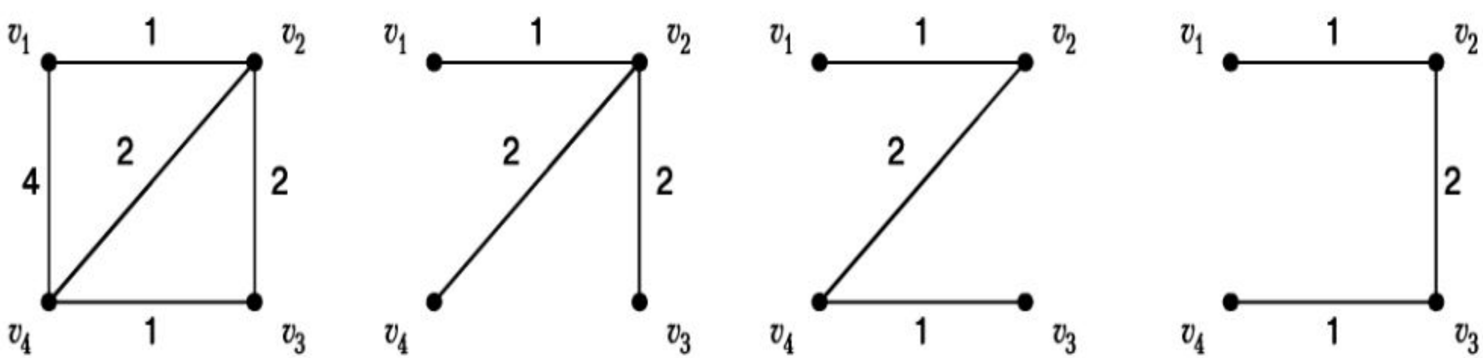
Остов графа
(покрывающее дерево)

Пусть G(V, E) — граф. Остовный подграф графа G(V, E) — это подграф, содержащий все вершины. Остовный подграф, являющийся деревом, называется остовом или каркасом (лес).
Букет – множество вершин, принадлежащих одной компоненте связности.



Алгоритм Краскала

Раскраска графа:

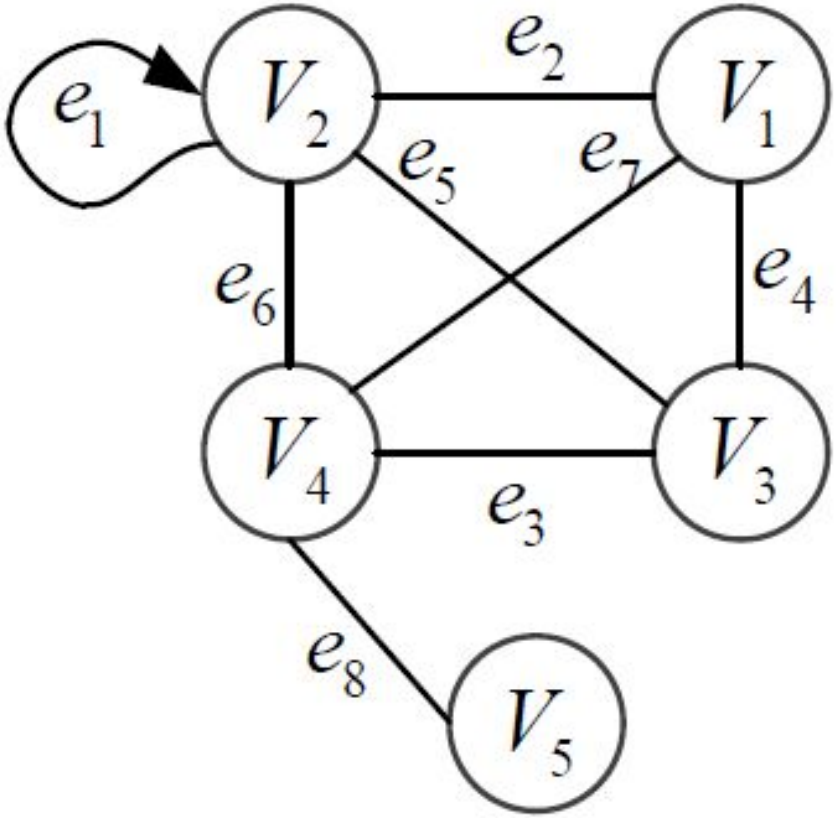
синим цветом окрашиваются ребра, включаемые в покрывающее дерево;

оранжевым цветом окрашиваются ребра, не включаемые в покрывающее дерево, т.к. они образуют цикл с синими ребрами или являются петлями.

Существует 2 случая связности графа:

1 Если G (**V**,**E**) – связный граф, то построение покрывающего дерева по алгоритму Краскала заканчивается, когда количество ребер, окрашенных в синий цвет, становится равным (р-1) .

2 Если G (**V**,**E**) – несвязный граф, то построение покрывающего дерева по алгоритму Краскала заканчивается после раскраски всех ребер графа. Число покрывающих деревьев в таком лесу будет равно числу


e_1
V_2
e_2
V_1
e_5
e_7
e_6
e_4
V_4
V_3
e_3
e_8
V_5

Начало: Все рёбра графа *G* (**V**,**E**) не окрашены и ни один из букетов не формирован.

Шаг 1. Все петли окрасить в оранжевый цвет.
Шаг 2. Из упорядоченного множества **E** выбирается первое ребро, не являющееся петлей. Это ребро окрашивается в синий цвет и формируется букет, в который включаются концевые вершины выбранного ребра.
Шаг 3. Из оставшихся ребер выбирается первое неокрашенное ребро, которое не является петлей. Если в графе такого ребра нет, следует закончить процедуру и перейти к шагу 4.
Шаг 4. Если все ребра окрашены, следует закончить процедуру. Синие ребра образуют покрывающее дерево (лес). В противном случае вернуться к началу шага 2.

Шаг 3. После выбора ребра возможны 4 случая:
А. Обе концевые вершины выбранного ребра принадлежат одному и тому же букету. В этом случае ребро окрашивается в оранжевый цвет.
Б. Одна из концевых вершин ребра принадлежит существующему букету, а другая – не принадлежит ни одному из уже сформированных букетов. В этом случае ребро окрашивается в синий цвет, и его вторая концевая вершина включается в букет, которому принадлежит первая концевая вершина.
В. Концевые вершины выбранного ребра принадлежат различным букетам. В этом случае ребро окрашивается в синий цвет, а оба букета, которым принадлежат его концевые вершины, объединяем в новый букет с меньшей нумерацией.
Г. Ни одна из концевых вершин не принадлежит ни одному из формированных букетов. В этом случае ребро окрашивается в синий цвет и формируется новый букет из

Алгоритм Краскала

Вход: список E рёбер графа G с длинами, упорядоченный в порядке возрастания длин.
Выход: множество T рёбер кратчайшего остова.

```
T = {}
k = 1 { номер рассматриваемого ребра }
for i from 1 to p-1 do
    while (T + E[k]) - цикл do
        k = k + 1 { пропустить это ребро }
    T= T + E[k] { добавить это ребро в остов }
    k = k + 1
end
```

Алгоритм Прима

Начало: Дан граф G (V,E) — связный и взвешенный. Все ребра упорядочены по возрастанию весов и по нумерации для ребер с одинаковыми весами. Петли удалены. Из кратных ребер оставляем ребро с минимальным весом.
Минимальное покрывающее дерево T не построено.
Букет не сформирован.

Шаг 1. Выбираем из упорядоченного множества E первое ребро (i,j) и включаем его в дерево T .
Обе концевые вершины включаем в букет { i, j} .
Удаляем из E выбранное ребро.

Шаг 2. Из множества **E** выбираем первое ребро, инцидентное какой-либо вершине из сформированного букета.

После выбора ребра возможны 2 случая:
А. Ребро составляет цикл с существующими ребрами дерева. Тогда удаляем его из E и возвращаемся к началу шага 2.
Б. Ребро не образует с существующими ребрами дерева цикл. Тогда новая вершина добавляется к вершинам сформированного букета. Ребро добавляется в дерево. Далее возвращаемся к началу шага 2.

Шаг 3. Если дерево (букет) включает в себя все вершины графа,
то минимальное покрывающее дерево построено.
(Если дерево не включает в себя все вершины, то граф не является связным.)

a[v] — это ближайшая к v вершина, уже включённая в остов, b[v] — это длина ребра, соединяющего v с остовом. Если вершину v ещё нельзя соединить с остовом одним ребром, то a[v]: = 0, b[v]: = ∞.

Вход: граф G(V, E), заданный матрицей длин рёбер C.
Выход: множество T рёбер кратчайшего остова.

Алгоритм:

select $u \in V$ { выбираем произвольную вершину }

S={u} { S - множество вершин, включённых в кратчайший остов }

$T = \varnothing$ { T - множество рёбер, включённых в кратчайший остов }

```
for v ∈ V - u do
    if v ∈ Γ(u) then
     a[v]: = u { u — ближайшая вершина остова }
     b[v]: = C[u, v] { C[u, v] — длина соответствующего ребра }
    else
     a[v]: = 0 { ближайшая вершина остова неизвестна }
     b[v]: = ∞ { и расстояние также неизвестно }
    end if
end for
```

```
 for i =1 to p - 1 do
   x: = ∞ {начальное значение для поиска ближайшей
вершины}
   for v ∈ V \ S do
      if b[v] < x then
         w = v { нашли более близкую вершину }
         x = b[v] { и расстояние до неё }
     end if
   end for
   S = S + w { добавляем найденную вершину в остов }
   T =T + (a[w],w) { добавляем найденное ребро в остов }
   for v ∈ Г(w) do
   if v ∉ S then
      if b[v] >C[v,w] then
            a[v]: = w { изменяем ближайшую вершину остова }
            b[v]: = C[v, w] { и длину ведущего к ней ребра }
      end if
   end if
```

Задача Штейнера