ЛЕКЦИЯ 13. ДЕРЕВЬЯ

ЧТО ТАКОЕ ДЕРЕВО

Определение 1. *Дерево* – это неориентированный связный граф без циклов.

Определение 2. *Лес* – неориентированный граф без циклов (т.е. объединение непересекающихся деревьев).

Определение 3. *Лист* дерева или висячая вершина – любая вершина степени 1.

Теорема 1. Дерево из p вершин содержит q=p-1 рёбер.

Доказательство. Докажем, что в любом дереве есть хотя бы один лист. Возьмём произвольную вершину и начнём из неё обход в глубину. Как только зайдём в тупик – это и есть лист (циклов-то нет!). Итак, в любом дереве есть хотя бы один лист. Уберём его вместе с инцидентным ребром – связность сохранится. Значения p и q уменьшатся на 1. Будем продолжать этот процесс до тех пор, пока не останется ни одного ребра. При этом полученный граф связный $\Rightarrow$ p=1.

Теорема 2. Для неориентированного графа G следующие условия эквиваленты:

1. G – связный граф без циклов (т.е. дерево).
2. G не имеет циклов и q=p-1.
3. G связен и q=p-1.
4. G связен и любое его ребро является мостом.
5. Для любых двух вершин графа G существует единственная цепь, которая их связывает.
6. В графе G нет циклов, но добавление любого нового ребра приводит к появлению цикла.

Доказательство.

1 $\Rightarrow$ 5. Пусть есть две цепи, соединяющие u и v. Тогда существуют такие w1 и w2, что участок от u до w1 у этих цепей общий, а от w1 до w2 все их рёбра различны. Но тогда w1→w2→w1 это цикл.

5 $\Rightarrow$ 1. Пусть есть цикл. Но тогда «половинки» цикла - различные цепи.

Остальное – в виде упражнений.

КОДИРОВАНИЕ ДЕРЕВЬЕВ

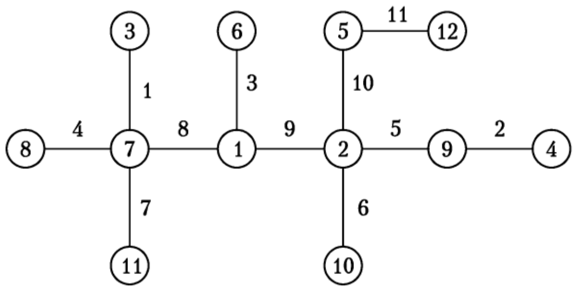
Описать структуру дерева можно гораздо компактнее, чем структуру произвольного графа.

КОД ПРЮФЕРА

Кодирование. Код Прюфера для дерева из p вершин представляет собой размещение с повторениями из p по (p-1), которое строится по следующем уалгоритму:

```
while (количество вершин > 2) do
    выбираем лист v с минимальным номером;
    в код Прюфера добавляем номер вершины, смежной с v (v на ней «висит»);
    лист v и инцидентное ей ребро удаляем из графа;
end do;
```

Пример: 7,9,1,7,2,2,7,1,2,5 – код Прюфера этого дерева (см. ниже)



Декодирование. Заметим, что все листы исходного дерева (и только они) в коде Прюфера отсутствуют. Каждая вершина встречается в коде ровно (deg(v)-1) раз.

1. Находим вершину с наименьшим номером y1, которой нет в КОДЕ. Она была стёрта первой $\Rightarrow$ она соединена с x1. Рисуем (y1,x1).
   - Вычёркиваем x1 из КОДА.
   - Выписываем y1 в СПИСОК восстановленных вершин.
2. Находим вершину с наименьшим номером y2, которой нет в КОДЕ и в СПИСКЕ. Она была стёрта первой из оставшихся $\Rightarrow$ она соединена с x2. Рисуем (y2,x2).
   - Вычёркиваем x2 из КОДА.
   - Выписываем y2 в СПИСОК восстановленных вершин.

и т.д. пока КОД не закончится. К этому моменту будет восстановлено (p-2)=(q-1) рёбер, т.е. не хватает одного ребра. В СПИСКЕ не хватает двух вершин – именно они и дают последнее ребро.

Примеры:

1. ТОТ ЖЕ, что в кодировании
2. 1,1,1,1,1
3. 1,2,3,4,5

Теорема (Кэли)**.** Количество *помеченных* деревьев с p вершинами равно p^{p-2}.

Доказательство – из кода Прюфера.

МИНИМАЛЬНЫЙ КАРКАС

Определение 1**.** *Каркас* или *остов* графа – это подграф, который содержит все вершины исходного графа и является деревом.

У связного графа может быть много разных каркасов. Можно, например, построить каркас обходом в глубину, можно обходом в ширину. Особый интерес представляют каркасы взвешенных графов.

Определение 2**.** *Минимальным* каркасом взвешенного графа называется каркас с наименьшим суммарным весом.

Задача: найти минимальный каркас взвешенного графа. Она интересна не только из-за своей практической значимости, но и тем, что это одна из немногих содержательных задач, которая решается *жадным алгоритмом*.

АЛГОРИТМ КРАСКАЛА (1953)

ОПИСАНИЕ АЛГОРИТМА

Обозначим текущее множество рёбер каркаса через X.

Инициализация. Упорядочим все рёбра по возрастанию весов.

$X := \varnothing$

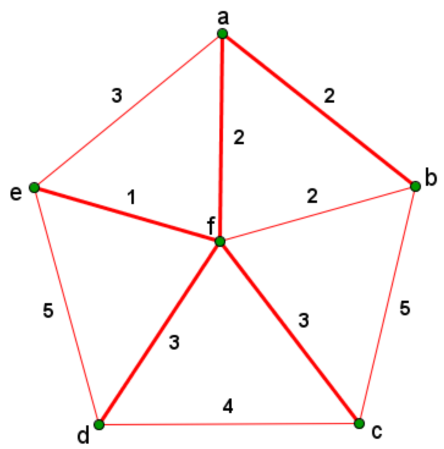
Шаг алгоритма.

Выберем очередное по весу ребро. Если при его добавлении образуется цикл – пропустим его, если нет цикла – добавим это ребро к X (*цикл можно искать, например, поиском в глубину*).

Как только мощность |X| станет равна (p-1), каркас будет получен.

Замечание: по ходу алгоритма X остаётся *лесом*.

ПРИМЕР.



Ребро		Вес	X
4	5	1	+
0	1	2	+
0	5	2	+
1	5	2	-
0	4	3	-
2	5	3	+
3	5	3	+
2	3	4	
1	2	5	
3	4	5	
		Σ=11	

ДОКАЗАТЕЛЬСТВО

Упорядочим веса рёбер найденного каркаса X по возрастанию: $x_1 \le \ldots \le x_{p-1}$. Пусть есть более дешёвый каркас $y_1 \le \ldots \le y_{p-1}$. Тогда до какого-то i ребра X и Y совпадают, а i-ые рёбра различны (может быть, i=1).

Но тогда $x_i < y_i$, поскольку оба ребра НЕ ОБРАЗУЮТ ЦИКЛА с предыдущими рёбрами, а x_i - самое дешёвое из таких рёбер. ДОБАВИМ ребро x_i в каркас Y. Получим цикл. В этот цикл обязательно входит какое-то из рёбер ПОСЛЕ i-го (ведь x_i не образует цикла с первым (i-1) ребром, которые общие у этих каркасов) . Выбросим одно из таких рёбер. Получим каркас, вес которого меньше Y – противоречие.

ПРОГРАММА

```
# Чтение данных
f = open("graph.txt")
p, q = map(int, f.readline().split())
edges = []
for i in range(q):
    start, end, weight = map(int, f.readline().split())
    edges.append([weight, start, end])
f.close()
# Сортировка
edges.sort()
```

```
# Сначала каждая вершина в своей компоненте связности
comp = [i for i in range(p)]
ans = 0
# Выбираем рёбра так, чтобы концы лежали в разных компонентах
for weight, start, end in edges:
    if comp[start] != comp[end]:
        print(start,end)
        ans += weight
        a = comp[start]
        b = comp[end]
        for i in range(p):
            if comp[i] == b:
                comp[i] = a
print(ans)
```

СЛОЖНОСТЬ

Сложность этой реализации = $O(pq) = O(p^3)$.

Если искать цикл поиском в глубину, а лес хранить как *систему непересекающихся множеств*, то можно понизить сложность до $O(q \log q) = O(p^2 \log p)$.

АЛГОРИТМ ПРИМА (1957)

ОПИСАНИЕ АЛГОРИТМА

Обозначим текущее множество рёбер каркаса через X.

Обозначим текущее множество вершин, включенных в каркас, через T.

Инициализация.

$X := \varnothing;$

$T := \{1\}$

(вместо 1 можно выбрать любую вершину)

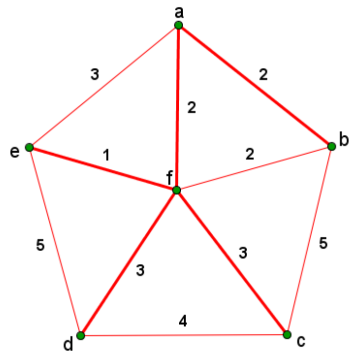
Шаг алгоритма.

Выберем минимальное ребро из тех, один конец которых лежит в T, а другой – не лежит в T (для этого придётся пройти с начала по всем невыбранным рёбрам), и добавим его к X, а второй его конец – к T.

После повторения этого шага (p-1) раз каркас будет получен.

Замечание: по ходу алгоритма X остаётся *деревом*.

ПРИМЕР.



Ребро		Вес	T
			a
a	b	2	b
a	f	2	f
f	e	1	e
f	c	3	c
f	d	3	d
		Σ=11	

ПРОГРАММА

```
# Чтение данных
… см. алгоритм Краскала …
# Помечаем начальную вершину каркаса
mark = [False]*p
mark[0] = True
# Добавляем минимальное из рёбер с разнопомеченными концами
from math import inf
ans = 0
for i in range(1,p):
    m = inf
    s = None
    e = None
    for weight, start, end in edges:
        if (mark[start] != mark[end]) and (weight<m):
            m = weight
            s = start
            e = end
    # Помечаем вершину и добавляем ребро в каркас
    print(s,e)
    ans += m
    mark[s] = True
    mark[e] = True
print(ans)
```

СЛОЖНОСТЬ

Сложность этой реализации = $O(pq) = O(p^3)$.

Если хранить рёбра как *очередь с приоритетом*, то можно понизить сложность до $O(p^2)$.