петли (рис. 19). В этом случае |BD| + |DD| можно интерпретировать как путь из двух ребер, первое из которых ребро BD, а второе – петля в вершине D.

Теперь можно построить матрицу расстояний по путям из двух ребер, применив формулу (1) ко всем парам вершин. Это будет довольно длинная процедура, так как в матрице графа с 4 вершинами 16 элементов, а для вычисления каждого надо произвести 4 сложения и найти минимальное из 4 чисел. Итого получится 64 сложения и 16 вычислений минимума. Такая операция в алгебре называется операцией умножения матриц. Если обозначить матрицу графа M, то полученную матрицу обозначают

$$M^2 = M \cdot M.$$

Замечание. Справедливости ради, нужно заметить, что в традиционном умножении матриц вместо операции сложения используется операция умножения, а вместо операции взятия минимума сложение. Однако это не меняет сути введенного определения.

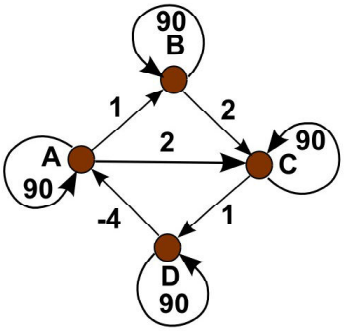


Рис. 19

После «умножения» матрицы графа на себя получим матрицу (рис. 20).

Теперь понятны дальнейшие действия по нахождению кратчайших путей: умножим полученную матрицу на исходную матрицу M, получим матрицу M^3, которая показывает кратчайшие расстояния по путям из 3 рёбер, затем ещё раз умножим результат на M и тогда числа в ней покажут расстояния по путям, состоящим из 4 рёбер, но, поскольку вершин тоже 4, такие пути будут замкнутыми, то есть, циклами в графе. Дальше умножать не нужно, так как более длинные пути обязательно будут включать циклы из 4 или меньшего числа рёбер.

Однако у этого алгоритма есть недостаток. Если сравнить длины путей из одного ребра и двух рёбер, то окажется, что для расстояния от A до C результат после умножения ухудшился: от d(A;C) = 2 мы перешли к d(A;C) = 3. Причина понятна: самые короткие пути не обязательно будут иметь большее число рёбер. Значит, на каждом шаге нужно выбирать минимальное из того расстояния, что имелось перед умножением матриц и полученного после умножения.

Так правило (1) перейдет в правило 2:

$$d(B;D) = \min\{d(B;D);\ |BA|+|AD|;\ |BB|+|BD|;\ |BC|+|CD|;\ |BD|+|DD|\} \quad (2)$$

Это означает, что после однократного умножения матриц мы получим не кратчайшие расстояния между вершинами по путям, состоящим из 2 рёбер, а по путям, состоящим из не более чем 2 рёбер. А после $n-1$ умножения – кратчайшие расстояния между вершинами по путям, состоящим из не более чем n рёбер. Этим будут исчерпаны все пути между вершинами, включая замкнутые, проходящие по каждому ребру не более одного раза, а также пути, содержащие один или более отрицательных циклов, которые могут проходить по одному ребру более одного раза.

Трудоемкость построенного алгоритма (число сложений) будет равна $n^3 \cdot (n-1)$, то есть иметь порядок роста n^4. Это не самый лучший алгоритм – алгоритм Флойда решает ту же задачу и имеет трудоемкость порядка n^3.

Упражнение. Найдите матрицу M^3.

После трёх умножений получим окончательную матрицу M^4 (рис. 21).

Наличие циклов отрицательной длины показывает появление отрицательных чисел на *главной диагонали* матрицы – диагонали из левого верхнего в нижний правый угол, на которой стоят веса замкнутых путей — циклов: d(AA), d(BB), d(CC), d(DD).

Найденный алгоритм определяет наличие циклов отрицательной длины, но не находит сами циклы. Как модернизировать его, чтобы решить исходную задачу? Заметим, что при выполнении операции нахождения минимума можно запоминать вершину, при прохождении через которую мы уменьшили расстояние. Например, на первом шаге (умножении матрицы графа на себя) при вычислении d(D;B) можно запомнить вершину C, через которую был найден короткий путь. Эти пометки можно для удобства записывать в тех же таблицах-матрицах. Тогда матрицы M, M^2, M^3, M^4 будут иметь следующий вид (рис. 22 *а*, *б*, *в*, *г* соответственно).

Если при выборе минимального значения есть несколько одинаковых минимальных значений, можно брать любой вариант, и тогда предыдущая вершина не определяется однозначно. Для определенности из нескольких вариантов будем брать первую вершину в алфавитном порядке, а если новое минимальное значение совпадает с уже полученным на предыдущем шаге, то сохраним этот вариант. Самой главной для нас будет первая матрица, имеющая отрицательные числа на главной диагонали. В нашем случае это матрица на рис. 22 *в*. У неё отрицательные числа появились в клетках AA, CC, DD. Как найти по

	A	B	C	D
A	90	1	3	3
B	90	90	2	3
C	-3	90	90	1
D	-4	-3	-2	90

Рис. 20

	A	B	C	D
A	-1	0	1	3
B	-1	0	1	3
C	-3	-2	-1	0
D	-5	-3	-2	-1

Рис. 21

а)

	A	B	C	D
A	90/A	1/A	2/A	90/A
B	90/B	90/B	2/B	90/B
C	90/C	90/C	90/C	1/C
D	-4/D	90/D	90/D	90/D

б)

	A	B	C	D
A	90/A	1/A	2/A	3/C
B	90/B	90/B	2/B	3/C
C	-3/D	90/C	90/C	1/C
D	-4/D	-3/A	-2/A	90/D

в)

	A	B	C	D
A	-1/D	1/A	2/A	3/C
B	-1/D	90/B	2/B	3/C
C	-3/D	-2/A	-1/A	1/C
D	-4/D	-3/A	-2/A	-1/C

г)

	A	B	C	D
A	-1/D	0/A	1/A	3/C
B	-1/D	0/A	1/C	3/C
C	-3/D	-2/A	-1/A	0/C
D	-5/A	-3/A	-2/A	-1/C

Рис. 22

ней, например, цикл, включающий вершину A?

На пересечении столбика A со строкой S стоит число –1, означающее что вес цикла (сумма весов рёбер, входящих в него) равен –1, и пометка D, означающая, что предпоследней в цикле была вершина D. Тогда найдем по этой же таблице-матрице, какая вершина была предпоследней на кратчайшем пути от A к D: на пересечении соответствующей строки и столбца стоит пометка C, значит, предпоследней была C, далее найдем предпоследнюю вершину на кратчайшем пути от A до C – это A. Это означает, что мы дошли до начала цикла, который запишется как ACDA, если записывать упомянутые выше вершины в обратном порядке.

Почему нужно брать первую матрицу с отрицательными числами на диагонали, а не использовать последнюю – итоговую – матрицу? Попробуйте построить цикл, содержащий вершину B. Последовательно появятся вершины B, A, D, C, C, C, … и «дойти» до вершины B не получится из-за зацикливания на цикле отрицательной длины ACDA. Такого не случится, если мы будем избавляться от первых же появившихся циклов отрицательной длины.

Замечание. При оформлении рисунков использованы картинки с сайта: http://www.clipproject.info/.

Акимушкин Василий Александрович,
аспирант математико-механического факультета СПбГУ,
программист АНО «КИО»,

Поздняков Сергей Николаевич,
доктор педагогических наук,
профессор кафедры ВМ-2 СПбГЭТУ «ЛЭТИ»,
научный руководитель Интернет-школы современной информатики и дискретной математики.