

63. Алгоритм Борувки: выбор минимального ребра из нескольких, корректность, реализация, асимптотика.

Algorithm.

Начало:

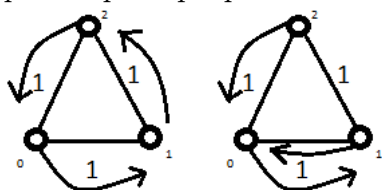
1) Изначально в лесу каждое дерево — одна вершина. Каждая вершина — отдельное дерево.

Итерация:

1) Для каждого дерева T_i из леса T выбираем самое легкое ребро, соединяющее T_i с остальным графом. Добавляем найденные ребра в лес, объединяя связанные им деревья.

Замечание:

Важно аккуратно выбирать минимальные рёбра для каждого дерева, чтобы в случае, когда из вершин исходит несколько рёбер наименьшего веса, не возникла ситуация, как на первой картинке (появился цикл). Избежать проблемы возможно, если из наименьших рёбер выбирать ребро с минимальным номером.



Теорема. Алгоритм Борувки корректен.

Доказательство:

- Будет построено дерево. Предположим, что это не так. (Будем проводить каждое неориентированное рёбро от дерева, в котором оно выбрано, ко второй вершине.) Тогда на некотором шаге будет построен простой цикл (хотя бы на 3-х вершинах), рёбра в котором были построены в одном направлении цикла (как пример - первая картинка выше). Если в цикле рёбра разной длины, то найдётся дерево, в котором было выбрано не минимальное по длине ребро. Если в цикле рёбра одной длины, то найдётся дерево, для которого было выбрано ребро большего номера, чем то, которое в него пришло. Противоречие. Следовательно, построено дерево.
- Будет построено дерево минимального веса. От противного. По аналогии с теоремой о разрезе рассмотреть первое добавленное ребро, не принадлежащее мин.остову.

Утверждение. Алгоритм Борувки работает за $O(E \log V)$

Доказательство: На каждом шаге алгоритма добавляется количество ребер равное n - количеству деревьев в лесу. Рёбра не образуют циклов, а мест, где произошли совпадения рёбер не более $\frac{n}{2}$. Следовательно, количество деревьев в лесу уменьшается как минимум в два раза. Следовательно, количество итераций не больше $\log V$. Один шаг выполняется за $O(E)$. Общее время работы — $O(E \log V)$. ■

64. Определение паросочетания в произвольном графе, двудольного графа, увеличивающего пути.

Определение. Двудольный граф — это граф, множество вершин которого можно разбить на две части таким образом, что каждое ребро графа соединяет каждую вершину из одной части с какой-то вершиной другой части, то есть не существует рёбер между вершинами одной и той же части графа.

Определение. Паросочетание в (обычном неориентированном) графе - произвольное множество ребер, такое что никакие два ребра не имеют общей вершины.

Определение. Паросочетание M в двудольном графе - произвольное множество рёбер двудольного графа, такое что никакие два ребра не имеют общей вершины.

Определение. Вершина v насыщена (инцидентна, покрыта) ребрами из паросочетания M , если v лежит в одном из рёбер M . Остальные вершины называются ненасыщенными (свободными).

Определение. Рёберное покрытие графа — множество рёбер, в котором каждая вершина графа инцидентна по меньшей мере одному ребру покрытия.

Определение. *Максимальное* паросочетание — это паросочетание в графе G , содержащее максимальное количество ребер

Определение. Паросочетание называется совершенным(полным), если оно покрывает все вершины графа

Определение. Чередующаяся цепь (путь) - цепь в двудольном графе, для любых двух соседних рёбер которой верно, что одно из них принадлежит паросочетанию, а другое нет

Определение. Увеличивающая цепь (путь) в G относительно паросочетания M - чередующаяся цепь, у которой оба конца свободны.

65. Лемма об устройстве неориентированного графа, в котором степени всех вершин не превосходят двух.

Теорема. Неориентированный граф, в котором степени всех вершин не превосходят двух, состоит из цепей и циклов.

Очевидно. Доказывается проходом по графу.

66. Теорема Бержа.

Теорема. (Бержа) Паросочетание M в двудольном графе G является максимальным тогда и только тогда, когда в G нет дополняющей цепи.

Доказательство:

⇒

Пусть в двудольном графе G с максимальным паросочетанием M существует дополняющая цепь. Тогда пройдя по ней и заменив вдоль неё все рёбра, входящие в паросочетание, на невходящие и наоборот, мы получим большее паросочетание. То есть M не являлось максимальным. Противоречие.

⇐

От противного. Пусть M — не максимальное. Пусть M' — другое паросочетание, $|M'| > |M|$. Рассмотрим их симметрическую разницу. В таком графе все вершины имеют степень не больше 2 (т.е. все компоненты будут цепями или циклами). Найдется компонента, в которой ребер M' больше. Это будет увеличивающей цепью для M , а не циклом, так как в цикле всегда одинаковое количество рёбер из M и M' . А это противоречие к условию отсутствия для M увеличивающих цепей.

67. Алгоритм Куна. Корректность, реализация, асимптотика.

Алгоритм Куна.

Задан граф $G(V, E)$, про который известно, что он двудольный, но разбиение не задано явно. Требуется найти наибольшее паросочетание в нём

Алгоритм можно описать так: сначала возьмём пустое паросочетание, а потом — пока в графе удаётся найти увеличивающую цепь, — будем выполнять чередование паросочетания вдоль этой цепи, и повторять процесс поиска увеличивающей цепи. Как только такую цепь найти не удалось — процесс останавливаем, — текущее паросочетание и есть максимальное.

В массиве `matching` хранятся паросочетания ($v, \text{matching}[v]$) (Если паросочетания с вершиной v не существует, то $\text{matching}[v] = -1$). А `used` — обычный массив "посещённостей" вершин в обходе в глубину (он нужен, чтобы обход в глубину не заходил в одну вершину дважды). Функция `dfs` возвращает `true`, если ей удалось найти увеличивающую цепь из вершины v , при этом считается, что эта функция уже произвела чередование паросочетания вдоль найденной цепи.

Внутри функции просматриваются все рёбра, исходящие из вершины v , и затем проверяется: если это ребро ведёт в ненасыщенную вершину to , либо если эта вершина to насыщена, но удаётся найти увеличивающую цепь рекурсивным запуском из `matching[to]`, то мы говорим, что мы нашли увеличивающую цепь, и перед возвратом из функции с результатом `true` производим чередование в текущем ребре: перенаправляем ребро, смежное с to , в вершину v .

В основной программе сначала указывается, что текущее паросочетание — пустое (массив `matching` заполняется числами -1). Затем перебирается вершина v , и из неё запускается обход в глубину `dfs`, предварительно обнулив массив `used`.

```
1 bool dfs(v: int):
2     if (used[v])
3         return ''false''
4     used[v] = ''true''
5     for to in g[v]
6         if (matching[to] == -1 or dfs(matching[to])):
7             matching[to] = v
8             return ''true''
9     return ''false''
10
11 function main():
12     fill(matching, -1)
13     for i = 1..n
14         fill(used, ''false'')
15         dfs(i)
16     for i = 1..n
17         if (matching[i] != -1)
18             print(i, " ", matching[i])
```

Итак, алгоритм Куна можно представить как серию из n запусков обхода в глубину на всём графе. Следовательно, всего этот алгоритм выполняется за время $O(nm)$, где m — количество рёбер, что в худшем случае есть $O(n^3)$. Если явно задано разбиение графа на две доли размером n_1 и n_2 , то можно запускать **dfs** только из вершин первой доли, поэтому весь алгоритм выполняется за время $O(n_1m)$. В худшем случае это составляет $O(n_1^2n_2)$.

68. Лемма об отсутствии увеличивающих путей из вершины при отсутствии таких путей относительного меньшего паросочетания.

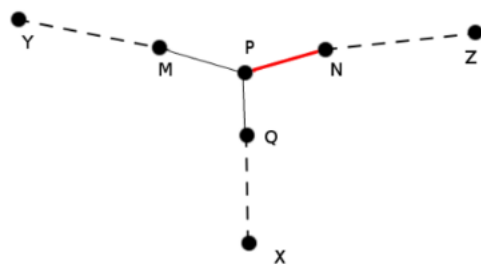
Теорема. Если из вершины x не существует увеличивающей цепи относительно паросочетания M , тогда из x не существует увеличивающей цепи в M' (паросочетание M' получается из M изменением вдоль увеличивающей цепи).

Доказательство:

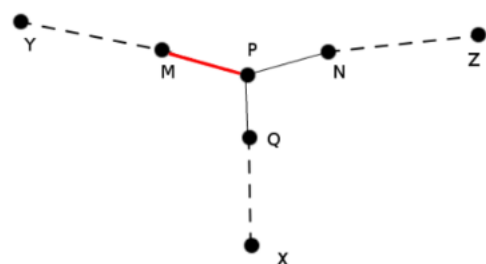
Доказательство от противного.

Допустим в паросочетание внесли изменения вдоль дополняющей цепи $(y \rightsquigarrow z)$ (цепь из y в z) и из вершины x появилась дополняющая цепь. Заметим, что эта дополняющая цепь должна вершинно пересекаться с той цепью, вдоль которой вносились изменения, иначе такая же дополняющая цепь из x существовала и в исходном паросочетании.

паросочетание M



паросочетание M'



Пусть p — ближайшая к x вершина, которая принадлежит и новой дополняющей цепи и цепи $(y \rightsquigarrow z)$. Тогда MP — последнее ребро на отрезке $(y \rightsquigarrow p)$ цепи $(y \rightsquigarrow z)$, NP — последнее ребро на отрезке $(z \rightsquigarrow p)$ цепи $(y \rightsquigarrow z)$, QP — последнее ребро лежащее на отрезке $(x \rightsquigarrow p)$ новой дополняющей цепи.

Допустим MP принадлежит паросочетанию M' , тогда NP ему не принадлежит. (Случай, когда NP принадлежит паросочетанию M' полностью симметричен.)

Поскольку паросочетание M' получается из M изменением вдоль дополняющей цепи $(y \rightsquigarrow z)$, в паросочетание M входило ребро NP , а ребро MP нет. Кроме того, ребро QP не лежит ни в исходном паросочетании M , ни в паросочетании M' , в противном случае оказалось бы, что вершина p инцидентна нескольким рёбрам из паросочетания, что противоречит определению паросочетания.

Тогда заметим, что цепь $(x \rightsquigarrow z)$, полученная объединением цепей $(x \rightsquigarrow p)$ и $(p \rightsquigarrow z)$, по определению будет дополняющей в паросочетании M , что приводит к противоречию, поскольку в паросочетании M из вершины x не существует дополняющей цепи.

69. Определения независимого множества, вершинного покрытия. Связь определений.

Определение. Подмножество вершин графа G называется независимым множеством, если не существует ребра графа соединяющего какие-либо две из них.

Определение. Подмножество вершин графа G называется вершинным покрытием G , если любое ребро графа содержит вершину из этого множества.

Связь. Подмножество вершин является независимым множеством тогда, и только тогда когда его дополнение является вершинным покрытием.

(лемма следует из определения)

Утверждение. Подмножество вершин является максимальным независимым множеством тогда, и только тогда когда его дополнение является минимальным вершинным покрытием.

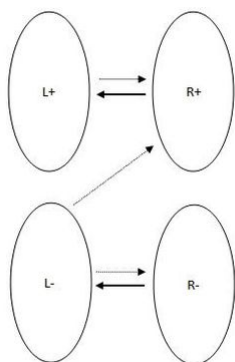
70. Алгоритм поиска максимального независимого множества и минимального вершинного покрытия в двудольном графе с помощью разбиения на доли $L^-; L^+; R^-; R^+$ (с доказательством).

Пусть G - двудольный, в нём M - максимальное паросочетание. Строим D - граф G , в котором рёбра из M ориентированы справа налево, а остальные слева направо. Пусть L, R - множество вершин левой и правой доли соответственно. Запустим обход из всех ненасыщенных вершин левой доли. Пусть L^- - посещённые обходом вершины левой доли, L^+ - непосещённые вершины левой доли. Для правой доли аналогично. Тогда

$L^- \cup R^+$ — минимальное вершинное покрытие

$L^+ \cup R^-$ — максимально независимое множество

Корректность:



Понятно, что не могут быть рёбра из L^+ в R^- и из R^+ в L^- . Нет рёбер из R^- в L^+ . Предположим противное и существует такое ребро из u в v . Единственные рёбра ведущие справа налево это рёбра из M . Следовательно, v насыщена, следовательно, из неё не запускали обход. Следовательно, существует путь до v из какой-нибудь ненасыщенной вершины левой доли w . Но единственный способ попасть в вершину v из правой доли это ребро (u, v) . При этом вершина u была посещена обходом, а v нет. Противоречие. Следовательно, нет рёбер из R^- в L^+

По картинке видно, что $L^+ \cup R^-$ — **независимое множество**, а $L^- \cup R^+$ — **вершинное покрытие**.

Осталось доказать, что $L^- \cup R^+$ — минимальное вершинное покрытие. Покажем, что $|L^- \cup R^+| \leq |M|$:

- 1) В L^- лежат только насыщенные паросочетанием M вершины.
- 2) В R^+ лежат только насыщенные вершины (иначе есть увеличивающий путь)
- 3) $\forall (u, v) \in M : u \in L^-$ или $v \notin R^+$.

Покажем, что множество вершинного покрытия не меньше мощности максимального паросочетания:

Понятно, что каждое ребро M будет покрыто хотя бы одной вершиной из вершинного покрытия.

Показали оценку снизу, показали оценку сверху. Следовательно, $|VC| \geq |M| \geq |L^- \cup R^+| \geq |VC|$, где VC — минимальное вершинное покрытие. Следовательно,

$L^- \cup R^+$ — минимальное вершинное покрытие

$L^+ \cup R^-$ — максимально независимое множество

71. Алгоритм поиска максимального независимого множества и минимального вершинного покрытия в двудольном графе с помощью задачи 2SAT.

В предыдущем доказательстве мы показали, что мощность минимального вершинного покрытия равна мощности максимального паросочетания. Следовательно, каждая вершина из минимального вершинного покрытия лежит на рёбре максимального паросочетания и ровно одна на каждом ребре. Значит нам необходимо решить задачу какую вершину каждого ребра максимального паросочетания нужно выбрать для вершинного покрытия. Введём обозначения: переменная $x_1 = 1$, если вершина минимального покрытия может лежать на правой стороне первого ребра максимального паросочетания, и $x_1 = 0$, если вершина минимального покрытия может лежать на левой стороне ребра. Для остальных рёбер та же паросочетания аналогично. Далее для каждого ребра выписываем логическую формулу, так чтобы при удовлетворении ей вершина минимального покрытия покрыла данное ребро. Получили систему логических формул, решив которую получим минимальное вершинное покрытие. Систему можно решить с помощью алгоритма 2SAT.