Министерство образования и науки Российской Федерации
Южно-Уральский государственный университет
Кафедра прикладной математики

519.1(07)
Э157

А. Ю. Эвнин

ЭЛЕМЕНТЫ ДИСКРЕТНОЙ ОПТИМИЗАЦИИ

Учебное пособие

Челябинск
Издательский центр ЮУрГУ
2012

УДК 519.1(076.1)
Э157

Рекомендовано учебно-методической комиссией
механико-математического факультета

Рецензенты:
доктор физ.-мат. наук *М. М. Кипнис,* ЧГПУ,
канд. физ.-мат. наук *С. М. Воронин,* ЧелГУ

Эвнин, А. Ю.

Э157 Элементы дискретной оптимизации: учебное пособие / А. Ю. Эвнин. – Челябинск: Издательский центр ЮУрГУ, 2012. – 92 с.

Учебное пособие соответствует курсу дискретной оптимизации для студентов специальностей «Прикладная математика», «Прикладная математика и информатика», «Программное обеспечение вычислительной техники и автоматизированных систем», «Программная инженерия». В книге приводятся типовые алгоритмы оптимизации на графах и в транспортных сетях; рассматриваются минимаксные теоремы, включая теорему Холла, и их приложения к различным задачам оптимизации; излагаются теория матроидов, служащая основой изучения жадных алгоритмов, и результаты по теории сложности алгоритмов.

УДК 519.1(076.1)

Оглавление

4. Сложность алгоритмов

Глава 1

Некоторые алгоритмы теории графов

В данном учебном пособии используются терминология и обозначения из книг [2, 17, 18]. Там же доказывается корректность алгоритмов из гл. 1.

1. Стягивающие деревья

Стягивающим (или **остовным**) **деревом** связного графа G называется произвольный его подграф, содержащий все вершины G и являющийся деревом.

Граф называется **взвешенным**, если каждому его ребру l поставлено в соответствие неотрицательное число $\mu(l)$ (**вес ребра**). **Вес графа** $G = \langle V, E\rangle$ — сумма весов всех его рёбер:

$$\mu(G) = \sum_{l \in E} \mu(l).$$

Существуют эффективные алгоритмы нахождения стягивающего дерева минимального веса в связном взвешенном графе.

Алгоритм Д. Краскала

Пусть $G = \langle V, E\rangle$ — исходный граф, а $T = \langle V, P\rangle$ — искомое дерево.

1. Положить $P = \varnothing$, $n = |V|$. Следующий шаг выполнять $n-1$ раз.
2. Включить в T ребро графа G наименьшего веса, обладающее тем свойством, что при добавлении его в графе T не образуется циклов. Исключить из G данное ребро.

На рис. 1 изображены взвешенный граф (числа показывают веса соответствующих рёбер) и стягивающее дерево, полученное в результате работы алгоритма.

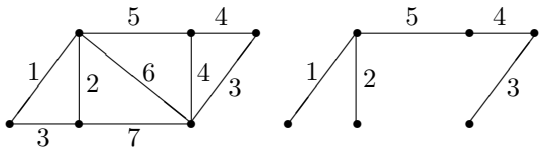


Рис. 1

Алгоритм Р. Прима

В этом алгоритме строится «разрастающееся» дерево, более точно: последовательность деревьев

$$S_1 \subset S_2 \subset \ldots \subset S_n,$$

где дерево $S_i = \langle V_i, E_i \rangle$ содержит i вершин $(i = 1, \ldots, n)$.

1. Положить $V_1 = \{x_1\}$, где $x_1 \in V$ — произвольная вершина G, $E_1 = \varnothing$. Следующий шаг выполнять для $i = 2, \ldots, n$.

2. Получить дерево S_i из дерева S_{i-1} добавлением ребра графа G наименьшего веса (среди тех рёбер, при добавлении которых к S_{i-1} вновь образуется дерево). Исключить из G данное ребро.

Опишем эффективную реализацию алгоритма Прима. На каждом шаге алгоритма каждой вершине x_i, еще не включённой в дерево, сопоставляется **пометка** — пара чисел $\langle a_i, b_i \rangle$, где b_i — наименьший вес ребра, соединяющего x_i с какой-либо вершиной, уже включённой в дерево, a_i — номер соответствующей вершины. Таким образом, $b_i = \mu(x_i a_i)$. Шаг алгоритма состоит в выборе $b_{i^*} = \min(b_i)$ и добавлении к дереву ребра $a_{i^*} x_{i^*}$. На рис. 2 дан набросок программной реализации алгоритма (с использованием конструкций языка программирования Си).

Пример. Работа алгоритма для графа, изображенного на рис. 1, показана в следующей таблице.

Шаг	1	2	3	4	5
V_r	$\{x_2, \ldots, x_6\}$	$\{x_3, \ldots, x_6\}$	$\{x_4, x_5, x_6\}$	$\{x_4, x_5\}$	$\{x_4\}$
$\langle a_2, b_2 \rangle$	$\langle x_1, 4 \rangle$	–	–	–	–
$\langle a_3, b_3 \rangle$	$\langle x_1, 4 \rangle$	$\langle x_2, 3 \rangle$	–	–	–
$\langle a_4, b_4 \rangle$	$\langle 0, \infty \rangle$	$\langle 0, \infty \rangle$	$\langle x_3, 7 \rangle$	$\langle x_6, 2 \rangle$	$\langle x_6, 2 \rangle$
$\langle a_5, b_5 \rangle$	$\langle 0, \infty \rangle$	$\langle 0, \infty \rangle$	$\langle 0, \infty \rangle$	$\langle x_6, 1 \rangle$	–
$\langle a_6, b_6 \rangle$	$\langle x_1, 5 \rangle$	$\langle x_1, 5 \rangle$	$\langle x_1, 5 \rangle$	–	–
$\min b_i$	b_2	b_3	b_6	b_5	b_4
Новое ребро	$x_1 x_2$	$x_2 x_3$	$x_1 x_6$	$x_6 x_5$	$x_6 x_4$

```
/* Алгоритм Прима нахождения минимального стягивающего дерева */

/* n ⩾ 2 - число вершин графа; */
/* V = {x_1, ..., x_n} - множество вершин; */
/* V_r - множество вершин, еще не включённых в дерево; */
/* E_s - множество рёбер строящегося дерева; */
V_r = V \ {x_1}; E_s = ∅;
/* Расстановка начальных пометок */
for(x_i ∈ V_r)
   if(x_i и x_1 смежны){a_i = x_1; b_i = μ(a_i x_i);};
   else {a_i = 0; b_i = ∞};
/* k - порядковый номер ребра, включаемого в дерево */
for(k = 1; k < n;){
   /* определение нового ребра */
   b_{i*} = min_{x_i ∈ V_r} b_i;
   E_s = E_s ∪ {a_{i*}, x_{i*}};
   V_r = V_r \ {x_{i*}};
   /* пересчёт пометок */
   if(++k < n)
     for(x_i ∈ V_r)
        if(x_i смежно с x_{i*} и μ(x_i x_{i*}) < b_i){
          b_i = μ(x_i, x_{i*}); a_i = x_{i*};
        }
}
```

Рис. 2

Сравним трудоемкость описанных алгоритмов для графа с n вершинами и m рёбрами. В первом из них основные затраты времени падают на сортировку рёбер по их весу; известно, что для выполнения сортировки m объектов требуется порядка $m \log_2 m$ операций сравнения. Во втором алгоритме на i-м шаге среди $n - i$ вершин, еще не включённых в дерево, нужно выбрать ту, чье подключение к дереву обойдется наиболее дёшево (ребро, соединяющее новую вершину с одной из «старых», должно быть наименьшего веса); для этого требуется порядка $n - i - 1$ операций. Суммируя по i, получим оценку трудоемкости алгоритма Прима: $O(n^2/2)$ операций сравнения. Понятно, что для полных графов (где число ребер $m = \frac{n(n-1)}{2}$) алгоритм Прима менее трудоемок, чем алгоритм Краскала.

2. Нахождение кратчайших путей в орграфе

Путь в орграфе — последовательность дуг вида (v_0, v_1), (v_1, v_2), $\ldots$, (v_{m-1}, v_m), которую будем записывать также в виде $v_0 \to v_1 \to \ldots \to v_m$ и называть путём из v_0 в v_m.

Пусть $G = \langle V, A \rangle$ — **взвешенный** орграф, т. е. каждой его дуге $e = (u, v) \in A$ поставлено в соответствие неотрицательное число $\mu(e) = \mu(u, v)$, называемое её **весом.** Вес пути P — сумма весов его дуг:

$$\mu(P) = \sum_{e \in P} \mu(e).$$

Кратчайший путь из вершины s в вершину t — путь минимального веса, ведущий из s в t, а **расстояние** между вершинами s и t — вес этого пути:

$$d(s, t) = \min_{P:\ s \to \ldots \to t} \mu(P).$$

Вес *тривиального*(т.е. не содержащего дуг) пути считаем равным нулю, поэтому $d(s, s) = 0$. Если не существует пути из s в t, то полагаем: $d(s, t) = \infty$.

Алгоритм Е. Дейкстры

Этот алгоритм находит кратчайшие пути от произвольной фиксированной вершины орграфа. Алгоритм представляет собой итерационную процедуру, на каждом шаге которой каждой вершине v сопоставляется пометка $l(v)$, которая является либо *постоянной* и равной при этом расстоянию $d(s, v)$ от начальной вершины s до данной вершины, либо *вре́менной* — числом, являющимся оценкой сверху для $d(s, v)$. В результате каждой итерации оценки уточняются, и при этом ровно одна временная пометка (а именно — наименьшая) переходит из разряда временных в разряд постоянных (после чего уже не меняется).

Перед первой итерацией начальная вершина имеет постоянную пометку $l(s) = 0$, у остальных вершин пометки временные и полагаются равными ∞. Итерация алгоритма состоит в просмотре вершин v с временными пометками, к которым ведут дуги из вершины p — вершины, последней получившей постоянную пометку (для первой итерации $p = s$). Пусть $a = l(p) + \mu(p, v)$. Если окажется, что $l(v) > a$, то a будет новым значением $l(v)$ (временная пометка данной вершины уменьшается).

```
/* Алгоритм Дейкстры */

/* Нахождение кратчайшего пути между двумя вершинами (s и t) */
/* V - множество вершин графа;*/
/* s - начальная вершина;     */
/* t - конечная вершина;      */
/* l - пометки вершин;        */

/* Расстановка начальных пометок */
l(s)=0;
for(v∈V) if(v!=s) l(v)=∞;
/* M - множество временных пометок */
M=V\{s};
/* p - вершина, последней получившая постоянную пометку */
p=s;

while(p!=t) {
     for(v∈ M ∩ Γ(p) ) {
        a=l(p)+μ(p,v);
        if(a<l(v)) {
          /* θ(v) - вершина, из которой идёт дуга в v*/
          /* (в кратчайшем пути)                     */
          l(v)=a; θ(v)=p;
        }
     }
     l(v*)=min_{v∈M} l(v);
          /* пометка v* становится постоянной */
     M=M\{v*}; p=v*;
}
d(s,t)=l(t); /* расстояние между s и t; */
/* кратчайший путь: s → ... → θ(θ(t)) → θ(t) → t */
```

Рис. 3

Алгоритм заканчивает работу, когда заданная конечная вершина t получает постоянную пометку.

Более детальное описание алгоритма (с использованием конструкций Си, а также математической символики) — на рис. 3.

Трудоемкость алгоритма Дейкстры. Пусть граф содержит n вершин. На каждой итерации число сложений не превышает количества временных пометок, которое в начале работы алгоритма равно $n-1$, а с каждой итерацией уменьшается на единицу. Таким образом, общее число сложений не более $(n-1)+(n-2)+\ldots+1=\frac{n(n-1)}{2}$. Операции сравнения выполняются как при пересчёте пометок, так и при нахождении минимальной временной пометки; нетрудно подсчитать, что их число не превосходит $n(n-1)$. Таким образом, трудоемкость алгоритма оценивается как $O(n^2)$ операций. Известна модификация алгоритма Дейкстры, имеющая трудоемкость $\mathrm{O}(m\log_2 n)$ (m — число дуг).

Для того, чтобы найти расстояния от заданной вершины s до *всех остальных* вершин орграфа алгоритм с рис. 3 модифицируется следующим образом:

- условие продолжения итераций `p!=t` заменяется на `M`$\neq\phi$.

Пример. Работа алгоритма для графа, изображенного на рис. 4

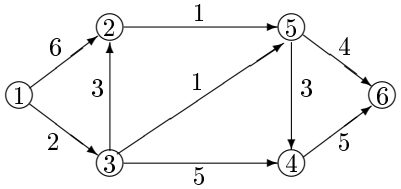


Рис. 4

(рядом с каждой дугой проставлен её вес), показана в следующей таблице (элементы таблицы — за исключением первого и последнего столбца — пометки вершин графа; в рамку заключены постоянные пометки вершин).

Итерация \вершина	1	2	3	4	5	6	p
0	**0**	∞	∞	∞	∞	∞	1
1	–	6	**2**	∞	∞	∞	3
2	–	5	–	7	**3**	∞	5
3	–	**5**	–	6	–	7	2
4	–	–	–	**6**	–	7	4
5	–	–	–	–	–	**7**	6

Анализ таблицы позволяет восстановить и сами кратчайшие пути. Найдём кратчайший путь из вершины 1 в вершину 6. Постоянная

пометка 6-й вершины равна 7, эта пометка впервые появилась в 6-м столбце, когда p равнялось 5. Значит, кратчайший путь содержит дугу $(5, 6)$. В свою очередь, постоянная пометка 5-й вершины равна 3, и эта пометка впервые появилась в 5-м столбце, когда p равнялось 3. Поэтому кратчайший путь содержит дугу $(3, 5)$. Аналогично получаем дугу $(1, 3)$. Итак, кратчайший путь из вершины 1 в вершину 6 таков: $1 \to 3 \to 5 \to 6$. Одновременно мы нашли и кратчайшие пути, ведущие из вершины 1 в вершины 3 и 5. Точно так же находятся кратчайшие пути $1 \to 3 \to 2$ и $1 \to 3 \to 5 \to 4$.

Нахождение расстояний между всеми парами вершин орграфа. В принципе, данную задачу можно решить n-кратным выполнением алгоритма Дейкстры, где n — количество вершин графа. Однако существует примерно вдвое менее трудоемкий алгоритм —

Алгоритм Р. Флойда

Строится последовательность матриц $C^{(k)} = (c_{ij}^{(k)})$, где $C^{(0)}$ — матрица весов дуг графа, т. е. для всех i и j $c_{ij}^{(0)} = \mu(v_i, v_j)$ (если в графе нет дуги (v_i, v_j), то полагаем: $\mu(v_i, v_j) = \infty$), а при $k = 1, \ldots, n$ $c_{ij}^{(k)}$ — длина кратчайшего пути из v_i в v_j такого, что в качестве промежуточных вершин могут быть лишь $v_1, v_2, \ldots, v_k$. Очевидно, что $c_{ij}^{(n)}$ — искомое расстояние между вершинами v_i и v_j. Несложно по индукции доказать следующую рекуррентную формулу, позволяющую по матрице $C^{(k-1)}$ построить матрицу $C^{(k)}$:

$$c_{ij}^{(k)} = \min(c_{ij}^{(k-1)}, c_{ik}^{(k-1)} + c_{kj}^{(k-1)}).$$

Действительно, кратчайший путь из v_i в v_j (где в качестве промежуточных вершин могут использоваться лишь $v_1, v_2, \ldots, v_k$) либо содержит вершину v_k, либо не содержит. В первом случае он имеет вес $c_{ij}^{(k-1)}$ (так как при этом промежуточными вершинами могут быть только $v_1, \ldots, v_{k-1}$), во втором — складывается из кратчайших путей из v_i в v_k и из v_k в v_j, и его вес равен $c_{ik}^{(k-1)} + c_{kj}^{(k-1)}$.

3. Потоки в сетях

Через $\Gamma(v)$ обозначим множество вершин, *к которым* ведут дуги с началом в вершине v: $\Gamma(v) = \{u \in V | (v,u) \in A\}$. Число таких дуг называют *полустепенью исхода* вершины v : $\rho^+(v) = |\Gamma(v)|$. Вершину с нулевой полустепенью исхода называют **стоком.**

Через $\Gamma^-(v)$ обозначим множество вершин, *из которых* ведут дуги к вершине v: $\Gamma^-(v) = \{u \in V | (u,v) \in A\}$. Число таких дуг называют *полустепенью захода* вершины v : $\rho^-(v) = |\Gamma^-(v)|$. Вершину с нулевой полустепенью захода называют **источником.**

Пусть $G = \langle V, A\rangle$ — орграф без петель, имеющий единственный источник (будем обозначать его v_1) и единственный сток (v_n). Все остальные вершины графа будем называть **промежуточными**. Если каждой дуге графа $a \in A$ поставлено в соответствие неотрицательное целое число $c(a)$ (**пропускная способность дуги**), то говорят, что задана **транспортная сеть** (или просто: **сеть**) $\langle G, c\rangle$.

Функция $\varphi : \ A \to \mathbb{N}_0$ (определенная на семействе дуг орграфа и принимающая неотрицательные целые значения) называется **потоком в сети** $\langle G, c\rangle$, если выполняются следующие условия:

1) для любой дуги $a \in A \ \ \varphi(a) \leqslant c(a)$;

2) для любой промежуточной вершины графа v

$$\sum_{u \in \Gamma^-(v)} \varphi(u,v) = \sum_{u \in \Gamma(v)} \varphi(v,u).$$

Величину $\varphi(a)$ будем называть **потоком по дуге** a. Таким образом, 1) поток по каждой дуге не должен превышать её пропускной способности; 2) сумма потоков по дугам, заходящим в произвольную промежуточную вершину, равен сумме потоков по дугам, исходящим из этой вершины.

Величина потока $W(\varphi)$ — сумма потоков по дугам, исходящим из источника:

$$W(\varphi) = \sum_{v \in \Gamma(v_1)} \varphi(v_1, v).$$

Она также равна сумме потоков по дугам, заходящим в сток. Поток в транспортной сети, имеющий наибольшую возможную величину, называют **максимальным потоком**. В одной и той же сети может быть несколько максимальных потоков (их величины, разумеется, должны совпадать).

Пусть φ — поток в сети $\langle G, c\rangle$. Дуга $a \in A$ называется **насыщенной**, если поток по ней равен её пропускной способности: $\varphi(a) = c(a)$.

Поток φ называется **полным**, если любой путь в орграфе G из v_1 в v_n содержит по меньшей мере одну насыщенную дугу.

В дальнейшем будем считать, что орграф $G = \langle V, A\rangle$ — *антисимметрический*, т. е. он не содержит кратных дуг и если $(u, v) \in A$, то $(v, u) \notin A$.

Максимальный поток можно найти с помощью следующего алгоритма.

Алгоритм Л. Форда – Д. Фалкерсона

1. *Построить произвольный поток* φ в сети $\langle G, c\rangle$ (можно и нулевой).

2. *Построить полный поток.* Если поток φ не полный, то в сети существует путь из v_1 в v_n, все дуги которого не насыщены. Увеличивая потоки через все дуги такого пути P на величину $\min\limits_{a\in P}(c(a) - \varphi(a))$, получаем путь, некоторая дуга которого является насыщенной. Такую операцию следует повторять до тех пор, пока не получится полный поток.

3. *Построить максимальный поток.*

   а) **Начальные пометки.** Присвоить источнику пометку 0: $l(v_1) = 0$, а остальным вершинам пометку ∞. Следующий шаг повторять до тех пор, пока в результате его выполнения не будет помечен сток v_n либо пока не перестанут появляться новые пометки.

   б) **Пересчёт пометок.** Для каждой помеченной вершины v_i, пометить символом $+i$ все непомеченные вершины v $(l(v) = \infty)$, для которых дуги (v_i, v) ненасыщенные (т. е. $\varphi(v_i, v) < c(v_i, v)$), и символом $-i$ все непомеченные вершины, для которых $\varphi(v, v_i) > 0$.

   в) **Увеличение потока.** Если сток v_n не получает новую пометку, то закончить работу алгоритма (максимальный поток найден), в противном случае существует цепь $v_0 \to \ldots \to v_n$, все вершины которой помечены. Если ориентация дуги $a = (v_i, v_j)$ совпадает с направлением прохождения цепи, будем обозначать её $\overrightarrow{a}$, в противном случае — $\overleftarrow{a}$. Если $l(v_j) = +i$ $(\overrightarrow{a} = (v_i, v_j))$, то положить $\lambda(a) = c(a) - \varphi(a)$. Если $l(v_j) = -i$ $(\overleftarrow{a} = (v_j, v_i))$, то положить $\lambda(a) = \varphi(a)$. Пусть $\varepsilon = \min(\lambda(a))$ (минимум вычисляется по всем дугам,

составляющим указанную цепь). По каждой дуге $\overrightarrow{a}$ поток увеличить на ε, а по каждой дуге $\overleftarrow{a}$ поток уменьшить на ε. (При этом величина потока в сети $W(\varphi)$ увеличится на ε). Перейти к шагу а).

Пример. На рис. 5 представлены транспортная сеть и все этапы работы алгоритма. Насыщенные дуги помечены значком "x". Начальный поток не является полным. В пути $v_1 \to v_3 \to v_4 \to v_6$

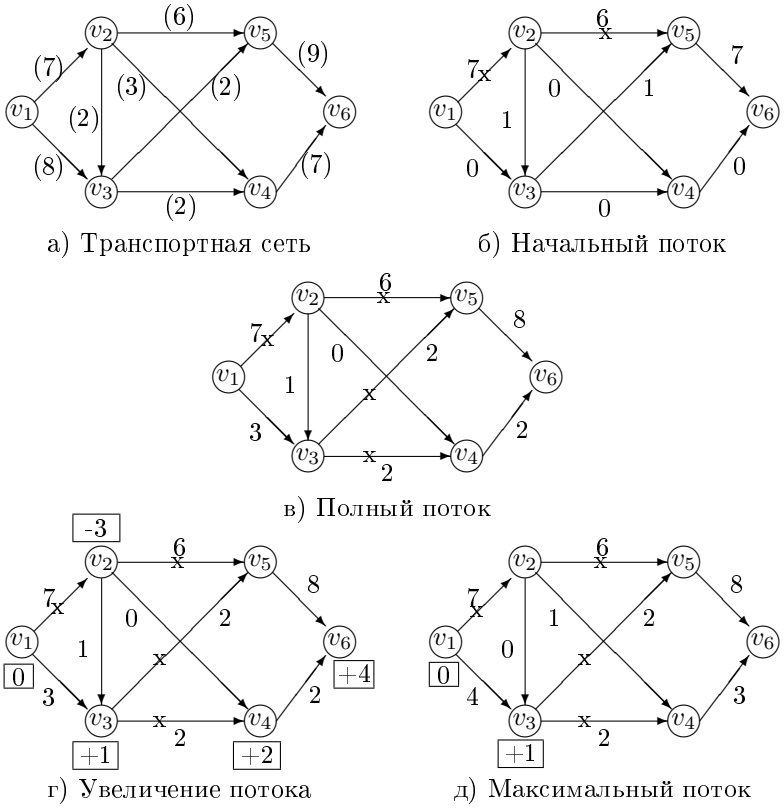


Рис. 5

все дуги не насыщены; увеличение потока, проходящего через этот путь, на 2 приводит к насыщению дуги $v_3 \to v_4$. Находим ещё один путь ($v_1 \to v_3 \to v_5 \to v_6$), состоящий из ненасыщенных дуг; увеличение потока через каждую из этих дуг на 1 делает дугу $v_3 \to v_5$ насыщенной. Теперь получен полный поток.

Переходим к 3-му этапу алгоритма Форда – Фалкерсона. Расставляем пометки вершин, как показано на рис. 5 г. Получаем цепь

$$v_1 \to v_3 \to v_2 \to v_4 \to v_6.$$

Имеем:

$$\lambda(\overrightarrow{v_1, v_3}) = 5; \quad \lambda(\overleftarrow{v_3, v_2}) = 1; \quad \lambda(\overrightarrow{v_2, v_4}) = 3; \quad \lambda(\overrightarrow{v_4, v_6}) = 5; \quad \varepsilon = 1.$$

Уменьшаем поток на 1 через дугу (v_2, v_3) и увеличиваем его на 1 через остальные дуги этой цепи. При повторении процедуры (рис. 5, д) удаётся пометить только две вершины (v_1 и v_3). Максимальный поток найден: $W(\varphi) = 11$.

Разрез в транспортной сети — множество дуг, после удаления которых в сети не будет пути из источника в сток. **Пропускная способность разреза** — сумма пропускных способностей составляющих его дуг. **Минимальный разрез** — разрез с минимальной пропускной способностью.

Теорема 1. (Л. Форд – Д. Фалкерсон, 1956 г.). *Величина максимального потока в сети равна пропускной способности минимального разреза.*

Один из минимальных разрезов составляют дуги, ведущие из помеченных вершин в непомеченные на последнем этапе работы алгоритма Форда – Фалкерсона. В рассмотренном выше примере минимальный разрез образуют дуги v_1v_2, v_3v_4, v_3v_5.

Глава 2

Паросочетания

Сначала приведём формулировки двух задач, первая из которых имеет практический характер, а вторая — скорее шуточный.

1. Пусть имеется несколько работников и несколько видов работ. Известно, какие работы каждый работник может выполнять. Можно ли так распределить работы между работниками, чтобы каждый выполнял какую-либо работу (при этом на каждую работу назначается не более одного работника)?

2. Имеется множество юношей, каждый из которых знаком с некоторыми девушками. При каких условиях можно одновременно женить всех юношей так, чтобы каждый из них женился на знакомой ему девушке?

С точки зрения математики эти задачи не отличаются друг от друга, поскольку их математическая модель одна и та же: выделение совершенного паросочетания в двудольном графе. *Совершенное паросочетание* — это множество попарно не смежных рёбер, покрывающих все вершины фиксированной доли двудольного графа. В первой задаче такую долю составляют работники, а во второй — юноши. Вторая доля в этих задачах состоит соответственно из работ и девушек. Наконец, произвольное ребро uv графа первой задачи описывает ситуацию «работник u может выполнять работу v», а графа второй задачи — «юноша u знаком с девушкой v».

В историю математики задача о существовании совершенного паросочетания в двудольном графе вошла под именем «задачи о свадьбах» (формулировка задачи 2 оказалась хорошо запоминающейся), а ответ к задаче дан Филипом Холлом (см. §1) в 1935 г.

В 20–50-х годах XX века были независимо друг от друга доказаны несколько теорем дискретной математики, носящих *минимаксный характер*: Форда – Фалкерсона (см. §3), Кёнига – Эгервари (§2), Дилворта (в другом написании — Дилуорса; §3), а также К. Менгера. В каждой из этих теорем утверждалось, что максимум одной величины совпадает с минимумом другой величины. Как выяснилось впоследствие, указанные теоремы, а также теорема Холла эквивалентны друг другу: каждая из них может быть выведена из любой другой. Было

также установлено, что эти теоремы являются проявлением принципа двойственности в линейном программировании.

В комбинаторном анализе и теории сложности вычислений (как и вообще в математике) важнейшей является **идея сведения** одной задачи к другой. Поэтому интересно показать, как внешне различные утверждения переходят друг в друга в результате построения некоторых дополнительных конструкций. Понимание таких переходов и преобразований важно и для программистов: зная, как одна задача сводится к другой, можно алгоритм решения первой задачи преобразовать в алгоритм решения второй задачи. В данной главе мы приводим наиболее простое из известных доказательств теоремы Холла, из теоремы Холла выводим теорему Кёнига – Эгервари, а из той, в свою очередь, теорему Дилворта.

Рассматриваются также некоторые приложения минимаксных теорем, в том числе числе теорема о возможности достраивания латинского прямоугольника до латинского квадрата (результат, полученный Маршаллом Холлом в 1945 г.) и теорема о *рёберно-хроматическом числе* двудольного графа (это — наименьшее число паросочетаний, на которое распадается множество рёбер графа).

Заключительные параграфы главы посвящены алгоритмам решения задач о двудольных графах. Базовой здесь является задача нахождения наибольшего (по мощности) паросочетания в двудольном графе. Излагаемый алгоритм её решения позволяет попутно найти наименьшее *вершинное покрытие* и наибольшее *независимое множество вершин*.

Вернёмся теперь к задаче 1 и рассмотрим следующее её (в некотором смысле) обобщение.

3. Пусть имеется несколько работников и столько же видов работ. Известна стоимость выполнения каждым работником каждой работы. Требуется так распределить работы между работниками, чтобы суммарная стоимость выполнения работ была наименьшей.

Это — один из вариантов **задачи о назначениях**. Её математическая формулировка: найти совершенное паросочетание наименьшего веса во взвешенном двудольном графе. Мы приводим алгоритм решения этой задачи, названный **венгерским** — в честь венгра Г. Куна, придумавшего это алгоритм в 1955 г.

1. Теорема Холла

Введём некоторые определения и обозначения.

Паросочетанием в графе называют множество попарно несмежных рёбер.

Двудольный граф G с фиксированным разбиением множества вершин на доли V_1 и V_2 будем обозначать $G(V_1, V_2)$.

Пусть $G(V_1, V_2)$ — двудольный граф. **Совершенным паросочетанием из V_1 в V_2** называется паросочетание в G, покрывающее V_1 (т.е. для всякой вершины из V_1 найдётся в паросочетании инцидентное ей ребро).

Пусть $A \subset V$ — подмножество вершин графа $G = \langle V, E \rangle$. Окружением множества A называют множество

$$\Gamma(A) = \bigcup_{v \in A} \Gamma(v) \setminus A,$$

где $\Gamma(v)$ — множество вершин, смежных с v.

Теорема 2. (Ф. Холл, 1935 г.) *Совершенное паросочетание из V_1 в V_2 в двудольном графе $G(V_1, V_2)$ существует тогда и только тогда, когда*

$$\forall A \subset V_1 \quad |\Gamma(A)| \geqslant |A|.$$

Теорема Холла даёт решение следующей задачи.

Задача о свадьбах. *Имеется множество юношей, каждый из которых знаком с некоторыми девушками. При каких условиях можно одновременно женить всех юношей так, чтобы каждый из них женился на знакомой ему девушке?*

Действительно, построим двудольный граф $G(V_1, V_2)$, в котором V_1 есть множество юношей, а V_2 — соответственно множество девушек, а знакомые юноши и девушки соединены рёбрами. Тогда одновременно женить всех юношей означает найти в данном графе совершенное паросочетание из V_1 в V_2. Ответ на вопрос задачи о свадьбах можно сформулировать так: *задача разрешима тогда и только тогда, когда любые k юношей из данного множества знакомы в совокупности не менее чем с k девушками.*

Доказательство теоремы Холла. *Необходимость* очевидна. Действительно, условия $|A| = k$ и $|\Gamma(A)| < k$ означают: некоторые k вершин из V_1 смежны в совокупности менее чем с k вершинами из V_2 — поэтому нет попарно несмежных ребер в $G(V_1, V_2)$,

покрывающих вершины даже из $A \subset V_1$, тем более нет совершеннного паросочетания из V_1 в V_2.

Достаточность будем доказывать индукцией по числу юношей. Пусть всего имеется m юношей. *База индукции* ($m = 1$) очевидна: уж одного-то юношу, знакомого хотя бы с одной девушкой, поженить можно.

Индукционный шаг. Пусть утвержение теоремы выполняется, если юношей меньше m, и докажем его для m юношей. Рассмотрим два возможных случая.

I. *Любые h юношей ($h < m$) в совокупности знакомы не менее чем с $h+1$ девушками.* Если в этом случае мы пожениим какого-то (любого) юношу на любой знакомой ему девушке, то для любых k оставшихся юношей общее число знакомых им (незамужних) девушек или не изменится, или уменьшится на единицу — значит, не превзойдёт числа k. По предположению индукции оставшихся $m-1$ юношей поженить можно.

II. *Некоторые h юношей ($1 \leqslant h < m$) знакомы в совокупности ровно с h девушками.* По предположению индукции их можно поженить. Попробуем устроить судьбу оставшихся $m-h$ юношей. Если какие-то k юношей из их числа знакомы в совокупности менее чем с k девушками, то вместе с указанными h юношами они составят группу из $k+h$ юношей, знакомых менее чем с $k+h$ девушками, что противоречит условию теоремы. Таким образом, и к оставшимся $m-h$ юношам можно применить предположение индукции. □

2. Венгерская теорема

Пусть имеется двоичная матрица (т. е. состоящая из нулей и единиц). Единичные элементы этой матрицы, никакие два из которых не лежат в одной строке или одном столбце, назовём **независимыми**. Строки и столбцы матрицы, содержащие в совокупности все её единицы, назовём **покрывающими линиями**.

Например, для матрицы

$$\begin{pmatrix} \boxed{1} & 1 & 1 & 0 & 0 & 0 \\ 1 & \boxed{1} & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & \boxed{1} & 0 & 0 & 0 \\ 1 & 0 & 0 & \boxed{1} & 1 & 1 \end{pmatrix}$$

можно указать четыре независимые единицы (они заключены в прямоугольники), и не больше, а минимальное по мощности множество покрывающих линий составляют первые три столбца и последняя строка.

Теорема 3. (Д. Кёниг – Э. Эгервари, 1931 г.) *Наибольшее число независимых единиц двоичной матрицы равно наименьшему числу покрывающих линий.*

Дадим сразу перевод условия теоремы на язык теории графов. Пусть в двоичной матрице $A = (a_{ij})$ m строк и n столбцов. Такая матрица задаёт двудольный граф $G(V_1, V_2)$, где $V_1 = \{v_1, \ldots, v_m\}, V_2 = \{u_1, \ldots, u_n\}$ и в графе есть ребро $v_i u_j$ тогда и только тогда, когда $a_{ij} = 1$. Матрицу A можно рассматривать как матрицу смежности двудольного графа $G(V_1, V_2)$. Независимые единицы матрицы соответствуют попарно несмежным рёбрам, т. е. паросочетанию графа, а покрывающие линии — **вершинному покрытию** графа — множеству таких его вершин, что всякое ребро графа инцидентно хотя бы одной из них.

На рис. 6 представлен граф, соответствующий матрице из примера, приведённого выше. Рёбра $v_1u_1, v_2u_2, v_4u_3, v_5u_4$ составляют максимальное паросочетание, а вершины u_1, u_2, u_3, v_5 образуют минимальное вершинное покрытие.

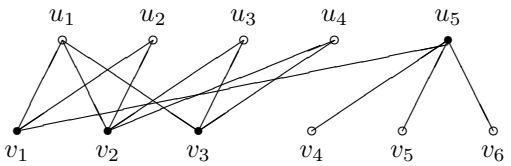


Рис. 6

Итак, венгерская теорема говорит о том, что
в двудольном графе наибольшая мощность паросочетания равна наименьшей мощности вершинного покрытия.

Доказательство венгерской теоремы. Обозначим через r наибольшее число независимых единиц, а через s наименьшее число покрывающих линий. Пусть все единицы матрицы находятся в некоторых x строках и y столбцах, причём $x + y = s$. Заметим, что если какие-то две единицы лежат (или не лежат) на одной линии, то это свойство сохранится и при любой перестановке строк или столбцов.

(В графовой модели задачи перестановки линий сводятся к перенумерованию вершин в долях графа). Поэтому при указанных преобразованиях мощности интересующих нас множеств не меняются. Значит, можно считать, что все единицы расположены в первых x строках и последних y столбцах, а матрица имеет следующий блочный вид:

$$\left(\begin{array}{c|c} A & B \\ \hline O & C \end{array}\right).$$

Рассмотрим двудольный граф $G'(V_1', V_2')$, который задаётся подматрицей A. В долях этого графа соответственно x и $n-y$ вершин. Если все единицы из каких-либо k строк матрицы A покрываются меньшим числом столбцов, то эти k строк можно заменить указанными столбцами и получить меньшее, чем $s = x + y$, число линий, покрывающих единицы матрицы, что приводит нас к противоречию с предположением о минимальности числа s. Таким образом, любые k вершин из доли V_1' смежны в совокупности не менее чем с k вершинами доли V_2' — выполнены условия теоремы Холла! Стало быть, в графе $G'(V_1', V_2')$ существует совершенное паросочетание из V_1' в V_2', то есть в подматрице A есть x независимых единиц.

Аналогичное рассуждение, применённое к подматрице C доказывает, что C содержит y независимых единиц.

Объединив независимые единицы из A и C, получим $x + y$ независимых единиц. Значит, максимальное число независимых единиц r не меньше числа $s = x + y$. С другой стороны, r независимых единиц нельзя покрыть менее чем r линиями (так как каждая линия покрывает не более одной независимой единицы), то есть r не больше s — минимального числа покрывающих линий. Итак, $r = s$.□

Терминологическое замечание. Наибольшее число независимых единиц двоичной матрицы называют её **словарным рангом**.

3. Теорема Дилворта

Сначала введём (или напомним) некоторые определения и обозначения. Пусть на некотором конечном множестве A введено **отношение порядка** $\prec$, т.е. отношение, обладающее свойствами **транзитивности** (состоящее в том, что если для элементов a, b и c множества A выполняются условия $a \prec b$ и $b \prec c$, то $a \prec c$) и **антисимметричности** (если $a \prec b$ и $b \prec a$, то $a = b$; другими словами, не существует двух различных элементов a и b, для которых одновременно выполняются условия $a \prec b$ и $b \prec a$). Всем известными примерами отношений

порядка являются отношения $<, \leqslant, >, \geqslant$, заданные на любом подмножестве числовой прямой, и отношение делимости на множестве натуральных чисел. Множество вместе с введённым на нём отношением порядка называют **упорядоченным**.

Подмножество упорядоченного множества называют **цепью**, если в нём любые два элемента сравнимы по данному отношению, и **антицепью**, если никакие два различные элемента этого подмножества не сравнимы по данному отношению. Например, в множестве $\{2, 3, 6, 7, 55\}$, упорядоченном отношением делимости, есть цепи $\{2, 6\}$, $\{3, 6\}$ и антицепи $\{2, 3, 7, 55\}$, $\{6, 7, 55\}$. Одноэлементное множество — одновременно и цепь, и антицепь. Число элементов цепи (антицепи) будем называть её **длиной**.

Очевидно, что любая цепь пересекается с любой антицепью не более чем по одному элементу. Отсюда следует, что если в упорядоченном множестве A имеется антицепь длины m, то число цепей, **покрывающих** A (т.е. их объединение есть A), не может быть меньше m.

Таким образом, *наименьшее количество цепей, покрывающих A, не меньше наибольшей длины антицепи.* Как показывает следующая теорема, на самом деле всегда имеет место равенство упомянутых величин.

Теорема 4. (Р. Дилворт, 1950 г.) *Минимальное число непересекающихся цепей, покрывающих упорядоченное множество, равно максимальной длине антицепи.*

Доказательство. Обозначим наименьшее количество покрывающих цепей через X, а наибольшую длину антицепи через Y. Уже установлено, что $X \geqslant Y$. Покажем, что имеет место и неравенство $X \leqslant Y$.

Пусть $A = \{a_1, a_2, \ldots, a_n\}$ — упорядоченное множество с отношением $\prec$. Построим двудольный граф $G = \langle V_1 \cup V_2, E\rangle$, где $V_1 = \{x_1, x_2, \ldots, x_n\}$, $V_2 = \{y_1, y_2, \ldots, y_n\}$ и $x_i y_j \in E \iff (a_i \prec a_j, i \neq j)$. В этом графе вершины x_i и y_i представляют элемент a_i исходного множества, а наличие ребра $x_i y_j$ свидетельствует о том, что i-й и j-й элементы множества A находятся в заданном отношении. Таким образом, построенный граф даёт графическое представление (на то он и граф!) упорядоченного множества.

Возьмём наибольшее (по мощности) паросочетание в этом графе. Пусть его мощность равна k. Это паросочетание порождает $m = n - k$

цепей, покрывающих множество A. Покажем, как их получить. Сначала имеем n одноэлементных цепей, соответствующих n элементам A. Каждое ребро паросочетания позволяет объединить две цепи в одну (например, ребро $x_i y_j$ объединяет цепи $\ldots \to a_i$ и $a_j \to \ldots$ в цепь $\ldots \to a_i \to a_j \to \ldots$). В итоге и получим m цепей, которые не пересекаются и покрывают множество A. Отсюда следует, что *наименьшее количество покрывающих цепей X не больше m.*

Теперь рассмотрим какое-либо минимальное вершинное покрытие графа

$$T = \{x_{i_1}, x_{i_2}, \ldots, x_{i_r}, y_{j_1}, y_{j_2}, \ldots, y_{j_s}\}.$$

Покажем, что индексы всех вершин в указанном множестве различны, то есть $\forall t, l \;\; i_t \neq j_l$. Ведём рассуждение от противного. Предположим, что для каких-то t и l имеет место равенство $i_t = j_l = h$. Существует ребро вида $x_h y_c$, где
$y_c \notin T$, так как в противном случае множество $T \setminus \{x_h\}$ также является вершинным покрытием, что противоречит минимальности T. Аналогично, существует и ребро вида $x_d y_h$ для некоторой вершины $x_d \notin T$. Для указанных c и d имеем $h \prec c$ и $d \prec h$, откуда в силу транзитивности $d \prec c$. Значит, в графе есть ребро $x_d y_c$, соединяющее вершины, не принадлежащие множеству T. Но тогда T не является вершинным покрытием — противоречие!

Таким образом, вершины множества T представляют в точности $r + s$ элементов множества A.

Венгерская теорема, напомним, говорит о том, что наибольшая мощность паросочетания равна минимальной мощности вершинного покрытия; в наших обозначениях получаем $k = r + s$. Заметим теперь, что элементы, не представленные в множестве T, попарно несравнимы, т.е. образуют антицепь, а число таких элементов равно

$$n - (r + s) = n - k = m.$$

Из существования антицепи мощности m вытекает, что *максимальная длина антицепи Y не меньше m.*

Итак, $X \leqslant m \leqslant Y$. Вспоминая, что $X \geqslant Y$, окончательно имеем: $X = m = Y$. $\square$

Замечание. Доказательство теоремы даёт алгоритм нахождения минимального цепного покрытия, основанный на выделении наибольшего паросочетания в соответствующем двудольном графе.

Теорема, двойственная к теореме Дилворта

Если в формулировке теоремы Дилворта заменить слово "цепи" на "антицепи", а "антицепь" на "цепь", то получим также верное утверждение, которое доказывается очень просто.

Теорема 5. *Минимальное число непересекающихся антицепей, покрывающих упорядоченное множество A, равно максимальной длине цепи.*

Доказательство. Пусть m — максимальная длина цепи, а P — цепь длины m. Так как любая антицепь имеет с цепью P не более одного общего элемента, исходное множество A покрывает не менее m антицепей. Обозначим через $l(a)$ наибольшую длину цепи, начинающейся с элемента a (то есть элемент a предшествует всем другим элементам данной цепи). Очевидно, если $a \prec b$, то $l(a) > l(b)$. Таким образом, элементы a и b, для которых $l(a) = l(b)$, не сравнимы, а множество

$$A_i = \{a \mid l(a) = i\}$$

является антицепью. Осталось заметить, что в наших предположениях функция l на элементах множества P принимает все значения от 1 до m, и m — наибольшее значение функции l. Значит, имеем m непересекающихся антицепей $A_1, A_2, \ldots, A_m$, покрывающих упорядоченное множество A. □

Из доказанной теоремы вытекает следующий интересный факт.

Теорема 6. *Если упорядоченное множество A содержит не менее $(p-1)(q-1)+1$ элементов, то в нём существует цепь длины не менее p или антицепь длины не менее q.*

Доказательство. Если в множестве A нет цепи длины p, то есть максимальная длина цепи не превосходит $p-1$, то, согласно предыдущей теореме, найдётся не более $p-1$ антицепей, покрывающих множество A. Если бы каждая из них содержала не более $q-1$ элементов, то в их объединении было бы не более $(p-1)(q-1)$ элементов, и они не покрывали бы множество A. Стало быть, длина некоторой антицепи не меньше числа q. □

Замечание. Теорема 6 легко выводится и из самой теоремы Дилворта, но, как уже заметил читатель, она доказывается не столь просто, как двойственная теорема.

В последующих параграфах данной главы мы рассмотрим некоторые приложения минимаксных теорем.

4. Совершенные паросочетания в регулярных двудольных графах

С помощью теоремы Холла получим результаты, связанные с темой параграфа.

Теорема 7. *В любом непустом регулярном двудольном графе $G(V_1, V_2)$ существует совершенное паросочетание.*

Доказательство. Пусть степень каждой вершины графа равна $q > 0$. Возьмём произвольные k вершин первой доли $b_1, b_2, \ldots, b_k$ и смежные с ними вершины $g_1, g_2, \ldots, g_l$ второй доли и рассмотрим порождённый этими $k + l$ вершинами подграф исходного графа. В полученном графе степень любой вершины из первой доли равна q, а из второй — не больше q. Число рёбер двудольного графа равно сумме степеней вершин любой из его долей. Поэтому

$$\sum_{i=1}^{k} \rho(b_i) = kq = \sum_{j=1}^{l} \rho(g_j) \leqslant lq,$$

откуда $l \geqslant k$. Таким образом, выполнены условия теоремы Холла о существовании в двудольном графе совершенного паросочетания. □

Следствие. *Любой непустой регулярный двудольный граф распадается на совершенные паросочетания.*

Доказательство. Пользуясь теоремой, будем последовательно одно за другим выделять в графе непересекающиеся паросочетания. □

5. Дважды стохастические матрицы

Числовая матрица называется **дважды стохастической**, если её элементы неотрицательны, и в каждой строке и каждом столбце сумма элементов равна единице.

Сумма всех элементов такой матрицы равна, с одной стороны, числу её строк, а, с другой стороны, числу столбцов. Значит, дважды стохастическая матрица является квадратной.

Подстановочная матрица состоит только из нулей и единиц, причём в каждой строке и каждом столбце содержится ровно одна единица. Ясно, что подстановочная матрица является дважды стохастической.

Подстановочным множеством матрицы называется множество её элементов, содержащее по одному элементу из каждой строки и каждого столбца.

Теорема 8. *Всякая дважды стохастическая матрица $A = (a_{ij})$ имеет подстановочное множество, состоящее из ненулевых элементов.*

Доказательство. Пусть матрица A имеет размер $n \times n$. Минимальное число линий, содержащих все ненулевые элементы матрицы A, равно n, поскольку сумма элементов всей матрицы равна n, а каждой линии — 1. Рассмотрим матрицу B, которая получается из матрицы A заменой ненулевых элементов на единицы. Минимальное число покрывающих линий матрицы B — такое же, как и для матрицы A, то есть n. По венгерской теореме в матрице B есть n независимых единиц. Эти единицы задают подстановочное множество матрицы A. □

Очевидно, что подстановочное множество существует и для матрицы из неотрицательных элементов с одинаковыми суммами по строкам и столбцам.

Теорема 9. (Г. Биркгоф, 1946 г.) *Всякая дважды стохастическая матрица представима в виде выпуклой комбинации подстановочных матриц.*

Доказательство. Пусть P_1 — подстановочная матрица, порождённая подстановочным множеством M_1 из ненулевых элементов дважды стохастической матрицы A, а число c_1 — наименьший элемент в M_1. Тогда матрица $A - c_1P_1$ состоит из неотрицательных элементов и имеет одинаковые суммы по строкам и столбцам, и в ней ненулевых элементов меньше, чем у исходной матрицы. Повторяя данное преобразование, через конечное число шагов приходим к равенству

$$A = c_1P_1 + c_2P_2 + \ldots + c_kP_k, \qquad (*)$$

где $c_i > 0$ и P_i — подстановочная матрица для каждого i. Пусть матрица A имеет размер $n \times n$. Так как сумма всех элементов каждой из матриц $A, P_1, P_2, \ldots, P_k$ равна n, из $(*)$ следует: $c_1 + c_2 + \ldots + c_k = 1$. Таким образом, найденная линейная комбинация является выпуклой. □

Замечание 1. Доказательство теоремы носит конструктивный характер. Описанный процесс получения разложения $(*)$ называют **алгоритмом Биркгофа**. Поскольку каждый шаг этого алгоритма даёт

требуемое значение по меньшей мере одному элементу матрицы A, а последний шаг — сразу n элементам матрицы, получаем неравенство $k \leqslant n^2 - n + 1$, где k — число подстановочных матриц, через которые линейно выражается дважды стохастическая матрица A. Известна и более точная оценка: $k \leqslant n^2 - 2n + 2$.

Замечание 2. Представим каждую дважды стохастическую матрицу размера $n \times n$ точкой в n^2-мерном пространстве. Геометрический смысл теоремы Биркгофа следующий: многогранник дважды стохастических матриц имеет своими вершинами подстановочные матрицы.

6. Латинские прямоугольники

Матрица размера $m \times n$ называется **латинским прямоугольником**, если элементы каждой строки этой матрицы образуют перестановку чисел от 1 до n, и в каждом столбце все числа разные.

Очевидно, что в силу определения число строк латинского прямоугольника m не превосходит числа его столбцов n. В случае $m = n$, как легко догадаться, латинский прямоугольник называют **латинским квадратом**.

Например, матрицы

$$\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 4 & 2 & 1 \\ 4 & 1 & 5 & 3 & 2 \end{pmatrix} \quad \text{и} \quad \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 3 & 5 & 4 & 2 & 1 \\ 4 & 1 & 5 & 3 & 2 \\ 2 & 3 & 1 & 5 & 4 \\ 5 & 4 & 2 & 1 & 3 \end{pmatrix}$$

представляют собой латинский прямоугольник размером 3×5 и латинский квадрат 5×5. В этом примере квадрат получается из прямоугольника приписыванием двух строк. Возникает вопрос: *всегда ли латинский прямоугольник $m \times n$, где $m < n$, можно дополнить до латинского квадрата $n \times n$, приписыванием новых $n - m$ строк?* Оказывается, *всегда!* **Докажем этот факт**.

Построим двудольный граф $G(V_1, V_2)$, в котором V_1 есть множество столбцов латинского прямоугольника размера $m \times n$, $V_2 = \{1, 2, \ldots, n\}$, а вершины $i \in V_1$ и $j \in V_2$ соединены ребром тогда и только тогда, когда в i-м столбце прямоугольника нет числа j.

Например, в случае прямоугольника из рассмотренного примера имеем граф, изображённый на рис. 7.

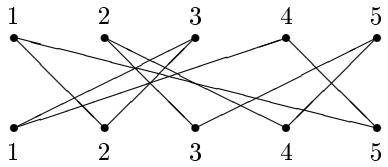


Рис. 7

Ясно, что степень любой вершины из доли V_1 равна $n - m$. С другой стороны, любое число j встречается в m строках исходного латинского прямоугольника m раз, значит, оно появляется в m столбцах и отсутствует в $n - m$ столбцах. Отсюда следует, что и степень любой вершины из V_2 также равна $n - m$.

Таким образом, двудольный граф $G(V_1, V_2)$ является непустым и регулярным. Как нам уже известно, такой граф распадается на совершенные паросочетания. Каждое совершенное паросочетание задаёт одну из новых строк латинского квадрата. □

В рассмотренном выше примере переходу от прямоугольника к квадрату соответствует выделение в двудольном графе совершенных паросочетаний: $\{12, 23, 31, 45, 54\}$ и $\{15, 24, 32, 41, 53\}$.

В заключение, отметим, что латинский прямоугольник $m \times n$ (при $m \leqslant n$) существует. Действительно, сначала заполним произвольно первую строку, затем дополним прямоугольник $1 \times n$ до квадрата, и, наконец, вычеркнем любые $n - m$ строк.

7. Рёберная раскраска графов

Рёберно-хроматическое число графа — это наименьшее число цветов, которыми можно раскрасить его рёбра так, чтобы смежные рёбра были разного цвета. Указанную раскраску будем называть **правильной**. Обозначают рёберно-хроматическое число графа G через $\chi_e(G)$ (оставляя обозначение $\chi(G)$ для **хроматического** числа графа — наименьшего числа цветов для раскраски вершин графа, при которой смежные вершины имеют разный цвет).

Поскольку рёбра одного цвета при правильной раскраске попарно не смежны, они составляют паросочетание. Таким образом, $\chi_e(G)$ — это наименьшее число паросочетаний, на которые распадается множество рёбер графа G.

Обозначим через $\Delta(G)$ наибольшую степень вершины графа G. Все рёбра, инцидентные фиксированной вершине графа, при правильной раскраске должны иметь разный цвет. Поэтому $\chi_e(G) \geqslant \Delta(G)$. Неожиданный факт обнаружил советский математик В.Г. Визинг:

Теорема 10. (В.Г. Визинг, 1964 г.) *Если в графе G нет петель, то*

$$\Delta(G) \leqslant \chi_e(G) \leqslant \Delta(G) + 1.$$

Вычислим рёберно-хроматические числа некоторых стандартных графов.

Легко видеть, что для циклического графа C_n при $n \geqslant 2$ имеем:

$$\chi_e(C_n) = \begin{cases} 2, & \text{если } n \text{ чётно;} \\ 3, & \text{если } n \text{ нечётно.} \end{cases}$$

В случае колеса W_n с $n \geqslant 4$ вершинами $\chi_e(W_n) = n - 1$.

Более сложно проверяются соотношения для полного графа K_n с $n \geqslant 2$ вершинами:

$$\chi_e(K_n) = \begin{cases} n-1, & \text{если } n \text{ чётно;} \\ n, & \text{если } n \text{ нечётно.} \end{cases}$$

Если $\chi_e(K_n) = \Delta(G) = n - 1$, то каждая вершина графа инцидентна ребру (и притом одному) каждого цвета. Отсюда следует, что количество рёбер одного цвета вдвое меньше общего числа вершин n, которое, стало быть, чётно. Значит, при нечётном n $\chi_e(K_n) > n - 1$ и, в силу теоремы Визинга, $\chi_e(K_n) = n$. Приведём, тем не менее,

Пример правильной раскраски рёбер K_n при нечётном n. Изобразим K_n правильным n-угольником с диагоналями. Выберем для каждой его стороны свой цвет. Любая диагональ будет параллельна какой-то стороне, в цвет этой стороны и выкрасим диагональ. Поскольку смежные отрезки (стороны и диагонали) не параллельны, полученная раскраска будет правильной.

В случае чётного n правильную раскраску в $n-1$ цветов можно получить следующим образом. Сначала раскрасим правильным образом подграф K_{n-1}. Вершину, не вошедшую в данный подграф, обозначим v. Для каждой вершины u подграфа будем иметь $n-2$ инцидентных рёбер, покрашенных во все цвета, кроме цвета противоположной (в $n-1$-угольнике) стороны. Используем этот цвет для покраски ребра uv. В результате любые два смежных ребра графа будут разного цвета.

Обратимся теперь к регулярному двудольному графу $K_{m,n}$.

Без потери общности предположим, что $m \leqslant n$. Составим латинский прямоугольник $A = (a_{ij})$ размера $m \times n$ и будем считать, что a_{ij} обозначает цвет ребра, соединяющего i-ю вершины первой доли с j-й вершиной второй доли. При этом получится правильная раскраска

$K_{m,n}$ в n цветов. Поскольку $\chi_e(K_{m,n}) \geqslant \Delta(K_{m,n}) = n$, делаем вывод: $\chi_e(K_{m,n}) = n$.

В общем случае, $\chi_e(K_{m,n}) = \max(m, n)$. Этот результат можно существенно усилить.

Теорема 11. *Пусть G — двудольный граф. Тогда $\chi_e(G) = \Delta(G)$.*

Доказательство. Положим $k = \Delta(G)$. Поскольку $\chi_e(G) \geqslant k$, достаточно указать способ правильной раскраски графа G в k цветов.

Покажем, как расширить граф G до регулярного двудольного графа, в котором степени всех вершин равны k.

Рассмотрим дизъюнктное объединение k экземпляров графа G, считая при этом, что первые доли указанных графов объединяются в первую долю нового графа, а вторые доли — во вторую. Возьмём произвольную вершину u первой доли исходного графа, степень которой меньше k. Введём во вторую долю строящегося графа $k - \rho(u)$ новых вершин, соединив каждую из них со всеми k "клонами" вершины u. Теперь степень клонов u станет равной k. Новые вершины также имеют эту степень. Затем аналогично поступим с вершинами второй доли, степень которых меньше k. В результате получится регулярный двудольный граф G', имеющий подграфом исходный граф G.

Результаты §4 показывают, что граф G' распадается на k совершенных паросочетаний. Покрасив рёбра каждого паросочетания в свой цвет, получим правильную раскраску графа G', а заодно и графа G. □

Заключительные параграфы данной главы посвящены алгоритмам решения различных задач, связанных с паросочетаниями.

8. Теорема Бёржа

Пусть M — паросочетание в графе G (не обязательно двудольном). Рёбра, входящие в M, назовём черными, а остальные рёбра графа — белыми. Простая цепь — **чередующаяся** (**относительно** M), если любые два соседних ребра в этой цепи разного цвета. Вершину, покрываемую некоторым ребром из M, назовём **насыщенной** (**относительно** M); в противном случае вершина — **ненасыщенная** (или **свободная**) (**относительно** M).

Если в графе имеется чередующаяся цепь, соединяющая две ненасыщенные вершины, то легко получить новое паросочетание, в котором на одно ребро больше, чем в M: достаточно поменять цвета

рёбер указанной цепи. Поэтому такую цепь называют **увеличивающей относительно** M, или M-**увеличивающей**. Очевидно теперь, что относительно наибольшего паросочетания увеличивающей цепи не существует. Оказывается, верно и обратное.

Теорема 12. (К. Бёрж, 1957 г.) *Паросочетание M в графе G является наибольшим тогда и только тогда, когда в G не существует увеличивающей относительно M цепи.*

Доказательство. *Необходимость* уже доказана. Убедимся в *достаточности.*

Пусть паросочетание M не допускает увеличивающей относительно себя цепи, а M' — некоторое наибольшее паросочетание. Рассмотрим подграф графа G, образованный рёбрами из симметрической разности $M \oplus M'$. В нём степень каждой вершины не превосходит 2, в силу чего этот подграф распадается на циклы и простые цепи. Поскольку в цикле рёбра из M и M' чередуются, цикл имеет чётную длину. Чередование происходит и в каждой цепи. Цепь не является увеличивающей ни относительно M (по условию теоремы), ни относительно M' (наибольшего паросочетания). Отсюда вытекает, что длина цепи является чётным числом. Стало быть, в симметрической разности $M \oplus M'$ поровну рёбер из M и M', а множества M и M' равномощны, то есть M, как и M', является наибольшим паросочетанием. □

9. Нахождение наибольшего паросочетания

Теорема Бёржа обосновывает корректность следующего способа нахождения наибольшего паросочетания в графе.

I. Построить произвольное паросочетание M.

II. Искать M-увеличивающую цепь.

III. Если такая цепь P найдена, то заменить M на $M \oplus P$ и перейти к шагу II. В противном случае закончить работу (текущее паросочетание M является наибольшим).

В случае **двудольного графа** поиск M-увеличивающих цепей происходит довольно просто. Используем для этого механизм пометок.

Алгоритм $\mathcal{A}$

1. Построить какое-нибудь паросочетание M.

2. Свободные относительно M вершины из доли V_1 пометить нулём, остальные вершины считаются непомеченными.

3. Для каждой помеченной на предыдущем шаге вершины $v_i \in V_1$ пометить её номером i все непомеченные вершины из V_2, которые соединяются с v_i белыми рёбрами.

   Если при этом окажется помеченной свободная вершина, то M-увеличивающая цепь P найдена; она восстанавливается по меткам, начиная с указанной вершины: метка вершины есть номер очередной вершины P. Увеличивающую цепь перекрасить (т.е. заменить M на $M \oplus P$) и перейти к шагу 2.

   Если же новых пометок на этом шаге не возникает, перейти к 5.

4. Для каждой помеченной на предыдущем шаге вершины $u_j \in V_2$ пометить её номером j непомеченную вершину из V_1, которая соединяется с u_j чёрным рёбром. Если новых пометок на этом шаге не возникает, перейти к шагу 5, иначе вернуться к шагу 3.

5. Текущее паросочетание M является наибольшим. Конец работы.

Проиллюстрируем работу изложенного алгоритма на следующем примере. Для графа на рис. 8 текущее паросочетание M состоит из

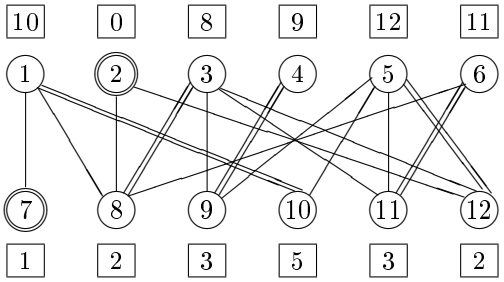


Рис. 8

рёбер, изображённых двойными линиями. Напомним, что рёбра из M мы называем чёрными, а остальные рёбра графа — белыми. Свободные вершины изображены двойными кружками.

В доле V_1 только одна свободная вершина — с номером 2; она помечается нулём (пометки — это числа в прямоугольниках). Белые рёбра

(изображены тонкими линиями) соединяют эту вершину с вершинами 8 и 12, которые получают пометку 2.

Чёрное ребро 8-3 даёт пометку 8 вершине 3; чёрное ребро 12-5 даёт пометку 12 вершине 5.

Из вершины 3 к непомеченным вершинам ведут два белых ребра, и их другие концы — вершины 9 и 11 — получают пометку 3. Аналогично, у вершины 10 возникает пометка 5.

Далее помечаются вершины из первой доли: 4, 1 и 6.

Наконец, удаётся пометить свободную вершину 7 из второй доли — номером 1. Идя по пометкам, начиная с этой вершины, находим M-увеличивающую цепь: 7–1–10–5–12–2. Перекрасим её ребра и получим новое (более мощное, чем ранее) паросочетание (рис. 9). Построенное

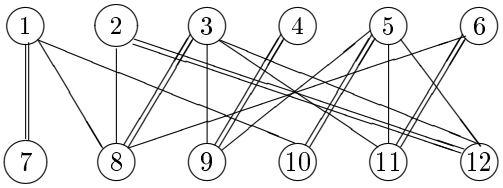


Рис. 9

паросочетание является совершенным (оно покрывает все вершины графа), значит, и наибольшим.

Замечание 1. Алгоритм $\mathcal{A}$ является, по сути, "проекцией" алгоритма Форда – Фалкерсона нахождения максимального потока в сети (см. гл. 1,§3).

Подробно поясним этот тезис. Превратим двудольный граф $G(V_1, V_2)$ в транспортную сеть следующим образом:

- ориентируем каждое ребро $u_i v_j$ в направлении от V_1 к V_2;
- к множеству вершин графа добавим источник a и сток b;
- проведём дуги из источника a ко всем вершинам из V_1;
- из каждой вершины доли V_2 проведём дугу к стоку b;
- положим пропускные способности всех дуг равными единице.

Существует взаимно однозначное соответствие между допустимыми целочисленными потоками в этой сети и паросочетаниями исходного двудольного графа: по каждой дуге сети, соответствующей ребру $G(V_1, V_2)$, входящему (не входящему) в паросочетание, идёт единичный (соответственно нулевой) поток. При этом максимальному

(по включению) паросочетанию будет отвечать полный поток. M-увеличивающей цепи

$$u_\alpha - v_\beta - u_\gamma - \ldots - v_\omega \tag{1}$$

соответствует цепь транспортной сети:

$$a \to u_\alpha \to v_\beta \leftarrow u_\gamma \to \ldots \to v_\omega \to b. \tag{2}$$

Перекраска рёбер в цепи (1) суть увеличение потока вдоль цепи (2). Например, ребро $u_\alpha - v_\beta$ было белым, а стало чёрным, соответственно поток по дуге $u_\alpha \to v_\beta$ из нулевого превратился в единичный; следующее ребро цепи $u_\gamma - v_\beta$ из чёрного стало белым, а поток по дуге $u_\alpha \to v_\beta$ стал нулевым.

Замечание 2. Несложно оценить трудоёмкость данного алгоритма. Пусть n — число вершин в доле V_1 (без ограничения общности можно считать, что в V_1 вершин не более, чем в V_2), а m — число рёбер графа $G(V_1, V_2)$. Тогда при работе алгоритма будет выполнено не более n итераций (итерацию составляют шаги 2, 3 и 4), а на каждой итерации порядка $O(m)$ действий. Стало быть, общее число операций есть $O(nm)$. Если $|V_1| = |V_2| = n$, то имеем оценку трудоёмкости, зависящую только от числа вершин: $O(n^3)$.

Замечание 3. Разработаны весьма изощрённые алгоритмы нахождения наибольшего паросочетания в произвольном графе (в общем случае, не двудольном), также имеющие трудоёмкость $O(n^3)$, где n — количество вершин графа.

10. Нахождение наименьшего вершинного покрытия

Вновь ограничимся случаем двудольного графа.

Оказывается, изложенный выше алгоритм $\mathcal{A}$ нахождения наибольшего паросочетания заодно позволяет обнаружить и наименьшее вершинное покрытие.

Пусть $A \subset V_1$ и $B \subset V_2$ — множества помеченных вершин обеих долей графа $G(V_1, V_2)$ по окончании работы алгоритма $\mathcal{A}$. Обозначим $\overline{A} = V_1 \setminus A$.

Теорема 13. *Множество $\overline{A} \cup B$ является минимальным вершинным покрытием графа $G(V_1, V_2)$.*

Доказательство. Пусть M — наибольшее паросочетание в графе $G(V_1, V_2)$, построенное в результате работы алгоритма $\mathcal{A}$. Рёбра, входящие в M, как и ранее, мы называем чёрными; а остальные рёбра графа для нас белые.

Доказательство теоремы разобьём на три этапа.

1. Докажем, что **концы любого чёрного ребра либо оба помечены, либо оба не помечены**.

   Возьмём произвольное чёрное ребро uv, где $u \in V_1,\ v \in V_2$.

   Если вершина v помечена, то на шаге 4 алгоритма $\mathcal{A}$ пометку получит и вершина u.

   Рассмотрим теперь случай, когда вершина v не помечена. Тогда непомеченной будет и вершина u. Действительно, u, будучи насыщенной вершиной, не помечается на 2-м шаге. Но и на 4-м шаге вершина u не получает пометку, так как она инцидентна единственному чёрному ребру — uv, а другой его конец не помечен.

   Утверждение доказано.

2. Проверим, что $C = \overline{A} \cup B$ — **вершинное покрытие графа.**

   Возьмём в графе $G(V_1, V_2)$ произвольное ребро $e = uv$, где $u \in V_1$, $v \in V_2$. Докажем, что $u \notin A$ или $v \in B$.

   Если это не так, то $u \in A$ (то есть u — помеченная вершина) и $v \notin B$ (v — непомеченная вершина). Тогда в силу утверждения 1 ребро e не может быть чёрным. Но оно и не белое, поскольку при помеченной вершине u и белом ребре uv на 3-м шаге алгоритма $\mathcal{A}$ вершина v получила бы пометку. Противоречие получено.

3. Заметим, что (благодаря утверждению 1)
   1) количество чёрных рёбер с помеченными концами равно $|B|$ — количеству помеченных вершин из доли V_2 (каждая из них инцидентна одному чёрному ребру);
   2) количество чёрных рёбер с непомеченными концами равно $|\overline{A}|$ — количеству непомеченных вершин из доли V_1 (любая такая вершина насыщенная, и из неё также исходит ровно одно чёрное ребро).

   Таким образом, $|M| = |B| + |\overline{A}| = |\overline{A} \cup B|$, то есть вершинное покрытие $\overline{A} \cup B$ равномощно наибольшему паросочетанию. В силу венгерской теоремы это означает минимальность вершинного покрытия.□

Как отмечалось в §3, дополнение к наименьшему вершинному покрытию графа является его наибольшим независимым множеством вершин. Значит, множество $A \cup \overline{B}$ — наибольшее независимое. Стало быть, алгоритм $\mathcal{A}$ (имеющий, как мы выяснили, полиномиальную трудоёмкость) позволяет наряду с наибольшим паросочетанием в двудольном графе найти также наименьшее вершинное покрытие и наибольшее независимое множество. Известно, что задачи определения наименьшего вершинного покрытия и наибольшего независимого множества для *произвольного графа* являются NP-полными, другими словами, для них нет эффективных алгоритмов решения (по крайней мере, они пока не известны науке). Очень важно, что для двудольных графов, возникающих во многих практических задачах, полиномиальные алгоритмы решения указанных задач существуют!

Пример. На рис. 10 изображён граф, в котором выделено наибольшее паросочетание и расставлены пометки, возникающие по окончании работы алгоритма $\mathcal{A}$. Имеем $A = \{1, 2, 3\}$; $B = \{6, 7\}$. Значит,

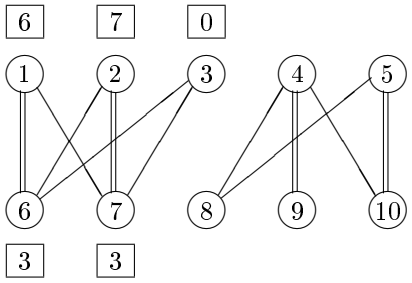


Рис. 10

вершины 4, 5, 6 и 7 образуют минимальное вершинное покрытие, а вершины 1, 2, 3, 8, 9, 10 — наибольшее независимое множество.

11. Венгерский алгоритм

Пусть $G(V_1, V_2)$ — взвешенный полный двудольный граф $K_{n,n}$ с матрицей весов рёбер $C = (c_{ij})$, где $c_{ij} \geqslant 0$ есть вес ребра, соединяющего i-ю вершину доли V_1 с j-й вершиной доли V_2.

В **задаче о назначениях** требуется найти совершенное паросочетание минимального веса (ниже будем называть такое паросочетание **оптимальным**). Название задачи объясняется следующей её трактовкой. Пусть u_i — i-й работник, v_j — j-я работа, а c_{ij} — стоимость

выполнения i-м работником j-й работы. Тогда совершенное паросочетание наименьшего веса описывает такое распределение работ между работниками (*назначение* работников на работы), при котором каждый работник выполняет ровно одну работу, а суммарная стоимость выполнения работ минимальна.

Заметим, что если к весам всех рёбер, инцидентных некоторой фиксированной вершине, прибавить одно и то же число (не обязательно положительное), то упорядоченность совершенных паросочетаний графа по их весу не изменится, поскольку каждое такое паросочетание содержит ровно одно ребро с изменённым весом. Назовём такую операцию **приведением**. Для матрицы весов приведение означает прибавление константы ко всем элементам некоторого столбца или строки.

С помощью операций приведения легко свести задачу к графу, в котором каждая вершина инцидентна некоторому ребру нулевого веса (будем называть такие рёбра **нулевыми**), и при этом веса всех рёбер остаются неотрицательными числами. Если в таком графе существует совершенное паросочетание нулевого веса (в матрице весов ему соответствуют n независимых нулей), очевидно, оно и будет решением задачи. Основная идея излагаемого ниже **венгерского алгоритма** (Г. Кун, 1955 г.) состоит в том, что в результате поиска M-увеличивающей цепи в $G(V_1, V_2)$, где M — паросочетание из нулевых рёбер, удаётся найти такие операции приведения, в результате которых количество нулевых рёбер увеличивается. В некоторый момент возникает граф, в котором есть совершенное паросочетание нулевого веса.

Алгоритм $\mathcal{B}$

1. Из каждой строки матрицы C вычесть её минимальный элемент. Так же поступить со столбцами.

2. Пусть N — множество нулевых рёбер текущего графа $G = G(V_1, V_2)$ (с матрицей весов C). Найти в графе $G' = \langle V_1 \cup V_2, N \rangle$ с помощью алгоритма $\mathcal{A}$ наибольшее паросочетание M, беря в качестве начального паросочетания текущее наибольшее паросочетание (если шаг 2 алгоритма $\mathcal{B}$ выполняется не впервые). Если найденное паросочетание M — совершенное, то конец работы.

3. Пусть $A \subset V_1$ и $B \subset V_2$ — множества помеченных вершин обеих долей графа G по окончании работы алгоритма $\mathcal{A}$ (напомним,

что пометки расставляются с целью найти M-увеличивающую цепь). Вычислить d — наименьший вес ребра, соединяющего помеченную вершину из V_1 с непомеченной вершиной из V_2. Из всех строк, отвечающих вершинам из A, вычесть число d. Ко всем столбцам, отвечающим вершинам из B, прибавить число d. Перейти к шагу 1.

Обоснуем **корректность** алгоритма $\mathcal{B}$. Во-первых, применяемые в нём операции приведения не меняют множества оптимальных паросочетаний. Во-вторых, алгоритм заканчивает работу, когда обнаруживает в двудольном графе с неотрицательными весами рёбер совершенное паросочетание нулевого веса. Осталось убедиться, что такой момент непременно наступит.

После первого шага работы алгоритма получаем граф, в котором каждая вершина инцидентна хотя бы одному нулевому ребру. Поэтому после шага 2 все вершины из V_1, не покрытые текущим паросочетанием, помечены. Значит, на шаге 3 множество A непусто. С другой стороны, если найденное в результате выполнения шага 2 паросочетание не является совершенным, это означает, что не все вершины из V_2 удалось пометить — значит, непусто и множество $\overline{B} = V_2 \setminus B$. Вес любого ребра, соединяющего вершину i из A с вершиной j из $\overline{B}$, положителен, так как в противном случае по алгоритму $\mathcal{A}$ вершина j должна была бы получить пометку. Таким образом, $d > 0$.

Посмотрим, что происходит с весами рёбер на шаге 3.

Веса рёбер из $G(\overline{A}, \overline{B})$ не меняются.

Рёбра из подграфа $G(A, B)$ сохраняют свой вес, поскольку первоначальное вычитание числа d компенсируется последующим его прибавлением.

Вес каждого ребра из $G(\overline{A}, B)$ увеличивается на d.

Поскольку $d > 0$ — наименьший вес ребра в графе $G(A, \overline{B})$, после шага 3 в этом подграфе появится хотя бы одно нулевое ребро.

Наконец, рёбер из $G(\overline{A}, \overline{B})$ никаких изменений не коснётся.

Значит, после шага 3 веса всех рёбер остаются неотрицательными. Как было отмечено в предыдущем параграфе, концы любого чёрного ребра либо оба помечены, либо нет. В первом случае ребро входит в $G(A, B)$, во втором — в $G(\overline{A}, \overline{B})$. Таким образом, после шага 3 рёбра текущего максимального паросочетания M остаются нулевыми. С другой стороны, рёбра из $G(A, \overline{B})$, ставшие нулевыми, при последующем выполнении шага 2 расширяют множество B. Поэтому на некоторой итерации будут найдены M-увеличивающая цепь и новое, более

мощное, паросочетание M. Рано или поздно в текущем графе будет будет существовать совершенное паросочетание из нулевых рёбер.□

Проиллюстрируем работу алгоритма $\mathcal{B}$ следующим **примером**.

Пусть веса рёбер графа $K_{4,4}$ задаются матрицей

$$\begin{pmatrix} 1 & 4 & 4 & 3 \\ 2 & 7 & 6 & 8 \\ 4 & 7 & 5 & 6 \\ 2 & 5 & 1 & 1 \end{pmatrix}.$$

После вычитания минимальных элементов строк и столбцов последовательно получим матрицы

$$\begin{pmatrix} 0 & 3 & 3 & 2 \\ 0 & 5 & 4 & 6 \\ 0 & 3 & 1 & 2 \\ 1 & 4 & 0 & 0 \end{pmatrix} \quad \text{и} \quad \begin{pmatrix} 0 & 0 & 3 & 2 \\ 0 & 2 & 4 & 6 \\ 0 & 0 & 1 & 2 \\ 1 & 1 & 0 & 0 \end{pmatrix}.$$

Построим граф G', образованный нулевыми рёбрами (текущего) графа G (рис. 11). Выделим в нём максимальное по включению паро-

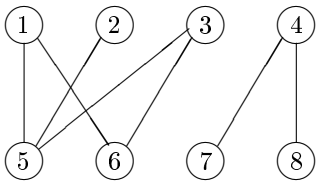


Рис. 11

сочетание (его рёбра изобразим двойными линиями) и осуществим процесс расстановки пометок согласно алгоритму $\mathcal{A}$. Пометки последовательно получают вершины 2, 5, 1, 6, 3 (рис. 12). Таким обра-

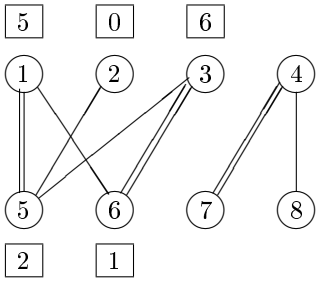


Рис. 12

зом, множество помеченных вершин из V_1 и V_2 есть $A = \{1, 2, 3\}$ и $B = \{5, 6\}$ соответственно. Теперь в матрице весов подграфа $G(A, \overline{B})$, образованной тремя верхними строками и двумя последними столбцами матрицы C, находим наименьший элемент: $d = 1$. В результате вычитания числа d из строк, отвечающих A, а затем прибавления d к столбцам, отвечающим B, возникают матрицы

$$\begin{pmatrix} -1 & -1 & 2 & 1 \\ -1 & 1 & 3 & 5 \\ -1 & -1 & 0 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} \quad \text{и} \quad \begin{pmatrix} 0 & 0 & 2 & 1 \\ 0 & 2 & 3 & 5 \\ 0 & 0 & 0 & 1 \\ 2 & 2 & 0 & 0 \end{pmatrix}.$$

Возвращаемся к шагу 2. Теперь имеем граф G', изображённый на рис. 13, в котором существует совершенное паросочетание. Итак, оп-

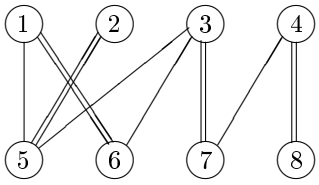


Рис. 13

тимальное паросочетание состоит из рёбер 1-6, 2-5, 3-7, 4-8, а его вес, как легко подсчитать, равен 12 (разумеется, в исходном графе).

В заключение — несколько замечаний.

Замечание 1. Трудоёмкость алгоритма $\mathcal{B}$ оценивается как $O(n^4)$ операций. Известны модификации венгерского алгоритма, имеющие трудоёмкость $O(n^3)$.

Замечание 2. Часто рассматривают также задачу нахождения во взвешенном двудольном графе совершенного паросочетания **наибольшего** веса. Читателю предлагается подумать над тем, как преобразовать матрицу весов исходного графа, чтобы перейти к графу, в котором оптимальное паросочетание (наименьшего веса) соответствует паросочетанию наибольшего веса в исходном графе.

12. Задача о назначениях на узкое место

Пусть на некотором конвейере (поточной линии) имеется n рабочих мест. Мастер должен расставить по этим местам n работников. Известна производительность труда c_{ij} i-го работника на j-м месте.

Производительность конвейера определяется минимальной производительностью занятых на нём работников. Рабочее место, на котором реализуется минимальная производительность при заданной расстановке работников (другими словами, при заданном *назначении*), и есть **узкое** место этого назначения.

Нужно помочь мастеру "сделать узкое место как можно шире": найти такое распределение работников по рабочим местам, при котором скорость конвейера (т.е. минимальная производительность занятых работников) будет максимальной.

На языке теории графов задача состоит в *выделении во взвешенном двудольном графе такого совершенного паросочетания, в котором наименьший вес ребра будет наибольшим.*

Другими словами, требуется найти такое наибольшее число f, при котором в графе существует совершенное паросочетание M, составленное из рёбер, чей вес не меньше f (паросочетание M и будет искомым).

Это можно сделать с помощью дихотомического поиска. Суть его в том, что на каждом шаге множество, содержащее искомое значение, сокращается вдвое. В описываемом ниже алгоритме индексы a и b задают границы указанного множества.

Алгоритм $\mathcal{C}$

1. Отсортировать рёбра по возрастанию веса, получив массив $w_1, w_2, \ldots, w_{n^2}$.

2. Положить $a = 1,\ b = n^2 - n + 2$.

3. Если $a = b - 1$, перейти к шагу 6.

4. Положить $k = \left[\frac{a+b}{2}\right],\ f = w_k$.

5. С помощью алгоритма $\mathcal{A}$ искать совершенное паросочетание в подграфе G' графа G, образованном рёбрами, чей вес не меньше f.

   Если такое паросочетание найдено, то объявить его текущим и положить $a = k$. В противном случае положить $b = k$.

   Перейти к шагу 3.

6. Текущее паросочетание является оптимальным. Конец работы.

Нетрудно оценить число операций при выполнении алгоритма $\mathcal{C}$. Наиболее трудоёмкий шаг 5 выполняется примерно $\log_2(n^2 - n + 1)$ раз. Поэтому общая трудоёмкость — $O(\log_2 n \cdot n^3)$ операций. В случае использования вместо $\mathcal{A}$ более эффективного алгоритма трудоёмкость алгоритма $\mathcal{C}$ соответственно понизится.

Глава 3

Матроиды

Матроиды были введены в 1935 г. Пионерская работа Х. Уитни называлась "On the abstract properties of linear dependence" ("К абстактным свойствам линейной зависимости"). Х. Уитни обнаружил, что можно с единых позиций рассматривать понятия зависимости в линейной алгебре и теории графов. Синтез идей различных областей математики является основой плодотворного развития теории матроидов.

Матроидные стуктуры естественным образом возникают в теории комбинаторной оптимизации, являясь основой применения так называемых "жадных" алгоритмов (см. §5). Исследования в этой области начались в конце 50-х годов прошлого века. Позднее теория матроидов нашла своё применение и при анализе надёжности электрических схем.

Конец XX века ознаменовался бурным развитием криптографии. В так называемых *идеальных схемах разделения секрета* также возникли матроидные структуры.

В настоящей главе представлены начальные понятия теории матроидов, включая связь матроидов с жадными алгоритмами и теорией трансверсалей.

1. Определения и примеры

Матроид — это упорядоченная пара $M = \langle E, J\rangle$, где
E — непустое конечное множество;
J — совокупность подмножеств множества E, удовлетворяющая следующим условиям (*аксиомам независимости*):

(J0) $\varnothing \in J$;

(J1) если $A \in J$ и $B \subset A$, то $B \in J$;

(J2) если $A \in J,\ B \in J$ и $|A| = |B| + 1$, то существует такой элемент e, принадлежащий A и не принадлежащий B, что $B \cup \{e\} \in J$.

Элементы множества J называют **независимыми множествами**. Таким образом, аксиома J1 говорит о том, что подмножество

независимого множества также является независимым, а аксиома J2 утверждает: если имеются два независимых множества, мощности которых отличаются на единицу, то в более мощном множестве есть элемент, который отсутствует в менее мощном, и при добавлении которого к последнему вновь получится независимое множество.

Базис — это максимальное по включению независимое множество (то есть если A — базис, $A \subset B$ и $A \neq B$, то множество B не является независимым).

В силу конечности множества E в матроиде существует хотя бы один базис.

Ранг матроида — количество элементов в любом его базисе.

Докажем корректность последнего определения: убедимся в том, что в любых двух базисах количество элементов одинаково. От противного: пусть имеются два различных базиса A и B, и в множестве A элементов больше, чем в B. Существует подмножество $A' \subset A$ такое, что $|A'| = |B| + 1$. По аксиоме J1 множество A' — независимое, а по аксиоме $J2$ найдётся элемент $e \in A' \setminus B$ такой, что множество $B \cup \{e\}$ независимо, но тогда множество B не является максимальным по включению независимым множеством, то есть базисом — противоречие!

Очевидно, что максимальное по мощности независимое множество является и максимальным по включению, то есть базисом. Таким образом, любое независимое множество в матроиде M мощности, равной его рангу, есть базис.

Взяв произвольное независимое множество B и некоторый фиксированный базис A с помощью свойства J2 можно в множестве A выбрать $|A| - |B|$ элементов, при добавлении которых к множеству B получится независимое множество мощности $|A|$, то есть базис. Мы доказали, что *всякое независимое множество можно дополнить до базиса.*

Множество $A \subset E$, которое не является независимым в матроиде $M = \langle E, J \rangle$, называется **зависимым**.

Цикл — это минимальное по включению зависимое множество (то есть если A — цикл, $B \subset A$ и $B \neq A$, то множество B — независимое).

Пусть дан матроид $M = \langle E, J \rangle$. Мощность множества E называют **порядком матроида** M.

Примеры матроидов

1. **Тривиальный матроид** — матроид $\langle E, \{\varnothing\} \rangle$; в нём единственным независимым множеством (значит, и единственным бази-

сом) является пустое множество. Ранг тривиального матроида равен нулю, а любое одноэлементное подмножество множества E является циклом.

2. **Дискретный матроид** — матроид $\langle E, \beta(E)\rangle$, где $\beta(E)$ — множество всех подмножеств множества E. В дискретном матроиде все множества являются независимыми, имеется единственный базис — само множество E, а циклов вовсе нет. Ранг дискретного матроида равен $|E|$.

3. **k-однородный матроид** — матроид $\langle E, J\rangle$, в котором любое k-элементное подмножество множества E является базисом. Здесь любое множество, в котором не более k элементов, является независимым. Проверка выполнения аксиом матроида тривиальна. Заметим, что дискретный матроид $\langle E, \beta(E)\rangle$ является $|E|$-однородным, а тривиальный матроид — 0-однородным. В k-однородном матроиде (при $k < |E|$) циклом является любое $k+1$-элементное множество. Ранг k-однородного матроида равен k. Имеется специальное обозначение для k-однородного матроида, заданного на n-элементном множестве: $U_{k,n}$.

4. Пусть E — конечная система векторов некоторого линейного пространства над полем F, а J состоит из всех линейно независимых систем векторов из E, а также пустого множества. Тогда, как известно из линейной алгебры, свойства J1 и J2 будут выполнены. Свойство J0 выполнено по определению. Поэтому $\langle E, J\rangle$ — матроид. Его называют **векторным**[1].

5. Пусть A — числовая матрица с элементами из поля F размера $m \times n$. Будем смотреть на столбцы этой матрицы как на векторы пространства F^m. Тогда столбцы матрицы A образуют векторный матроид; будем называть его **матричным матроидом**, или (точнее) **матроидом столбцов матрицы** A. Обозначение: $M[A]$. Аналогично вводится **матроид строк матрицы**. Легко видеть, что ранг матрицы A равен рангу соответствующего матричного матроида.

6. Пусть G — граф, E — множество его рёбер. Объявим независимыми те подмножества E, которые состоят из рёбер некоторого леса. Как известно из теории графов, свойства J0, J1 и J2 будут

[1] Иногда дают более узкое определение векторного матроида, предполагая, что векторы из системы E попарно различны.

при этом выполнены. Полученный матроид называют **матроидом циклов** графа G и обозначают $M(G)$. Цикл матроида будут составлять рёбра, образующие простую замкнутую цепь в графе G (то есть цикл, в котором каждая вершина встречается ровно один раз).

7. Пусть G — граф, E — множество его рёбер. Объявим зависимыми те подмножества E, которые являются разделяющими множествами. Опять же из результатов теории графов следует выполнение аксиом матроида. Полученный матроид называют **матроидом разрезов** графа G и обозначают $M^*(G)$. Циклу этого матроида будет соответствовать разрез графа G, а базису — дополнение к множеству рёбер любого остовного леса графа.

Изоморфизм матроидов

Матроиды $M_1 = \langle E_1, J_1 \rangle$ и $M_2 = \langle E_2, J_2 \rangle$ называются **изоморфными**, если существует биекция (взаимно однозначное отображение) $\varphi : E_1 \to E_2$, сохраняющая независимость; другими словами, множество $A \subset E_1$ является независимым в матроиде M_1 тогда и только тогда, когда образ этого множества при заданном отображении $\varphi(A)$ есть независимое множество в матроиде M_2.

Пример. Матроид циклов графа G, изображённого на рис. 14,

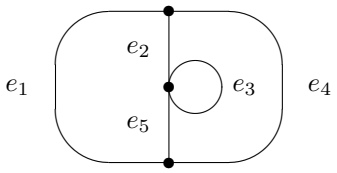


Рис. 14

изоморфен матроиду столбцов матрицы $A = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}$, рассматриваемой над произвольным числовым полем.

Введём ещё несколько определений.

Матроид — **графический**, если он изоморфен матроиду циклов некоторого графа.

Матроид — **кографический**, если он изоморфен матроиду разрезов некоторого графа.

Наконец, если матроид является одновременно графическим и кографическим, то его называют **планарным**. Это название объясня-

ется тем, что планарный матроид изоморфен матроиду циклов планарного графа.

2. Двойственность

Пусть $M = \langle E, J\rangle$ — матроид. Обозначим через J^* множество всех подмножеств дополнений к базисам матроида M. Оказывается, что $M^* = \langle E, J^*\rangle$ — также матроид; его называют **двойственным** к M матроидом. Легко видеть, что базисы двойственного матроида — это дополнения к базисам исходного матроида. Отсюда сразу вытекает, что $(M^*)^* = M$ — матроид, двойственный двойственному M, есть M.

Теорема 14. *Для любого графа G матроид его разрезов является двойственным матроиду циклов.*

Доказательство. Пусть E — множество рёбер графа G. Можно считать, что в G нет изолированных вершин.

Рассмотрим произвольный базис B в матроиде циклов $M(G)$. Нужно доказать, что $E \setminus B$ — базис в матроиде разрезов $M^*(G)$. Базис в $M(G)$ есть остовный лес графа G. Базис в $M^*(G)$ — максимальное по включению неразделяющее множество в графе G. При удалении из графа рёбер, составляющих $E \setminus B$, остаётся остовный лес, т.е. число компонент связности графа не изменяется — поэтому множество $E \setminus B$ не является разделяющим. Если оно не максимально по включению, то для некоторого ребра $e \notin E \setminus B$ неразделяющим будет множество $B' = E \setminus B \cup \{e\}$. Но тогда дополнение к B' содержит остовный лес W. Заметим теперь, что $|E \setminus B'| < |B|$, откуда $|W| < |B|$, в то время как любые два остовных леса (одного и того же графа) имеют одинаковую мощность.

Обратно. Пусть D — базис в $M^*(G)$. Нужно убедиться в том, что $E \setminus D$ — остовный лес. Поскольку D — неразделяющее множество, $E \setminus D$ покрывает все вершины графа G. Если в $E \setminus D$ есть цикл, возьмём ребро e, входящее в него. Тогда $D \cup \{e\}$ — также неразделяющее множество вопреки предположению. Значит, $E \setminus D$ — лес, и притом — остовный. □

3. Представимые матроиды

Матроид **представим над полем** F, если он изоморфен некоторому векторному матроиду над этим полем. Если матроид представим

над любым полем, его называют **регулярным**. В случае представимости матроида над полем $GF(2)$ его называют **бинарным**, а над полем $GF(3)$ — **тернарным**.

Теорема 15. *Графический матроид является бинарным.*

Доказательство. Нужно убедиться в том, что матроид циклов произвольного графа G представим над полем $GF(2)$. Составим матрицу $A = (a_{ij})$ инцидентности графа G. Строки этой матрицы соответствуют вершинам графа, а столбцы — рёбрам. Если j-е ребро есть петля, инцидентная i-му ребру, то $a_{ij} = 2$, иначе

$$a_{ij} = \begin{cases} 1, & \text{если } i\text{-я вершина инцидентна } j\text{-му ребру;} \\ 0 & \text{в противном случае.} \end{cases}$$

Заменим в этой матрице двойки нулями, оставив для матрицы прежнее обозначение.

Итак, мы имеем составленную из нулей и единиц матрицу A. Петле графа соответствует нулевой столбец, а столбцы, отвечающие кратным рёбрам, — одинаковые. Докажем, что матроид столбцов этой матрицы над полем $GF(2)$ изоморфен $M(G)$, то есть что столбцы линейно зависимы над $GF(2)$ тогда и только тогда, когда соответствующие им рёбра графа G содержат цикл.

Нетривиальная линейная комбинация векторов над указанным полем есть просто сумма некоторых из данных векторов. Значит, если некоторые столбцы матрицы A линейно зависимы, то среди них можно выделить столбцы с нулевой суммой. Рассмотрим подграф G' графа G с рёбрами, соответствующими этим столбцам. Очевидно, степень каждой вершины в этом подграфе есть чётное число. Стало быть, в G' есть цикл (хотя бы потому, что G' есть объединение эйлеровых графов, а в эйлеровом графе есть эйлеров цикл; впрочем, и непосредственное доказательство указанного факта весьма несложно).

Обратно. Пусть некоторое множество рёбер содержит цикл. Если среди них есть петля, то отвечающий ей нулевой столбец обеспечивает линейную зависимость столбцов.

Рассмотрим теперь столбцы, отвечающие рёбрам простого цикла длины больше 1. Любая строка матрицы A содержит в этих столбцах ровно две единицы. Поэтому сумма указанных столбцов (по модулю 2) равна нулевому столбцу, что означает линейную зависимость исходного множества столбцов.□

Заметим, что имеется существенное усиление доказанной теоремы. Оказывается,

любой графический матроид является регулярным; то же верно и для любого кографического матроида.

Кроме того, достаточным условием регулярности матроида является его представимость над $GF(2)$ и $GF(3)$, то есть
матроид регулярен тогда и только тогда, когда он одновременно является бинарным и тернарным.

4. Ранговая функция

Пусть $M = \langle E, J\rangle$ — матроид. Для множества $A \subset E$ определим **сужение матроида** M **на множество** A как матроид $M|A = \langle A, J'\rangle$, где множество J' образовано всеми подмножествами множества A, являющимися независимыми множествами матроида M:

$$J' = \{X \mid X \subset A, X \in J\}.$$

То, что $M|A$ — действительно матроид, очевидно.

Назовём ранг матроида $M|A$ **рангом** множества A (обозначение $\rho(A)$), а каждый базис этого матроида — **базой** множества A. Таким образом, $\rho(A)$ — это наибольшая мощность независимого подмножества множества A. Заметим, что $M|E = M$, и поэтому ранг матроида $M = \langle E, J\rangle$ равен рангу множества E, а базис матроида есть база множества, на котором он определён. Очевидно также, что $\rho(\varnothing) = 0$, и если $A \in J$, то $\rho(A) = |A|$ (ранг независимого множества равен его мощности).

Отметим следующие свойства ранговой функции ρ:

(ρ1) $\forall A \subset E \;\; 0 \leqslant \rho(A) \leqslant |A|$;

(ρ2) если $A \subset B$, то $\rho(A) \leqslant \rho(B)$ (*монотонность*);

(ρ3) $\forall A, B \subset E \;\; \rho(A \cup B) + \rho(A \cap B) \leqslant \rho(A) + \rho(B)$ (*полумодулярность*).

Выполнение свойств ρ1 и ρ2 очевидно. Докажем ρ3.

Пусть X — база множества $A \cap B$. С помощью свойства J2 множество X можно дополнить до базы множества A; обозначим полученную базу через Y. Аналогично, Y дополняется до базы Z множества $A \cup B$. Итак, имеем

$$X \subset Y \subset Z \;\; \text{и} \;\; \rho(A \cap B) = |X|, \;\; \rho(A) = |Y|, \;\; \rho(A \cup B) = |Z|.$$

Необходимо доказать, что

$$|Z| + |X| \leqslant |Y| + \rho(B). \qquad (*)$$

Заметим, что по построению $X \cup (Z \setminus Y) \subset B$. Отсюда

$$\rho(X \cup (Z \setminus Y)) \leqslant \rho(B).$$

Кроме того, множество $X \cup (Z \setminus Y)$ является подмножеством независимого множества Z, и, в силу J1, само независимо. Значит,

$$\rho(X \cup (Z \setminus Y)) = |X \cup (Z \setminus Y)| = |X| + |Z| - |Y|.$$

Таким образом, $|X| + |Z| - |Y| \leqslant \rho(B)$, что равносильно неравенству $(*)$.$\square$

Покажем теперь, что матроид можно задать через ранговую функцию.

Теорема 16. *Пусть на подмножествах множества E определена целозначная функция ρ, удовлетворяющая условиям $\rho 1$, $\rho 2$, $\rho 3$, а множество J состоит из всех тех подмножеств A множества E, для которых $\rho(A) = |A|$. Тогда $M = \langle E, J\rangle$ суть матроид.*

Доказательство. Свойство J0 следует непосредственно из $\rho 1$.

Докажем J1. Пусть $\rho(A) = |A|$ и $B \subset A$. Благодаря полумодулярности функции ρ имеем

$$\rho(B \cup (A \setminus B)) + \rho(B \cap (A \setminus B)) \leqslant \rho(B) + \rho(A \setminus B).$$

Но $B \cup (A \setminus B) = A$, и $\rho(A) = |A|$, а $B \cap (A \setminus B) = \varnothing$, и $\rho(\varnothing) = 0$. Поэтому, учитывая также свойство $\rho 1$ и то, что $B \subset A$, получаем

$$|A| = \rho(A) \leqslant \rho(B) + \rho(A \setminus B) \leqslant |B| + |A \setminus B| = |B| + |A| - |B| = |A|.$$

Поскольку на концах этой цепочки соотношений стоит одно и то же число, всюду в ней на самом деле выполняются равенства, в частности, $\rho(B) = |B|$. Таким образом, $B \in J$.

Свойство J2 будем доказывать от противного. Пусть $A,\ B \in J,\ |B| = k,\ |A| = k + 1$. Если $B \subset A$, то доказывать нечего. Поэтому можно считать, что $|A \setminus B| \geqslant 2$. Предположим, что для любого элемента $e \in A \setminus B$ неверно, что $B \cup \{e\} \in J$. Тогда $\rho(B \cup \{e\}) \neq k + 1$ и, в силу соотношений $k = \rho(B) \leqslant \rho(B \cup \{e\}) \leqslant |B \cup \{e\}| = k + 1$ и целочисленности функции ρ, имеем $\rho(B \cup \{e\}) = k$. Возьмём в качестве e два различных элемента c и d и пусть $C = B \cup \{c\},\ D = B \cup \{d\}$. Тогда $C \cup D = B \cup \{c, d\}$ и $C \cap D = B$. Поэтому

$$\rho(B \cup \{c, d\}) + \rho(B) = \rho(C \cup D) + \rho(C \cap D) \leqslant \rho(C) + \rho(D) = k + k,$$

откуда $\rho(B \cup \{c,d\}) \leqslant k$. Так как $\rho(B \cup \{c,d\}) \geqslant \rho(B) = k$, получаем $\rho(B \cup \{c,d\}) = k$. Итак, при добавлении к множеству B двух элементов из $A \setminus B$ мы получили множество с прежним значением функции ρ.

Если в множестве $A \setminus B$, кроме c и d, есть ещё, например, элемент f, положим $C = B \cup \{c,d\}$, $D = B \cup \{f\}$ и, повторив предыдущие выкладки, получим, что $\rho(B \cup \{c,d,f\}) = k$.

Добавляя к множеству B последовательно элементы из $A \setminus B$, мы рано или поздно придём к множеству $B \cup A$, и при этом окажется, что $\rho(B \cup A) = k$, в то время как $\rho(B \cup A) \geqslant \rho(A) = k + 1$. Противоречие получено. □

5. Жадный алгоритм

Весьма общей является следующая задача оптимизации.

Пусть каждому элементу e непустого конечного множества E поставлено в соответствие неотрицательное число $w(e)$, называемое **весом** этого элемента. **Вес подмножества** $X \subset E$ определяется как сумма весов его элементов:

$$w(X) = \sum_{e \in X} w(e).$$

Рассматривается некоторая совокупность J подмножеств множества E. Требуется найти в J подмножество максимального веса.

Подобный вид имеют или сводятся к нему многие задачи: например, задача коммивояжёра, задача о рюкзаке, задача о минимальном стягивающем дереве и другие.

Жадный алгоритм решения описанной задачи состоит в последовательном, элемент за элементом, формировании искомого множества, причём на каждом шаге из всех элементов множества E, добавление которых к ранее выбранным возможно (то есть приводит к к некоторому множеству из J), выбирается элемент наибольшего веса.

Формально алгоритм можно описать так.

1. В качестве e_1 выбрать элемент, удовлетворяющий условию

$$w(e_1) = \max_{\{e\} \in J} w(e).$$

Следующий шаг выполнять до тех пор, пока он приводит к расширению формируемого множества S.

2. Если $S = \{e_1, e_2, \ldots, e_{k-1}\}$, то в качестве очередного элемента множества S выбрать элемент e_k такой, что

$$w(e_k) = \max\{w(e) \mid \{e_1, e_2, \ldots, e_{k-1}, e\} \in J,\ e \notin \{e_1, e_2, \ldots, e_{k-1}\}\}.$$

Конкретная реализация жадного алгоритма зависит во многом от того, что представляет собой множество J.

Пример 1. Рассмотрим задачу о назначениях, которая состоит в выделении совершенного паросочетания наименьшего веса в двудольном графе. В данном случае E — множество рёбер двудольного графа, а J — множество всех его паросочетаний. Жадный алгоритм на каждом шаге выбирает ребро наименьшего веса, не смежное с ранее выбранными рёбрами. Весовая функция задаётся матрицей весов рёбер графа. Пусть для графа $K_{3,3}$ матрица весов такова:

$$\begin{pmatrix} 4 & 5 & 5 \\ 5 & 8 & 9 \\ 5 & 9 & 12 \end{pmatrix}.$$

Очевидно, что жадный алгоритм последовательно выберет рёбра веса 4, 8 и 12, в результате чего формируется паросочетание веса 24. Легко проверить, что любое другое совершенное паросочетание имеет меньший вес. Таким образом, *жадная стратегия привела к наихудшему из возможных решений*[2].

Пример 2. В **задаче коммивояжёра** требуется найти замкнутый маршрут наименьшей длины, проходящий через заданные города. Здесь E — множество дорог между городами, в роли веса дороги выступает её длина. Жадная стратегия для задачи коммивояжёра состоит в том, что, начав маршрут в произвольном городе, в качестве очередного города на каждом шаге выбираем такой ранее не посещённый город, к которому ведёт самая короткая дорога.

Пример 3. В задаче о минимальном стягивающем дереве требуется в заданном связном взвешенном графе $G = \langle V, E\rangle$ выделить стягивающее дерево минимального веса. Здесь E — множество рёбер графа, а J состоит из всех его стягивающих деревьев. В §1 гл. 1 описаны два жадных алгоритма решения данной задачи.

Пример 4. **Задача о рюкзаке.** Имеется рюкзак объёма V и n предметов, для каждого из которых известны объём и стоимость. Тре-

[2]Задачи, при решении которых жадным алгоритмом может возникнуть такая парадоксальная ситуация, называют **антиматроидами.** Подробно об этом написано в статье [15].

буется заполнить рюкзак предметами наибольшей суммарной стоимости. Здесь E — множество предметов, J состоит из наборов предметов, суммарный объём которых не превосходит V. Общее количество допустимых вариантов заполнения рюкзака не больше 2^n. Жадный алгоритм, состоит, очевидно, в том, что на каждом шаге мы помещаем в рюкзак наиболее дорогую вещь из тех, которые ещё можно туда разместить.

Известно, что жадный алгоритм для задачи из примера 3 всегда даёт оптимальное решение, а для задач из примеров 1, 2 и 4 — не всегда.

В этом разделе мы выясним, каким требованиям должна удовлетворять совокупность множеств J, чтобы жадная стратегия приводила к оптимальному решению.

Теорема 17. *Пусть $M = \langle E, J\rangle$ — матроид, а на множестве E определена функция веса $w : E \to \mathbb{R}^+$. Тогда жадный алгоритм выделяет независимое подмножество E наибольшего веса.*

Доказательство. Пусть в результате работы жадного алгоритма сформировано множество $I = \{e_1, e_2, \ldots, e_s\}$. По смыслу алгоритма элементы этого множества проиндексированы в порядке убывания их веса:

$$w(e_1) \geqslant w(e_2) \geqslant \ldots \geqslant w(e_s).$$

Возьмём в E произвольное независимое подмножество $L = \{e'_1, e'_2, \ldots, e'_t\}$ максимального веса, считая,что

$$w(e'_1) \geqslant w(e'_2) \geqslant \ldots \geqslant w(e'_t).$$

Заметим, что множество L максимально по включению среди множеств, входящих в J (иначе его можно расширить, а вес при этом не уменьшится). Множество I максимально по включению по самому смыслу жадного алгоритма. Таким образом, и L, и I — базисы матроида. Значит, как нам уже известно, в них поровну элементов: $t = s$.

Докажем по индукции, что для любого i выполняется неравенство $w(e_i) \geqslant w(e'_i)$. *База индукции* обеспечена первым шагом жадного алгоритма.

Пусть теперь для любого $k < n$ уже установлено, что $w(e_k) \geqslant w(e'_k)$. Нужно доказать, что $w(e_n) \geqslant w(e'_n)$.

Предположим, что это не так: $w(e_n) < w(e'_n)$. Сформируем множество

$$A = \{e \in E \mid w(e) \geqslant w(e'_n)\}.$$

Рассмотрим сужение матроида $M = \langle E, J \rangle$ на множество A — матроид $M' = M|A$. Поскольку

$$w(e_1) \geqslant w(e_2) \geqslant \ldots \geqslant w(e_{n-1}) \geqslant w(e'_{n-1}) \geqslant w(e'_n),$$

множество $B = \{e_1, e_2, \ldots, e_{n-1}\}$ является подмножеством множества A. Это множество независимо (будучи подмножеством независимого множества I) и максимально по включению. Действительно, если в A имеется более широкое независимое подмножество $\{e_1, e_2, \ldots, e_{n-1}, e\}$, то $w(e) \geqslant w(e'_n) > w(e_n)$, и жадный алгоритм должен бы был на n-м шаге включать в формируемое множество вместо элемента e_n элемент e. Итак, B — базис в матроиде M'. Однако, в M' содержится и независимое множество $C = \{e'_1, e'_2, \ldots, e'_n\}$ (оно является подмножеством независимого множества L). Получилось, что в некотором независимом множестве матроида M' элементов больше, чем в его базисе. Противоречие!

Итак, для всех i справедливо неравенство $w(e_i) \geqslant w(e'_i)$. Тогда

$$w(I) = \sum_{i=1}^{s} w(e_i) \geqslant \sum_{i=1}^{s} w(e'_i) = w(L).$$

Значит, I — независимое подмножество максимального веса. □

Доказательство теоремы показывает, что для задачи с матроидной структурой в оптимальном решении I по сравнению с произвольным независимым множеством L больше (если выразиться более аккуратно, не меньше) не только вес всего множества, но также вес i-го по весу элемента для каждого i. Интересной особенностью оптимального решения является также то, что оно целиком определяется упорядочением весов элементов множества E, но не их конкретными значениями.

Здесь же отметим, чем жадные алгоритмы привлекательны для программистов. Во-первых, эти алгоритмы обычно легки для программирования (вспомним, например алгоритм Прима). Во-вторых, они имеют, как правило, полиномиальные оценки трудоёмкости. Это объясняется тем, что искомое множество формируется элемент за элементом, в отличие от алгоритмов типа перебора с возвратом, для которых характерна экспоненциальная трудоёмкость.

Итак, в случае матроидной структуры подмножеств жадный алгоритм приводит к оптимальному решению. Оказывается, справедливо и обратное утверждение, возникающее при естественном предположении о замкнутости системы множеств относительно включения.

Теорема 18. *Пусть J — непустая система подмножеств множества E такая, что выполняется условие J1. Если для любой весовой функции $w : E \to \mathbb{R}^+$ жадный алгоритм находит подмножество E наибольшего веса, то $\langle E, J\rangle$ — матроид.*

Доказательство. Достаточно привести пример весовой функции, для которой жадный алгоритм, применённый к задаче, в которой выполняются условия J0 и J1 и не выполняется условие J2, не даёт оптимального решения.

Пусть подмножества A и B множества E таковы, что $|A| = k + 1$, $|B| = k$, и для любого элемента $e \in A \setminus B$ множество $B \cup \{e\}$ не входит в J. Определим весовую функцию следующим образом:

$$w(e) = \begin{cases} k+2, & \text{если} \quad e \in B; \\ k+1, & \text{если} \quad e \in A \setminus B; \\ 0, & \text{если} \quad e \notin A \cup B. \end{cases}$$

Жадный алгоритм сначала выберет все k элементов множества B, после чего не сможет добавить к нему ни одного элемента ненулевого веса. Таким образом, будет сформировано подмножество веса $(k + 2)k = k^2 + 2k$. С другой стороны, множество A имеет заведомо бóльший вес, поскольку вес каждого из $k+1$ его элементов не меньше, чем $k + 1$, и

$$w(A) \geqslant (k + 1)^2 > k^2 + 2k.$$

Жадный алгоритм не привёл к оптимальному решению задачи. □

6. Одна задача планирования эксперимента

Рассмотрим задачу практического характера, в которой возникает структура матроида.

Пусть некоторый объект подвергается воздействию нескольких независимых факторов. В результате единичного эксперимента можно найти некоторую числовую характеристику объекта, которая является функцией значений указанных факторов. В случае линейной модели эта функция имеет вид:

$$f = c_1x_1 + c_2x_2 + \ldots + c_nx_n,$$

где f — числовая характеристика объекта, n — общее количество факторов, x_i — значения факторов, c_i — коэффициенты, которые надлежит определить в результате проведения серии экспериментов. По

техническим причинам факторы могут встречаться только в определённых комбинациях. Будем считать, что таких комбинаций m, причём $m \geqslant n$.

Пример. Пусть изучается влияние содержания различных минералов в почве на повышение урожайности некоторой зерновой культуры. При этом в распоряжении экспериментаторов имеется m различных удобрений, каждое из которых представляет собой смесь указанных минералов в определённых пропорциях.

В принципе можно провести m различных экспериментов и получить m уравнений:

$$\begin{array}{ccccccccc} a_{11}c_1 & + & a_{12}c_2 & + & \ldots & + & a_{1n}c_n & = & f_1; \\ a_{21}c_1 & + & a_{22}c_2 & + & \ldots & + & a_{2n}c_n & = & f_2; \\ & & & & \ldots & & & & \\ a_{m1}c_1 & + & a_{m2}c_2 & + & \ldots & + & a_{mn}c_n & = & f_m. \end{array}$$

Предположим, что каждый эксперимент имеет свою цену, известную экспериментаторам, а те желают провести наиболее дешёвую серию опытов для определения искомых коэффициентов c_i. Для этого им нужно в матрице $A = (a_{ij})$ выбрать n линейно независимых строк с наименьшим суммарным весом, где вес i-й строки есть стоимость i-го эксперимента. Эта задача разрешима, если ранг этой матрицы равен n; будем предполагать, что это условие выполняется.

Рассмотрим матроид строк матрицы A. Каждый базис матроида состоит из n линейно независимых векторов-строк. Независимое множество наименьшего веса как раз и есть то, что нам требуется: оно содержит строки, определяющие наиболее дешёвую систему экспериментов.

В заключение заметим, что известен вариант жадного алгоритма для матричного матроида, основанный на классическом методе Гаусса приведения матрицы к треугольному виду. Трудоёмкость этого алгоритма — порядка m^2n операций.

7. Трансверсали

Пусть E — непустое конечное множество, $P = (S_1, S_2, \ldots, S_m)$ — некоторая последовательность его непустых подмножеств (они могут пересекаться и даже совпадать). **Трансверсалью** (или **системой различных представителей**) для совокупности множеств P называют множество $T = \{t_1, t_2, \ldots, t_m\}$ такое, что для каждого числа i

элемент t_i принадлежит множеству S_i; при этом при различных i и j элементы t_i и t_j также различны.

Другими словами, трансверсаль состоит из m различных представителей m множеств. Заметим, что при этом представителю одного множества не возбраняется быть членом и других множеств.

Трансверсаль любой подпоследовательности последовательности $P = (S_1, S_2, \ldots, S_m)$ называют **частичной трансверсалью** для P. Пустое множество также будем считать частичной трансверсалью: тогда *любое подмножество частичной трансверсали есть частичная трансверсаль*.

Пример 1. Имеется несколько работников, каждый из которых может выполнять определённое (своё для каждого работника) множество работ. При этом для выполнения каждой работы требуется ровно один человек. Требуется так распределить имеющуюся рабочую силу, чтобы каждая работа выполнялась. Формализуя эту задачу, имеем: E — множество работников, множество S_i состоит из тех работников, которые могут выполнять i-ю работу. Назначения на каждый вид работ сводятся к отысканию трансверсали для последовательности множеств $P = (S_1, S_2, \ldots, S_m)$, где m — общее количество работ.

Пример 2. В некотором учреждении имеется m комиссий. Требуется из состава каждой комиссии назначить их председателей так, чтобы ни один человек не председательствовал более чем в одной комиссии. Здесь трансверсаль комиссий составят их председатели.

Пример 3. Пусть $S_1 = S_2 = \{1, 2\}$, $S_3 = S_4 = \{2, 3\}$, $S_5 = \{1, 2, 3, 4, 5, 6\}$. Легко видеть, что не существует трансверсали для $P = (S_1, S_2, S_3, S_4, S_5)$, однако, например, элементы 1, 2, 3, 6 составляют трансверсаль для последовательности $P' = (S_1, S_2, S_3, S_5)$, то есть частичную трансверсаль для P.

Необходимое и достаточное условие существования трансверсали даёт следующая

Теорема 19. *Пусть E — непустое конечное множество. Последовательность его непустых подмножеств $P = (S_1, S_2, \ldots, S_m)$ имеет трансверсаль тогда и только тогда, когда объединение любых k подмножеств из этой последовательности содержит не менее k элементов, где k — произвольное натуральное число, не превосходящее m.*

Доказательство. Сначала дадим краткую запись условия существования трансверсали из формулировки теоремы:

$$\forall A \subset \{1, 2, \ldots, m\} \quad \Big|\bigcup_{i \in A} S_i\Big| \geqslant |A|. \tag{1}$$

Необходимость данного условия очевидна. Перейдём к достаточности.

Предварительно докажем

Утверждение. *Если в некотором множестве, например, в* S_1, *не менее двух элементов, то из этого множества можно удалить один элемент, не нарушив при этом условия* (1).

От противного: пусть $|S_1| \geqslant 2$ и, какой элемент ни удалить из S_1, условие (1) не будет выполнено. Возьмём два элемента x и y из множества S_1. Для них найдутся такие множества индексов $A' = \{1\} \cup A$ и $B' = \{1\} \cup B$, где $A, B \subset \{2, 3, \ldots, m\}$, что

$$\Big|\bigcup_{i\in A} S_i \cup (S_1 \setminus \{x\})\Big| < |A'| = |A|+1 \text{ и } \Big|\bigcup_{i\in B} S_i \cup (S_1 \setminus \{y\})\Big| < |B'| = |B|+1. \tag{2}$$

Положим:

$$X = \bigcup_{i\in A} S_i \cup (S_1 \setminus \{x\}),\ Y = \bigcup_{i\in B} S_i \cup (S_1 \setminus \{y\}).$$

Соотношения (2) перепишем в виде:

$$|X| \leqslant |A|;\ \ |Y| \leqslant |B|,$$

откуда

$$|X| + |Y| \leqslant |A| + |B|. \tag{3}$$

Для дальнейшего нам понадобится следующий вариант формулы включения-исключения:

$$|C| + |D| = |C \cup D| + |C \cap D|, \tag{4}$$

где C и D — произвольные множества.

С помощью условия (1) оценим снизу мощности объединения и пересечения множеств X и Y. Поскольку

$$X \cup Y = \bigcup_{i\in A\cup B} S_i \cup (S_1 \setminus \{x\}) \cup (S_1 \setminus \{y\}) = \bigcup_{i\in A\cup B} S_i \cup S_1,$$

выполняется неравенство

$$|X \cup Y| \geqslant |A \cup B| + 1. \tag{5}$$

В силу того, что

$$X \cap Y \supset \bigcup_{i\in A\cap B} S_i,$$

имеем

$$|X \cap Y| \geqslant |A \cap B|. \tag{6}$$

Сложим неравенства (5) и (6), дважды использовав тождество (4):

$$|X| + |Y| = |X \cup Y| + |X \cap Y| \geqslant |A \cup B| + |A \cap B| + 1 = |A| + |B| + 1.$$

Полученное противоречие с неравенством (3) завершает доказательство утверждения.

Будем применять процедуру из утверждения до тех пор, пока у нас не останутся лишь одноэлементные множества. При этом в объединении любых k из них содержится k элементов. Значит, все эти множества попарно не пересекаются, а их объединение и есть искомая трансверсаль. □

Замечание. Искушённый читатель, наверное, сразу узнал в теореме о существовании трансверсали теорему Холла. Действительно, построим двудольный граф с долями $V_1 = \{v_1, v_2, \ldots, v_m\}$ и $V_2 = S_1 \cup S_2 \cup \ldots \cup S_m$, в котором для каждого i множество S_i есть множество вершин, смежных с вершиной v_i. Тогда трансверсаль задаёт совершенное паросочетание из V_1 в V_2. В матримониальной трактовке теоремы Холла множество S_i состоит из девушек, знакомых i-му юноше, а трансверсаль есть множество счастливых невест. Идея изложенного выше доказательства, принадлежащего Р. Радо, позволяет получить более общий результат. Об этом пойдёт речь в §9. А сейчас установим два следствия из доказанной теоремы.

Следствие 1. *Пусть E — непустое конечное множество. Последовательность его непустых подмножеств $P = (S_1, S_2, \ldots, S_m)$ имеет частичную трансверсаль мощности t тогда и только тогда, когда объединение любых k подмножеств из этой последовательности содержит не менее $k + t - m$ элементов, где k — произвольное натуральное число, не превосходящее m, т.е.*

$$\forall A \subset \{1, 2, \ldots, m\} \quad |\bigcup_{i \in A} S_i| \geqslant |A| + t - m.$$

Доказательство. Чтобы иметь возможность применить утверждение теоремы, возьмём множество D, имеющее мощность $m - t$ и не пересекающееся с E, и образуем новое семейство множеств $P' = (S'_1, S'_2, \ldots, S'_m)$, где $S'_i = S_i \cup D$. С помощью $m - t$ элементов множества D любая частичная трансверсаль мощности t дополняется до (полной) трансверсали. Обратно: если имеется трансверсаль для P', то выбросив из неё элементы множества D, получим частичную трансверсаль мощности не меньше t, а, значит, есть и частичная трансверсаль, мощность которой равна t. Таким образом, существование частичной трансверсали мощности t для

P равносильно существованию полной трансверсали для P', а последнее имеет место тогда и только тогда, когда

$$\forall A \subset \{1,2,\ldots,m\} \quad |\underset{i\in A}{\cup} S'_i| = |\underset{i\in A}{\cup} S_i \cup D| = |\underset{i\in A}{\cup} S_i| + m - t \geqslant |A|.$$

Доказательство следствия 1 завершено. □

Следствие 2. *Пусть E — непустое конечное множество, $P = (S_1, S_2, \ldots, S_m)$ — последовательность его непустых подмножеств. Множество $X \subset E$ содержит частичную трансверсаль мощности t для P тогда и только тогда, когда*

$$\forall A \subset \{1,2,\ldots,m\} \quad |(\underset{i\in A}{\cup} S_i) \cap X| \geqslant |A| + t - m.$$

Доказательство. Положим $S'_i = S_i \cap X$ (для каждого i) и применим к последовательности $P' = (S'_1, S'_2, \ldots, S'_m)$ предыдущее следствие. Получим условие

$$\forall A \subset \{1,2,\ldots,m\} \quad |\underset{i\in A}{\cup} S'_i| \geqslant |A| + t - m.$$

Но $\cup S'_i = \cup(S_i \cap X) = (\cup S_i) \cap X$. □

8. Трансверсальный матроид

Теорема 20. (Дж. Эдмондс, Д. Фалкерсон, 1965 г.)
Пусть E — непустое конечное множество, $P = (S_1, S_2, \ldots, S_m)$ — некоторая последовательность его непустых подмножеств, а J — множество всех частичных трансверсалей для P. Тогда $\langle E, J\rangle$ — матроид.

Доказательство. Свойства J0 и J1, очевидно, имеют место. Проверим выполнение аксиомы независимости J2.

Прибегнем к наглядному представлению семейства множеств $P = (S_1, \ldots, S_m)$ с помощью двудольного графа G, который строится следующим образом. Вершины первой доли V_1 будут соответствовать множествам $S_1, \ldots, S_m$, вершины второй доли V_2 — элементам множества E, и (для каждого i) рёбра, инцидентные вершине S_i, соединяют её со всеми элементами множества S_i. Выделению системы различных представителей соответствует некоторое паросочетание (т.е. множество попарно несмежных рёбер) в этом графе: если элемент t_i представляет множество S_i, то в паросочетание войдёт ребро $S_i t_i$.

Заметим, кстати, что в случае трансверсали имеем совершенное паросочетание из V_1 в V_2.

Пусть A и B — частичные трансверсали и $|A| = |B| + 1$. Нужно доказать, что найдётся элемент $e \in A \setminus B$ такой, что $B \cup \{e\}$ — частичная трансверсаль. Паросочетания, отвечающие указанным частичным трансверсалям, обозначим соответственно через W_A и W_B. Покрасим рёбра из $W_A \setminus W_B$ в красный цвет, из $W_B \setminus W_A$ — в синий, а из $W_A \cap W_B$ — в зелёный. Красных рёбер будет на одно больше, чем синих. Заметим также, что зелёное ребро не смежно ни с одним покрашенным ребром.

Рассмотрим подграф G' исходного графа, образованный красными и синими рёбрами. Так как два ребра одного цвета не могут быть смежны, степень каждой вершины в G' равна 1 или 2. Легко видеть, что компоненты связности G' представляют собой циклы и цепи. В каждом цикле и каждой цепи цвета рёбер чередуются. Поэтому в цикле, а также цепи чётной длины одинаковое количество красных и синих рёбер. Поскольку красных рёбер больше, чем синих, найдётся цепь C нечётной длины $v_1 \to v_2 \to \ldots \to v_{2k}$, в которой первое и последнее ребро — красные. Ровно одна из концевых вершин цепи C лежит в V_2, пусть это вершина v_1. Эта вершина инцидентна только красному ребру и изображает элемент из $A \setminus B$. Каждая из вершин $v_3, v_5, \ldots, v_{2k-1}$ изображает элемент множества E и инцидентна как красному, так и синему ребру. Значит, $\{v_3, v_5, \ldots, v_{2k-1}\} \subset A \cap B$. Перекрасим теперь цепь C, заменив красный цвет синим, а синий красным.

Вернёмся к графу G. В результате произведённой перекраски множество вершин из V_2, покрытых синими и зелёными рёбрами, пополнилось вершиной v_1, то есть частичная трансверсаль B удлинилась за счёт элемента из $A \setminus B$. □

Матроид, образованный частичными трансверсалями фиксированного семейства множеств P, будем называть **трансверсальным** и обозначать $M[P]$.

Приведём пример практической задачи, в которой возникает трансверсальный матроид.

Пример. Имеется m работников; i-й работник может выполнять работы из множества работ S_i, где $i = 1, 2, \ldots, m$. Пусть общее множество работ $E = \cup S_i = \{e_1, e_2, \ldots, e_n\}$, а прибыль от выполнения работы e_i равна $w(e_i)$. Требуется так распределить работы между работниками (при этом каждый выполняет не более одной работы, а для каждой работы нужен один работник; какие-то рабо-

ты могут оказаться невыполненными, а какие-то работники могут остаться без работы), чтобы общая прибыль от выполнения работ была максимальной. Математически, задача сводится к отысканию частичной трансверсали наибольшего веса для последовательности множеств $P = (S_1, S_2, \ldots, S_m)$. Другими словами, нужно найти независимое множество наибольшего веса в трансверсальном матроиде. Для этого, как нам уже известно, годится жадный алгоритм.

В заключение раздела, используем понятие трансверсального матроида для решения одной теоретической задачи.

Выясним, *каким требованиям должны удовлетворять множество $A \subset E$ и семейство множеств P, чтобы первое можно было дополнить до трансверсали второго*, т.е. чтобы семейство P имело трансверсаль, содержащую множество A. Очевидно, необходимыми являются следующие условия:

1. P имеет хотя бы одну трансверсаль.
2. A — частичная трансверсаль для P.

Удивительно, но эти условия являются и достаточными. **Доказательство** проводится очень просто, если опираться на теорию матроидов. Действительно, множество A, будучи частичной трансверсалью, является независимым множеством трансверсального матроида. Любое независимое множество можно расширить до базиса. Все базисы матроида имеют одну и ту же мощность. В силу условия 1 мощность базиса равна m. Значит, базис суть трансверсаль. □

9. Независимые трансверсали

Ранее мы установили необходимое и достаточное условие существования трансверсали для семейства подмножеств $P = (S_1, S_2, \ldots, S_m)$ множества E. Теперь пусть на множестве E задан некоторый матроид. **Независимой трансверсалью** для P назовём трансверсаль, которая является независимым множеством в смысле указанного матроида. В частности, если матроид — дискретный, то любая трансверсаль — независимая. Следующая теорема даёт критерий существования независимой трансверсали.

Теорема 21. (Р. Радо, 1942 г.) *Пусть $M = \langle E, J \rangle$ — матроид. Последовательность $P = (S_1, S_2, \ldots, S_m)$ непустых подмножеств множества E имеет независимую трансверсаль тогда и только тогда, когда объединение любых k подмножеств из этой последовательности содержит независимое множество, в котором не менее*

k элементов, где k — произвольное натуральное число, не превосходящее m.

Доказательство. Условие теоремы удобно сформулировать, используя понятие ранга множества (наибольшей мощности независимого подмножества):

$$\forall A \subset \{1, 2, \ldots, m\} \quad \rho(\bigcup_{i \in A} S_i) \geqslant |A|. \tag{1}$$

Необходимость. Если имеется независимая трансверсаль, то её пересечение с множеством $\cup_{i \in A} S_i$ имеет $|A|$ элементов, откуда $\rho(\bigcup_{i \in A} S_i) \geqslant |A|$.

Достаточность. Предварительно докажем

Утверждение. *Если в некотором множестве (например, в S_1) не менее двух элементов, то из этого множества можно удалить один элемент, не нарушив при этом условия (1).*

От противного: пусть $|S_1| \geqslant 2$ и, какой элемент ни удалить из S_1, условие (1) не будет выполнено. Возьмём два элемента x и y из множества S_1. Для них найдутся такие множества индексов $A' = \{1\} \cup A$ и $B' = \{1\} \cup B$, где $A, B \subset \{2, 3, \ldots, m\}$, что

$$\rho(\bigcup_{i \in A} S_i \cup (S_1 \setminus \{x\})) < |A'| = |A| + 1 \text{ и } \rho(\bigcup_{i \in B} S_i \cup (S_1 \setminus \{y\})) < |B'| = |B| + 1. \tag{2}$$

Положим:

$$X = \bigcup_{i \in A} S_i \cup (S_1 \setminus \{x\}),\ Y = \bigcup_{i \in B} S_i \cup (S_1 \setminus \{y\}).$$

Соотношения (2) перепишем в виде:

$$\rho(X) \leqslant |A|; \quad \rho(Y) \leqslant |B|,$$

откуда

$$\rho(X) + \rho(Y) \leqslant |A| + |B|. \tag{3}$$

С помощью условия (1) оценим снизу ранги объединения и пересечения множеств X и Y. Поскольку

$$X \cup Y = \bigcup_{i \in A \cup B} S_i \cup (S_1 \setminus \{x\}) \cup (S_1 \setminus \{y\}) = \bigcup_{i \in A \cup B} S_i \cup S_1,$$

выполняется неравенство

$$\rho(X \cup Y) \geqslant |A \cup B| + 1. \tag{4}$$

В силу того, что

$$X \cap Y \supset \bigcup_{i \in A \cap B} S_i,$$

имеем

$$\rho(X \cap Y) \geqslant |A \cap B|. \tag{5}$$

Используя свойство полумодулярности ранговой функции, после сложения (4) и (5) получим:

$$\rho(X)+\rho(Y) \geqslant \rho(X \cup Y)+\rho(X \cap Y) \geqslant |A \cup B|+|A \cap B|+1 = |A|+|B|+1. \tag{6}$$

Неравенство (6) противоречит неравенству (3). Утверждение доказано.

Будем применять процедуру из утверждения до тех пор, пока у нас не останется m одноэлементных множеств $\{t_1\}, \{t_2\}, \ldots, \{t_m\}$. При этом ранг их объединения $T = \{t_1, t_2, \ldots, t_m\}$ равен m. Значит, T и есть искомая независимая трансверсаль. □

Следствие. *Пусть $M = \langle E, J \rangle$ — матроид. Последовательность $P = (S_1, S_2, \ldots, S_m)$ непустых подмножеств множества E имеет независимую частичную трансверсаль мощности t тогда и только тогда, когда объединение любых k подмножеств из этой последовательности содержит независимое подмножество мощности не менее $k + t - m$, т.е.*

$$\forall A \subset \{1, 2, \ldots, m\} \quad \rho(\bigcup_{i \in A} S_i) \geqslant |A| + t - m.$$

Доказательство вполне аналогично доказательству следствия 1 из теоремы 19.

10. Общие трансверсали

Критерий существования независимой трансверсали позволяет получить необходимое и достаточное условия существования общей трансверсали у двух различных систем подмножеств одного и того же множества. Имеет место

Теорема 22. *Два семейства $P = (S_1, S_2, \ldots, S_m)$ и $Q = (R_1, R_2, \ldots, R_m)$ непустых подмножеств конечного множества E обладают общей трансверсалью тогда и только тогда, когда для любых подмножеств A и B множества $\{1, 2, \ldots, m\}$ выполняется неравенство*

$$|(\bigcup_{i \in A} S_i) \cap (\bigcup_{i \in B} R_i)| \geqslant |A| + |B| - m.$$

Доказательство. Рассмотрим матроид частичных трансверсалей для P. Общая трансверсаль P и Q есть независимая (в указанном матроиде) трансверсаль Q. По теореме Радо независимая трансверсаль Q существует в том и только том случае, когда объединение любых k множеств R_i содержит независимое множество из k элементов, которое в нашем случае суть частичная трансверсаль мощности k. Применяя следствие 2 из теоремы 19, имеем

$$\forall A, B \subset \{1, 2, \ldots, m\} \quad |(\bigcup_{i \in A} S_i) \cap X| \geqslant |A| + k - m,$$

где $X = \bigcup\limits_{i \in B} R_i, \ k = |B|$. □

Покажем, как свести нахождение общей трансверсали к нахождению максимального потока в сети.

Итак, имеем множество $E = \{e_1, e_2, \ldots, e_n\}$ и два семейства его подмножеств $P = (S_1, S_2, \ldots, S_m)$ и $Q = (R_1, R_2, \ldots, R_m)$. Построим ориентированный граф, содержащий следующие вершины:

- a — источник, b — сток;
- вершины, изображающие подмножества S_1, S_2,..., S_m, R_1, R_2,..., R_m;
- по две вершины на каждый элемент множества E: $v'_1, v''_1, v'_2, v''_2, \ldots, v'_n, v''_n$

и следующие дуги:

- aS_i и $R_i b$ для $i = 1, 2, \ldots, m$;
- $S_i v'_j$, если $e_j \in S_i$;
- $v''_j R_i$, если $e_j \in R_i$;
- $v'_i v''_i$ для $j = 1, 2, \ldots, n$.

Положим пропускные способности всех дуг равными единице. Если существует общая трансверсаль $t_1, t_2, \ldots, t_m$ для P и Q, то в построенном графе есть m непересекающихся путей из источника в сток вида

$$a \to S_i \to v'_k \to v''_k \to R_j \to b,$$

где элемент e_k является представителем множеств S_i и R_j. Эти m путей формируют максимальный поток (его величина m) в построенной транспортной сети. После нахождения максимального потока легко указать общую трансверсаль.

Пример. Для множеств $S_1 = \{1, 2\}$, $S_2 = \{1, 2, 3, 4\}$, $S_3 = \{2, 4\}$, $R_1 = \{2, 3\}$, $R_2 = \{1, 4\}$, $R_3 = \{1, 2, 3\}$ имеем сеть (с единичными пропускными способностями всех дуг), изображённую на рис. 15.

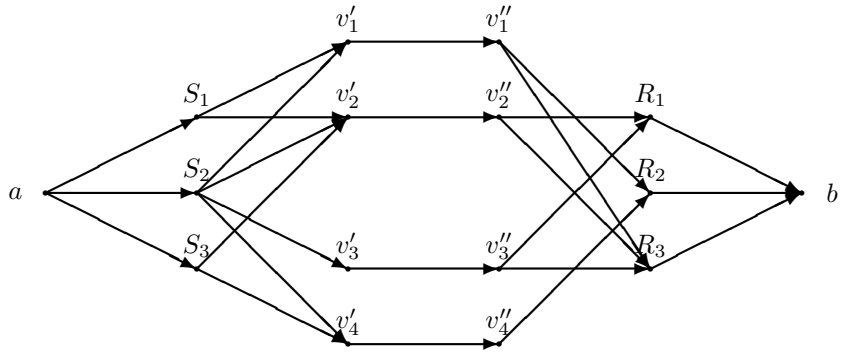


Рис. 15

После нахождения максимального потока можно увидеть общую трансверсаль $\{1, 2, 3\}$, причём 1 является представителем множеств S_1 и R_2, 2 представляет S_3 и R_1, а 3 — множества S_2 и R_3.

Глава 4

Сложность алгоритмов

1. О вычислительной сложности алгоритмов

Рост могущества вычислительной техники иногда порождает иллюзию, что компьютерам всё подвластно: за счёт колоссального быстродействия они могут, используя даже самый примитивный алгоритм (например, прямого перебора), решить любую задачу.

Пусть n — параметр, определяющий размерность задачи (например, число городов в задаче коммивояжёра), а $f(n)$ — функция временной сложности (число элементарных шагов) алгоритма при решении этой задачи.

Приведём несколько примеров функций временной сложности алгоритмов.

В задаче коммивояжёра с n городами прямой перебор требует рассмотрения $(n-1)!$ различных маршрутов. Метод ветвей и границ требует в среднем $1{,}26^n$ операций (см. [13]).

В задаче нахождения минимального стягивающего дерева в полном графе с n вершинами (напомним, это математическая модель задачи нахождения наиболее дешёвой сети дорог, связывающих некоторые населённые пункты) при полном переборе рассматривается n^{n-2} различных деревьев (теорема Кэли). Алгоритм Прима требует порядка $n^2/2$ операций.

В монографии [4], вышедшей в 1982 г., приводится таблица, которая показывает время работы алгоритма в зависимости от функции временной сложности и размерности задачи в предположении, что за 1 секунду выполняется 1 млн шагов алгоритма. С учётом роста быстродействия современных компьютеров примем, что за одну секунду выполняется 1 млрд операций, и после несложного пересчёта получим такую таблицу:

Время работы различных алгоритмов

$f(n) \setminus n$	10	20	50	60
n	10^{-8} с	$2 \cdot 10^{-8}$ с	$5 \cdot 10^{-8}$ с	$6 \cdot 10^{-8}$ с
n^2	10^{-7} с	$4 \cdot 10^{-7}$ с	$25 \cdot 10^{-7}$ с	$36 \cdot 10^{-7}$ с
n^3	10^{-6} с	$8 \cdot 10^{-6}$ с	$125 \cdot 10^{-6}$ с	$216 \cdot 10^{-6}$ с
2^n	10^{-6} с	10^{-3} с	13 дней	36,6 лет
3^n	$5{,}9 \cdot 10^{-6}$ с	3,5 с	$2 \cdot 10^5$ столетий	$1{,}3 \cdot 10^{10}$ столетий

Следующая таблица показывает влияние роста быстродействия ЭВМ на наибольшую размерность задачи, которую можно решить за фиксированный промежуток времени.

Размерность задачи, разрешимой за час

$f(n)$	Современная ЭВМ	ЭВМ, в 100 раз более быстрая	ЭВМ, в 1000 раз более быстрая
n	N_1	$100N_1$	$1000N_1$
n^2	N_2	$10N_2$	$31{,}6N_2$
n^3	N_3	$4{,}64N_3$	$10N_3$
2^n	N_4	$N_4 + 6{,}64$	$N_4 + 9{,}97$
3^n	N_5	$N_5 + 4{,}19$	$N_5 + 6{,}29$

Эта таблица (также, как и предыдущая) вскрывает принципиальное отличие полиномиальных алгоритмов (время работы которых есть многочлен от размерности задачи) от экспоненциальных. Например, если трудоёмкость алгоритма равна n^2, то при повышении скорости работы ЭВМ в 1000 раз размерность разрешимой задачи увеличивается в 32 раза, а в случае трудоёмкости 2^n всего на 10 единиц.

Таким образом, исследователь, собирающийся применить тот или иной алгоритм для решения возникающей перед ним задачи, должен иметь представление представление о трудоёмкости этого алгоритма.

2. Задача выбора

Пусть имеется некоторое конечное множество (допустимых решений) F и некоторый предикат $P(f)$. **Задача выбора в слабом смысле** состоит в нахождении решения $f \in F$ такого, что $P(f)$ — истина, либо в обнаружении того, что решения f, обладающего свойством $P(f)$, нет. В **задаче выбора в сильном смысле** требуется найти всю область истинности предиката $P(f)$.

Различают **индивидуальную** задачу выбора, в которой выбирается нужное решение из некоторого фиксированного множества и **массовую** задачу выбора, состоящую из индивидуальных задач, порождаемых одинаковым образом.

В **задаче дискретной оптимизации** свойство $P(f)$ означает минимальность или максимальность некоторого функционала стоимости решения (целевой функции)

$$c : F \to \mathbb{R}.$$

Например, в случае задачи минимизации

$$P(\hat{f}) = \left(\forall f \in F \quad c(\hat{f}) \leqslant c(f)\right).$$

Как представляется индивидуальная задача оптимизации на входе ЭВМ? Будем считать, что F и c заданы неявно с помощью алгоритмов $\mathcal{A}_F$ и $\mathcal{A}_c$.

Алгоритм $\mathcal{A}_F$ по комбинаторному объекту f и некоторому множеству параметров S определяет, является ли f элементом F. Алгоритм $\mathcal{A}_c$ по допустимому решению f и множеству параметров Q вычисляет $c(f)$.

Индивидуальная задача оптимизации задаётся представлением множеств S и Q с помощью некоторого стандартного способа кодирования с использованием некоторого фиксированного алфавита.

Примеры.

1. Задача коммивояжёра с n городами. Множество S состоит только из параметра n; а Q содержит элементы матрицы расстояний $(c_{i,j})$ между городами. Алгоритм $\mathcal{A}_F$ по объекту f определяет, является ли f циклической перестановкой чисел $1, 2, \ldots, n$. Алгоритм $\mathcal{A}_c$ по маршруту f и матрице $(c_{i,j})$ находит длину маршрута.

2. Минимальное стягивающее дерево. Множество S и Q содержат параметры, описывающие соответственно множество рёбер и их веса.

3. Задача о максимальной клике [1]. Параметры из множества S описывают граф G, а множество Q пусто. $\mathcal{A}_F$ определяет, является ли множество вершин f кликой; $\mathcal{A}_c$ находит мощность f.

[1] Клика — множество попарно смежных вершин.

4. Задача целочисленного линейного программирования.

   $c'x \to \min$, если $Ax \leqslant b, x \in \mathbb{Z}_+^n$.

   Здесь S состоит из матрицы A и столбца b, а Q содержит столбец c. Для данного целочисленного вектра x алгоритм $\mathcal{A}_F$ проверяет выполнение условий $Ax = b$ и $x \geqslant 0$. Алгоритм $\mathcal{A}_c$ вычисляет стоимость $c'x$ допустимого решения.

Чаще всего алгоритмы $\mathcal{A}_F$ и $\mathcal{A}_c$ полиномиальны.

Варианты задачи оптимизации

I. Оптимизационный. Найти оптимальное решение (т. е. допустимое решение $\hat{f} \in F$, максимизирующее или минимизирующее целевую функцию).

II. Вычислительный. Найти стоимость оптимального решения.

III. Вариант распознавания. По числу $L \in \mathbb{Z}$ определить, существует ли решение $f \in F$ такое, что $c(f) \leqslant L$ (если решается задача минимизации), либо $c(f) \geqslant L$ (в случае задачи максимизации).

В предположение малой трудоёмкости $\mathcal{A}_c$ III не сложнее II, II не сложнее I.

Если $\log_2 c(\hat{f}) = O(P(n))$, где n — «размер» задачи, то задача II сводится к задаче III за полиномиальное время с помощью бинарного поиска. Действительно, если известно, что $0 \leqslant c(\hat{f}) \leqslant b$, где $c(\hat{f}) \in Z$, то получая ответы на вопросы типа: «Верно ли, что $c(\hat{f}) \leqslant d$?», можно с каждым шагом сужать вдвое промежуток, которому принадлежит число $c(\hat{f})$, и за $\log_2 b$ шагов найти это число.

В некоторых случаях и задачу I можно свести к задаче II.

Пример. Покажем, как задачу нахождения максимальной клики в графе свести к вычислительному варианту этой задачи.

Процедура max_clique (рис. 16) — рекурсивная: после нахождения вершины v, входящей в некоторую максимальную клику, она обращается к самой себе, оставляя в графе лишь вершины, смежные с вершиной v.

Пусть $T(n)$ и $C(n)$ — время работы процедур max_clique и size_of_clique соответственно. Первая из них, работая с графом, в котором n вершин, $n + 1$ раз обращается ко второй, а затем вызывает саму себя, передевая в качестве аргумента процедуры граф с меньшим числом вершин. Поэтому

$$T(n) \leqslant (n+1)C(n) + O(n) + T(n-1) \leqslant (n+1)C(n) +$$

```
/* Нахождение максимальной клики в графе G */

/* size_of_clique(G) - размер максимальной клики графа G; */
/* G(v) - подграф G, порождённый v и смежными с нею вершинами; */
max_clique(graph G){
      if(G==∅)return(∅);
      else { найти такую вершину v, что
             size_of_clique(G(v))=size_of_clique(G);
             return({v}∪ max_clique(G(v)\ v));
      }
}
```

Рис. 16

$$+ nC(n-1) + 2O(n) + T(n-2) \leqslant \ldots \leqslant \frac{(n+2)(n+1)}{2} C(n) + O(n^2),$$

откуда $T(n) = O(n^2 \cdot C(n))$. Таким образом, если бы за полиномиальное время можно было найти размер максимальной клики, то это же было бы справедливо и относительно нахождения самой максимальной клики.

3. Классы P и NP

В предыдущем параграфе мы говорили о распознавательном варианте задачи дискретной оптимизации. Однако многие интересные задачи дискретной математики уже по своей постановке являются **задачами распознавания**: в них относительно некоторого объекта нужно определить, обладает ли он некотором свойством. Таковыми являются, например, следующие задачи:

- СВЯЗНОСТЬ: является ли данный граф связным;
- ГАМИЛЬТОНОВОСТЬ: является ли данный граф гамильтоновым;
- ПРОСТОЕ ЧИСЛО: является ли заданное число простым;

- ВЫПОЛНИМОСТЬ: существует ли набор значений переменных, на котором заданная КНФ (конъюктивная нормальная форма) принимает значение ИСТИНА.

В теории сложности вычислений рассматривают именно задачи распознавания.

P — множество задач распознавания, для каждой из которых существует полиномиальный алгоритм решения. Другими словами, P — класс задач, решаемых эффективными алгоритмами.

К классу P относятся такие известные задачи, как СВЯЗНОСТЬ, МИНИМАЛЬНОЕ СТЯГИВАЮЩЕЕ ДЕРЕВО, МАКСИМАЛЬНОЕ ПАРОСОЧЕТАНИЕ.

*Задача распознавания A входит в класс NP , если существуют такие многочлены $p_1(n)$ и $p_2(n)$ и алгоритм $\mathcal{A}$ (***алгоритм проверки удостоверения***), что индивидуальная задача A, задаваемая строкой символов x, имеет ответ «да» тогда и только тогда, когда существуют такая строка $q(x)$ (***удостоверение***), длина которой есть $O(p_1(|x|))$ ($|x|$ — количество символов в строке x), что алгоритм $\mathcal{A}$, имея на входе строку $x\#q(x)$ (здесь $\#$ — некоторый дополнительный символ), приходит к ответу «да» после не более $p_2(|x|)$ шагов.*

Таким образом, принадлежность задачи распознавания классу NP означает возможность по некоторому *краткому удостоверению* за полиномиальное время проверить справедливость утвердительного ответа (если он действительно таков) к этой задаче.

Примеры.

КЛИКА. В распознавательном варианте эта задача ставится так: существует ли в данном графе G клика мощностью не меньше L? В строке x здесь представлена информация о графе, а в качестве удостоверения $q(x)$ естественным образом выступает список вершин, составляющих клику. Алгоритм $\mathcal{A}$ проверяет их попарную смежность.

ЗАДАЧА КОММИВОЯЖЕРА. В варианте распознавания требуется определить, существует ли в данном графе G гамильтонов цикл веса не больше L. Удостоверением будет служить последовательность вершин графа в некотором гамильтоновом цикле нужного веса. Алгоритм $\mathcal{A}$ находит вес этого цикла и сравнивает его с числом L.

ВЫПОЛНИМОСТЬ. Очевидно, удостоверением будет набор значений истинности логических переменных. Алгоритм $\mathcal{A}$ вычисляет значение истинности КНФ на этом наборе.

СОВЕРШЕННОЕ ПАРОСОЧЕТАНИЕ. В распознавательном варианте этой задачи требуется определить, существует ли в заданном двудольном графе совершенное паросочетание. В качестве удостове-

рения естественно предъявить список рёбер, попарную несмежность которых проверит алгоритм $\mathcal{A}$.

Во всех приведённых задачах удостоверением фактически служил ответ. Но так бывает не всегда! Если в задаче спрашивается не о выполнении, а, наоборот, о невыполнении некоторого свойства, то что может быть в роли удостоверения к ответу «да»? Рассмотрим пример.

ОТСУТСТВИЕ СОВЕРШЕННОГО ПАРОСОЧЕТАНИЯ. Если в двудольном графе $G = \langle V_1, V_2 \rangle$ нет совершенного паросочетания, то, по теореме Холла, существует такое множество вершин $A \subset V_1$, окружение которого $\Gamma(A)$ имеет меньшую мощность, само множество A. Значит, в качестве удостоверения разумно предъявить это множество A. [2]

Докажем, что $P \subset NP$.

Действительно, если некоторый алгоритм $\mathcal{B}$ решения задачи A работает полиномиальное время, то запись работы этого алгоритма имеет полиномиальную длину и может выступать в роли удостоверения, по которому алгоритм проверки $\mathcal{A}$ будет убеждаться в правильности работы алгоритма $\mathcal{B}$.

Все *разумные* дискретные задачи входят в NP.

---

[2] Приведём длинную цитату из книги [8]. «При дворе короля Артура жили 150 рыцарей и 150 дам. Король хотел переженить их, но беда была в том, что некоторые из них настолько сильно ненавидели друг друга, что о женитьбе нечего было и говорить! Король Артур неоднократно пытался поделить их на пары, но всякий раз нарывался на скандал. Он призвал чародея Мерлина и повелел ему организовать пары так, чтобы все согласились пожениться. Мерлин обладал сверхъестественными способностями и сразу же увидел, что ни одно из 150! возможных делений на пары не является приемлемым, и сказал об этом королю. Однако Мерлин был не только великим чародеем, но и не совсем искренней личностью, поэтому полностью ему король не доверял. «Найди мне разделение на пары, или я прикажу посадить тебя навечно в пещеру!» – сказал Артур. К счастью для Мерлина, он смог использовать свои сверхъестественные способности, чтобы найти способ продемонстрировать, почему такое разделение на пары неосуществимо. Он попросил 56 дам встать с одной стороны от короля и 95 рыцарей с другой стороны и спросил: «Есть ли среди вас хоть одна дама, которая хотела бы выйти замуж за кого-нибудь из этих рыцарей?» И когда все они ответили: «Нет», Мерлин сказал: «О, Король! Как ты сможешь приказать мне найти мужа для каждой из 56 дам среди оставшихся 55 рыцарей?» Так король, чьё изысканное образование не позволяло прибегать к принуждению, увидел, что на этот раз Мерлин говорил правду, и милостиво простил его.»

4. Полиномиальная сводимость

Пусть A — некоторая (массовая) задача распознавания. Через $A(x)$ будем обозначать индивидуальную задачу A с входными данными, представленными строкой символов x.

Задача A_1 **сводится** к задаче A_2 (обозначение: $A_1 \propto A_2$), если

$$\exists A \in P \;\; \forall x \;\; A_1(x) = A_2(A(x)).$$

Подробнее: A_1 сводится к A_2, если по произвольной строке x за полиномиальное (относительно $|x|$) время можно построить такую строку y, что задача $A_1(x)$ имеет такой же ответ, что и задача $A_2(y)$. Примеры сведения одних задач к другим приведены в следующем параграфе.

Легко видеть, что отношение (полиномиальной) сводимости $\propto$ является транзитивным. Действительно, если $A_1 \propto A_2$ и $A_2 \propto A_3$, то существуют такие полиномиальные алгоритмы A' и A'', что

$$\forall x, y \;\; A_1(x) = A_2(A'(x)), \;\; A_2(y) = A_3(A''(y)).$$

Отсюда

$$\forall x \;\; A_1(x) = A_3(A''(A'(x)).$$

Осталось заметить, что композиция полиномиальных алгоритмов A' и A'' входит в класс P.

Пусть $A_1 \propto A_2$.

Если $A_2 \in P$, то, очевидно, и $A_1 \in \mathbf{P}$. По закону контрапозиции: если $A_1 \notin P$, то и и $A_2 \notin P$.

Если $A_2 \in NP$, то $A_1 \in NP$.

Задача $A \in NP$ называется **NP-полной**, если произвольная задача $B \in NP$ сводится к A.

5. Примеры NP-полных задач

Если A — NP-полная задача, $A \propto C$ и $C \in NP$, то задача C также является NP-полной. Это утверждение позволяет доказывать NP-полноту какой-либо задачи сведением к ней какой-либо NP-полной задачи.

Первой задачей, для которой была доказана её NP-полнота стала задача ВЫПОЛНИМОСТЬ.

Теорема 23. (С. Кук, Л. Левин, 1971 г.) *Задача ВЫПОЛНИМОСТЬ является NP-полной.*

Суть доказательства теоремы состоит в следующем. Алгоритм решения произвольной задачи из класса NP реализуется некоторой программой машины Тьюринга. Далее составляется КНФ, интерпретирующая работу машины Тьюринга.

Частным случаем задачи ВЫПОЛНИМОСТЬ является задача 3-КНФ, в которой каждая дизъюнкция состоит из трёх литералов.

Теорема 24. *Задача 3-КНФ является NP-полной.*

Доказательство. Покажем, как к задаче 3-КНФ сводится задача ВЫПОЛНИМОСТЬ. Пусть КНФ имеет вид $F = \&C_i$, где $C_i = \vee t_j$, $t_j = x_j^{\sigma_j}$ — литерал (т. е. переменная или её отрицание). Построим 3-КНФ F', выполнимую тогда и только тогда, когда выполнима F. Рассмотрим различные случаи в зависимости от количества литералов в дизъюнкции C_i.

1. $C_i = t$. Положим

$$D_i = (t \vee y \vee z)(\overline{z} \vee \alpha \vee \beta)(\overline{z} \vee \overline{\alpha} \vee \beta)(\overline{z} \vee \alpha \vee \overline{\beta})(\overline{z} \vee \overline{\alpha} \vee \overline{\beta})$$

$$(\overline{y} \vee \alpha \vee \beta)(\overline{y} \vee \overline{\alpha} \vee \beta)(\overline{y} \vee \alpha \vee \overline{\beta})(\overline{y} \vee \overline{\alpha} \vee \overline{\beta}).$$

   Если $D_i = 1$, то $z = 0$ (иначе одна из четырёх дизъюнкций, содержащих $\overline{z}$, примет значение ЛОЖЬ) и $y = 0$ (по аналогичной причине), а $t = 1$. Поэтому выполнимость C_i равносильна выполнимости D_i.

2. $C_i = t_1 \vee t_2$. Положим

$$D_i = (t_1 \vee t_2 \vee z)(\overline{z} \vee \alpha \vee \beta)(\overline{z} \vee \overline{\alpha} \vee \beta)(\overline{z} \vee \alpha \vee \overline{\beta})(\overline{z} \vee \overline{\alpha} \vee \overline{\beta}).$$

   Вновь выполнимость C_i равносильна выполнимости D_i.

3. $C_i = t_1 \vee t_2 \vee t_3$. Здесь ничего делать не нужно: $D_i = C_i$.

4. $C_i = \bigvee\limits_{j=1}^{k} t_j$, где $k > 3$. Положим

$$D_i = (t_1 \vee t_2 \vee y_1)(\overline{y}_1 \vee t_3 \vee y_2)(\overline{y}_2 \vee t_4 \vee y_3) \ldots (\overline{y}_{k-3} \vee t_{k-1} \vee t_k).$$

   Если $t_1 = t_2 = \ldots = t_k = 0$ и $D_i = 1$, то $y_1 = 1$, откуда последовательно получаем $y_2 = 1, \ldots, y_{k-3} = 1$ и $D_i = 0$ — противоречие. Значит, при $C_i = 0$ имеем и $D_i = 0$.

Если $C_i = 1$, то для некоторого j выполняется $t_j = 1$. При $j \leqslant 2$ положим $y_1 = y_2 = \ldots = y_{k-3} = 0$. В случае $j > 2$ формула D_i будет выполнима, если $y_m = 0$ при $m \geqslant j - 1$ и $y_m = 1$ при $m < j - 1$. Таким образом, и в этом случае выполнимость C_i равносильна выполнимости D_i.

Очевидно, что 3-КНФ $F' = \&D_i$ обладает нужным свойством. Осталось заметить, что переход от F к F' занимает линейное время относительно длины исходной КНФ (длина увеличивается не более чем в 27 раз). □

Теорема 25. *Задача КЛИКА является NP-полной.*

Доказательство. Покажем, как к задаче КЛИКА сводится задача 3-КНФ. Каждой дизъюнкции трёх литералов $C = t_i \vee t_j \vee t_k$ сопоставим 7 последовательностей длины n (где n — общее число логических переменных в КНФ), в которых на i-м, j-м и k-м местах стоят такие значения истинности переменных x_i, x_j и x_k, при которых $C = 1$, а на остальных местах стоит символ d. Каждая такая последовательность будет вершиной графа G. Две вершины этого графа соединим ребром, если соответствующие им последовательности могут совпасть (при подходящих заменах символов d на 0 или 1). Если в 3-КНФ m дизъюнкций, то в графе $7m$ вершин. Выполнимость 3-КНФ равносильна существованию в G клики мощности m.□

Теорема 26. *Задача НЕЗАВИСИМОЕ МНОЖЕСТВО является NP-полной.*

Доказательство. Данная задача (распознавательный вариант которой состоит в ответе на вопрос, существует ли в графе m попарно несмежных вершин) сводится к задаче КЛИКА, поскольку независимое множество в графе G будет кликой в $\overline{G}$ (дополнении к графу G). □

В распознавательном варианте задачи ВЕРШИННОЕ ПОКРЫТИЕ требуется выяснить, существует ли в данном графе вершинное покрытие, в котором не более L вершин.

Теорема 27. *Задача ВЕРШИННОЕ ПОКРЫТИЕ является NP-полной.*

Доказательство. И в этом случае задача легко сводится к предыдущей. Достаточно вспомнить, что множество C вершин графа является вершинным покрытием тогда и только тогда, когда дополнение $\overline{C}$ (к этому множеству) является независимым. □

Задача 3-РАСКРАСКА отвечает на вопрос, существует ли раскраска вершин данного графа G в три цвета, при которой смежные вершины имеют разный цвет (другими словами, нужно выяснить, верно ли, что хроматическое число графа $\chi(G) \leqslant 3$).

Теорема 28. *Задача 3-РАСКРАСКА является NP-полной.*

Доказательство. Покажем, как к задаче 3-РАСКРАСКА сводится задача 3-КНФ. Для этого по 3-КНФ построим граф с $7m + 2n + 3$ вершинами. Вершины 0, 1, 2 образуют треугольник. Для каждого i имеется также треугольник из вершин x_i, $\overline{x}_i$ и 2. Каждой дизъюнкции $C_i = y_1 \vee y_2 \vee y_3$ из трёх литералов отвечает подграф G_i (рис. 17).

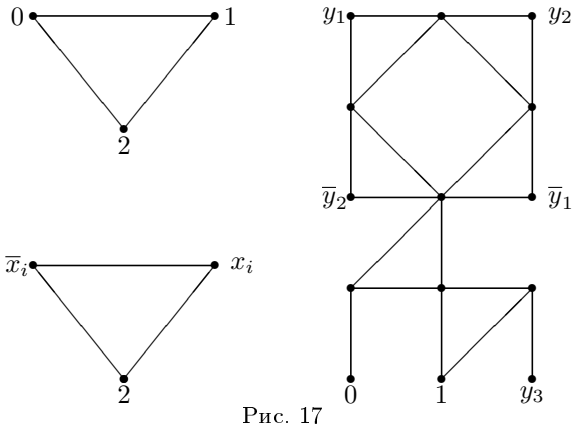


Рис. 17

Пусть вершины 0, 1, 2 покрашены соответственно в цвета 0, 1 и 2. Тогда при правильной раскраске вершин графа в три цвета, вершины, отвечающие переменным и их отрицаниям будут покрашены в цвета 0 и 1, причём вершины x_i и $\overline{x}_i$ будут разного цвета.

Непосредственно проверяется, что правильная раскраска подграфа G_i существует тогда и только тогда, когда $C_i = 1$.

Теорема 29. *Задача ГАМИЛЬТОНОВОСТЬ является NP-полной.*

Доказательство. Покажем, как к задаче ГАМИЛЬТОНОВОСТЬ сводится задача 3-КНФ. Пусть имеется КНФ $F(x_1, x_2, \ldots, x_n) = C_1 \& C_2 \& \ldots \& C_m$, где C_i — дизъюнкция трёх литералов. Построим такой граф, в котором гамильтонов цикл (ГЦ) будет существовать тогда и только тогда, когда формула F выполнима.

Пусть граф A — подграф G, причём из его вершин лишь вершины u, u', v, v' могут быть смежны каким-либо вершинам, не входящим в A (рис. 18, слева).

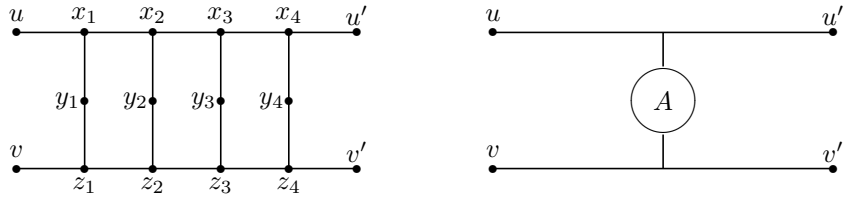


Рис. 18

Тогда если в G есть гамильтонов цикл c, то он проходит через A одним из двух способов. Действительно, для вершины степени 2 оба инцидентных ей ребра обязательно входят в гамильтонов цикл, в силу чего вертикальные отрезки, исходящие из вершин y_1, y_2, y_3 и y_4, входят в цикл c. А концы этих отрезков могут соединяться в цикле c только двумя способами. В одном варианте мы имеем маршрут

$$u \to x_1 \to y_1 \to z_1 \to z_2 \to y_2 \to x_2 \to x_3 \to \ldots \to y_4 \to x_4 \to u',$$

а в другом

$$v \to z_1 \to y_1 \to x_1 \to x_2 \to y_2 \to z_2 \to z_3 \to \ldots \to y_4 \to z_4 \to u'.$$

Таким образом, можно считать, что A состоит из двух рёбер uu' и vv', причём ГЦ содержит ровно одно из этих рёбер. Подграф A мы будем изображать так, как показано на рис. 18 справа.

Граф B (рис. 19, слева) — подграф G, причём из его вершин лишь вершины w_1 и w_4 могут быть смежны каким-либо вершинам, не входящим в B.

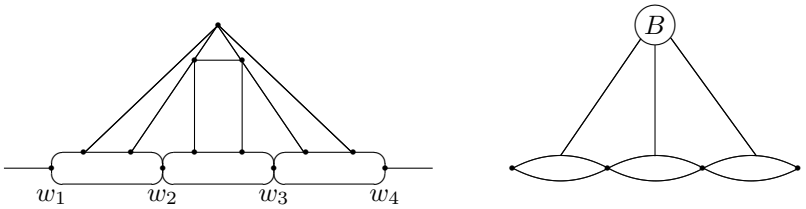


Рис. 19

Тогда если в G есть ГЦ, то он не может одновременно проходить по всем трём рёбрам w_1w_2, w_2w_3 и w_3w_4, но может содержать любые другие их комбинации. Подграф B будем изображать так, как показано на рис. 19 справа.

Граф G строится так (рис. 20).

Каждой дизъюнкции C_i соответствует своя копия графа B. Они соединены последовательно.

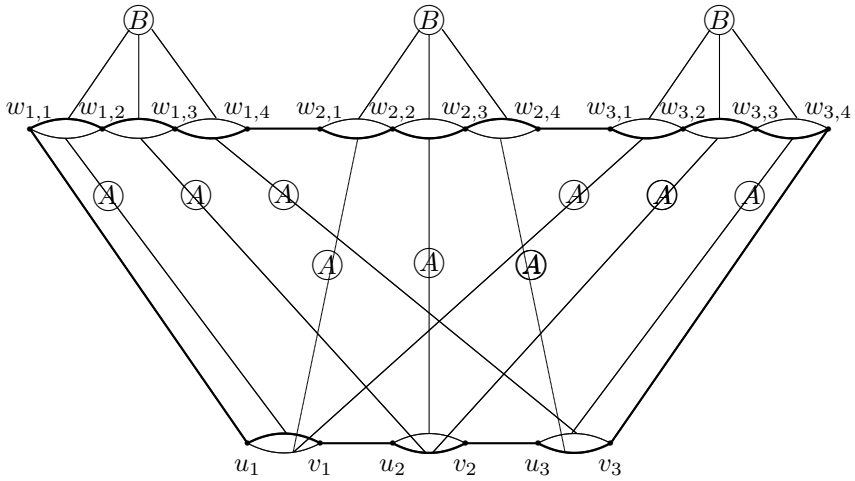


Рис. 20

Каждой переменной x_i соответствуют два кратных ребра u_iv_i, одно из которых будем называть верхним, а другое нижним. Кроме того, имеются рёбра v_iu_{i+1}, а также $w_{1,1}u_1$ и $v_nw_{m,4}$.

Ребро $w_{i,j}w_{i,j+1}$ соединяется A-связью с верхним ребром u_kv_k, если j-й литерал в C_i есть x_k, и с нижним ребром u_kv_k, если j-й литерал в C_i есть $\overline{x}_k$.

Пусть в G есть ГЦ c. Он имеет вид:

$$w_{1,1} \to u_1 \to v_1 \to u_2 \to v_2 \to \ldots \to u_n \to v_n \to w_{m,4} \to \ldots \to w_{1,1}.$$

Если в c входит верхнее ребро u_iv_i, будем считать, что $x_i = 1$; если в c входит нижнее ребро u_iv_i, будем считать, что $x_i = 0$.

Цикл c проходит по ребру $w_{i,j}w_{i,j+1}$ тогда и только тогда, когда он не проходит по соответствующему ребру u_kv_k, т. е. когда соответствующий литерал $t_j = 0$. Поскольку по всем трём ребрам $w_{i,1}w_{i,2}$, $w_{i,2}w_{i,3}$, $w_{i,3}w_{i,4}$ цикл проходить не может, дизъюнкция C_i принимает значение ИСТИНА при соответствующем наборе значений переменных.□

На рис. 20 изображён граф, соответствующий 3-КНФ

$$F = (x_1 \vee \overline{x}_2 \vee x_3)(\overline{x}_1 \vee x_2 \vee \overline{x}_3)(\overline{x}_1 \vee \overline{x}_2 \vee x_3).$$

В нём жирными линиями выделен гамильтонов цикл, соответствующий такому набору значений переменных: $x_1 = 1$, $x_2 = 0$, $x_3 = 0$.

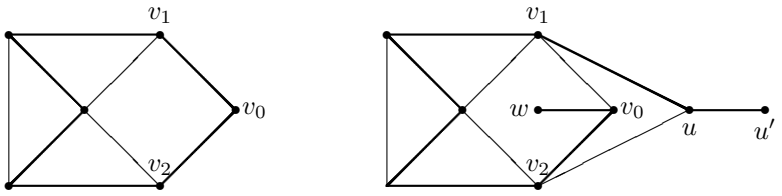
Задача ПОЛУГАМИЛЬТОНОВОСТЬ, как это следует из её названия, отвечает на вопрос, является ли данный граф полугамильтоновым, т. е. есть ли в нём гамильтонова цепь — маршрут (не обязательно замкнутый), проходящий через все вершины графа по одному разу.

Теорема 30. *Задача ПОЛУГАМИЛЬТОНОВОСТЬ является NP-полной.*

Доказательство. Покажем, как к задаче ПОЛУГАМИЛЬТОНОВОСТЬ сводится задача ГАМИЛЬТОНОВОСТЬ. По данному графу G построим граф G', обладающий свойством:

- G — гамильтонов $\iff$ G' — полугамильтонов.

Для этого добавим к графу G вершины u, u', w. Выберем в графе G произвольную вершину v_0; пусть вершины, смежные с v_0 есть v_1, v_2,... Добавим также к графу рёбра uu', v_0w и v_iu (для каждого i). В результате получится граф G' (иллюстрация на рис. 21).



Граф G Граф G'

Рис. 21

В графе G' есть висячие вершины u' и w. Поэтому если в G' есть гамильтонова цепь, то её концевые рёбра uu' и wv_0. После удаления этих рёбер получим (для некоторого i) цепь $u \to v_i \to \ldots \to v_0$. Но, по построению, вершины v_i и v_0 смежны, в силу чего указанная цепь легко преобразуется в гамильтонов цикл графа G заменой ребра uv_i ребром v_0v_i. Так же легко по гамильтонову циклу графа G строится гамильтонова цепь графа G'.

Поскольку переход от G к G' требует линейного (относительно числа вершин графа) времени, доказана искомая полиномиальная сводимость одной задачи к другой. Задача ГАМИЛЬТОНОВОСТЬ NP-полна. Значит, таким же свойством обладает и задача ПОЛУГАМИЛЬТОНОВОСТЬ. □

Теорема 31. *ЗАДАЧА КОММИВОЯЖЕРА NP-полна.*

Доказательство. По графу G с множеством вершин $V = \{v_1, \ldots, v_n\}$ составим матрицу расстояний $D = (d_{i,j})$ между вершинами графа, в которой $d_{i,j} = 1$, если вершины v_i и v_j смежны, и $d_{i,j} = 2$, если вершины v_i и v_j не являются смежными. Тогда вопрос «Существует ли маршрут коммивояжёра длины $|V|$?» равносилен вопросу о гамильтоновости графа G.□

Замечание. Приведённое только что доказательство показывает, что NP-полной будет даже метрическая ЗК (в которой расстояния между вершинами удовлетворяют аксиомам метрики — свойству симметричности и неравенству треугольника).

В этом параграфе из NP-полноты задачи ВЫПОЛНИМОСТЬ мы вывели NP-полноту ещё нескольких важных задач дискретной оптимизации. Список NP-полных задач весьма велик. В классической монографии М. Гэри и Д. Джонсона [4] приводится около 300 таких задач, относящихся к теории графов, построению сетей, хранению и поиску данных, теории расписаний, математическому программированию, алгебре, теории чисел, математической логике, теории автоматов и даже к играм и головоломкам. В настоящее время насчитывают уже более 3000 NP-полных задач. См. Википедию:
`http://en.wikipedia.org/wiki/List_of_NP-complete_problems`.

6. Частные случаи NP-полных задач

6.1. Доказательство NP-полноты сужением

Чем более общей является задача, тем труднее её решить[3]. Если некоторая задача A входит в класс NP, а некоторый частный случай этой задачи обладает свойством NP-полноты, то задача A также NP-полна.

Начнём с простого примера.

Задачи ГАМИЛЬТОНОВОСТЬ и ПОЛУГАМИЛЬТОНОВОСТЬ для орграфов являются NP-полными. Действительно, неориентированный граф можно рассматривать как частный случай орграфа в котором вместе с каждой дугой uv присутствует и дуга vu.

Задача ИЗОМОРФИЗМ ПОДГРАФУ состоит в следующем. Даны два графа G и G'. Требуется выяснить, есть ли в графе G подграф, изоморфный графу G'.

[3]Здесь мы оставляем в стороне так называемый *парадокс исследователя*.

Можно указать сразу две NP-полные задачи, обобщением которых является данная задача.

Если в качестве G' взять полный граф K_m, то наличие в G подграфа, изоморфного K_m, говорит о том, что в G есть клика мощности m. Значит, КЛИКА — частный случай рассматриваемой задачи.

А если в качестве G' взять циклический граф C_n, где n — количество вершин графа G, то приходим к задаче ГАМИЛЬТОНОВОСТЬ.

Рассмотрим задачу ПОСТРОЕНИЕ НАДЕЖНОЙ СЕТИ. Имеется граф G с n вершинами; $D = (d_{i,j})$ — матрица расстояний между его вершинами; $R = (r_{i,j})$ — матрица избыточности; L — некоторое положительное число. Требуется узнать, существует ли такой подграф G' графа G, вес которого не больше L, и для любых двух вершин с номерами i и j найдётся в этом подграфе не менее $r_{i,j}$ соединяющих их цепей, не имеющих общих вершин, кроме начальной и конечной. (Разумеется, матрицы D и R должны быть симметричными).

Покажем, что задача ГАМИЛЬТОНОВОСТЬ является частным случаем этой задачи. Действительно, пусть

- $d_{i,j} \geqslant 1$ для всех i и j;
- $r_{i,j} = 2$ при $i \neq j$;
- $L = n$.

Тогда любые две вершины графа входят в некоторый цикл. Так как суммарный вес рёбер равен числу вершин, все рёбра искомого подграфа должны быть единичного веса и их должно быть ровно n. Отсюда следует, что в подграфе G, составленном из рёбер единичного веса, должен быть гамильтонов цикл.

6.2. Трудные и лёгкие частные случаи NP-полных задач

Итак, если наша задача входит в класс NP и является обобщением некоторой NP-полной задачи, то она также является NP-полной. Отсюда, конечно, не следует, что любой частный случай NP-полной задачи также сложная задача.

Например, из NP-полноты задачи 3-РАСКРАСКА следует NP-полнота задачи РАСКРАСКА, в которой для заданного графа G и заданного числа m требуется выяснить, существует ли *правильная* раскраска вершин графа в m цветов (при такой раскраске смежные вершины должны быть разного цвета). Однако, если в данной задаче

положить $m = 2$, то есть ограничиться двумя цветами, то приходим к полиномиальной задаче, которая сводится к ответу на вопрос, является ли граф двудольным.

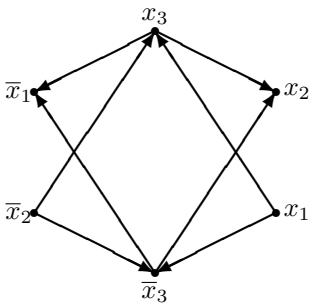
В предыдущем параграфе мы убедились в NP-полноте задачи 3-КНФ, являющейся частным случаем задачи ВЫПОЛНИМОСТЬ. А что можно сказать о задаче 2-КНФ? Оказывается, она является полиномиальной! [4] Докажем это.

В основе доказательства лежат логические равносильности

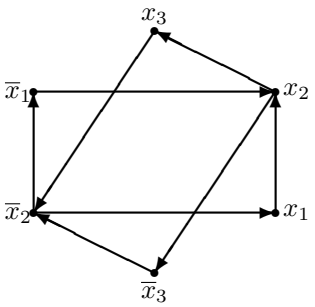
$$p \vee q = \overline{p} \to q = \overline{q} \to p.$$

Пусть $F = \&C_i$, где для каждого i дизъюнкция C_i состоит из двух литералов, а логическими переменными являются $x_1, x_2, \ldots, x_k$. Построим орграф $G = \langle V, A \rangle$, в котором $V = \{x_1, x_2, \ldots, x_k, \overline{x}_1, \overline{x}_2, \ldots, \overline{x}_k\}$, а каждой дизъюнкции $p \vee q$ из F соответствуют две дуги $\overline{q}p$ и $\overline{p}q$.

Пример. 2-КНФ $F_1 = (\overline{x}_1 \vee \overline{x}_3)(\overline{x}_1 \vee x_3)(x_2 \vee \overline{x}_3)(x_2 \vee x_3)$ и $F_2 = (x_1 \vee x_2)(\overline{x}_1 \vee x_2)(\overline{x}_2 \vee x_3)(\overline{x}_2 \vee \overline{x}_3)$ порождают графы G_1 и G_2, изображённые на рис. 22.



Граф G_1



Граф G_2

Рис. 22

Теорема 32. *2-КНФ F выполнима тогда и только тогда, когда в графе G ни для какого i вершины x_i и $\overline{x}_i$ не являются взаимно достижимыми.*

Для иллюстрации утверждения теоремы заметим, что в приведённом выше примере 2-КНФ F_1 выполнима, поскольку в графе G_1 вообще нет двух взаимно достижимых вершин, а 2-КНФ F_2 не выполнима, так как вершины x_2 и $\overline{x}_2$ взаимно достижимы.

[4] Задачи 2-РАСКРАСКА и 2-КНФ полиномиальны, а задачи 3-РАСКРАСКА и 3-КНФ NP-полные. Не говорит ли это о той огромной пропасти, которая отделяет двойку от тройки?!

Доказательство. *Необходимость.* Пусть F выполнима, т. е. на некотором наборе значений логических переменных $F = 1$. Предположим, что для некоторого i вершины x_i и $\overline{x}_i$ взаимно достижимы. Обозначим через t_1 ту из этих двух вершин, которая принимает значение 1. Рассмотрим путь $t_1 \to t_2 \to t_3 \to \ldots \to \overline{t}_1$, ведущий от литерала t_1 к его отрицанию $\overline{t}_1$. Дуга $t_1 t_2$ изображает дизъюнкцию $\overline{t}_1 \vee t_2$ (или, что то же самое, импликацию $t_1 \to t_2$). Из того, что $\overline{t}_1 \vee t_2 = 1$ и $t_1 = 1$, получаем, что $t_2 = 1$. Точно так же получаем истинность всех литералов, входящих в указанный путь, в том числе и $\overline{t}_1$, что приводит нас к противоречию.

Достаточность. Напомним, что компонента сильной связности орграфа — это максимальный по включению подграф, в котором любые две вершины взаимно достижимы. (Можно описать компоненту сильной связности и по-другому. Несложно проверить, что отношение взамной достижимости на множестве вершин орграфа является отношением эквивалентности. Соответствующие классы эквивалентности и порождают компоненты сильной связности.)

Компоненты сильной связности можно пронумеровать таким образом, чтобы из компоненты с бóльшим номером нельзя было попасть в компоненту с меньшим номером. Алгоритм решения такой задачи, описанный в [13], имеет трудоёмкость $O(|V| + |A|)$.

Пусть в соответствии с указанной иерархией имеем компоненты сильной связности $D_1, D_2, \ldots, D_s$. Будем расставлять пометки вершин графа (первоначально все они считаются непомеченными) в порядке убывания номеров компонент сильной связности по следующим правилу:

- если в компоненте D_i имеется такая вершина v, что вершина $\overline{v}$ уже имеет пометку 1, то все вершины из D_i получают пометку 0; в противном случае все вершины из D_i помечаются 1.

Покажем, что логическая функция F будет принимать значение 1 на наборе значений переменных, полученном в результате расстановки пометок.

Рассуждая от противного, предположим, что 2-КНФ содержит дизъюнкцию литералов $t_1 \vee t_2$, а вершины t_1 и t_2 имеют пометку 0. Пусть

$$t_1 \in D_a, \quad \overline{t}_1 \in D_b, \quad t_2 \in D_c, \quad \overline{t}_2 \in D_d.$$

Согласно описанному алгоритму, $a < b$ и $c < d$. Кроме того, наличие дуги $\overline{t}_1 t_2$ говорит о том, что $b \leqslant c$. Из трёх полученных неравенств

выводим неравенство $a < d$. Вместе с тем, в нашем графе должна быть и дуга $\bar{t}_2 t_1$, откуда $d < a$. Противоречие получено! □

Пример. Для графа G_1, построенного по 2-КНФ F_1, возможно такое распределение вершин по компонентам сильной связности:

$$D_1 = \{x_1\},\ D_2 = \{\overline{x}_2\},\ D_3 = \{\overline{x}_3\},\ D_4 = \{x_3\},\ D_5 = \{x_2\},\ D_6 = \{\overline{x}_1\}.$$

Применяя алгоритм из доказательства теоремы, последовательно получаем:

$$\overline{x}_1 = 1,\ x_2 = 1,\ x_3 = 1,\ \overline{x}_3 = 0, \overline{x}_2 = 0,\ x_1 = 0.$$

Легко проверить, что на наборе $x_1 = 0, x_2 = 1, x_3 = 1$ логическая функция F_1 принимает значение 1.

Очевидно, что трудоёмкость описанного алгоритма является линейной функцией от длины 2-КНФ. Стало быть, действительно задача 2-КНФ входит в класс P.

7. Структура класса NP

Для класса NP-полных задач часто используют обозначение NPC. Как известно, NPC — множество наиболее «трудных» задач класса NP: к любой из этих задач за *полиномиальное* время сводится любая задача из NP. В то же время, P — множество наиболее «лёгких» задач: для них существуют полиномиальные алгоритмы решения. Вполне правдоподобной кажется гипотеза $P \neq NP$ (в этом случае NPC есть собственное подмножество NP, не пересекающееся с P)[5]. В её пользу говорит тот факт, что несмотря на предпринимавшиеся гигантские усилия многих поколений математиков не было найдено ни одного полиномиального алгоритма ни одной из многочисленных NP-полных задач.

В 1975 г. было доказано, что в предположении $P \neq NP$ множество

$$NPI = NP \setminus (P \cup NPC)$$

[5]Особую популярность эта гипотеза получила после того, как она была включена в так называемый список *семи задач тысячелетия*, за решение каждой из которых Математический институт Клэя в Массачусетсе установил премию в 1 млн долларов. Одна из этих задач была решена российским математиком Григорием Перельманом. На сайте `http://www.win.tue.nl/~gwoegi/P-versus-NP.htm` собраны ссылки на десятки научных работ, авторы которых считают, что они доказали (или опровергли) гипотезу $P \neq NP$. Однако ни один из этих авторов не убедил научное сообщество в своей правоте.

задач, промежуточных по сложности (между P и NPC), непусто. В этом случае структуру класса NP можно изобразить диаграммой на рис. 23.

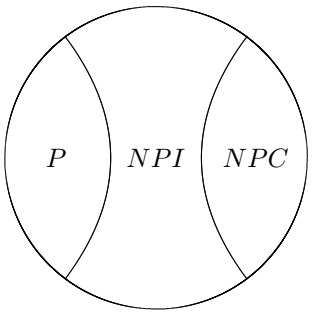


Рис. 23

В классической монографии [4] приведены три задачи — наиболее вероятные кандидаты на попадание в класс NPI.

- ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ. В распознавательном варианте эта задача ставится так. Пусть A — целочисленная матрица размера $n \times n$, c и b — целочисленные векторы с n координатами, L — некоторое целое число. Существует ли такой n-мерный вектор x с рациональными координатами, для которого $Ax \leqslant b$ и $c'x \geqslant L$?

- ПРОСТЫЕ ЧИСЛА. Является ли заданное число n простым? При оценке эффективности алгоритма решения данной задачи интересуются зависимостью трудоёмкости от длины записи числа n, т. е. фактически от $\lg n$.

- ИЗОМОРФИЗМ ГРАФОВ. Являются ли два заданных графа изоморфными? Это частный случай задачи ИЗОМОРФИЗМ ПОДГРАФУ. Но, как мы знаем, частный случай NP-полной задачи не обязательно трудная задача.

Монография [4] вышла на языке оригинала более 30 лет назад — в 1979 г. Всё это время теория сложности алгоритмов интенсивно развивалась.

Уже в 1979 г. советский математик Л. Г. Хачиян доказал, что известный ранее (и разработанный другими советскими математиками А. С. Немировским и Д. Б. Юдиным) алгоритм эллипсоидов за полиномиальное время решает задачу ЛИНЕЙНОЕ ПРОГРАММИРОВАНИЕ.

6 августа 2002 г. индийские математики М. Агравал, Н. Кайал и Н. Саксена опубликовали статью под названием *PRIMES is in P*, в которой был предложен полиномиальный детерминированный тест простоты чисел. В 2006 г. они получили премию Гёделя, которая присуждается организациями ACM SIGACT (Special Interest Group on Algorithms and Computation Theory) и EATCS (European Association for Theoretical Computer Science) за выдающиеся труды по логике и теоретической информатике. Алгоритм AKC (получивший своё название по первым буквам фамилий его авторов), определяющий, является ли число n простым, имеет трудоёмкость $O(\lg^{12} n)$. В 2005 г. была получена модификация этого алгоритма, имевшая трудоёмкость $O(\lg^6 n)$.

Лишь относительно статуса задачи ИЗОМОРФИЗМ ГРАФОВ нет никакой ясности: по-прежнему не найдено полиномиального алгоритма решения данной задачи, но и не доказано, что эта задача является NP-полной.

8. Приближённые алгоритмы

Нужда в приближённых алгоритмах возникает при исследовании задач, имеющих трудоёмкие *точные* алгоритмы решения. Между тем, при решении практических задач могут оказаться приемлемыми и решения, не являющиеся оптимальными, но в определённом смысле близкие к ним.

Пусть A — задача оптимизации с целевой функцией (или *функцией стоимости*) c. Рассматривается алгоритм $\mathcal{A}$, который для индивидуальной задачи I даёт решение $f(I)$. Через $\hat{f}(I)$ обозначим оптимальное решение задачи I. Алгоритм $\mathcal{A}$ называется **ε-приближённым алгоритмом** для задачи A, если

$$\forall I \qquad \left| \frac{c(f(I)) - c(\hat{f}(I))}{c(\hat{f}(I))} \right| \leqslant \varepsilon.$$

Другими словами, алгоритм является ε-приближённым, если он даёт такое решение задачи, что если заменить этим решение точное решение, мы получим относительную погрешность в стоимости решения, не превосходящую ε. Таким образом, приближённый алгоритм хоть и не даёт оптимального решения, но гарантирует определённое *качество* решения.

Как мы знаем, задача коммивояжёра (ЗК) относится к классу NP-полных задач. То же относится и к её частному случаю — метрической ЗК (см. замечание на стр. 81). Оказывается, существует полиномиальный приближённый алгоритм решения данной задачи — **алгоритм Кристофидеса** [12].

Назовём данный алгоритм $\mathcal{K}$. На входе алгоритма $\mathcal{K}$ мы имеем полный граф G с матрицей расстояний $(d_{i,j})$, которые удовлетворяют неравенству треугольника.

Алгоритм $\mathcal{K}$

1. Найти в графе G минимальное стягивающее дерево T.
2. Выделить в T все вершины нечётной степени и найти совершенное паросочетание P наименьшего веса в полном подграфе графа G, порождённом этими вершинами.
3. В графе, составленном из рёбер дерева T и паросочетания P, найти эйлеров цикл τ.
4. Устраняя повторное вхождение вершин, получить из эйлерова цикла τ «вложенный» в него гамильтонов цикл τ'.

Шаги 1, 2, 3, 4 данного алгоритма можно выполнить за время $O(n^2), O(n^4), O(n)$ и $O(n)$ соответственно, где n — число вершин графа. Поэтому алгоритм $\mathcal{K}$ является полиномиальным.

Теорема 33. *Алгоритм $\mathcal{K}$ является $\frac{1}{2}$-приближённым для метрической задачи коммивояжёра.*

Доказательство. Пусть $\hat{\tau}$ — оптимальное решение задачи коммивояжёра, т. е. гамильтонов цикл наименьшей длины в графе G. Вес произвольного подграфа G' будем обозначать $c(G')$.

Во-первых, заметим, что

$$c(\hat{\tau}) \geqslant c(T), \tag{1}$$

поскольку удаление любого ребра из $\hat{\tau}$ даёт нам некоторое стягивающее дерево графа G, а T — стягивающее дерево минимального веса.

Во-вторых, $\tau \subset T \cup P$. Поэтому

$$c(\tau) \leqslant c(T) + c(P). \tag{2}$$

Пусть $v_1, v_2, \ldots, v_{2m}$ — вершины нечётной степени дерева T (таких вершин, по следствию из леммы о рукопожатиях, всегда чётное число), взятые в том порядке, в каком они появляются в маршруте $\hat{\tau}$.

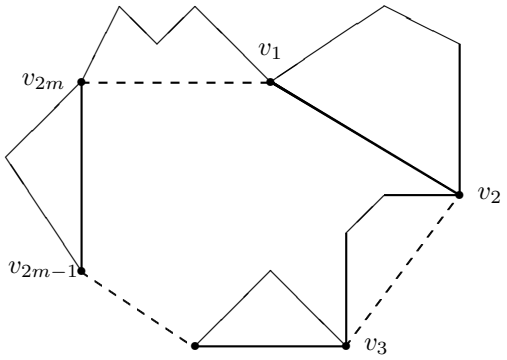


Рис. 24

Рассмотрим два паросочетания $P_1 = \{v_1v_2, v_3v_4, \ldots, v_{2m-1}, v_{2m}\}$ и $P_2 = \{v_2v_3, v_4v_5, \ldots, v_{2m}, v_1\}$ (на рис. 24 рёбра, входящие в P_1, P_2 и $\hat{\tau}$ изображены соответственно жирными, пунктирными и тонкими линиями). В силу неравенства треугольника

$$c(\hat{\tau}) \geqslant \sum \mu(v_iv_{i+1}) = c(P_1) + c(P_2). \tag{3}$$

Поскольку P — совершенное паросочетание наименьшего веса в соответствующем графе, имеем $c(P_1)+c(P_2) \geqslant 2c(P)$. С учётом неравенств (1)–(3), получаем

$$c(\tau) \leqslant c(T) + c(P) \leqslant c(\hat{\tau}) + \frac{c(\hat{\tau})}{2}.$$

Поэтому $c(\tau) - c(\hat{\tau}) \leqslant \dfrac{c(\hat{\tau})}{2}$. Очевидно также, что $c(\tau') \leqslant c(\tau)$. Отсюда получаем требуемое:

$$\left|\frac{c(\tau') - c(\hat{\tau})}{c(\hat{\tau})}\right| \leqslant \frac{1}{2}.\square$$

Заметим, что оценка $\varepsilon = \frac{1}{2}$ является точной, о чём говорит пример с рис. 25.

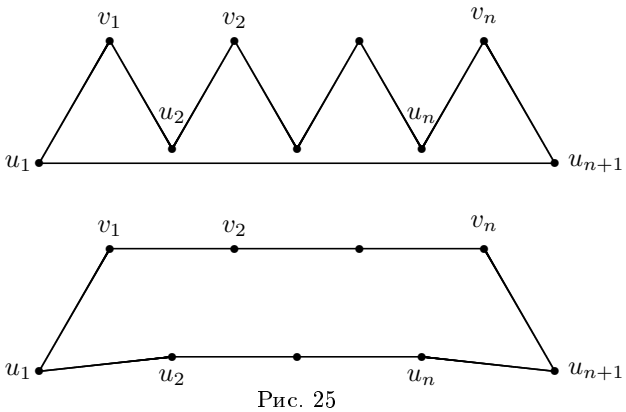


Рис. 25

Здесь расстояния между вершинами графа равны расстояниям между соответствующими точками плоскости. В ломаной $u_1v_1u_2v_2\dots u_nv_nu_{n+1}$ углы между соседними звеньями по $60°$,

$$v_1u_2 = u_2v_2 = \dots = u_nv_n = 1, \quad u_1v_1 = v_nu_{n+1} = 1{,}1.$$

Минимальное стягивающее дерево представляет собой цепь $u_1 \to v_1 \to u_2 \to \dots \to u_n \to v_n \to u_{n+1}$, в которой всего две вершины нечётной степени u_1 и u_{n+1}. Поэтому алгоритм Кристофидеса построит гамильтонов цикл $u_1 \to v_1 \to u_2 \to \dots \to u_n \to v_n \to u_{n+1} \to u_1$ длины $3n + 0{,}3$. В то же время длина гамильтонова цикла $u_1 \to v_1 \to v_2 \to \dots \to v_n \to u_{n+1} - u_n \to \dots \to u_2 \to u_1$ приблизительно равна $2n + 1$. За счёт выбора n относительную ошибку, возникающую при замене оптимального решения приближённым, можно сделать сколь угодно близкой к $\frac{1}{2}$.

В общем случае (т. е. если не требовать выполнения неравенства треугольника) задача нахождения ε-приближённого решения является NP-полной (каким бы большим ни было число ε).

Для доказательства этого утверждения покажем, что к данной задаче сводится NP-полная задача ГАМИЛЬТОНОВОСТЬ. По заданному графу G с множеством вершин $V = \{v_1, \dots, v_n\}$ составим матрицу расстояний $D = (d_{i,j})$ между вершинами графа, в которой $d_{i,j} = 1$, если вершины v_i и v_j смежны, и $d_{i,j} = 1 + \varepsilon n$, если вершины v_i и v_j не являются смежными. Тогда вопрос «Существует ли маршрут коммивояжёра длины меньше $n(1+\varepsilon)$?» равносилен вопросу о гамильтоновости графа G. Действительно, если граф гамильтонов, то найдётся гамильтонов цикл длины n. Если же граф не является гамильтоновым, то длина маршрута коммивояжёра не меньше $n - 1 + 1 + \varepsilon n = n(1+\varepsilon)$. □

Библиографический список

1. Асанов, М. О. *Дискретная математика: графы, матроиды, алгоритмы* / М. О. Асанов, В. А. Баранский, В. В. Расин. — СПб.: Лань, 2010. – 368 с.

2. *Вся высшая математика:* учебник / М. Л. Краснов, А. И. Киселёв, Г. И. Макаренко и др. — М.: КомКнига, 2012. – Т.7 – 208 с.

3. Вялый, М.Н. *Сложность вычислительных задач* / М. Н. Вялый // Математическое просвещение. – 2000. Сер. 3, вып. 4. – С. 81–114.

4. Гэри, М. *Вычислительные машины и труднорешаемые задачи* / М. Гэри, Д. Джонсон. — М.: Мир, 1982. – 416 с.

5. Кормен, Т. *Алгоритмы: построение и анализ* / Т. Кормен, Ч. Лейзерсон, Р. Ривест. — М.: МЦНМО, 2000. – 960 с.

6. *Лекции по теории графов* / В. А. Емеличев, О. И. Мельников, В. И. Сарванов и др. — М.: Наука, 1990. – 384 с.

7. Липский, В. *Комбинаторика для программистов* / В. Липский. — М.: Мир, 1988. – 213 с.

8. Ловас, Л. *Прикладные задачи теории графов* / Л. Ловас, М. Пламмер. — М.: Мир, 1998. – 653 с.

9. Ляхов, А. Ф. *Трудно решаемые задачи* / А. Ф. Ляхов // Математическое образование. – 2009. – № 3(51). – С. 27–38.

10. Нефедов, В. Н. *Дискретные задачи оптимизации* / В. Н. Нефедов. — М.: Изд-во МАИ, 1993. – 60 с.

11. Новиков, Ф. А. *Дискретная математика для программистов* / Ф. А. Новиков. — СПб.: Питер, 2000. – 304 с.

12. Пападимитриу, Х. *Комбинаторная оптимизация. Алгоритмы и сложность* / Х. Пападимитриу, К. Стайглиц. — М.: Мир, 1985. – 512 с.

13. Рейнгольд, Э. *Комбинаторные алгоритмы. Теория и практика* / Э. Рейнгольд, Ю. Нивергельт, Н. Део. — М.: Мир, 1980. – 478 с.

14. Уилсон, Р. *Введение в теорию графов* / Р. Уилсон. — М.: Мир, 1977. — 208 с.

15. Эвнин, А. Ю. *Антиматроиды* // Математическое образование. – 2008. – № 1(45). – С.2–8.

16. Эвнин, А. Ю. *Вокруг теоремы Холла* / А. Ю. Эвнин. — М.: ЛИБРОКОМ, 2012. – 88 с.

17. Эвнин, А. Ю. *Дискретная математика: задачник* / А. Ю. Эвнин. — Челябинск: Издательский центр ЮУрГУ, 2009. – 265 с.

18. Эвнин, А. Ю. *Дискретная математика: Конспект лекций* / А. Ю. Эвнин. — Челябинск: Изд-во ЮУрГУ, 1998. – 176 с.

19. Эвнин, А. Ю. *Элементарное введение в матроиды* / А. Ю. Эвнин. — Челябинск: Изд-во ЮУрГУ, 2004. – 40 с.

20. Bang-Jensen, J. *Digraphs. Theory, Algorithms and Applications* / J. Bang-Jensen, G. Gutin. — Springer-Verlag, 2007. – 772 p.

21. Oxley, J.G. *Matroid Theory* / J. G. Oxley. — Oxford Univ. Press, 1992. – 532 p.

Александр Юрьевич Эвнин

ЭЛЕМЕНТЫ ДИСКРЕТНОЙ ОПТИМИЗАЦИИ

Учебное пособие

Техн. редактор А. В. Миних

Издательский центр Южно-Уральского
государственного университета

Подписано в печать 14.05.2012. Формат 60 × 84 1/16. Печать цифровая. Усл. печ. л. 5,11. Тираж 100 экз. Заказ 69/170. Цена С.

Отпечатано в типографии Издательского центра ЮУрГУ.
454080, г. Челябинск, пр. им. В. И. Ленина, 76.