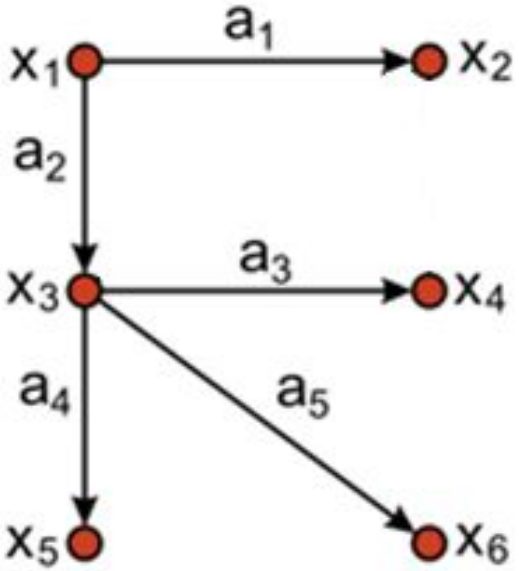
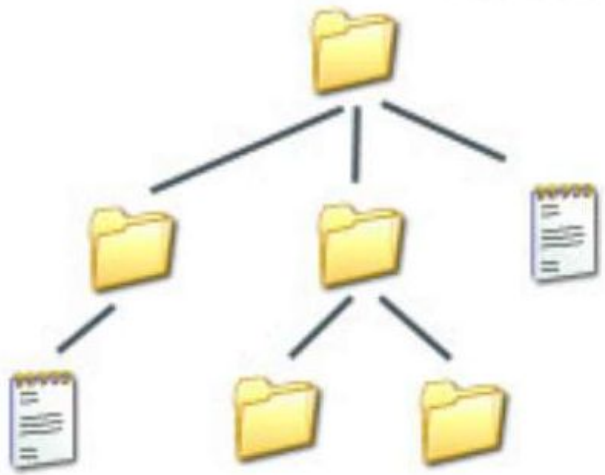
Остовные деревья.

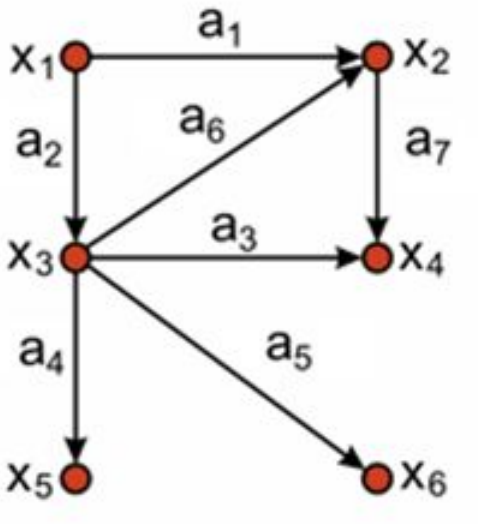
Деревом называется связный ациклический граф.

Т.е. в таком графе есть путь между любой парой вершин, причём этот путь – единственный.





Остовное дерево графа - ациклический связный подграф в который входят все его вершины.



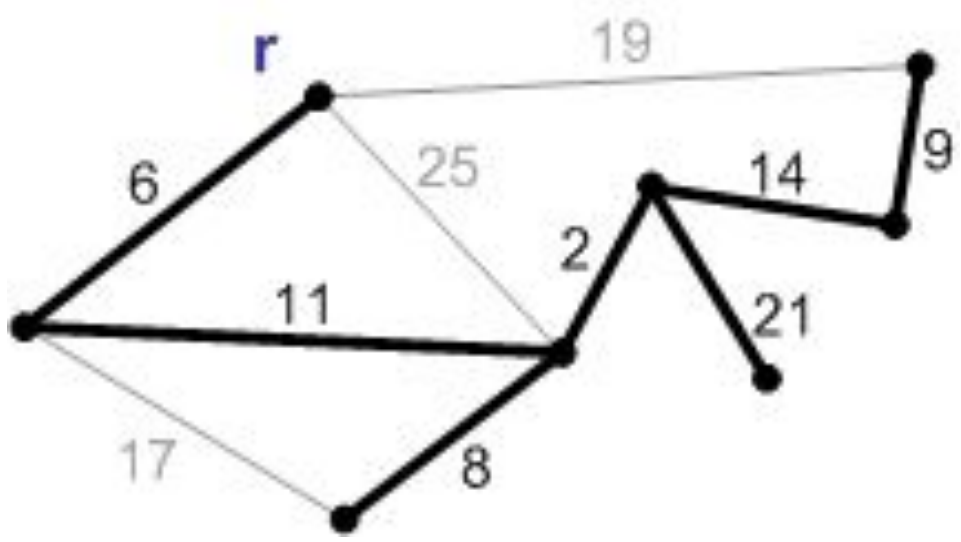
Матричная теорема о деревьях (теорема Кирхгофа). Алгоритм Прима.

Для связного помеченного графа *G* с матрицей Кирхгофа *M* количество остовных деревьев равно алгебраическому дополнению матрицы Кирхгофа *M*.

Для взвешенного графа можно поставить задачу нахождения остовного дерева минимальной длины (веса). Пример – граф дорог между пунктами, где вес ребра – стоимость построения дороги.

Алгоритм Прима строит минимальное остовное дерево, добавляя на каждом шаге к строящемуся остову *безопасное ребро* минимальной длинны.

Ребро называется безопасным, если при добавлении его к строящемуся остову не нарушается свойство ацикличности.



Алгоритм Прима.

Алгоритм поиска минимального остовного дерева

Вход: G=(V, A) - неориентированный граф, представленный списками смежностей: для каждой вершины v список **Lv** содержит перечень всех смежных с v вершин, r – номер корневой вершины .

Выход: Cw – список с номерами вершин минимального остовного дерева.

Алгоритм ПМД:

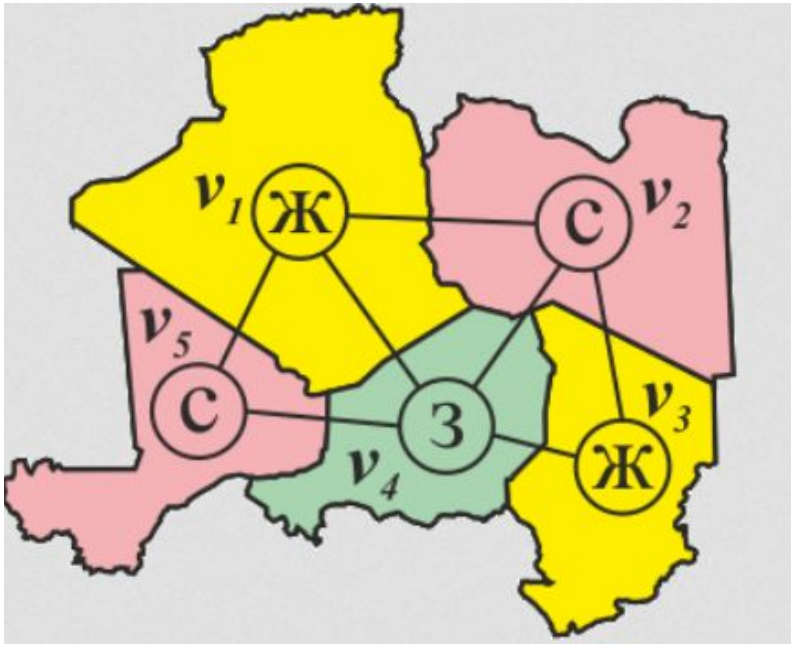
```
ДЛЯ ВСЕХ v dist[v] = ∞;
dist[r] = 0;
Cr += r;
Создать список непосещённых вершин Q=V;
 ПОКА Q != ∅ ;
        Выбрать из Q элемент u с минимальным расстоянием до r;
        Поместить u в список Cr += u;
        ДЛЯ ВСЕХ w являющихся соседями u;
                ЕСЛИ dist[w] > W(u,w);
                    ТО
                                {
                                 dist[w] = W(u,w);
                                 Cw +=u;
                                 }
```

Раскраска графа.

Раскраской вершин графа называется назначение цветов (в общем случае - меток) его вершинам.

Ставится задача раскраски в наименьшее число цветов так, что бы любые две смежные вершины имели разные цвета.

Наименьшее число цветов раскраски *хроматическим числом графа* и обозначается.



Другие применения раскраски графа:

1 Составление расписаний. Например, лекции – вершины, которые смежны тогда, когда они не могут проходить одновременно. Необходимо найти наименьшее число красок – пар.

2 Распределение регистров процессора в процессе компиляции. Регистры – цвета, переменные времена жизни которых пересекаются – вершины. Необходимо раскрасить граф таким образом, чтобы ни одна пара соседних узлов не имела одинаковый цвет.

Решение задачи о раскраске графа.

Переборный алгоритм

Вход: G=(V, A) - неориентированный граф, представленный матрицей смежности.

Выход: num[v] – вектор цветов вершин.

Алгоритм ПРГ:

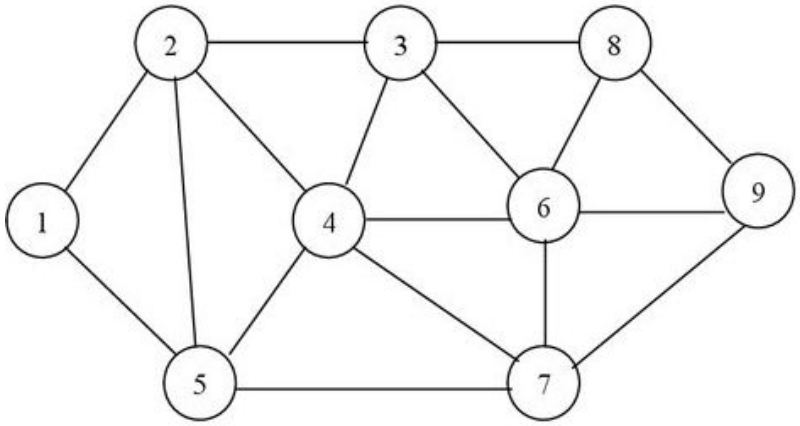
```
1 Создать список вершин Q=V, упорядоченных по убыванию степеней;
2 Для всех v num[v] =0;
3 Выбрать первый цвет r = 1;
4 Выбрать первую v из Q;
5 Окрасить вершину num[v] = r;
6 ПОКА не окрашены все вершины;
4           Выбрать из Q элемент u;
5           ЕСЛИ u НЕ СМЕЖНА с окрашенными в использованные цвета r = 1…;
6                       ТО
7                         {
8                           окрашиваем в num[u] = min ( из использованных r);
9                           Q -= u;
10                        }
11                      ИНАЧЕ
12                        {
13                          r++;
14                        }
```

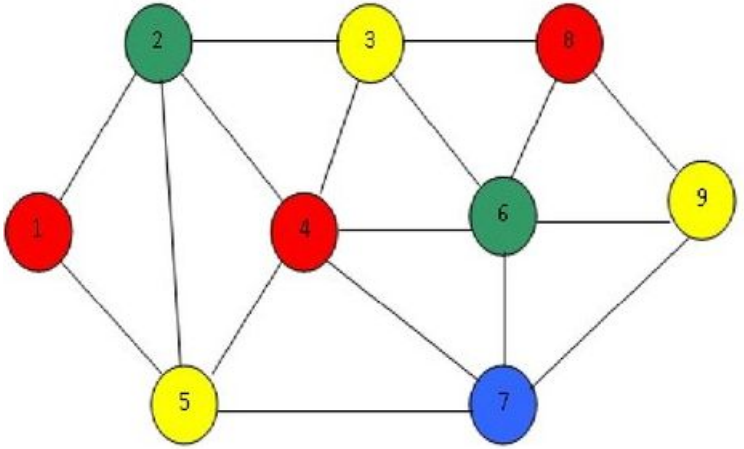
Решение задачи о раскраске графа.

Сведение к задаче о независимом множестве

Суть метода состоит в последовательном нахождении максимальных независимых множеств вершин с последующей их раскраской.

1 Найти в графе максимальное независимое множество вершин.
2 Раскрасить найденное множество в один цвет.
3 Удалить найденные вершины из графа.
4 Если остались не раскрашенные вершины, то повторять п. 1…3.





Потоки в графах.

Сетью называется ориентированный граф G = (V, A), в котором каждому ребру приписано два числа - неотрицательная *пропускную способность* c(u,v)>0 и *поток* f(u,v).

В сети выделяют некоторые особенные вершины: *источник* s и *сток* t, обладающие свойством, что любая другая вершина сети лежит на пути из s в t.

Свойства потока

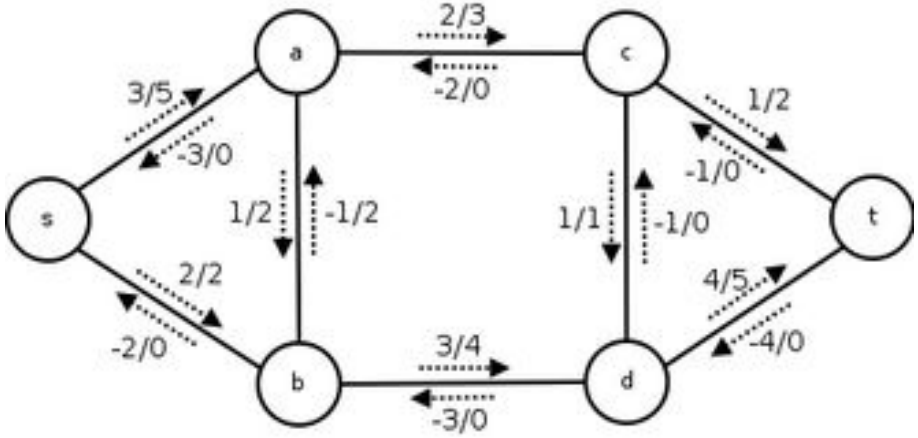
1 *Ограничение пропускной способностью*. Поток не может превысить пропускную способность: f(u,v) ≤ c(u,v).

2 *Антисимметричность*. Поток из u в v противоположен потоку из v в u: f(u,v) = - f(v,u).

3 *Сохранение потока*: sum (f(u,w)) = 0 для всех u из V, кроме источника и стока.

Величина потока - сумма потоков из источника |f| = sum (f(s,v)).

Сумма потоков из источника равна сумме потоков в сток.

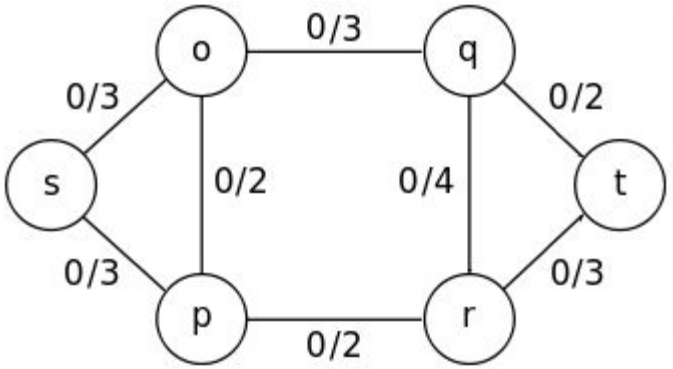


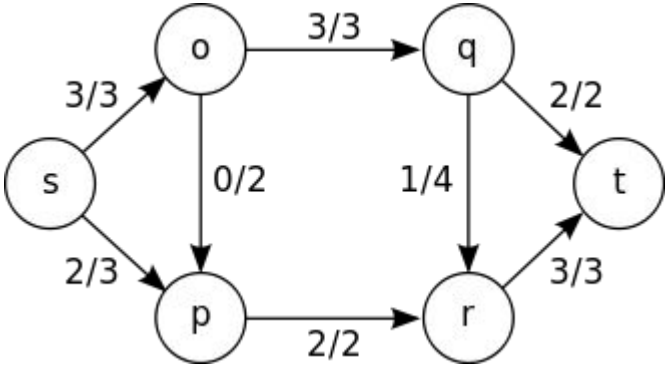
Потоки в графах.

Дуга называется *насыщенной*, если поток по ней равен ее пропускной способности.

Поток называется *полным*, если любой путь в сети из источника в сток содержит, по крайней мере, одну насыщенную дугу.

Задача о максимальном потоке заключается в нахождении такого потока по транспортной сети, что сумма потоков из истока, или, что то же самое, сумма потоков в сток максимальна.





Алгоритм Форда - Фалкерсона.

Дан граф G(V,A) с пропускной способностью c(u,v) и потоком f(u,v) = 0. Необходимо найти максимальный поток из источника s в сток t при действующих ограничениях на поток:

1 $f(u,v) \leq c(u,v)$;

2 $f(u,v) = - f(v,u)$;

3 $f_{in}(u) = f_{out}(u)$.

Идея алгоритма *Форда - Фалкерсона* заключается в следующем. Изначально величине потока присваивается значение 0: f(u,v) = 0 для всех $u, v \in V$.

Затем величина потока итеративно увеличивается посредством поиска увеличивающего пути (путь от источника *s* к стоку *t*, вдоль которого можно послать больший поток). Процесс повторяется, пока можно найти увеличивающий путь.

Алгоритм Форда - Фалкерсона.

Вход: G=(V, A, c, f, s, t) - ориентированный граф, в котором f(u,v) = 0 для всех u, v ∈ V.

Выход: f – максимальный поток из s в t.

Алгоритм НПФФ:

1 **ПОКА** есть путь p из s в t, такой, что c(u,v) > 0 для всех (u,v) ∈ p:
2 Найти минимальную $c_{min}(u,v)$ для ребер (u,v) ∈ p;
3 ДЛЯ КАЖДОГО ребра (u,v) ∈ p:
4 $f(u,v) = f(u,v) + c_{min}(u,v)$;
5 $f(v,u) = f(v,u) - c_{min}(u,v)$.

Для нахождения пути можно использовать любой из способов обхода.