Алгоритмы и структуры данных, лекции

Мы начнём с паросочетаний и потоков.

1 Паросочетания

Паросочетание графа — это набор его ребер, не имеющих общих вершин.

1.1 Паросочетание в двудольном графе

Паросочетания в двудольных графах ищутся гораздо проще, чем в произвольных. Алгоритмы решают эту задачку понемногу. Берется маленькое (пустое) паросочетание и расширяется с помощью дополняющих цепочек.

Дополняющая цепочка — это путь, который начинается и заканчивается в вершинах вне паросочетания, и ребра в нём чередуются (принадлежит - не принадлежит пар. сочетанию). Из такой цепочки можно получить паросочетание большего размера (на 1).

Алгоритм начинает строить цепочки и удаляет дополняющие цепочки такой заменой.

Теорема 1. В графе нет дополняющего пути тогда и только тогда, когда паросочетание максимально.

Это утверждение сформулировано для произвольных графов: для двудольных и не очень. А в чём проблема? Проблема в поиске дополняющих путей.

В двудольном графе это делается просто. Это алгоритм Куна.

Двудольный граф можно хранить не как нормальный граф. "Алгоритм КУНА. Это не связано с аниме никак!" "Код, на самом деле, за пять копеек." Асимптотика: $O(nm)$.

Алгоритм никогда не освобождает вершины — если вершшина попала в пар. соч., она там останется, можно dfs из неё не запускать. Чуть сложнее: если dfs однажда не нашел доп. пути из одной вершины, он никогда его не найдёт — можно его не запускать.

Немного о полных сочетаниях.

Теорема 2. В двудольном графе $G_{n,n}$ существует полное паросочетание в том и только том случае, когда для любого подмножества A вершин одной доли выполнено $|N(A)| \geqslant |A|$.

Паросочетания позволяют решать кучу прикольных задач.

1.2 Вершинное покрытие

Вершинное покрытие в графе — это набор вершин, таких, что никаие две вершины не лежат на одном ребре. Это понятие двойственное понятию паросочетания. Мы ищем минимальное по мощности вершинное покрытие. Это NP-полная задача в произвольном графе. В двудольном, однако, всё очень мило.

Размер покрытия в графе всегда не превосходит размер вершинного покрытия: $M \leqslant S$.

1.3 Парасочетание в недвудольном графе

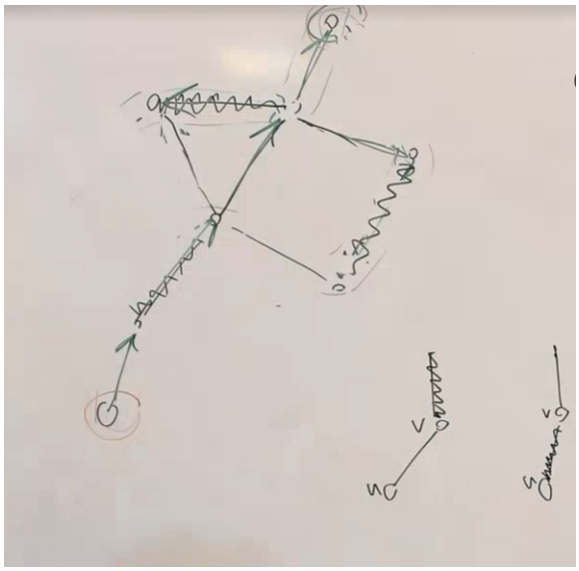
Так же понемногу будем искать дополняющие пути и достраивать наше паросочетание.

Как уже было доказано, M — max парсоч $\iff$ не. т дополняющих путей.

Как искать дополняющие пути? Может быть заюзать `dfs`?

```
    dfs(v, state)
        ...
    for n
        if state:
            (u,v) ∈ M
        else:
            (u,v) ∈ M
        dfs(n, !state)
```

Но так нельзя!
На таком графе не сработает:

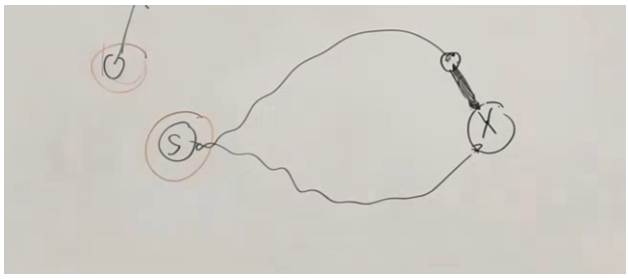


Проблема в том, что в какую-то вершину мы можем прийти с неправильной четностью.

Утверждение: если такого не случилось, то все будет ок.

Пусть есть стартовая вершина. Назовем вершину сомнительной, если до нее есть четный и нечетный пути.

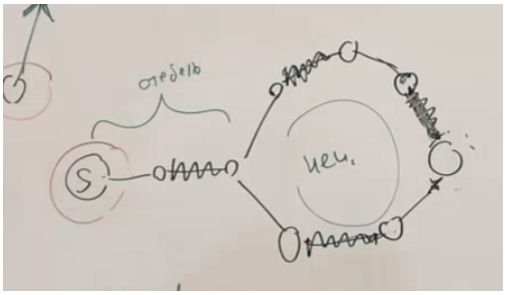
Если в графе нет сомнительных вершин, такой граф можно переделать в двудольный и легко в нем постить паросочетание.



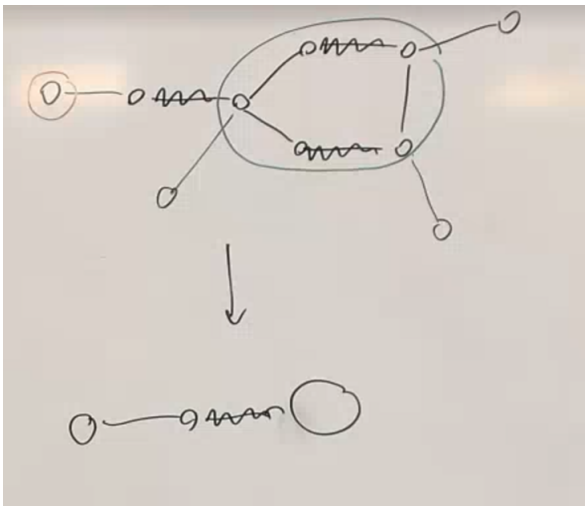
Что делать? Давайте запустим такой dfs. Может быть три случая:

1. Нашли дополняющий путь;
2. Не нашли дополняющий путь и нет сомнительных вершин $\Longrightarrow$ паросочетание максмальное;
3. Нашли сомнительную вершину.

Если случилось третье, будем веселиться. При помощи того же dfs-а найдем соцветие (blossom). Внутри соцветия цикл нечетной длинны.



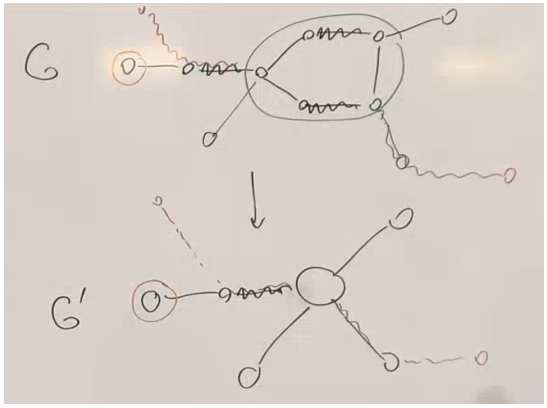
Алгоритм называется blossom cut. Возьмем все вершины соцветия и заменим на большую вершину.



Утверждение: если в исходном графе G был дополняющий путь, то и в полученном графе G'.

Доказательство. Докажем в две стороны

($\Leftarrow$) Либо дополняющий путь вообще не проходит через большую вершину соцветия, тогда вообще все просто, ничего точно не поломается.
Если дополгяющий путь проходит через большую вершину соцветия, то выберем кусок цикла нужной четности и соседеним кусочки дополняющего пути.



($\Rightarrow$) Все сложно. Могут быть всякие сложные пути и после сжатия не понятно, что с ними делать. Конструктивно доказать сложно.

Докажем при помощи теоремы о дополняющем пути. От противного.

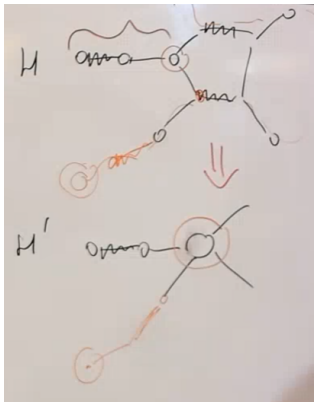
1. Возьмем граф G вместе с соцветием и инвертируем ему стебель, получим граф H с валидным парсочем.

2. Возьмем граф H вместе с соцветием и инвертируем ему стебель, получим граф H' с валидным парсочем.

3. Если в G есть дополняющий путь, то и в G' тоже есть дополняющий путь (очевидно, так как и то и другое паросочетание не максимального размера).

   Если в H' есть дополняющий путь, то и в H тоже есть дополняющий путь (очевидно, так как и то и другое паросочетание не максимального размера).

   Если в G' есть дополняющий путь, то и в H' есть дополняющий путь (конструктивно).



□

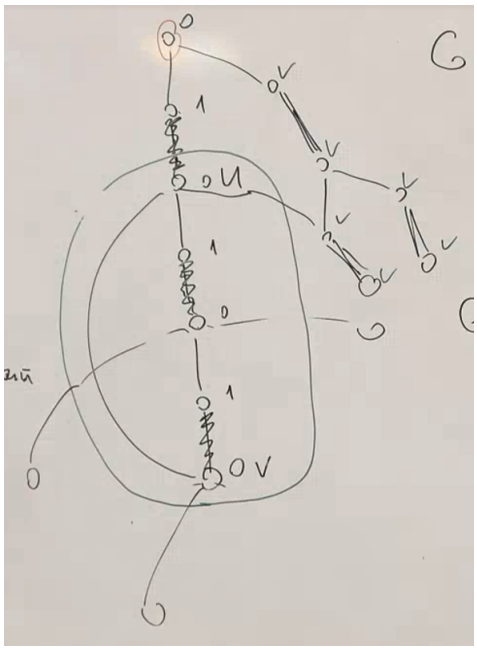
Алгоритм

```
M = ∅
while True:
    dfs
    if нашли доп путь:
        разжать соцветие
        M++
        continue
    if нет додоп пути, не соцветий:
        break
    if соцветие:
        сжать соцветие
```

Итого, время работы алгоритма — $\mathcal{O}(n^2m)$.

Как реализовать побыстрее? *Немного пострадать*.

Пооптимизируем DFS. Прямо проходя в DFS–е можно сжимать вершины. Для этого можно быстро мерджить списки ребер.



В общем, если постараться, можно получить решение $\mathcal{O}(nm\alpha(m,n))$. А вообще, пацаны умеет делать за $\mathcal{O}(m\sqrt{n})$.