Минимальное остовное дерево (MST)

(*англ*. **m**inimum **s**panning **t**ree)

https://github.com/larandaA/alg-ds-snippets

Для связного взвешенного графа (G, w) **задача о минимальном остовном дереве** заключается в построении остова минимального веса.

Определение 1.

Граф H называется подграфом графа *G*, если выполняются включения $V(H) \subseteq V(G)$ и $E(H) \subseteq E(G)$ (если *H* - подграф графа *G*, то говорят, что *H* содержится в *G*).

Определение 2.

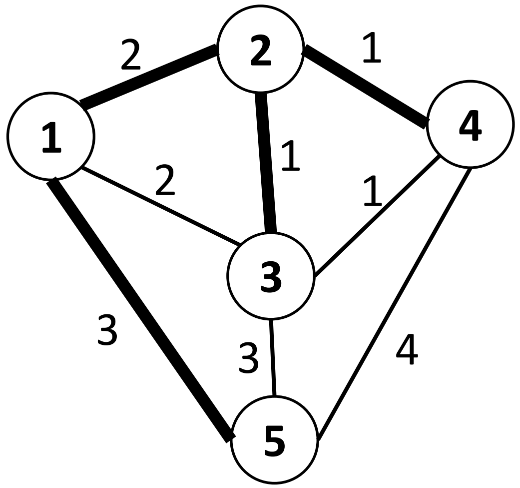
Если *H* содержится в *G* и $V(H) = V(G)$, то граф H называют остовным подграфом или фактором графа G.

Определение 3.

Пусть *H* - остовный подграф графа *G*. Если на каждой компоненте связности графа *G* множеством рёбер подграфа *H* порождается дерево, то *H* называют остовом или каркасом графа *G.*

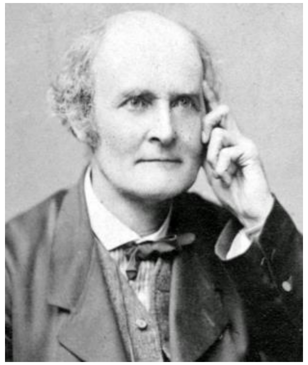
Определение 4.

Остовный подграф, который является деревом, называют остовным деревом.



$\boldsymbol{w(T) = 7}$

Число остовных деревьев полного графа



А́ртур Кэ́ли (*Arthur Cayley*) 1821-1895, английский математик в честь которого названа серия теорем, которые называются теоремами Кэли.

Теорема Кэли о числе деревьев (1889 г.)

На n вершинах, пронумерованных числами от 1 до n существует ровно $\boldsymbol{n^{n-2}}$ различных деревьев (т.е. число остовных деревьев в полном графе) .

Хайнц Прюфер, нем. Ernst Paul Heinz Prüfer, 1896 -1834, Германия

В 1918 году Прюфер использовал код для доказательства формулы Кэли о числе остовных деревьев.

Код Прюфера –

способ однозначного кодирования n-вершинного помеченного дерева упорядоченной последовательностью из $(n-2)$ номеров его вершин.

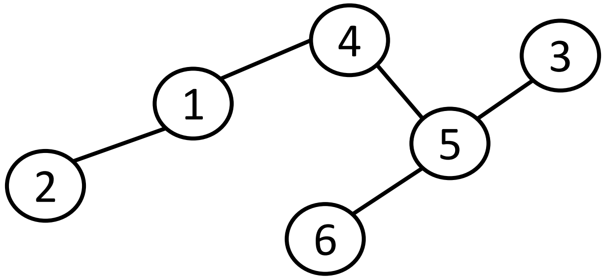
Код Прюфера –

способ однозначного кодирования $n-$вершинного помеченного дерева упорядоченной последовательностью из $(n-2)$ номеров его вершин.

Построение кода Прюфера

Пока вершин более 2-х:

1. Выбрать лист v с минимальным номером (*лист - вершина степени 1*).
2. Добавить в код Прюфера номер вершины, смежной с v.
3. Удалить из дерева вершину v и инцидентное ей ребро.



Код Прюфера: {1, 4, 5, 5}

1 4 5 5

Восстановление дерева по коду Прюфера

1. Код: {1, 4, 5, 5}
2. Вершины: V= {1, 2, 3, 4, 5}
3. Просматриваем код Прюфера слева направо.

- Пусть v – вершина в коде.
- Пусть w – вершина из множества V с минимальным номером, которой нет в коде.
- Добавляем ребро $\{v, w\}$ в дерево.
- Удаляем v из кода.
- Удаляем w из списка вершин V.

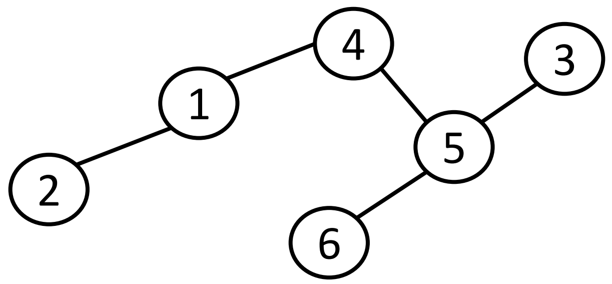
Оставшиеся в V две вершины соединяем ребром и добавляем в дерево.

Код Прюфера: $\overbrace{1 \quad 4 \quad 5 \quad 5}^{n-2}$

v: 1 4 5 5

w: 1 2 3 4 5 6

Восстановленное по коду Прюфера дерево



Теорема Кэли о числе деревьев (1889 г.)
На n вершинах, пронумерованных числами от 1 до n существует ровно $\boldsymbol{n^{n-2}}$ различных деревьев.

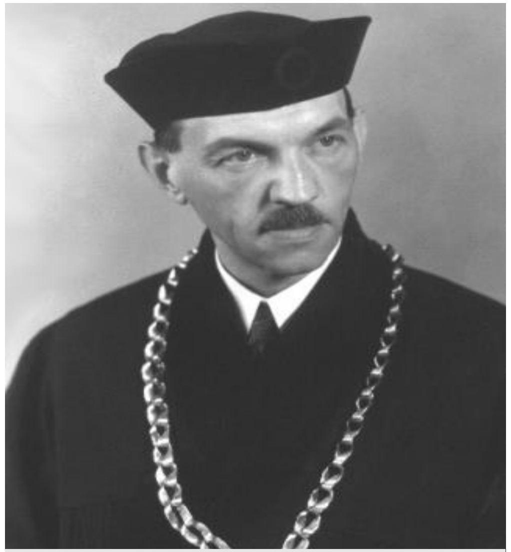
Алгоритм Крускала

1956 год

Джозеф Бернард Крускал-младший (англ. *Joseph Bernard Kruskal, Jr.*)
1928-2010
США
Научная сфера – математик, статистик, программист и психометрик
Доктор наук

Алгоритм **Прима** (или **Ярника**, или **DJP**)

1930 год

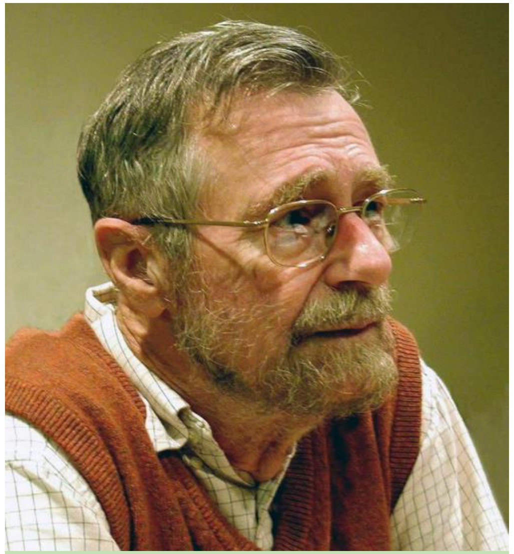


Войтек Ярник
Vojtěch Jarník
1897 - 1970
Чехословакия
Научная сфера – математическая физика, теория чисел, математический анализ.
Академик

1957 год

Роберт Клей Прим
Robert Clay Prim
1921 -
США
Научная сфера – математик, информатик
(фото 1971 г.)
Доктор философии по математике

1959 год



Э́дсгер Ви́бе Де́йкстра
Edsger Wybe Dijkstra
1930 – 2002
Нидерланды
Научная сфера – информатик
Награждён премией Тьюринга

Алгоритм Крускала (1956 г.)

Джозеф Бернард Крускал-младший (англ. *Joseph Bernard Kruskal, Jr.*)
1928-2010
США
Научная сфера – математик, статистик, программист и психометрик
Доктор наук

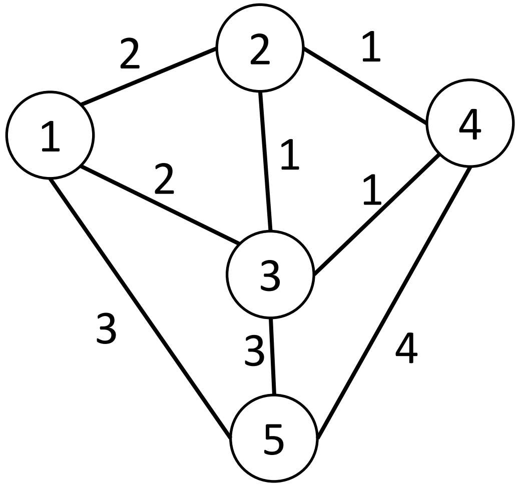
Алгоритм Крускала

1. Упорядочим все рёбра E графа G по неубыванию веса.
2. Минимальное остовное дерево графа G – пустое.
3. Просматриваем все рёбра графа G в порядке неубывания их веса:

    пусть $\{v, u\}$ – текущее ребро;

    - если ребро $\{v, u\}$ не образует цикла с ранее выбранными в $\boldsymbol{MST}$ рёбрами, то добавляем его к $\boldsymbol{MST}$;
    - если ребро $\{v, u\}$ образует цикл с ранее выбранными рёбрами – игнорируем его.

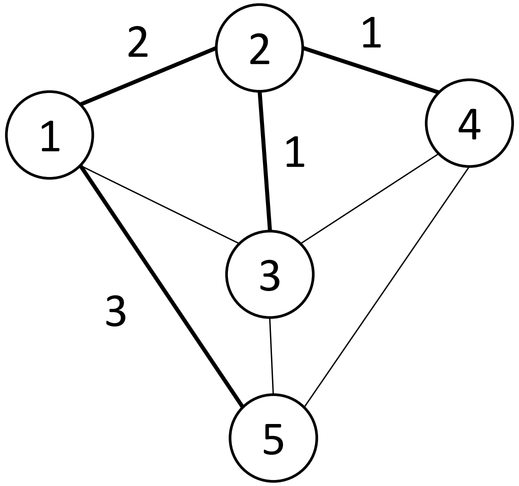
4. $\boldsymbol{MST}$ – минимальное остовное дерево графа G.



$$w_{\{2,3\}} = 1$$
$$w_{\{2,4\}} = 1$$
$$w_{\{3,4\}} = 1$$
$$w_{\{1,2\}} = 2$$
$$w_{\{1,3\}} = 2$$
$$w_{\{1,5\}} = 3$$
$$w_{\{3,5\}} = 3$$
$$w_{\{4,5\}} = 4$$

$\boldsymbol{MST}$

$w(\boldsymbol{MST}) = 7$



1. **Для просмотра рёбер по неубыванию веса**

   можно использовать следующие структуры данных:

   массив

   добавить в массив все рёбра, затем отсортировать их, используя, например, сортировку слиянием или быструю сортировку Хоара, которые работают за время $\mathbf{O}(\boldsymbol{m} \cdot \log \boldsymbol{m})$;

   бинарную кучу `min_heap`

   создать кучу из m рёбер – $\mathbf{O}(\boldsymbol{m})$;

   пока куча не станет пустой, удалять из кучи ребро с наименьшим весом;

   т.к. удаление минимального элемента из кучи выполняется за время $\mathrm{O}(\log m)$, то все удаления будут выполнены за время $\mathbf{O}(\boldsymbol{m} \cdot \log \boldsymbol{m})$.

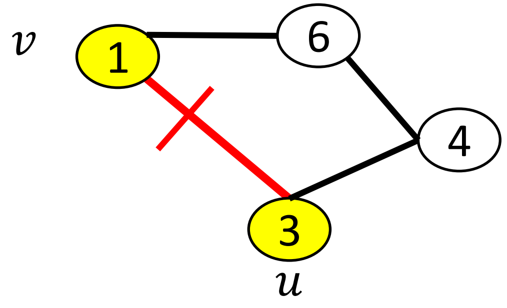
2. <u>Для определения наличия цикла с ранее выбранными в остов рёбрами</u>

можно использовать структуру данных **система непересекающихся множеств (DSU).**

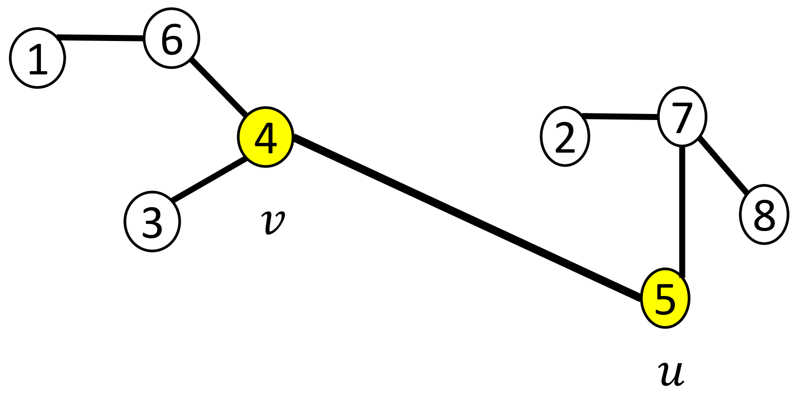
Одному множеству будут принадлежать вершины, которые в формируемом ***MST*** уже соединены некоторой цепью. Изначально имеем n одноэлементных множеств (время создания – $\mathbf{O}(\boldsymbol{n})$).

<u>Просматриваем все рёбра графа, для каждого ребра $\{\boldsymbol{v},\boldsymbol{u}\}$ проверяем:</u>

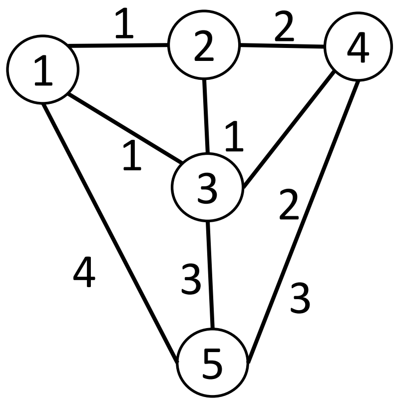
Если вершины $\boldsymbol{v}$ и $\boldsymbol{u}$ принадлежат одному множеству (проверка можно выполнить за время $\mathbf{O}(\log \boldsymbol{n})$), то они в *MST* соединены цепью и добавление ребра $\{v,u\}$ приведёт к циклу;



Если $\boldsymbol{v}$ и $\boldsymbol{u}$ принадлежат разным множествам, то добавляем ребро к *MST*, а множества, которым принадлежали вершины $\boldsymbol{v}$ и $\boldsymbol{u}$ объединяем в одно множество (поиск и объединение можно выполнить за время $\mathbf{O}(\log \boldsymbol{n})$).



ПРИМЕР

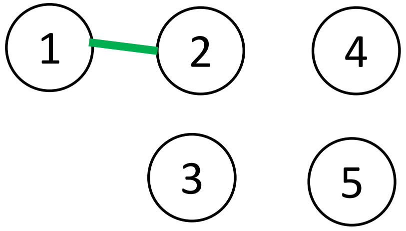


$w_{\{1,2\}}=1$
$w_{\{2,3\}}=1$
$w_{\{1,3\}}=1$
$w_{\{2,4\}}=2$
$w_{\{3,4\}}=2$
$w_{\{4,5\}}=3$
$w_{\{3,5\}}=3$
$w_{\{1,5\}}=4$

$w(\textbf{MST})=7$

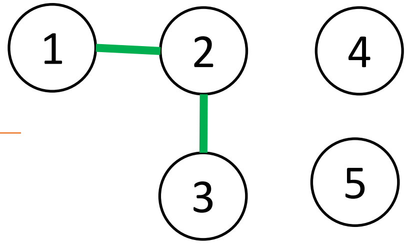
MST

$w_{\{1,2\}}=1$



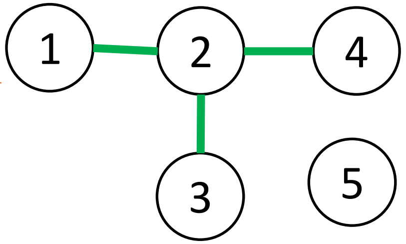
$w_{\{2,3\}}=1$

$\cancel{w_{\{1,3\}}=1}$



$w_{\{2,4\}}=2$

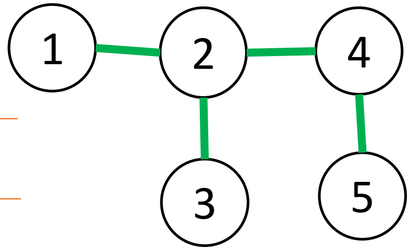
$\cancel{w_{\{3,4\}}=2}$



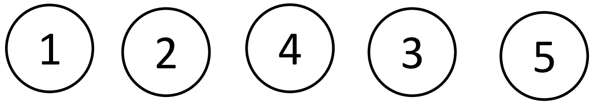
$w_{\{4,5\}}=3$

$\cancel{w_{\{3,5\}}=3}$

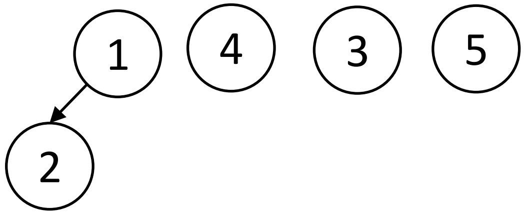
$\cancel{w_{\{1,5\}}=4}$



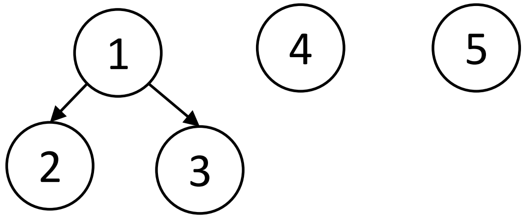
DSU



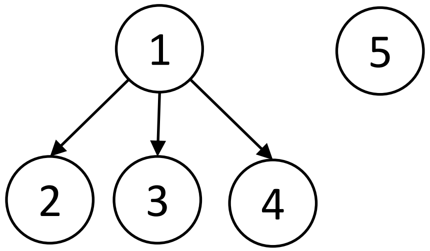
1	2	3	4	5
-1	-1	-1	-1	-1



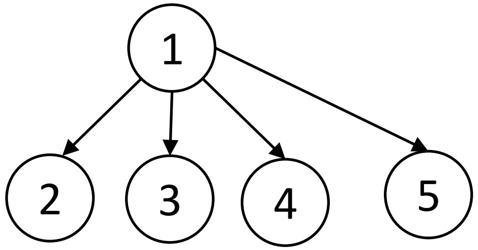
1	2	3	4	5
-2	1	-1	-1	-1



1	2	3	4	5
-3	1	1	-1	-1



1	2	3	4	5
-4	1	1	1	-1



1	2	3	4	5
-5	1	1	1	1

```python
mst = []
components = dsu(n)

def finished():
    return len(mst) == n - 1

def add_edge(edge):
    v, _, u = edge
    components.union(v, u)
    mst.append(edge)

def adds_cycle(edge):
    v, _, u = edge
    cv = components.find(v)
    cu = components.find(u)
    return cv == cu

def cost(edge):
    return edge[1]

def build_mst():
    edges = sorted(edges, key=cost)

    next_edge = 0
    while not finished() and next_edge < m:
        edge = edges[next_edge]
        if not adds_cycle(edge):
            add_edge(edge)
        next_edge += 1
```

$O(m \cdot \log m)$ обработка рёбер графа по неубыванию веса

$+$

$O(m \cdot \log n)$ проверка наличия цикла при попытке добавить очередное ребро к построенной части MST; использование DSU с эвристикой объединения по размеру

Обоснование корректности алгоритма Крускала

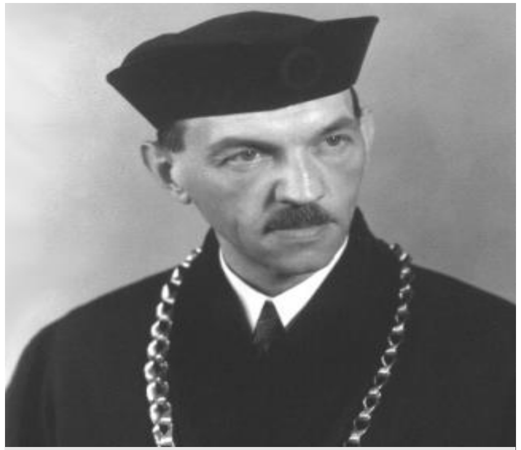
Почему этот жадный алгоритм дает верное решение? Рассмотрим случай с уникальными весами ребер, так как в этом случае минимальное остовное дерево единственно. Докажем от противного. Пусть алгоритм построил дерево `T`, в то время как оптимальным деревом является `T*`, и `T != T*`. Тогда рассмотрим ребра графа в отсортированном по весу порядке, проверяя их принадлежность `T` и `T*`. Возьмем первое встреченное ребро `e`, по которому решения разошлись (то есть по всем ребрам `e'`, `c(e') < c(e)`, решения совпали). У нас имеется два случая:

- `e` принадлежит `T*` и не принадлежит `T`. Так как мы рассматриваем ребра в порядке возрастания весов, то `e` не может принадлежать `T` только в случае, если оно образует цикл с ребрами, добавленными ранее. Но так как все ребра с весом `< c(e)` у `T` совпадают с `T*`, то и в `T*` ребро `e` тоже должно образовать цикл, что невозожно. Получаем противоречие.
- `e` принадлежит `T` и не принадлежит `T*`. Рассмотрим вершины `v` и `u`, инцидентные `e`. В дереве `T*` эти вершины также должны быть соединены некоторой цепью, поэтому рассмотрим ребра этой цепи. У нас опять два случая:
    - Веса всех ребер этой цепи меньше `c(e)`. В таком случае, как и выше, все эти ребра принадлежали бы и `T`, и добавление ребра `e` создало бы цикл, что по построению невозможно. Получили противоречие.
    - В цепи есть хотя бы одно ребро `e'`, что `c(e') > c(e)`. В таком случае в дереве `T*` мы можем заменить ребро `e'` на `e` и и получить новое дерево меньшего веса, чем `T*`. Получаем противоречие с утверждением, что `T*` -- оптимальное решение.

Доказательство можно расширить и для случая с неуникальными весами ребер.

Алгоритм Прима (или Ярника, или DJP)

1930 год



Войтек Ярник

Vojtěch Jarník

1897 - 1970

Чехословакия

Научная сфера – математическая физика, теория чисел, математический анализ.
Академик

1957 год

Роберт Клей Прим

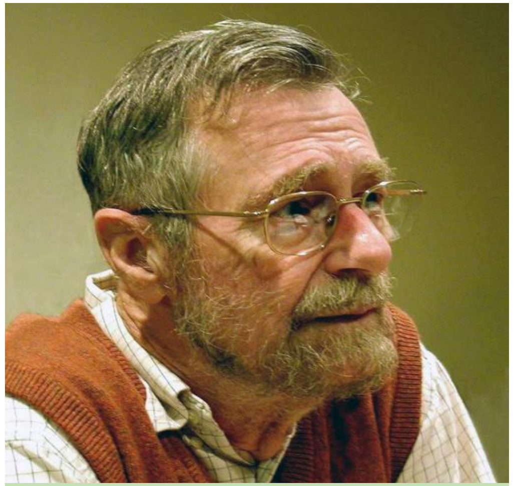
Robert Clay Prim

1921 -

США

Научная сфера – математик, информатик
(фото 1971 г.)
Доктор философии по математике

1959 год



Э́дсгер Ви́бе Де́йкстра
Edsger Wybe

Dijkstra

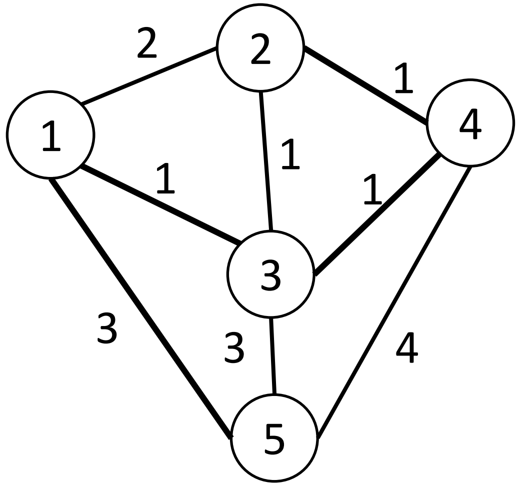
1930 – 2002

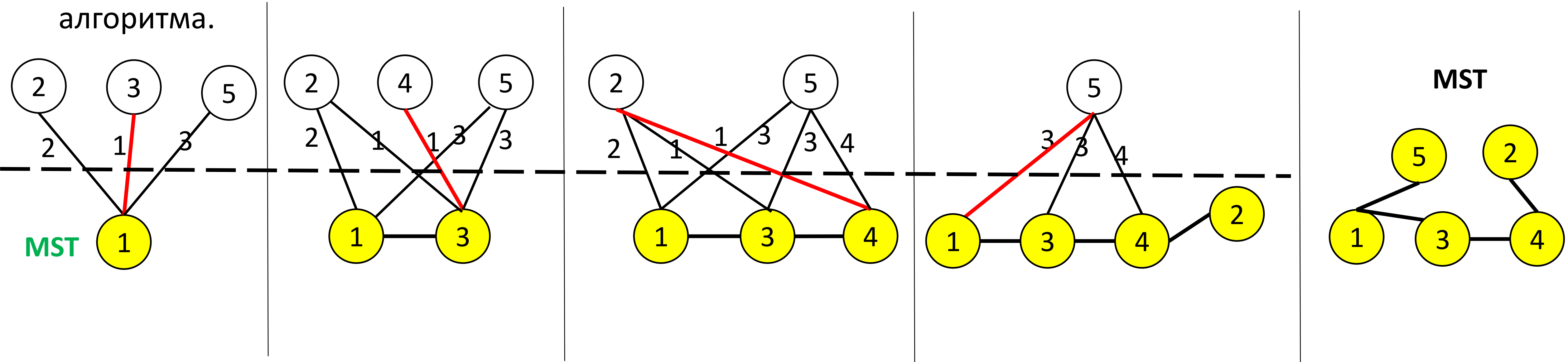
Нидерланды

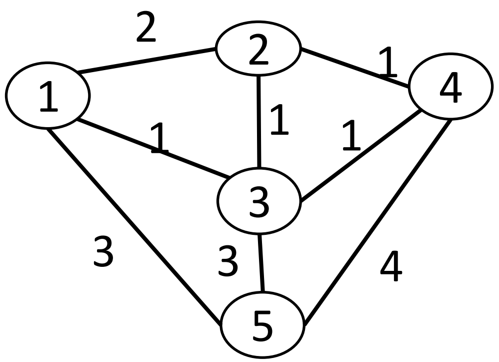
Научная сфера – информатик
Награждён премией Тьюринга

Алгоритм Прима (Ярника, DJP)

1. Выбираем произвольную вершину $\boldsymbol{v}$ графа G.
2. Добавляем вершину $\boldsymbol{v}$ в минимальное остовное дерево *MST*.
3. Рассмотрим все рёбра графа G у которых ровно один конец которых лежит в *MST* и выберем из них то, вес которого наименьший. Предположим, что выбрано ребро $\{v, u\}$.
4. Добавляем вершину u вместе с ребром $\{v, u\}$ в *MST.*
5. Если все вершины графа G перенесены в остов, то алгоритм завершает свою работу, а MST – минимальное остовное дерево графа G.
6. Если не все вершины графа G перенесены в остов, то возвращаемся к шагу 3 алгоритма.







1. Реализация DJP на структуре данных **массив**

$\boldsymbol{O(n^2 + m)}$

$\boldsymbol{n}$ – число итераций;

$\boldsymbol{O(n \cdot n)}$ – поиск всех минимумов в массиве;

$\boldsymbol{O(m)}$ – пересчёты элементов массива, содержащего кратчайшее ребро, соединяющего вершину с построенной частью MST;

в остове?

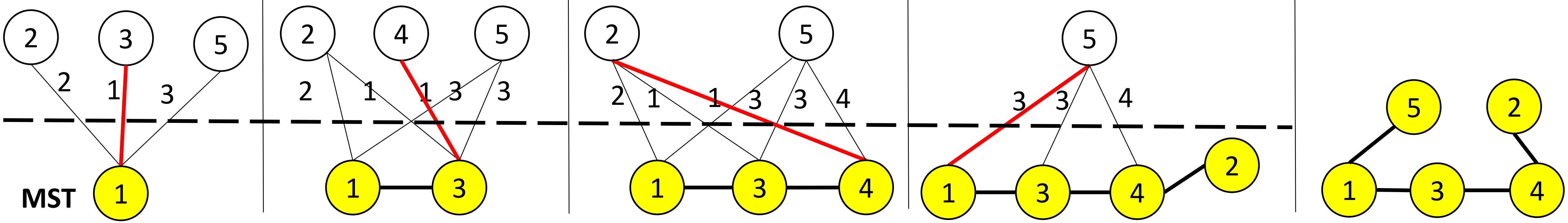
1	2	3	4	5
+	-	-	-	-
-	2	1	-	3

1	2	3	4	5
+	-	+	-	-
-	1	-	1	3

1	2	3	4	5
+	-	+	+	-
-	1	-	-	3

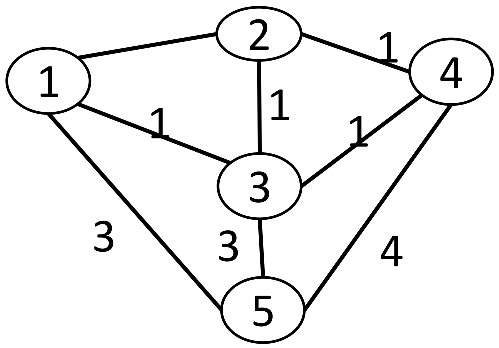
1	2	3	4	5
+	+	+	+	-
-	-	-	-	3

1	2	3	4	5
+	+	+	+	+
-	-	-	-	-



2. Реализация DJP

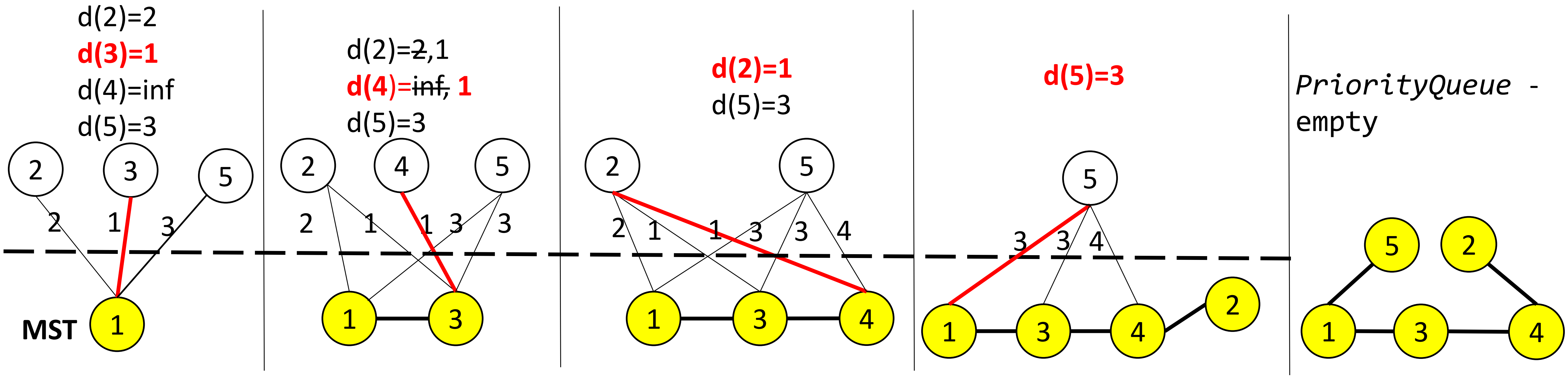
на структуре данных **бинарная куча + модификация ключа**



- построим бинарую кучу (min_heap) из n вершин:

  каждая вершина $\boldsymbol{v}$ хранится с меткой – вес наименьшего ребра, которое соединяет вершину $\boldsymbol{v}$ с построенной частью ***MST***;
- для каждой вершины поддерживаем её индекс в массиве, на котором реализована бинарная куча;
- пока куча не станет пустой, удаляем из кучи вершину $\boldsymbol{w}$ с минимальной меткой, добавляем соответствующее ребро в остов и при необходимости уменьшаем в куче метки вершин, которые смежны с $\boldsymbol{w}$ и ещё не добавлены в остов.

min_heap



Реализация DJP
на структуре данных **куча + модификация ключа**

- создание кучи из n вершин (в куче каждая вершина $\boldsymbol{v}$ хранится с меткой – вес наименьшего ребра, которое соединяет вершину $\boldsymbol{v}$ с построенной частью MST):

  $\mathbf{O}(\boldsymbol{n})$

- пересчёты меток вершин (уменьшение ключа):

  $\mathbf{O}(\boldsymbol{m} \cdot \log \boldsymbol{n})$ – БИНАРНАЯ КУЧА

  $\mathbf{O}(\boldsymbol{m})$ – КУЧА Фибоначчи (усреднённо)

  !!! необходимо поддерживать для каждой вершины её индекс в массиве, на котором реализована бинарная куча;

- удаление вершин из кучи:

  $\mathbf{O}(\boldsymbol{n} \cdot \log \boldsymbol{n})$

$\mathbf{O}(\boldsymbol{m} \cdot \log \boldsymbol{n}) + \mathbf{O}(\boldsymbol{n} \cdot \log \boldsymbol{n})$ – бинарная куча
$\mathbf{O}(\boldsymbol{m}) + \mathbf{O}(\boldsymbol{n} \cdot \log \boldsymbol{n})$ –куча Фибоначчи (усреднённо)

3. Реализация DJP

(приоритетная очередь без модификации ключа)

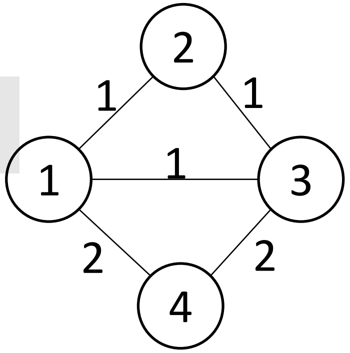
$$O(m \cdot \log m)$$

```
mst = []

def build_mst(start):
    q = PriorityQueue()

    in_mst[start] = True
    for u, cu in g[start]:
        q.enqueue((cu, start, u))

    while not q.empty():
        cu, v, u = q.dequeue()
        if in_mst[u]:
            continue

        in_mst[u] = True
        mst.append((v, cu, u))
        for w, cw in g[u]:
            q.enqueue((cw, u, w))
```

уже в *MST*

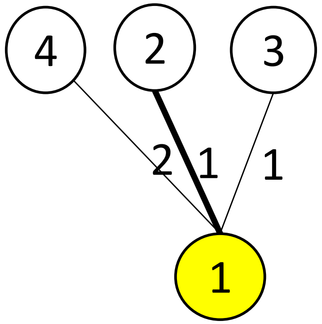
пока ещё не в *MST*

можно добавить проверку, что w не находится в *MST* и только тогда добавлять



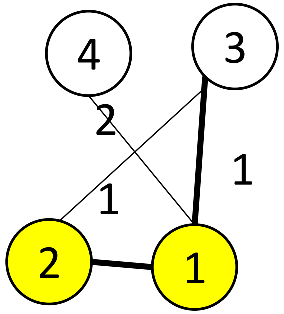
PriorityQueue

$c_{1,2}=1$
$c_{1,3}=1$
$c_{1,4}=1$



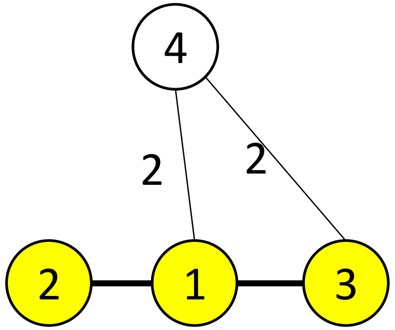
$c_{1,3}=1$
$c_{1,4}=2$

$c_{2,3}=1$
$c_{2,1}=1$



$c_{1,4}=2$
$c_{2,3}=1$
$c_{2,1}=1$

$c_{3,1}=1$
$c_{3,2}=1$
$c_{3,4}=2$



Алгоритм	*Оценка*	*Структура данных*
Крускала	$O(m \cdot \log m) + O(m \cdot \log n)$	массив/бинарная куча, DSU (объединение по размеру)
	$O(m \cdot \log m) + O(m \cdot \alpha(n))$, α(n) - обратная функция Аккермана	массив/бинарная куча, DSU (объединение по размеру, сжатие пути)
DJP	$O(n^2 + m)$	массив
	$O(m \cdot \log n) + O(n \cdot \log n)$	бинарная куча с модификацией ключа
	$O(m) + O(n \cdot \log n)$	куча Фибоначчи с модификацией ключа (усреднённая оценка)
	$O(m \cdot \log m)$	бинарная куча без модификации ключа

Жадный алгоритм оптимально решает задачу о минимальном остовном дереве.

Какие задачи можно решить оптимально жадным алгоритмом?

Матроиды

Системой подмножеств $S = (E, J)$ называется конечное множество E вместе с семейством J подмножеств множества E, <u>которое замкнуто относительно включения</u> (т. е. если $I \in J$ и $I' \subseteq I$, то $I' \in J$).

Элементы семейства J называются ***независимыми***.

Пусть задано конечное множество

$$\boldsymbol{E = \{1, 2, 3, 4\}}$$

и семейства подмножеств этого множества:

$$\boldsymbol{J_1 = \{(2, 3), 2, 3\}}$$

$$\boldsymbol{J_2 = \{(2, 3), (1, 2), 3\}}.$$

Семейство J_1 является замкнутым семейством подмножеств и $S = (E, J_1)$ – система подмножеств.

Семейство J_2 не является замкнутым.

Пусть для каждого элемента $e \in E$ задан вес $w(e) \geq 0$.

Комбинаторная задача оптимизации для системы подмножеств $(\boldsymbol{E}, \boldsymbol{J})$ состоит в нахождении независимого подмножества, которое имеет наибольший общий вес.

> Возникает вопрос: как задавать систему подмножеств? Можно предложить выписывать все независимые подмножества из E, используя подходящее представление. Однако такой способ представления может оказаться очень неэффективным, так как семейство подмножеств J может содержать до $2^{|E|}$ подмножеств. Поэтому все рассматриваемые системы подмножеств будем представлять с помощью алгоритма, который по данному подмножеству I множества E решает, верно ли, что $I \in J$ (т.е. является независимым).

Общая схема «жадного» алгоритма

$1. I = \emptyset$.

2. Пока $E \neq \{\emptyset\}$ выполнять следующую последовательность шагов:

а) пусть $e \in E$ и имеет наибольший вес $w(e)$;

б) удалить e из E, т. е. $E = E \backslash e$;

в) если $I \cup e \in J$, то $I = I \cup e$.

Некоторые комбинаторные задачи для систем подмножеств могут быть решены корректно (точно) «жадным» алгоритмом, а некоторые нет (получается некоторое приближенное решение).

Пусть $\boldsymbol{M} = (\boldsymbol{E}, \boldsymbol{J})$ – система подмножеств.

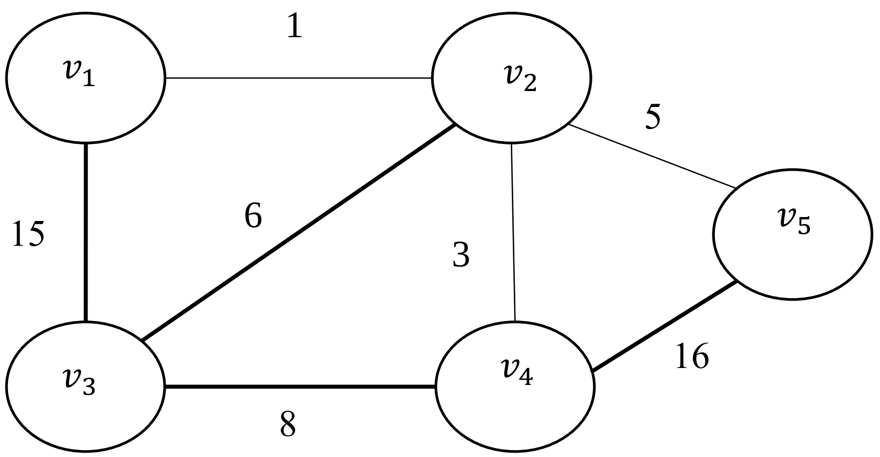
Будем называть $\boldsymbol{M}$ **матроидом**, если «жадный» алгоритм корректно решает любую индивидуальную комбинаторную задачу для системы подмножеств M.

Задача

Задан граф $G = (V, E)$.

Каждому ребру графа поставлен в соответствие некоторый вес $w(e) \geq 0$.

Требуется найти лес – **ациклический подграф графа $\boldsymbol{G}$, имеющий максимальный общий вес**.



Решение

Предположим, что граф является связным.
Так как веса положительны, то любой оптимальный лес можно сделать максимальным по включению, т. е. остовным деревом графа G.
Кроме того, все остовные деревья графа G имеют $|V| - 1$ ребер.
Поэтому можно построить функцию расстояний для E, полагая

$$\boldsymbol{d_e = W - w(e), \forall e \in E,}$$

где

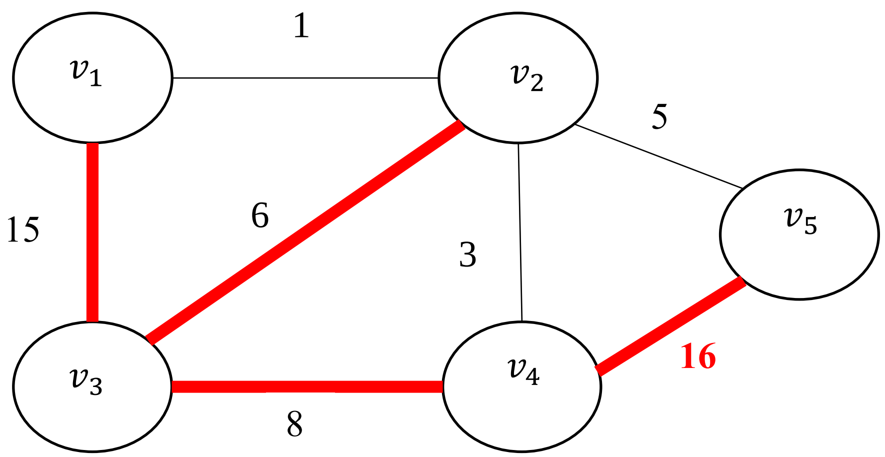
$$\boldsymbol{W = \max_{e \in E} w(e).}$$

При этом сумма расстояний $d(T)$ любого остовного дерева T будет связана с общим весом $w(T)$ дерева T следующим соотношением:

$$\boldsymbol{w(T) = (|V| - 1) \cdot W - d(T).}$$

Отсюда сразу следует, что минимальное остовное дерево в графе G с расстояниями d совпадает с лесом максимального веса в графе G с весами w.

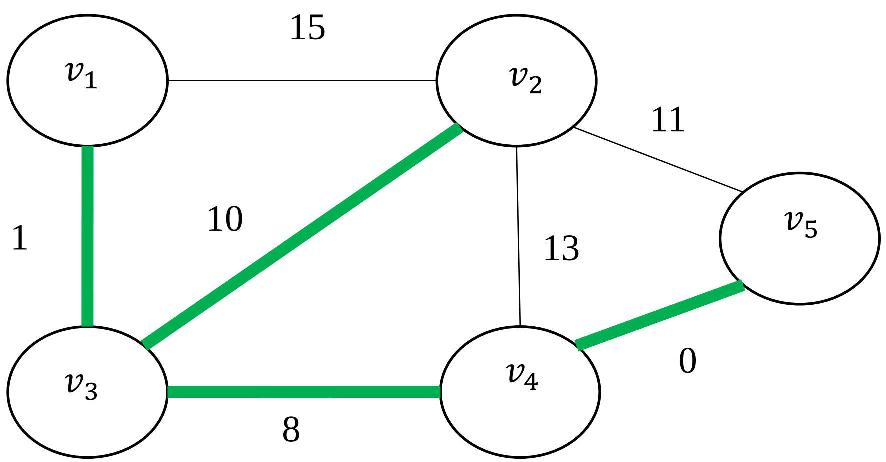
Максимальный ациклический подграф



Функция весов w

$w(T)$=45

Минимальное остовное дерево



Функция весов d

$d(T)$=19

$$W = \max_{e \in E} w(e) = 16$$

$$d_e = W - w(e), \forall e \in E$$

$$w(T) = (|V| - 1) \cdot W - d(T)$$

«Жадный» алгоритм корректно решает задачу о лесе максимального веса.

Рассмотренная в задаче система подмножеств является **матроидом**, который называют ***графическим***.

Задача о минимальном остовном дереве также решалась «жадным» алгоритмом.

Однако эта задача не может быть сформулирована в терминах системы подмножеств, так как если обозначить через J семейство остовных деревьев некоторого графа, то это семейство не будет являться замкнутым относительно включения (любое подмножество ребер остовного дерева уже не будет являться остовным деревом исходного графа).

Задача о лесе максимального веса по существу, может рассматриваться как задача о минимальном остовном дереве, но в то же время может быть сформулирована как комбинаторная задача оптимизации для системы подмножеств.

Спасибо за внимание