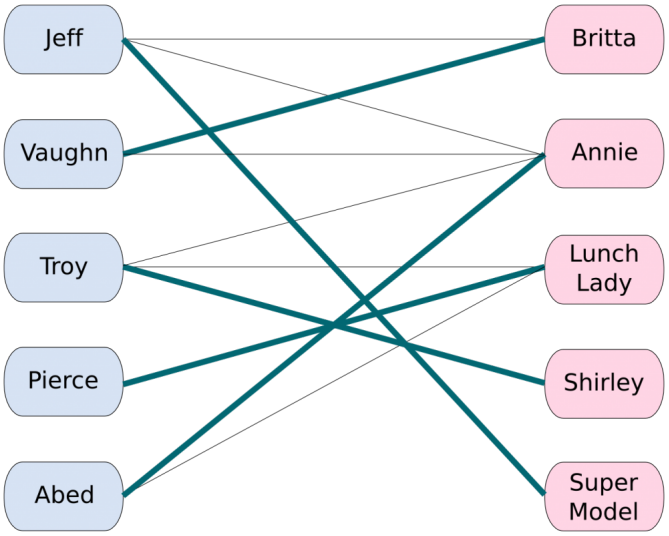
Паросочетания и всё что с ними связано

Сапожников Денис

Contents

1 Паросочетания

Задача. Пусть есть мальчиков и девочек. Про каждого мальчика и про каждую девочку известно, с кем они не против танцевать. Нужно составить как можно больше пар, в которых партнёры хотят танцевать друг с другом.



Формализуем эту задачу, представив мальчиков и девочек как вершины в двудольном графе, рёбрами которого будет отношение «могут танцевать вместе».

Определение. Паросочетанием M называется набор попарно несмежных рёбер графа (иными словами, любой вершине графа должно быть инцидентно не более одного ребра из M).

Определение. Все вершины, у которых есть смежное ребро из паросочетания (т.е. которые имеют степень ровно один в подграфе, образованном M), назовём насыщенными этим паросочетанием.

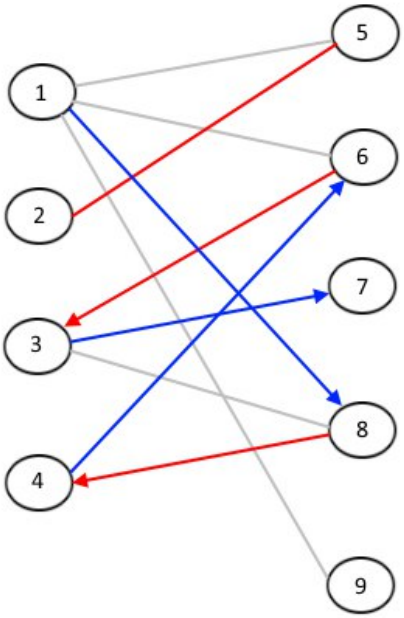
Определение. Мощностью паросочетания назовём количество рёбер в нём. Наибольшим (максимальным) паросочетанием назовём паросочетание, мощность которого максимальна среди всех возможных паросочетаний в данном графе, а совершенным — где все вершины левой доли им насыщенны.

Паросочетания можно искать в любых графах, однако этот алгоритм неприятно кодить, и он работает за $O(n^3)$, так что сегодня мы сфокусируемся только на двудольных графах. Будем в дальнейшем обозначать левую долю графа как L, а правую долю как R.

Цепью длины k назовём некоторый простой путь (т.е. не содержащий повторяющихся вершин или рёбер), содержащий ровно k рёбер.

Чередующейся цепью относительно некоторого паросочетания назовём простой путь длины k в которой рёбра поочередно принадлежат/не принадлежат паросочетанию.

Увеличивающей цепью относительно некоторого паросочетания назовём чередующуюся цепь, у которой начальная и конечная вершины не принадлежат паросочетанию.



Здесь красными помечены вершины паросочетания, а в графе есть увеличивающая цепь:
$1 \to 8 \to 4 \to 6 \to 3 \to 7$.

1.1 Лемма Берже и применение

Зачем нужны увеличивающие цепи? Оказывается, можно с их помощью увеличивать паросочетание на единицу (отсюда и название). Можно взять такой путь и провести чередование — убрать из паросочетания все рёбра, принадлежащие цепи, и, наоборот, добавить все остальные. Всего в увеличивающей цепи нечетное число рёбер, а первое и последнее были не в паросочетании. Значит, мощность паросочетания увеличилась ровно на единицу.

В примере добавятся синие рёбра $(1,8), (3,7)$, и $(4,6)$, а удалятся красные $(3,6)$ и $(4,8)$. С ребром $(2,5)$ ничего не случится — оно не в увеличивающей цепи. Таким образом, размер паросочетания увеличится на единицу.

Для доказательства алгоритма нам будет достаточно ещё доказать, что если увеличивающие цепи уже не ищутся, то паросочетание в принципе нельзя увеличить.

Лемма (Берже). *Паросочетание без увеличивающих цепей является максимальным.*

Её важное следствие состоит в том, что максимальное паросочетание можно строить инкрементально, на каждом шаге делая поиск увеличивающей цепи и проводя её чередование.

Proof. Доказательство проведём от противного: пусть есть два паросочетания вершин $|A| < |B|$, и для A нет увеличивающих путей, и покажем, как найти этот путь и увеличить $|A|$ на единицу.

Раскрасим ребра из паросочетания, соответствующего A в красный цвет, B — в синий, а ребра из обоих паросочетаний — в пурпурный. Рассмотрим граф из только красных и синих ребер. Любая компонента связности в нём представляет собой либо путь, либо цикл, состоящий из чередующихся красных и синих ребер. В любом цикле будет равное число красных и синих рёбер, а так как всего синих рёбер больше, то должен существовать путь, начинающийся и оканчивающийся синим ребром — он и будет увеличивающей цепью для A, а значит A не оптимальное, и мы получили противоречие. □

1.2 Алгоритм Куна

Алгоритм Куна ровно в этом и заключается — начнем с пустого паросочетания и будем искать увеличивающие цепи, пока они ищутся.

Для поиска увеличивающей цепи можно мысленно построить граф, в котором из правой доли в левую можно идти только по рёбрам паросочетания, а из левой в правую — по любым. Тогда можно запустить поиск любого пути из свободной вершины левой доли в какую-нибудь свободную вершину правой доли в измененном графе, и ровно такой путь и будет увеличивающим.

Это можно делать как угодно — для упражнения автор рекомендует явно строить такой граф, искать путь любимым алгоритмом поиска и явно проводить чередования — однако устоялась эффективная реализация в виде dfs на 20 строчек кода, приведённая ниже.

```
int n, k;
vector < vector<int> > g;
vector<int> mt;
vector<char> used;

bool try_kuhn (int v) {
  if (used[v])  return false;
  used[v] = true;
  for (int u : g[v]) {
    if (mt[u] == -1 || try_kuhn (mt[u])) {
      mt[u] = v;
      return true;
    }
  }
  return false;
}
```

Теперь, для нахождения самого паросочетания нужно просто запустить этот поиск от всех вершин левой доли, откатывая состояние вспомогательного массива *used*:

```
int main() {
  // read graph

  mt.assign (k, -1);
  for (int v = 0; v < n; ++v) {
    fill(used.begin(), used.end(), false);
    try_kuhn(v);
  }

  for (int i = 0; i < k; ++i) {
    if (mt[i] != -1)
      cout << mt[i]+1 << ' ' << i+1 << '\n';
  }
}
```

Алгоритм ровно n раз ищет увеличивающий путь, каждый раз просматривая не более m рёбер, а значит суммарно отработает за $O(nm)$.

1.3 Оптимизации

Что примечательно, алгоритм можно не бояться запускать на ограничениях и побольше ($n, m \approx 10^4$), если сделать некоторые неасимптотические оптимизации:

- Паросочетание можно жадно инициализировать — например, если просто заранее пройтись по вершинам левой доли и сматчить их со свободной вершиной правой, если она есть.
- Можно не заполнять нулями на каждой итерации массив $used$, а использовать следующий трюк: хранить в нём вместо булева флага версию последнего изменения, а конкретно — номер итерации, на которой это значение стало $true$. Если этот номер меньше текущего номера итерации, то мы можем воспринимать это значение как $false$. В каком-то смысле это позволяет эмулировать очищение массива за константу.
- Очень часто граф приходит из какой-то другой задачи, природа которой накладывает ограничения на его вид. Например, в задачах на решетках (когда есть двумерный массив, и соседние клетки связаны друг с другом) граф двудольный, но степень каждой вершины маленькая, и граф имеет очень специфичную структуру. На таких графах алгоритм Куша часто работает быстрее, чем ожидается из формулы $n \times m$. Контрпримеры в таких задачах почти всегда возможно сгенерировать, но авторы редко заморачиваются.
- Можно пройтись по ребрам в dfs дважды: в первом проходе попробовать найти ненасыщенную паросочетанием вершину. Во втором – если такой не нашлось, то уже запускать dfs.

Подробнее про оптимизации можно почитать обсуждение здесь с примером кода. Формально, существует способ решить задачу быстрее, чем алгоритмом куна, а именно задача нахождения максимального паросочетания — частный случай задачи о максимальном потоке, и если применить алгоритм Диница к двудольным графам с единичной пропускной способностью, то работать он будет за $O(m\sqrt{n})$. Но на практике Кун всё равно разносит этот алгоритм.

1.4 Покрытие путями DUGa

Сводить задачи к поиску максимального паросочетания обычно не очень трудно, но в некоторых случаях самому додуматься сложно. Разберём одну такую известную задачу. Дан ориентированный ациклический граф G (англ. *directed acyclic graph*). Требуется покрыть его наименьшим числом путей, то есть найти наименьшее множество простых путей, где каждая вершина принадлежит ровно одному пути.

Построим соответствующие изначальному графу G два двудольных графа H и $\hat{H}$ следующим образом:

- В каждой доле графа H будет по n вершин. Обозначим их через a_i и b_i соответственно.
- Для каждого ребра (i, j) исходного графа G проведём соответствующее ребро (a_i, b_j) в графе H.
- Теперь из графа H сделаем граф $\hat{H}$, добавив обратное ребро (b_i, a_i) для каждого i.

Если мы рассмотрим любой путь $v_1, v_2, \dots v_k$ в исходном графе G, то в графе $\hat{H}$ ему будет соответствовать путь $a_{v_1}, a_{v_2}, \dots, a_{v_k}$. Обратное тоже верно: любой путь, начинающийся в левой доле $\hat{H}$ и заканчивающийся в правой будет соответствовать какому-то пути в G.

Итак, есть взаимно однозначное соответствие между путями в G и путями $\hat{H}$, идущими из левой доли в правую. Заметим, что любой такой путь в G — это паросочетание в $\hat{H}$ (напомним, это без обратных рёбер). Получается, любому пути из G можно поставить в соответствие паросочетание в $\hat{H}$, и наоборот. Более того, непересекающимся путям в G соответствуют непересекающиеся паросочетания в $\hat{H}$.

Заметим, что если есть p непересекающихся путей, покрывающих все n вершин графа, то они вместе содержат $r = n - p$ рёбер. Отсюда получаем, что чтобы минимизировать число путей p, мы должны максимизировать число рёбер r в них.

Мы теперь можем свести задачу к нахождению максимального паросочетания в двудольном графе $\hat{H}$. После нахождения этого паросочетания мы должны преобразовать его в набор путей в G. Это делается тривиальным алгоритмом: возьмем a_1, посмотрим, с какой b_k она соединена, посмотрим на b_k и так далее. Некоторые вершины могут остаться ненасыщенными — в таком случае в ответ надо добавить пути нулевой длины из каждой из этих вершин.

Задача 1 (Кубики)**.** Дано n кубиков, у каждого из них 6 граней, на каждой гране написана какая-то буква. Дано слово s, и требуется каждой букве слова s сопоставить уникальный кубик, так чтобы мы могли повернуть этот кубик и получить нужную нам букву.

Пример — даны три кубика, на гранях которых написаны буквы:
(a, a, a, a, a, a)
$(a, a, a, a, a, б)$
$(a, a, a, б, б, c)$
Если задано слово «acb», тогда ответ ровно один: 132.

Решение. Сделаем двудольный граф, одна доля которого — номера кубиков, а другая — номер буквы в слове s. Проведем ребра из номера кубика в номер буквы только если мы можем взять этот кубик на эту позицию в строке, то есть если у него есть грань с соответствующей буквой. Теперь ответ — максимальное паросочетание в этом графе.

По определению паросочетания мы не сопоставим ни один кубик нескольким буквам, но так как наше паросочетание — максимально, то мы покроем максимально возможное количество букв.

Задача 2 (Доминошки)**.** Есть прямоугольное поле $n \times m$, которое содержит какие-то выколотые клетки. Надо положить на это поле как можно больше костей домино (прямоугольников размера 1×2), но с условием, что поверх выколотых полей ничего лежать не должно.

Решение. Составим граф, в котором вершины будут полями, а рёбрами будут возможности положить доминошку на два соседних свободных поля. Заметим, что такой граф будет двудольным: можно его покрасить как шахматную доску, и тогда черные клетки будут вершинами первой доли, а белые — второй.

Ответ — максимальное паросочетание в таком графе. Асимптотика с алгоритмом Куна $O(n^2m^2)$, потому что у нас будет $O(nm)$ вершин и рёбер.

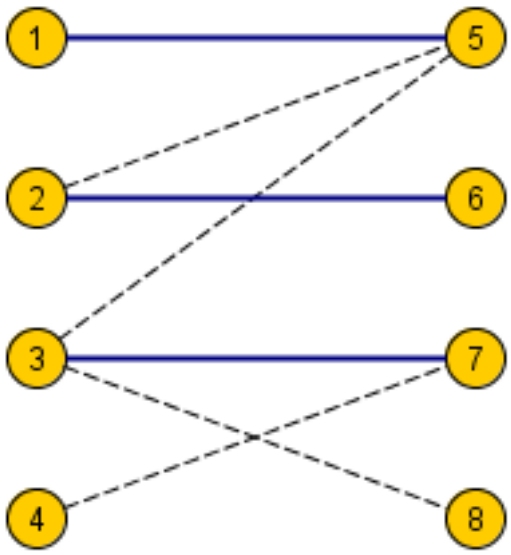
Задача 3 (Взвешенное по одной доле паросочетание)**.** Дан двудольный граф, у которого у каждой вершины левой доли есть вес. Вес паросочетания — это сумма весов на левых концах ребёр паросочетания. Необходимо найти максимальное паросочетание минимального веса.

Решение. Отсортируем все вершины по весу по возрастанию. Далее в таком порядке будем запускать алгоритм Куна. Утверждается, что на каждом шаге Куна после успешного увеличения паросочетания, мы будем иметь паросочетание минимального веса, состоящего из ровно k рёбер.

Задача 4. Необходима раскрасить двудольный граф в наименьшее число цветов так, чтобы смежные ребра были разных цветов.

2 Лемма Холла

Лемма Холла (или: теорема о свадьбах) — очень удобный критерий в задачах, где нужно проверить, что паросочетание существует, но при этом не требуется строить его явно.



Лемма (Холла)**.** *Полное (или совершенное) паросочетание существует тогда и только тогда, когда любая группа вершин левой доли соединена с не меньшим количеством вершин правой доли.*

Некоторая несимметричность в теореме связана с тем, что в долях может быть разное число вершин.

Proof. В одну сторону понятно — если совершенное паросочетание есть, то для любого подмножества вершин левой доли можно взять вершины правой, соединенные с ним паросочетанием.

В другую сложнее — нужно воспользоваться индукцией. Будем доказывать, что если паросочетание не полное, то можно в таком графе найти увеличивающую цепь, и с её помощью увеличить паросочетание на единицу.

База индукции: одна вершина из L, которая по условию соединена с хотя бы одной вершиной из R.

Индукционный переход: пусть после k шагов построено паросочетание $k < |L|$. Докажем, что в можно добавить вершину x из L, не насыщенную паросочетанием, при том утверждение верно для любой такой x.

Рассмотрим множество вершин H — все вершины, достижимые из x, если можно ходить из правой доли в левую только по рёбрам паросочетания, а из левой в правую — по любым (мы такой граф по сути строим, когда ищем увеличивающую цепь в алгоритме Куна)

Тогда в H найдется вершина y из R, не насыщенная паросочетанием. Иначе, если такой вершины нет, то получается, что если рассмотреть вершины H_L (вершины левой доли, насыщенные паросочетанием), то для них не будет выполнено условие, что $|H_L| \leq N(H_L)$ (здесь $N(X)$ — множество вершин, соединенным паросочетанием с X).

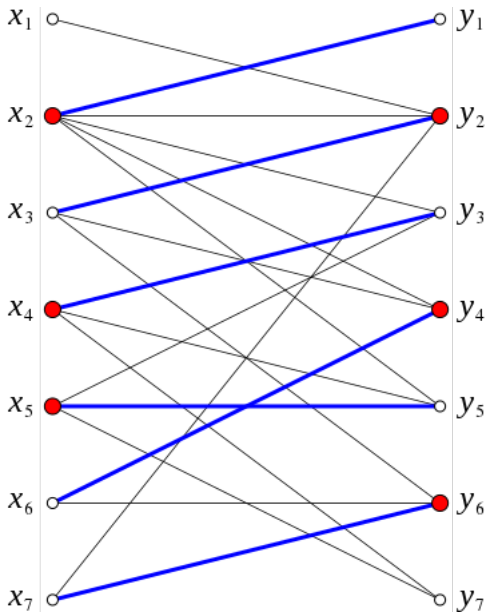
Тогда должен существовать путь из x в y, и он будет увеличивающим для паросочетания M, потому что из R в L мы всегда шли только по ребрам паросочетания. Проведя чередование вдоль этого пути, получим большее паросочетание, следовательно предположение индукции верно. □

Зачем это нужно в контексте задач по олимпиадному программированию — я не знаю. Вернее когда-то было пару задач на эту тему, но сейчас я справился найти ровно одну: CF 628

3 Минимальное вершинное покрытие

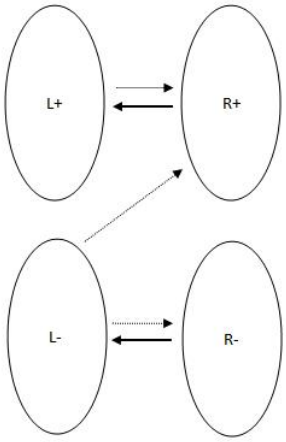
Определение. Вершинное покрытие графа — это множество вершин, такое, что любое ребро графа имеет хотя бы одну конечную вершину из этого множества. Вершинное покрытие называется наименьшим, если никакое другое вершинное покрытие не имеет меньшего числа вершин.

Теорема (Кёнига). *В любом двудольном графе число рёбер в наибольшем паросочетании равно числу вершин в наименьшем вершинном покрытии.*



Двудольный граф на рисунке вверху имеет по 7 вершин в каждой из долей. Паросочетание с 6 рёбрами выделено синим цветом, а вершинное покрытие $\{x_2, x_4, x_5, y_2, y_4, y_6\}$ выделено красным. Это покрытие является наименьшим по размеру, поскольку любая вершина в покрытии должна включать по меньшей мере одну конечную вершину ребра паросочетания. Таким же образом, нет паросочетания большего размера, поскольку любое ребро паросочетания должно содержать по меньшей мере одну конечную вершину из вершинного покрытия, так что это паросочетание является наибольшим. Теорема Кёнига как раз и утверждает равенство размеров паросочетания и покрытия (в данном примере оба числа равны шести).

Proof. Пусть в G построено максимальное паросочетание. Ориентируем ребра паросочетания, чтобы они шли из правой доли в левую, ребра не из паросочетания — так, чтобы они шли из левой доли в правую. Запустим обход в глубину из всех не насыщенных паросочетанием вершин левой доли. Разобьем вершины каждой доли графа на два множества: те, которые были посещены в процессе обхода, и те, которые не были посещены в процессе обхода. Тогда $L = L_+ \cup L_-, R = R_+ \cup R_-$, где L, R — правая и левая доли соответственно, L_+, R_+ — вершины правой и левой доли, посещенные обходом, L_-, R_- — не посещенные обходом вершины. Тогда в G могут быть следующие ребра:



- Из вершин L_+ в вершины R_+ и из вершин R_+ в вершины L_+.
- Из вершин L_- в вершины R_- и из вершин R_- в вершины L_-.
- Из вершин L_- в вершины R_+.

Очевидно, что ребер из L_+ в R_- и из R_+ в L_- быть не может по алгоритму (dfs). Ребер из R_- в L_+ быть не может, т.к. если такое ребро uv существует, то оно — ребро паросочетания. Тогда вершина v насыщена паросочетанием. Но т.к. $v \in L_+$, то в нее можно дойти из какой-то ненасыщенной вершины левой доли. Значит, существует ребро $wv, w \in R_+$. Но тогда v инцидентны два ребра из паросочетания. Противоречие.

Заметим, что минимальным вершинным покрытием G является либо L, либо R, либо $L_- \cup R_+$. В R_+ не насыщенных паросочетанием вершин быть не может, т.к. иначе в G существует дополняющая цепь, что противоречит максимальности построенного паросочетания. В L_- свободных вершин быть не может, т.к. все они должны находиться в L_+. Тогда т.к. ребер из паросочетания между R_+ и L_- нет, то каждому ребру максимального паросочетания инцидентна ровно одна вершина из $L_- \cup R_+$.

Тогда $|L_- \cup R_+|$ равна мощности максимального паросочетания. Множество вершин $L_- \cup R_+$ является минимальным вершинным покрытием. Значит мощность максимального паросочетания равна мощности минимального вершинного покрытия. $\square$

Из доказательства сразу следует алгоритм: запускаем dfs из всех ненасыщенных вершин левой доли. В качестве ответа берем $L_- \cup R_+$.

4 Цепи и антицепи

Ох, дети, сейчас вам покажется, что эта тема вообще не нужна и зачем я вас гружу математикой, но так оно и есть.

4.1 Определения

Определение. Частично упорядоченное множество — это пара $(A, \preceq)$, состоящая из множества элементов A и отношения частичного порядка $\preceq$, то есть отношения между парой объектов которое удовлетворяет следующим трем критериям:

1. $\forall a \in A : a \preceq a$
2. $\forall a, b, c \in A : a \preceq b, b \preceq c \Rightarrow a \preceq c$
3. $\forall a, b : a \preceq b, b \preceq a \Rightarrow a = b$

Например, пара $(\mathbb{R}, \leq)$ удовлетворяет этим критериям.

Определение. Два объекта $a, b \in A$ называются сравнимыми, если $a \preceq b$ или $b \preceq a$.

Обратите внимание, в определении частично упорядоченного множества не требуется, чтобы любые два объекта были сравнимы, это значит, что пара $(\mathbb{Z}, \vdots)$ тоже является частично упорядоченным множеством. Таких пар можно придумать много, и как раз о таких парах мы далее будем говорить.

Замечание. Данное отношение, задаваемое выше называется отношением нестрого порядка из-за свойства 1. Можно попросить задать строго отношение $\prec$, в котором первый пункт заменится на $\forall a \in A : \neg(a \prec a)$, также немного поменяется пункт 3. Вообще говоря, далее нам не важно, отношение будет строгим или не строгим, от этого структура задачи не меняется! Т.е. пару $(\mathbb{R}, \prec)$ мы тоже будем называть частично упорядоченным множеством.

4.2 Теорема Мирского

Теорема (Мирского). *В конечном частично упорядоченном множестве размер максимальной цепи равен минимальному количеству антицепей, на которое можно разбить множество.*

Proof. В графе частично упорядоченного множества в качестве максимальной цепи будет путь наибольшей глубины от корня. А в качестве антицепей мы возьмем вершины на одной и той же глубине. □

Задача 5. На какое минимальное количество неубывающих подпоследовательностей можно разбить данную последовательность?

Решение. Ответ равен длине НВП.

4.3 Теорема Дилуорса

Теорема (Дилуорса). *В конечном частично упорядоченном множестве размер максимальной антицепи равен минимальному количеству цепей, на которое можно разбить множество.*

Proof. Доказательство полностью повторяет разложение DUG на минимальное количество путей и задачу о максимальном независимом множестве. □

Задача 6 (нечто немного математическое)**.** В процессе эксперимента исследуется потомство из $nm + 1$ мышей. Доказать, что существует

a) либо $n + 1$ мышь, каждая из которой является потомком другой

b) либо $m + 1$ мышь, ни одна из которых не является потомком другой.

После доказательства можете придумать алгоритм построения искомого множества мышей для задачи.