Объектно-ориентированное программирование

Петрусевич Денис Андреевич

Доцент кафедры Проблем управления

petrusevich@mirea.ru

Тема 7. Задача о максимальном потоке

В невзвешенном графе каждому ребру присвоена пропускная способность: количество груза, которое можно перевезти по нему, за единицу времени

Требуется найти максимальное количество грузов, которое можно провезти через транспортную систему, выражаемую графом, за единицу времени

Желательно предоставить маршруты, по которым нужно перевозить грузы; дать величину груза, отправляемую по каждому маршруту

Тема 7. Максимальный поток

Одно из первых решений – алгоритм Форда-Фалкерсона (1956)

В алгоритме Диница вводятся псевдомаксимальные потоки и вспомогательные контурные сети (1970)

Карзанов улучшает метод Диница, вводя предпоток (1974)

Голдберг и Тарьян вводят push-relabel метод (1986)

Одна из самых последних «больших» работ – Голдберг и Рао (1997)

Тема 7. Решения задачи

n = |V| - число вершин

m = |E| - число ребер

U – максимум пропускной способности сети

см. ссылку на таблицу в литературе

Год	Автор	Оценка
1951	Dantzig	$O(n^2mU)$
1956	**Ford & Fulkerson**	**$O(nmU)$**
1970	**and Karp**	**$O(nm^2)$**
1970	**Dinic**	**$O(n^2m)$**
1972	**and Karp**	**$O(m^2LogU)$**
1973	**Dinic Gabow**	**O(nmLogU)**
1974	**Karzanov**	**$O(n^3)$**
1977	Cherkasky	$O(n^2m^{1/2})$
1978	**Malhotra, Pramodh Kumar, and Maheshwari**	**$O(n^3)$**
1978	Galil	$O(n^{5/3}m^{2/3})$
1978	**Galil & Naamad Shiloach**	**$O(nmLog^2n)$**
1980	**Sleator and Tarjan**	**O(nmLogn)**
1982	Shiloach & Vishkin	$O(n^3)$
1984	Tarjan	$O(n^3)$
1985	Goldberg	$O(n^3)$
1986	**Goldberg & Tarjan**	**$O(nmLog(n^2/m))$**
1987	Ahuja and Orlin	$O(nm + n^2LogU)$
1987	Ahuja et al.	$O(nmLog(n(Log^{1/2}U)/m))$
1989	**Cheriyan, Hagerup & Mehlhorn**	**$E(nm + n^2Log^2n)$**
1990	Cheriyan et al.	$O(n^3/logn)$
1990	Alon	$O(nm + n^{8/3}Logn)$
1992	**King, Rao & Tarjan**	**$O(nm + n^{2+e})$**
1993	Phillips & Westbrook	$O(nm(Log_{m/(Logn)}n))$
1997	**Goldberg & Rao**	**$O(min\{n^{2/3},n1^{/2}\}m Log(n^2/m)LogU)$**

Тема 7. Алгоритм Форда-Фалкерсона

- Дана матрица пропускной способности *c*: эквивалентна матрице смежности, но вместо весов содержит пропускные способности (сколько груза можно провезти по ребрам за единицу времени)
- Матрица потока *f* показывает, сколько единиц груза перевозится по каждому ребру
- *Допустимая дуга* e = (u, v) относительно потока *f* :

  а) e = (u, v) и f(e) ≤ c(e) (согласованная дуга) или

  б) e = (v, u) и f(e) > 0 (несогласованная дуга).
- *Увеличивающая цепь* (длины l) из вершины-источника s (source) в вершину-сток t (target) - последовательность допустимых дуг, начинающаяся в s и заканчивающаяся в t
- Для простоты опустим работу с несогласованными дугами

Тема 7. Алгоритм Форда-Фалкерсона

- *Увеличивающая цепь* (длины l) из вершины-источника s (source) в вершину-сток t (target) - последовательность допустимых дуг, начинающаяся в s и заканчивающаяся в t

1. Найти новую увеличивающую цепь (если она не найдена, – выход)
2. По увеличивающей цепи пустить поток величины F, где $F = \min\{c(e_i), 1 \le i \le l\}$ (минимум пропускной способности составляющих цепь ребер):

   для всех ребер цепи обновляем пропускную способность $c(e_i) = c(e_i) – F, 1 \le i \le l$

   *запомнить цепь и величину F
3. Перейти к шагу 1

Тема 7. Алгоритм Форда-Фалкерсона

Алгоритм поиска новой увеличивающей цепи?

В невзвешенном графе необходимо найти путь из источника в сток

Требования к пути:

- ребра должны иметь ненулевую пропускную способность $c(e_i)>0$ (быть допустимыми дугами)
- в классическом понимании алгоритма применяется «жадный» подход: из набора ребер, выходящих из текущей вершины, выбираем ребро с максимальной пропускной способностью
- отмечаем использованные в цепи вершины, чтобы не возвращаться назад

Тема 7. Алгоритм Форда-Фалкерсона

Алгоритм поиска новой увеличивающей цепи?

Текущая вершина - источник

Обрабатываем все вершины, смежные с текущей, если ребра, ведущие к ним не насыщенны (добавляем в структуру данных)

Пока структура данных не пуста

Достаем вершину

Обрабатываем все смежные вершины (если к ним ведут допустимые дуги / ребра с ненулевой пропускной способностью, и они ещё не использовались в этой цепи)

Если одна из них сток, - строим поток по полученной цепи

Желательно первой обрабатывать вершину, к которой ведет ребро с максимальной пропускной способностью («жадность»)

Тема 7. Алгоритм Форда-Фалкерсона

Алгоритм поиска новой увеличивающей цепи – модификация поиска в ширину

К условию не использовать пройденные вершины добавляем условие допустимости дуг / используем ребра с ненулевой пропускной способностью

Гарантируется, что, когда не найден новый путь в графе (их допустимых дуг), определен максимальный поток

Не гарантируется, что каждая цепь обладает максимальной пропускной способностью из всех возможных цепей

Тема 7. Поиск максимального паросочетания

Дан двудольный граф (вершины двух типов: «мальчики и девочки»)

Граф содержит информацию о том, как можно объединить вершины первого и второго типа в пары (вершины одного типа не соединяются)

Каждая вершина в итоговом паросочетании может участвовать только в одной паре

Какое максимальное количество пар можно составить?

*Какие пары могут быть в максимальном паросочетании)

*Выгоднее сделать левую долю меньшей по числу вершин

Тема 7. Сведение к поиску максимального потока

Все ребра ориентируем и направляем из левой доли в правую. Добавляем источник s и сток t

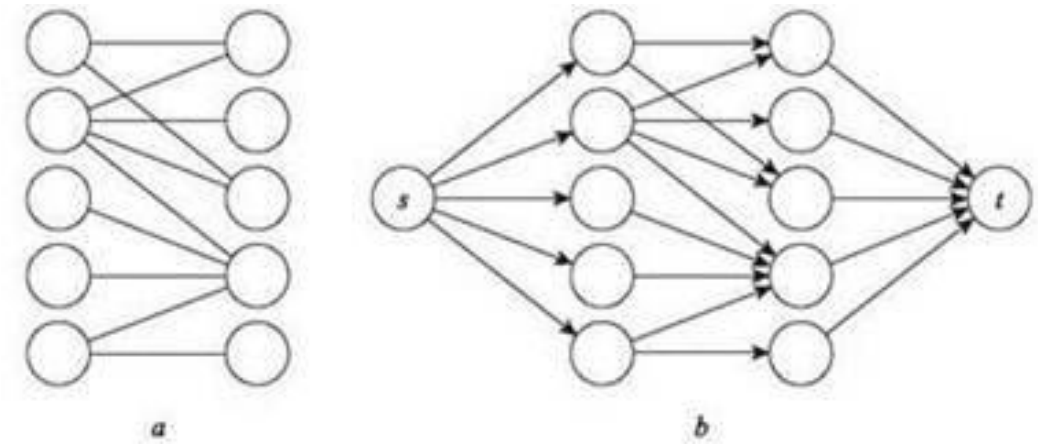
Каждое ребро имеет пропускную способность 1 (будет участвовать не более, чем в одном паросочетании / будет участвовать не более, чем в одной цепи)

Запускаем алгоритм поиска максимального потока

Величина максимального потока =

максимальному числу пар

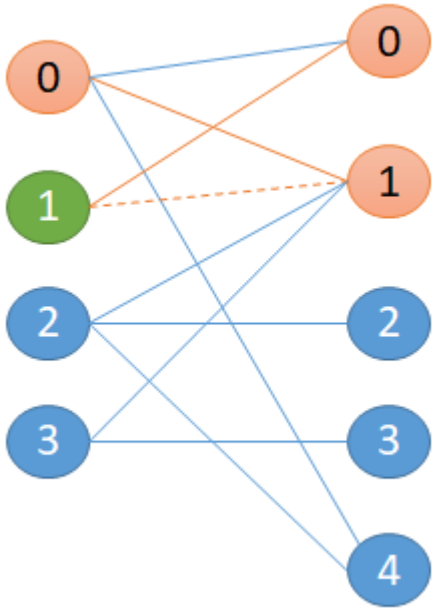
*Как определить списки пар?



Тема 7. Алгоритм Куна

Цепь длины – некоторый простой путь (т.е. не содержащий повторяющихся вершин или рёбер).

Чередующаяся цепь (в двудольном графе, относительно некоторого паросочетания) – цепь, в которой рёбра поочередно принадлежат/не принадлежат паросочетанию (фиктивными ребрами соединяем конец ребра в предыдущей паре и начало ребра в следующей).



Тема 7. Алгоритм Куна

Увеличивающей цепь (в двудольном графе, относительно некоторого паросочетания) - чередующаяся цепь, у которой начальная и конечная вершины не принадлежат паросочетанию.

Теорема Бержа: паросочетание является максимальным тогда и только тогда, когда не существует увеличивающих относительно него цепей.

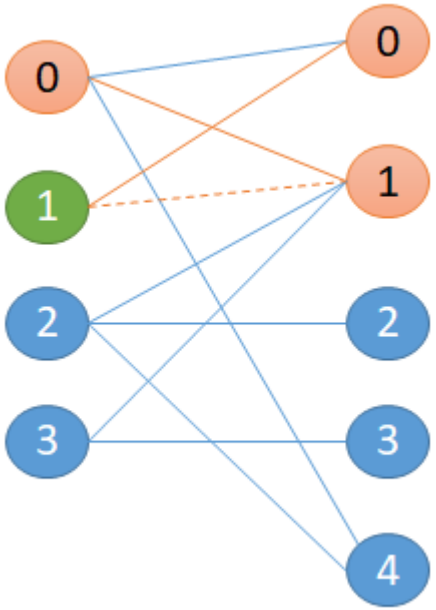
Алгоритм Куна

Инициализация пустого паросочетания

Пока найдена увеличивающая цепь,

Выполнять чередование паросочетания вдоль неё

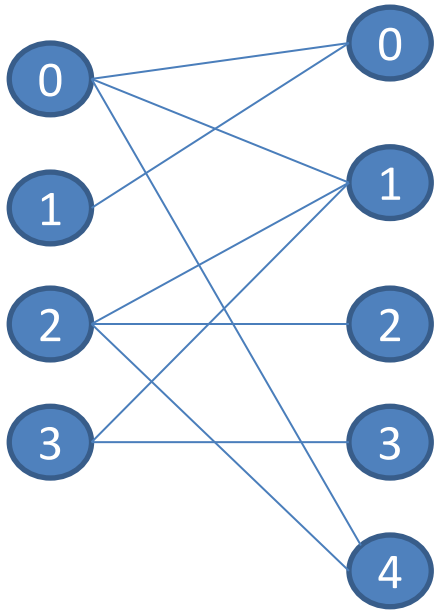
(добавить новую пару)



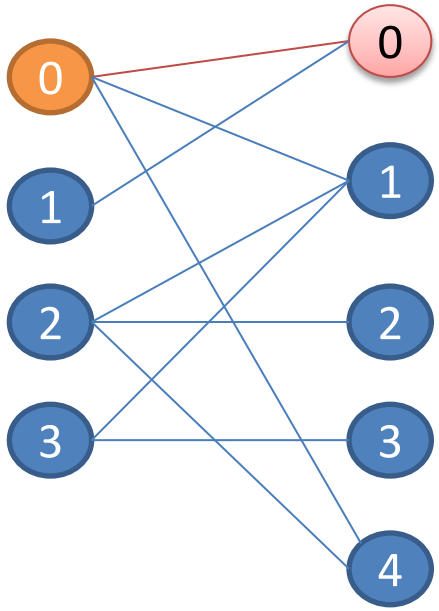
Когда цикл закончился, паросочетание и есть максимальное.

Тема 7. Алгоритм Куна

Начало: с 0 вершины в левой доле



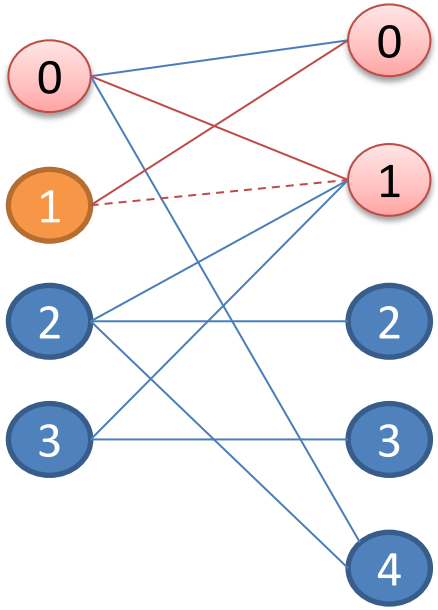
Итерация 0: пара (0, 0)



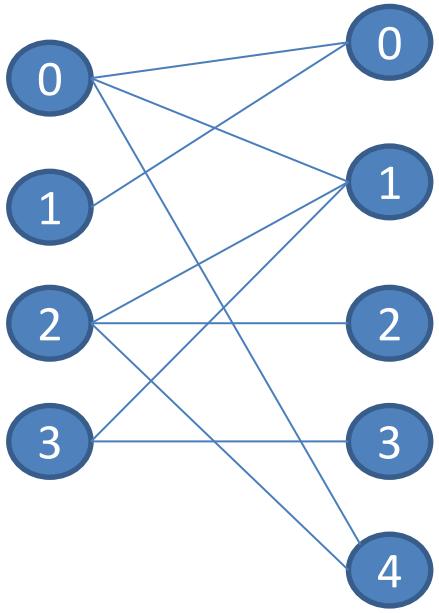
Итерация i: начинаем подбор пар с вершины 1 слева

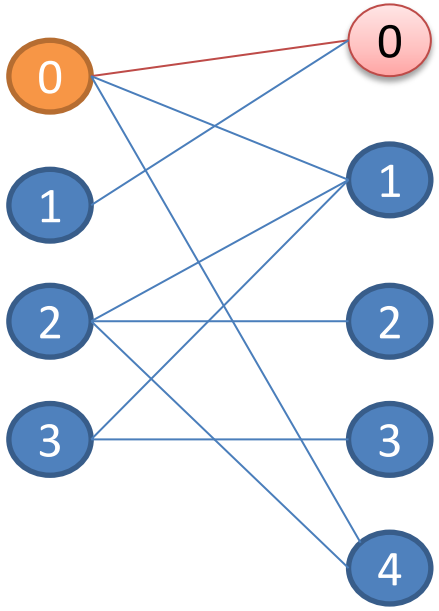
Пара (1, 0)

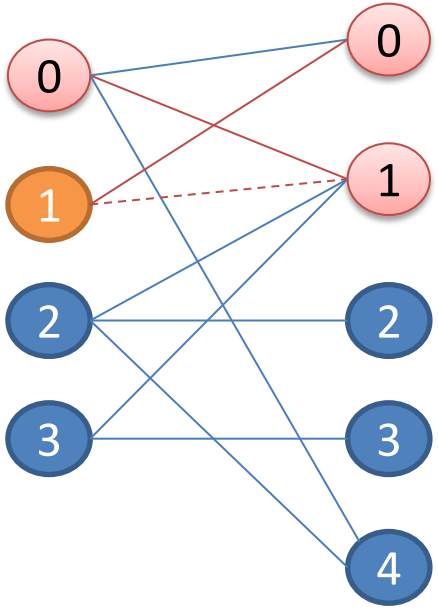
Пара (0, 1)



Тема 7. Алгоритм Куна








```
g = {       //список
смежности
{ 0, 1, 4},
{ 0 },
{ 1, 2, 4 },
{ 1, 3 },
};
```

mt=[0,-1,-1,-1]

mt=[1,0,-1,-1]

used=[0,0,0,0] – использованные вершины

mt=[-1,-1,-1,-1] – текущее решение

Тема 7. Оценка времени работы

Подбор пар – поиск в глубину, запускаемый из каждой вершины слева

Рационально в первую очередь рассматривать самые «разборчивые» вершины (узкое место процесса)

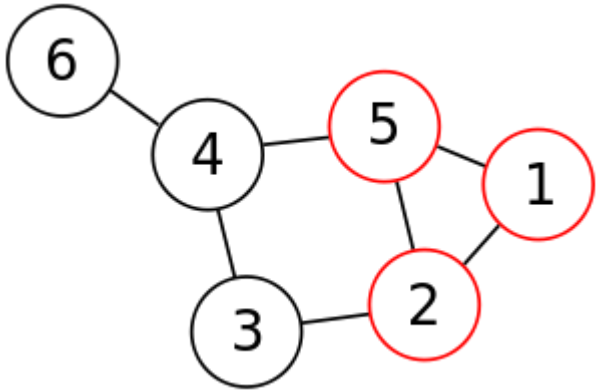
Оценка алгоритма по производительности: $O(v_{left}^2 v_{right})$, здесь v – число вершин в левой/правой доле

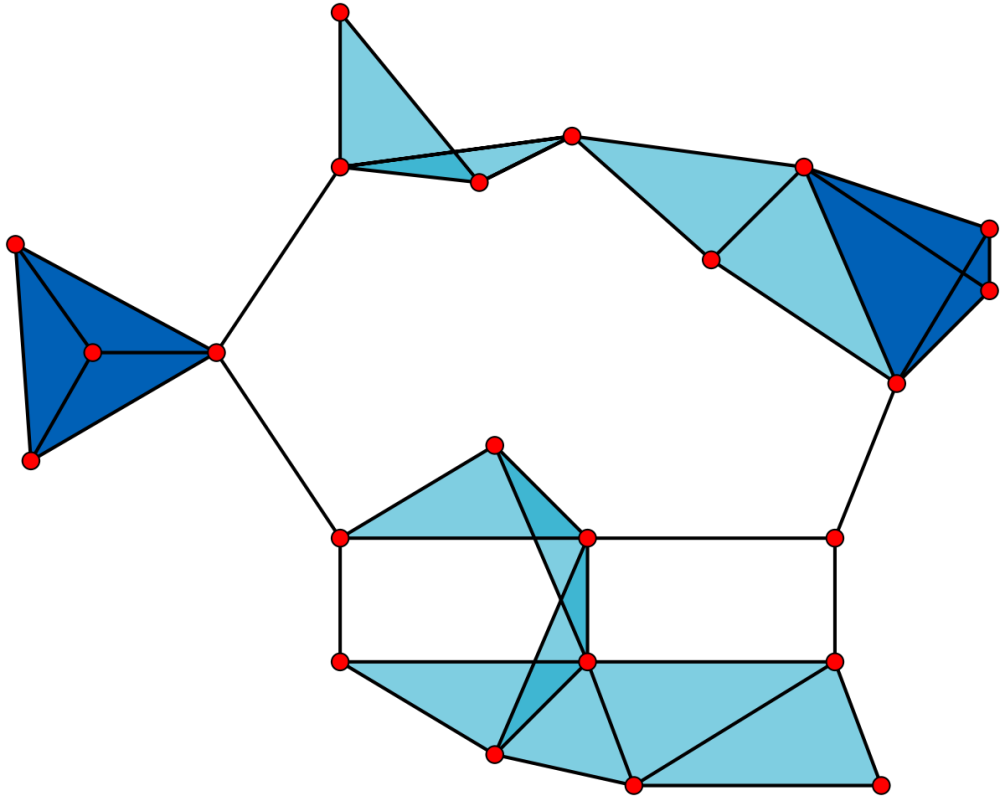
Тема 7. Поиск клик в графе

Клика – полносвязное подмножество в графе

Поиск клик – NP-полная задача, эффективных способов решения нет

Можно сокращать перебор, делая его «умным»

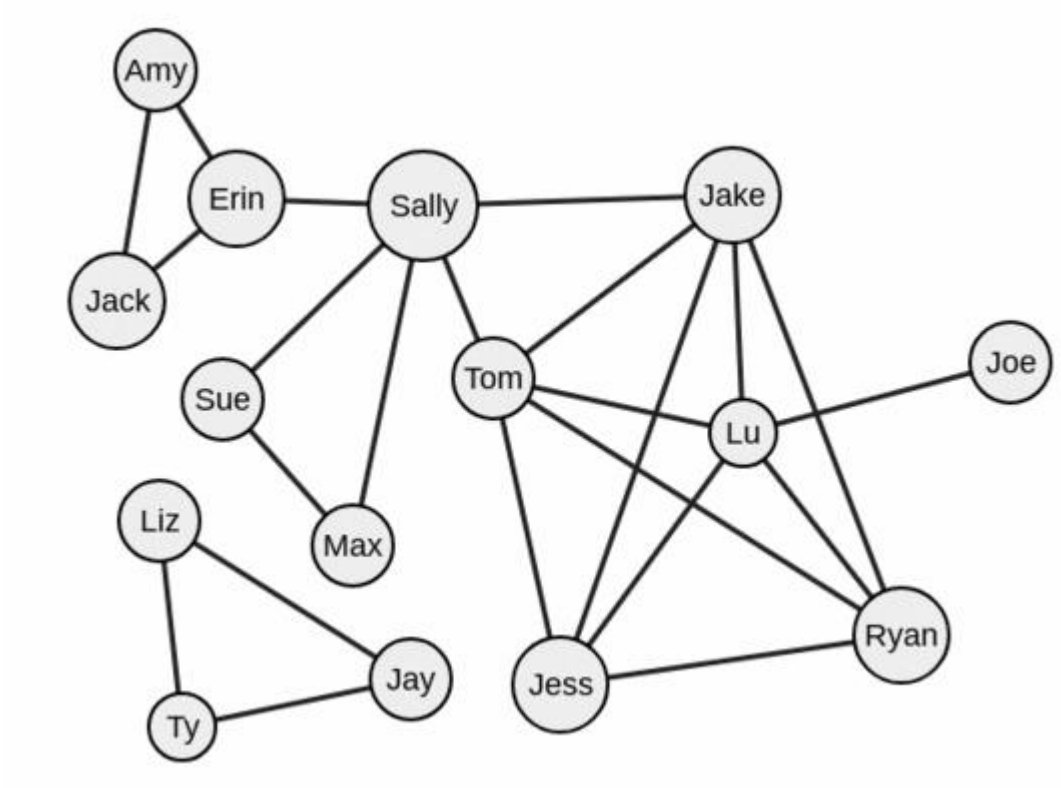




Тема 7. Алгоритм Брона-Кербоша

Найти максимальные клики: здесь множества людей, где все знакомы друг с другом

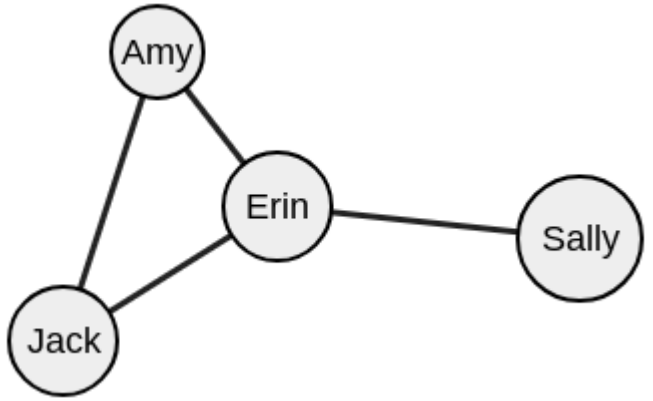
Метод рекурсивного поиска с возвратом. Сложность: $O(3^{n/3})$

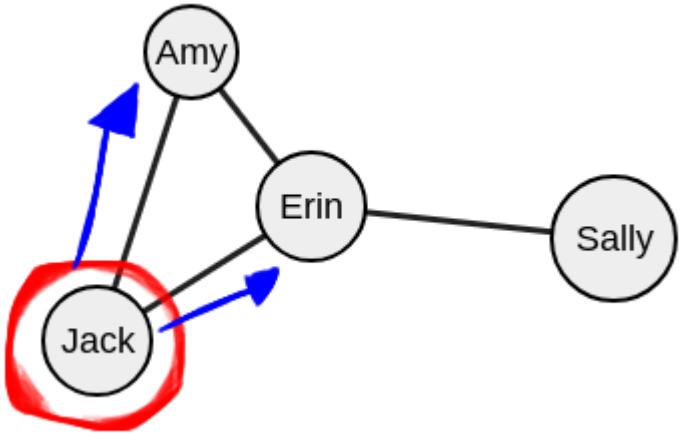


Тема 7. Алгоритм Брона-Кербоша

Пример подграфа:

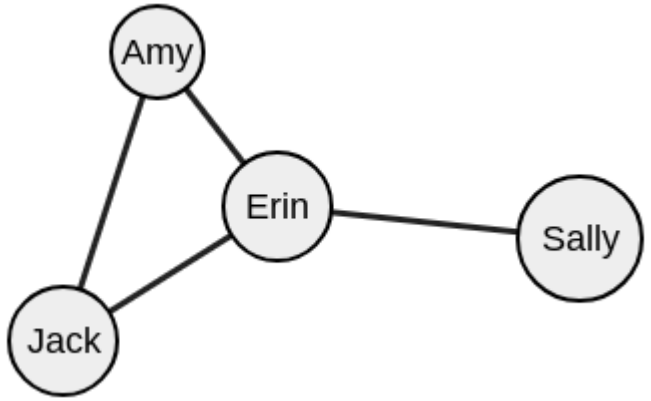
1. Рассмотрим только знакомых (соседей) Джека





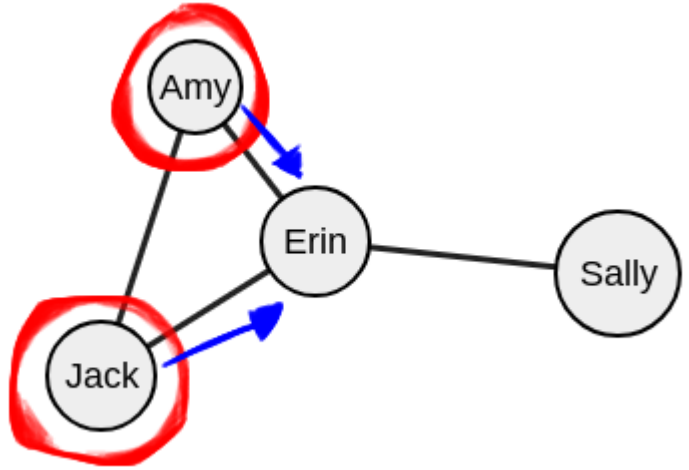
Тема 7. Алгоритм Брона-Кербоша

Пример подграфа:



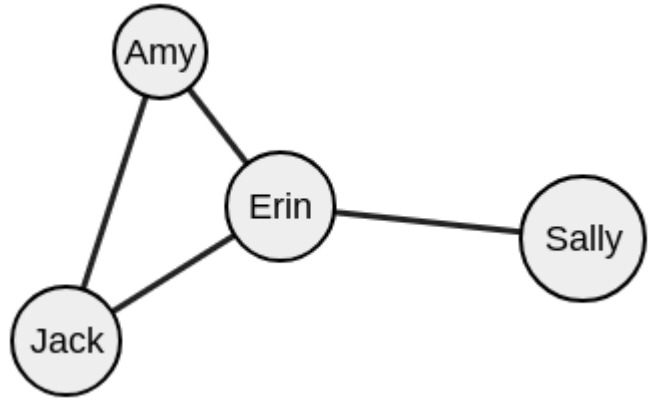
2. Среди соседей вершин начнем перебор

Например, рассматриваем Эми



Тема 7. Алгоритм Брона-Кербоша

Пример подграфа:

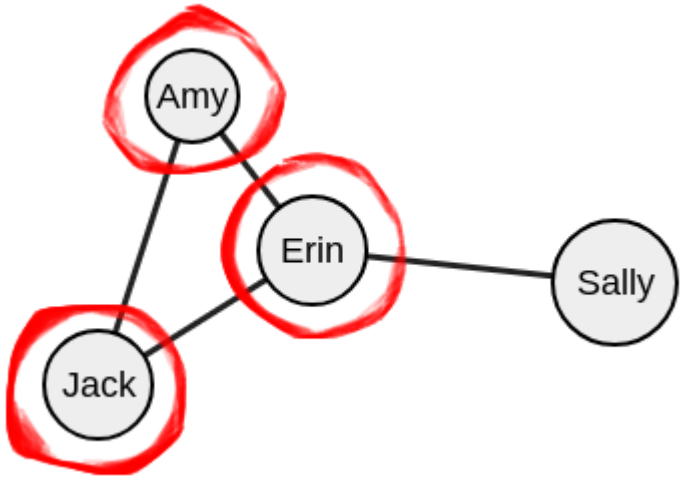


2. Рассмотрим соседей Эми

Причём, нас интересует пересечение множеств

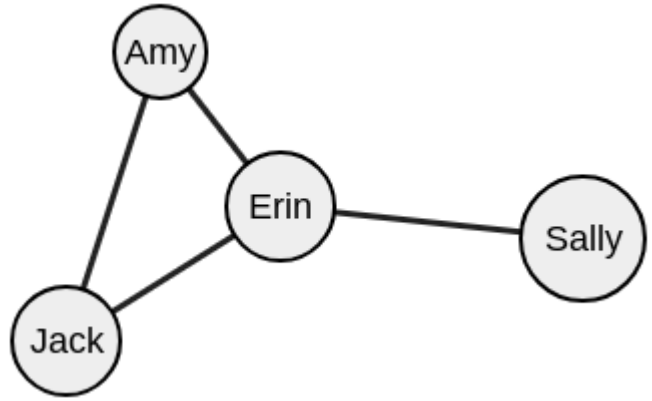
«знакомые Эми», «знакомые Джека»

3. Среди них здесь есть только Эрин



Тема 7. Алгоритм Брона-Кербоша

Пример подграфа:

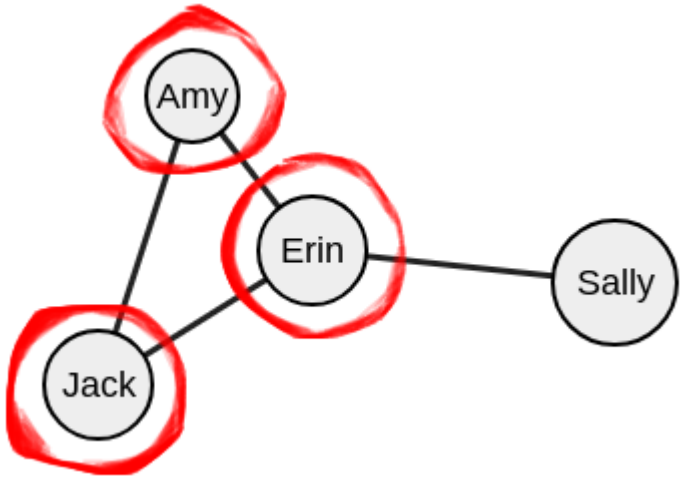


2. Рассмотрим соседей Эми

Причём, нас интересует пересечение множеств «знакомые Эми», «знакомые Джека»

3. Среди них здесь есть только Эрин

4. Так же можно обнаружить клику «Эрин – Салли»



Ищем множество: 1) каждая пара вершин связана ребрами, 2) при добавлении новых вершин в множество теряется связность

Тема 7. Алгоритм Брона-Кербоша

R := множество вершин максимальной клики.
P := множество возможных вершин максимальной клики.
X := множество исключенных вершин.
R ⋃ {v} := объединение R с единичным множеством (singleton) v.
P ⋂ N(v) := пересечение множества P с соседями v.
X ⋂ N(v) := пересечение множества X с соседями v.
P \ {v} := разность множества P и единичного множества v.
X ⋃ {v} := объединение множества X и единичного набора v.

BronKerbosch1(R, P, X):
Если и P, и X окажутся пустыми,
то R — максимальная клика

для каждой вершины v в P:
BronKerbosch1(R ⋃ {v}, P ⋂ N(v), X ⋂ N(v))
P := P \ {v}
X := X ⋃ {v}

Тема 7. Алгоритм Брона-Кербоша

Пример последовательности вызовов

BronKerbosch2(Ø, {1,2,3,4,5,6}, Ø)

BronKerbosch2({2}, {1,3,5}, Ø)

BronKerbosch2({2,3}, Ø, Ø): output {2, 3}

BronKerbosch2({2,5}, {1}, Ø)

BronKerbosch2({1,2,5}, Ø, Ø): output {1,2,5}

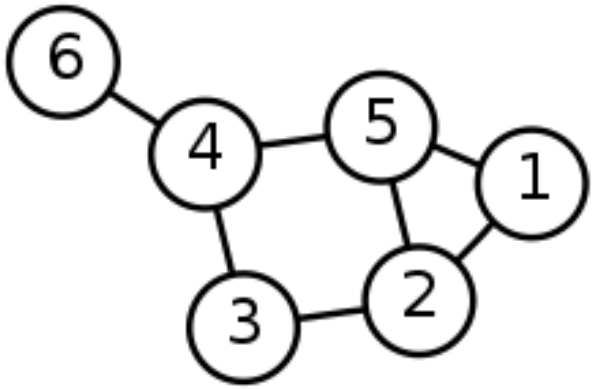
BronKerbosch2({4}, {3,5,6}, Ø)

BronKerbosch2({3,4}, Ø, Ø): output {3,4}

BronKerbosch2({4,5}, Ø, Ø): output {4,5}

BronKerbosch2({4,6}, Ø, Ø): output {4,6}

BronKerbosch2({6}, Ø, {4}): no output



Тема 7. Гамильтонов цикл

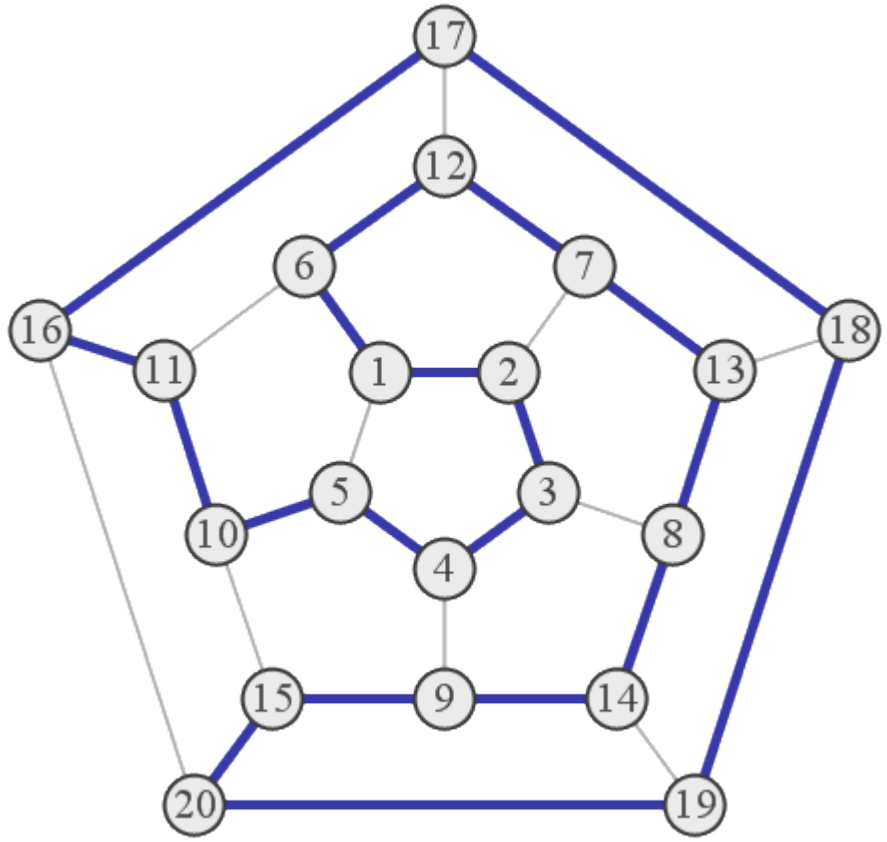
Гамильтонов цикл содержит все вершины графа один раз. Начинается и заканчивается на одной и той же вершине (аналогично можно рассматривать эйлеров цикл, граф и т.д.)

Гамильтонов путь содержит все вершины графа один раз, но не замыкается

Перебор всех последовательностей вершин =
числу перестановок 1…n

Перебор всех путей с помощью алгоритма в глубину
не уменьшает сложность на порядок

Сложность поиска цикла – O(n!)

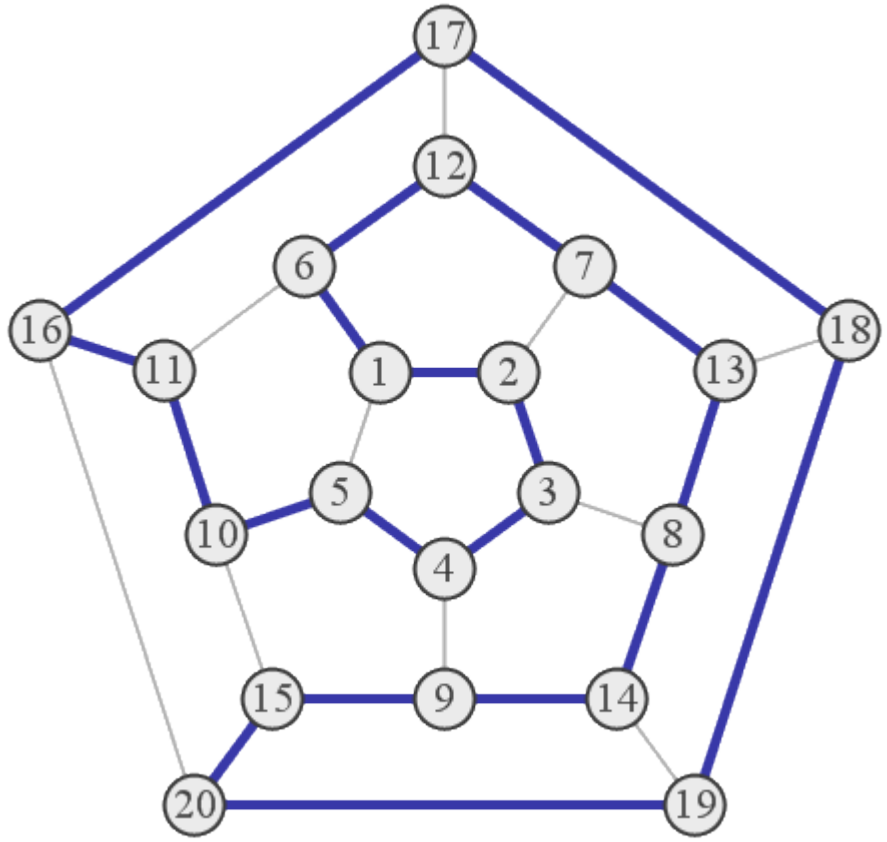


Тема 7. Гамильтонов цикл

Близкие задачи

Задача коммивояжёра: поиск цикла минимальной стоимости

Задача о 7 мостах Кёнигсберга



Тема 7. Гамильтонов цикл

Необходимого и достаточного условия существования цикла нет

Достаточные условия: Оре, Дирака, Бонди-Хватала, Поша

Условие Оре: пусть $V>2$ — количество вершин в данном графе. Если для любой пары несмежных вершин x, y выполнено неравенство

$\deg x + \deg y >= V$,

то данный граф — гамильтонов

(сумма степеней любых двух несмежных вершин

не меньше общего числа вершин в графе)

Тема 7. Гамильтонов цикл

Достаточные условия: условие Дирака

Условие Дирака: пусть V — число вершин в данном графе и $V > 3$.

Если степень каждой вершины не меньше, чем $V/2$, то данный граф — гамильтонов.

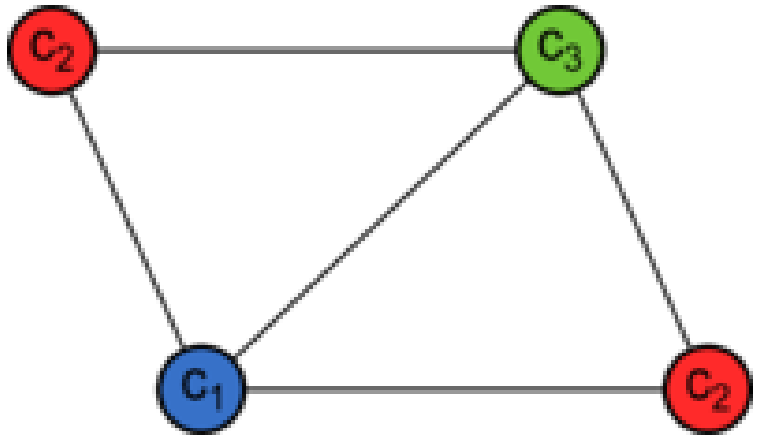
Для всех вершин x: $\deg x >= V/2$,

то данный граф — гамильтонов

Тема 7. Раскраска графа

Закрасить вершины графа так, чтобы ни одна пара смежных вершин не имела одинаковый цвет

Сколько красок надо?



Задача встречается при раскраске карт

(соседние области или государства должны

получить различные цвета)

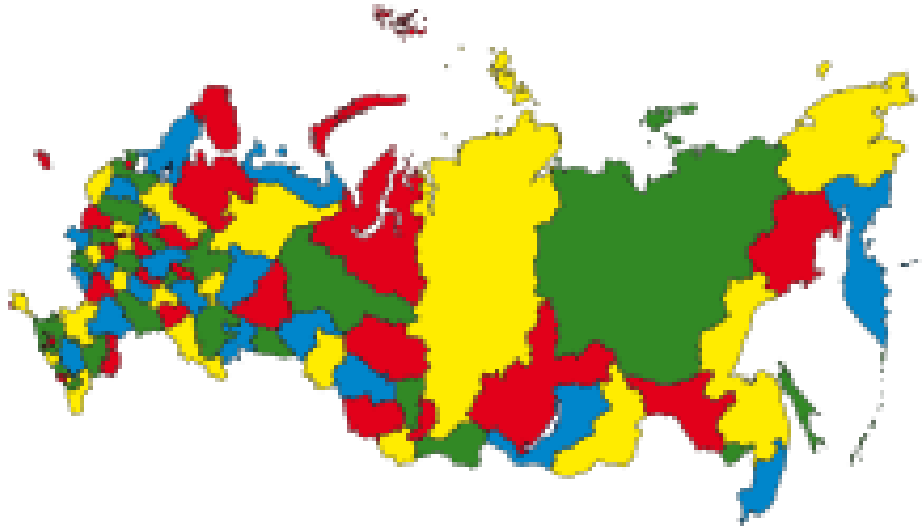
Определение хроматического числа (min числа красок) в общем случае не разрешима за полиномиальное время

Двудольный граф: две краски

Полный граф из V вершин: V красок

Тема 7. Раскраска графа

Теорема о 4 красках: всякую расположенную на плоскости (планарный граф) или на сфере карту можно раскрасить не более чем четырьмя разными цветами (красками) так, чтобы любые две области с общим участком границы были раскрашены в разные цвета. При этом области должны быть односвязными, а под общим участком границы понимается часть линии, то есть стыки нескольких областей в одной точке не считаются общей границей для них.



Тема 7. Раскраска графа

Теорема о 4 красках: всякую расположенную на плоскости или на сфере карту можно раскрасить не более чем четырьмя разными цветами (красками) так, чтобы любые две области с общим участком границы были раскрашены в разные цвета. При этом области должны быть односвязными, а под общим участком границы понимается часть линии, то есть стыки нескольких областей в одной точке не считаются общей границей для них.

Примечание: планарный граф можно изобразить на плоскости, причём так, что никакая пара рёбер не пересечётся

Существует определённый набор из 1936 карт, ни одна из которых не может содержать карту меньшего размера, которая опровергала бы теорему (примеры проверялись с помощью специальной компьютерной программы)

Не существует наименьшего контрпримера к теореме, потому что иначе он должен бы содержать какую-нибудь из этих 1936 карт, чего нет. Это противоречие говорит о том, что контрпримера нет вообще

Литература

Гамильтонов цикл

Кормен. Часть VI

https://foxford.ru/wiki/informatika/postroenie-gamiltonova-tsikla

Литература

Задача о раскраске графа

Кормен. Часть VI

https://habr.com/ru/post/346710/

Литература

Поиск максимального потока

Кормен. Часть VI

https://www.youtube.com/watch?v=u9NigdVHUr0

http://algolist.manual.ru/maths/graphs/maxflows/Ford_Fulkerson.php

http://algolist.manual.ru/maths/graphs/maxflows/

Литература

Поиск максимального паросочетания

Кормен. Часть VI

https://e-maxx.ru/algo/kuhn_matching

https://habr.com/ru/post/102367/

https://algorithmica.org/ru/matching

https://wiki.algocode.ru/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%9A%D1%83%D0%BD%D0%B0

Литература

Поиск клик в графе

Кормен. Часть VI

https://medium.com/nuances-of-programming/%D0%B3%D1%80%D0%B0%D1%84%D1%8B-%D0%B8-%D0%BF%D1%83%D1%82%D0%B8-%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC-%D0%B1%D1%80%D0%BE%D0%BD%D0%B0-%D0%BA%D0%B5%D1%80%D0%B1%D0%BE%D1%88%D0%B0-%D0%BC%D0%B0%D0%BA%D1%81%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B5-%D0%BA%D0%BB%D0%B8%D0%BA%D0%B8-857d7b7e829d

https://ru.wikipedia.org/wiki/Алгоритм_Брона_—_Кербоша

https://ru.qaz.wiki/wiki/Bron–Kerbosch_algorithm

Спасибо за внимание!

Петрусевич Денис Андреевич

Доцент Кафедры Проблем управления

petrusevich@mirea.ru