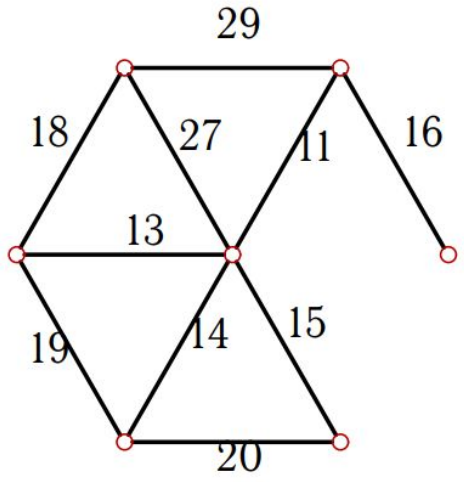
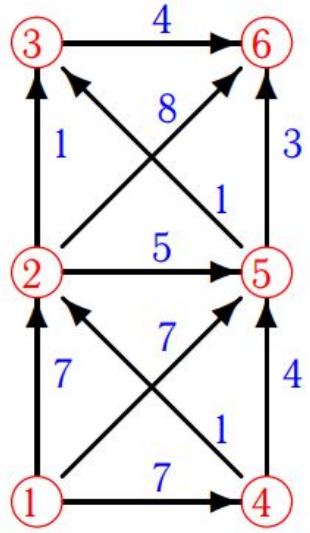
Взвешенные графы
Остовные деревья
Кратчайшие пути

Взвешенный граф

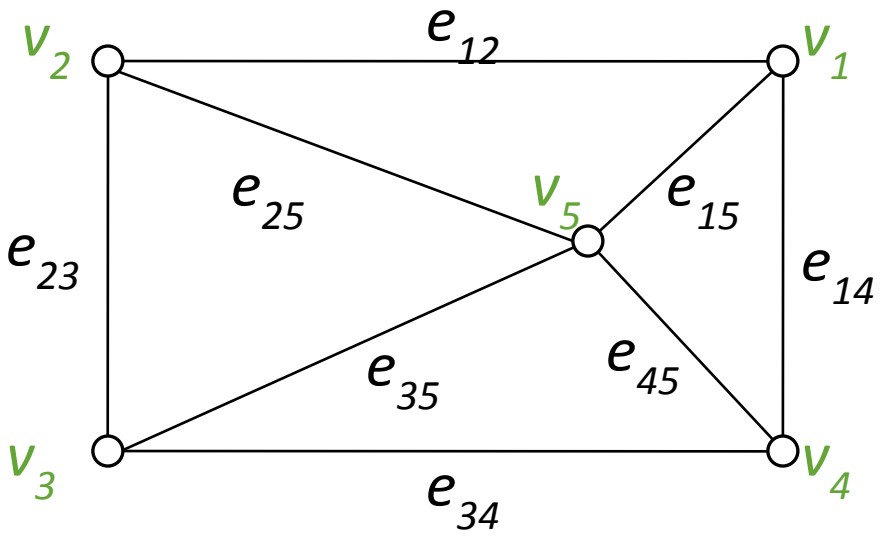
Если каждому ребру в графе / дуге в орграфе сопоставлено некоторое число, называемое весом ребра / дуги, то граф / орграф называется взвешенным.

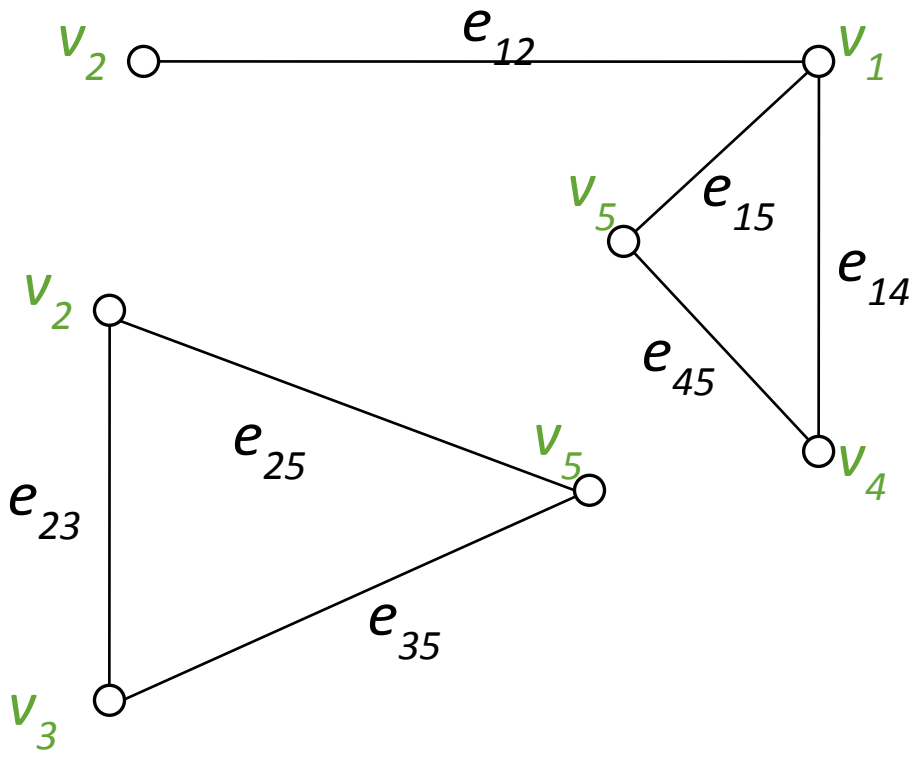




Подграф

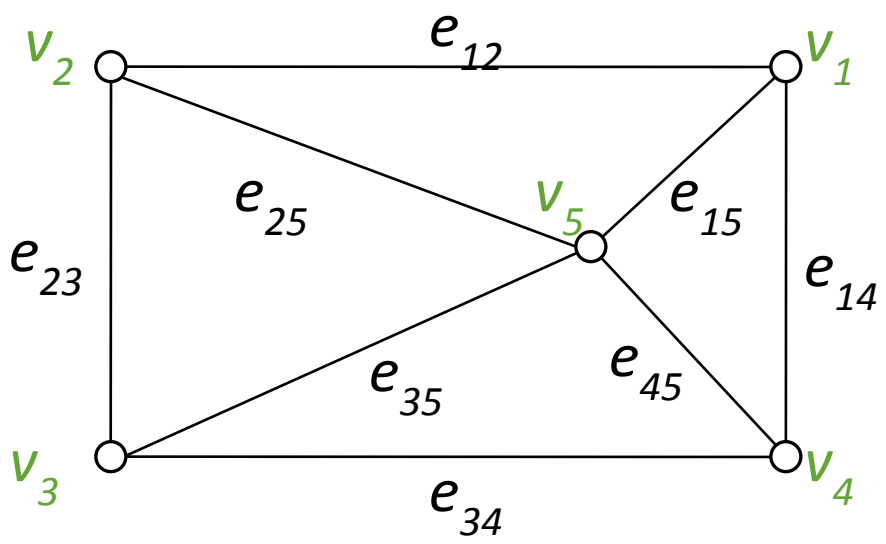
Граф, все вершины и ребра которого принадлежат исходному графу.

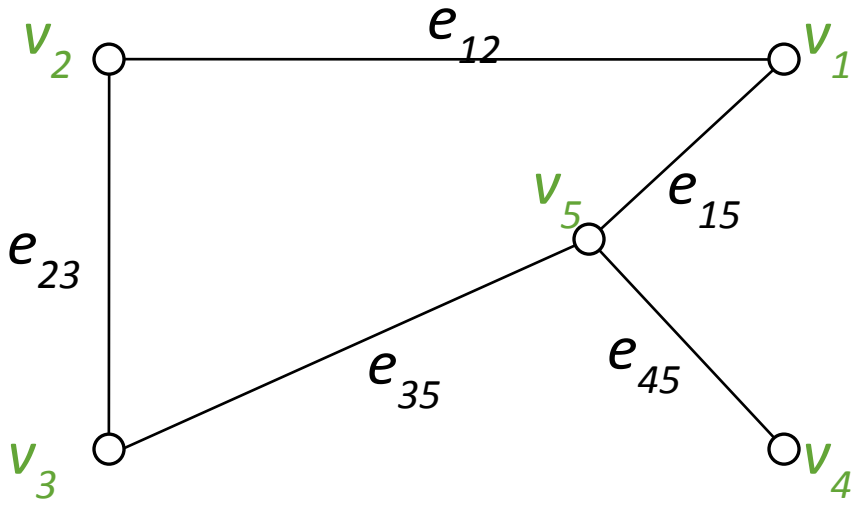




Остовной подграф

Остовным называется подграф, которое содержит все вершины исходного графа.

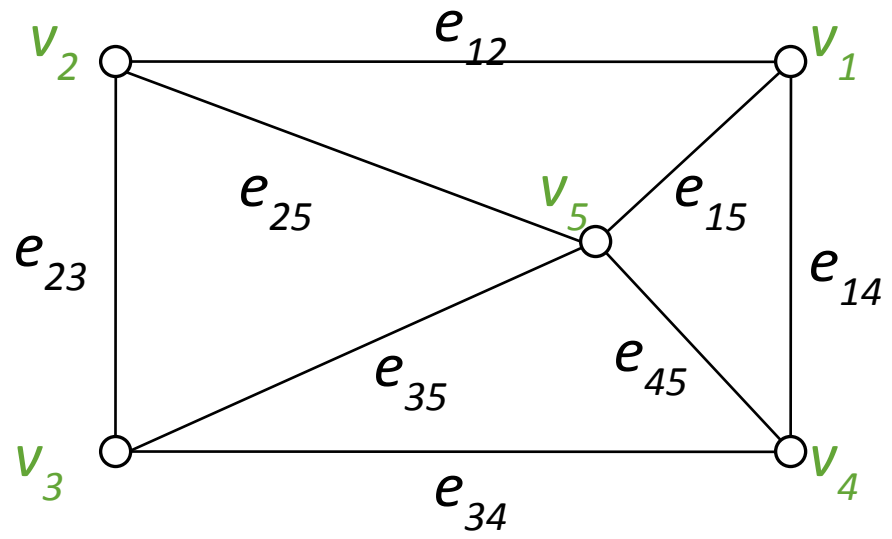


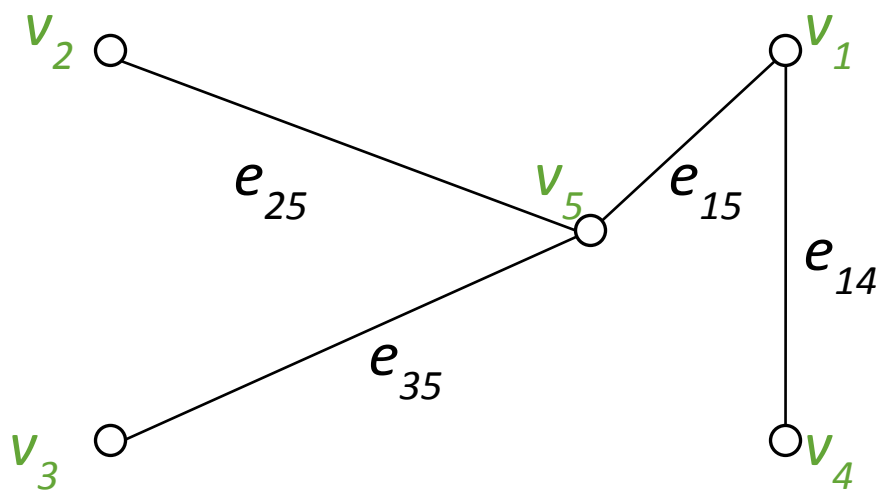


Остовное дерево графа

Остовное дерево графа (остов) – ациклический связный подграф, содержащий все вершины исходного графа.

Остовное дерево, у которого суммарный вес его рёбер минимален, называется минимальным остовным деревом.





Теорема Кирхгофа

Число остовных деревьев графа равно алгебраическому дополнению любого элемента матрицы Кирхгофа.

Матрица Кирхгофа графа *G* определяется следующим образом:

$$k_{ij} = \begin{cases} -1, & e_{ij} \in E, i \neq j, \\ 0, & e_{ij} \notin E, i \neq j, \\ \deg(i), & i = j. \end{cases}$$

Свойства матрицы Кирхгофа:

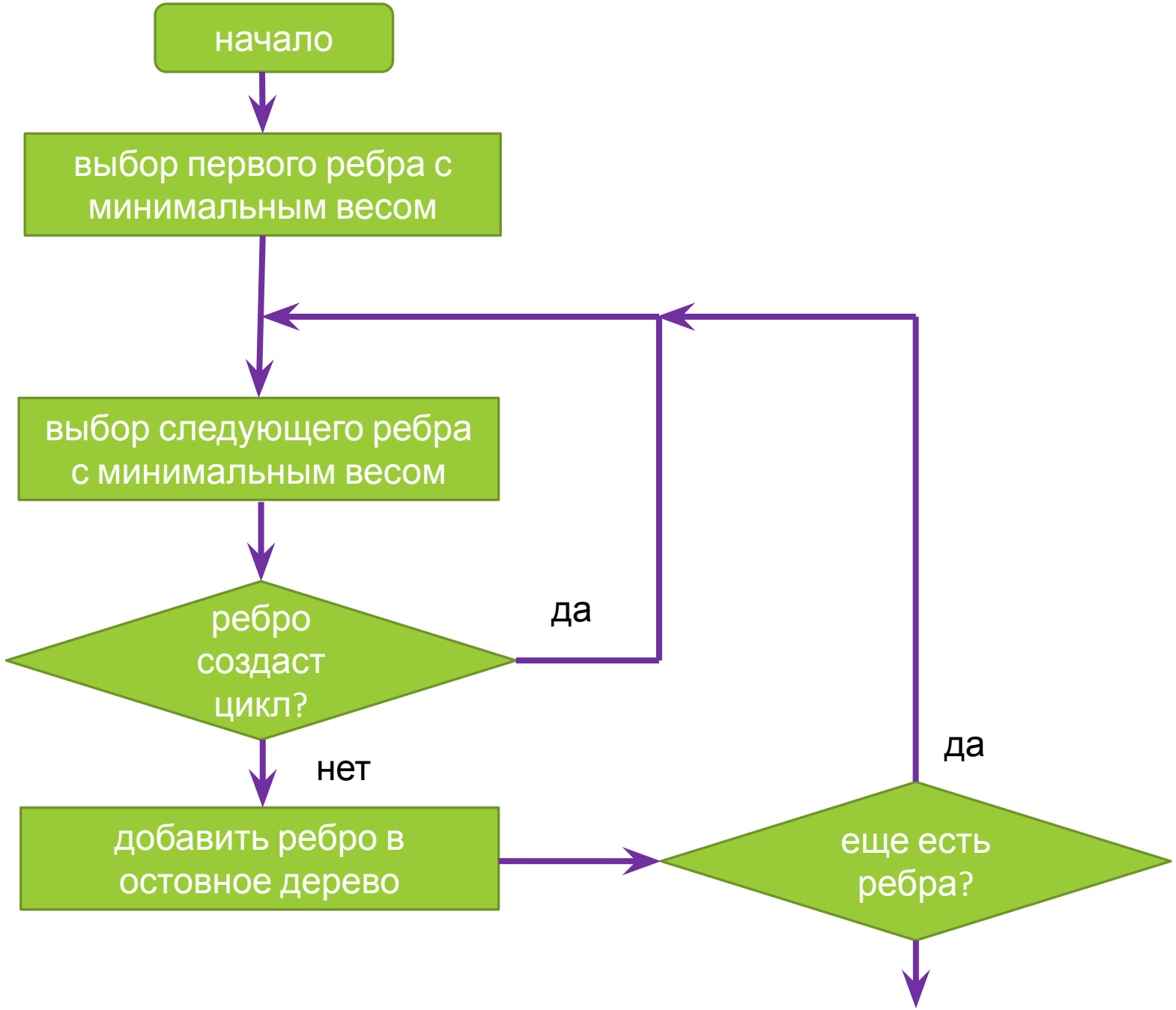
1) *сумма* элементов в каждой строке и каждом столбце матрицы Кирхгофа равна 0;

2) алгебраические дополнения всех элементов матрицы Кирхгофа равны;

Алгоритм Краскала, 1956

1. Сортируем ребра графа по возрастанию весов.
2. Полагаем, что каждая вершина относится к своей компоненте связности.
3. Проходим ребра в "отсортированном" порядке. Для каждого ребра выполняем:
   a) если вершины, соединяемые данным ребром, лежат в разных компонентах связности, то объединяем эти компоненты в одну, а рассматриваемое ребро добавляем к минимальному остовному дереву;
   b) если вершины, соединяемые данным ребром лежат в одной компоненте связности, то исключаем ребро из рассмотрения.
4. Если есть еще нерассмотренные ребра и не все компоненты связности объединены в одну, то переходим к шагу 3, иначе выход.

Алгоритм Краскала



Алгоритм Прима

На каждом шаге вычеркиваем из графа дугу максимальной стоимости с тем условием, что она не разрывает граф на две или более компоненты связности, т.е. после удаления дуги граф должен оставаться связным.

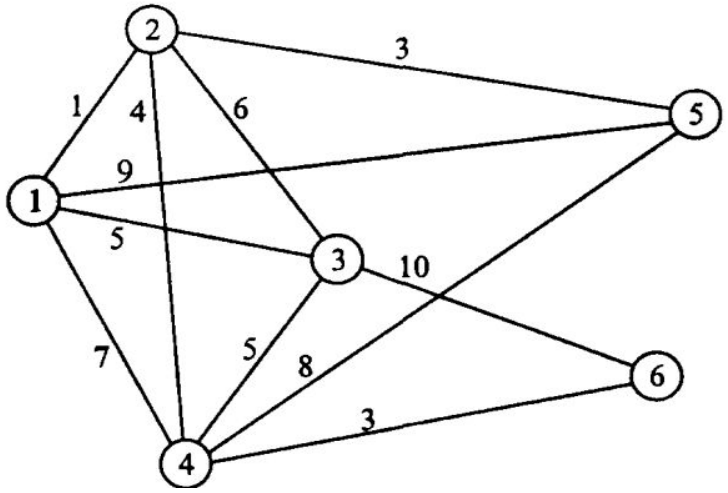
Для того, чтобы определить, остался ли граф связным, достаточно запустить поиск в глубину от одной из вершин, связанных с удаленной дугой.

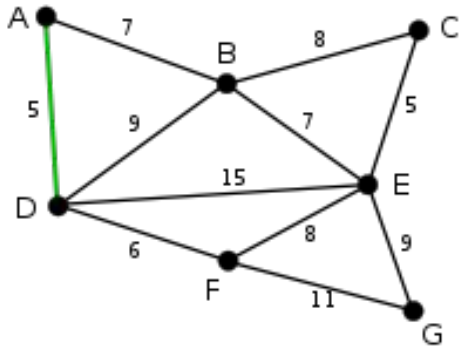
Условие окончания алгоритма?

Например, пока количество ребер больше либо равно количеству вершин, нужно продолжать, иначе – остановиться.

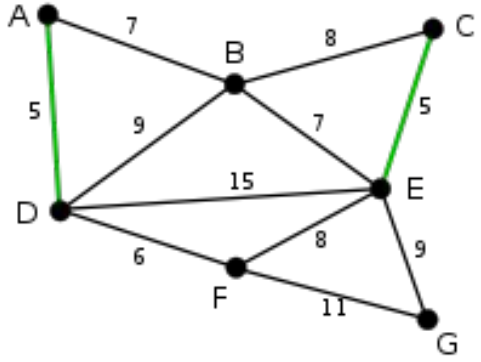
Задача

Телевизионная компания планирует подключение к своей кабельной сети пяти новых районов. На рисунке показана структура планируемой сети и расстояния (в км) между районами и телецентром. Необходимо спланировать наиболее экономичную кабельную сеть.

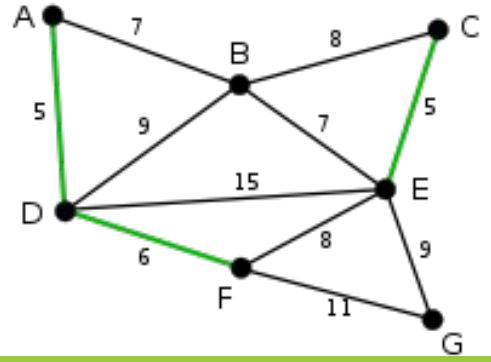




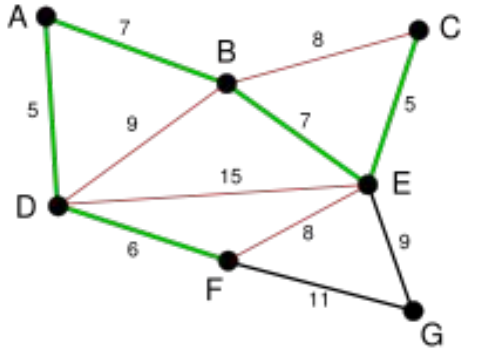
Ребра AD и CE имеют минимальный вес, равный 5. Произвольно выбирается ребро AD (выделено на рисунке).



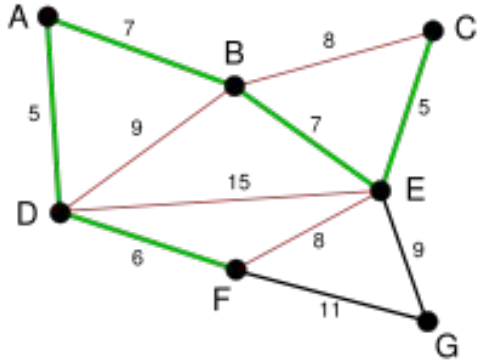
Теперь наименьший вес, равный 5, имеет ребро CE.
Так как добавление CE не образует цикла, то выбираем его в качестве второго ребра.



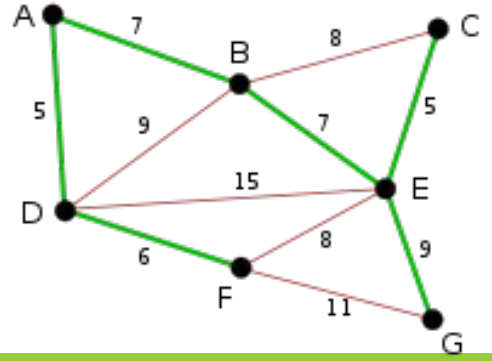
Аналогично выбираем ребро DF, вес которого равен 6.



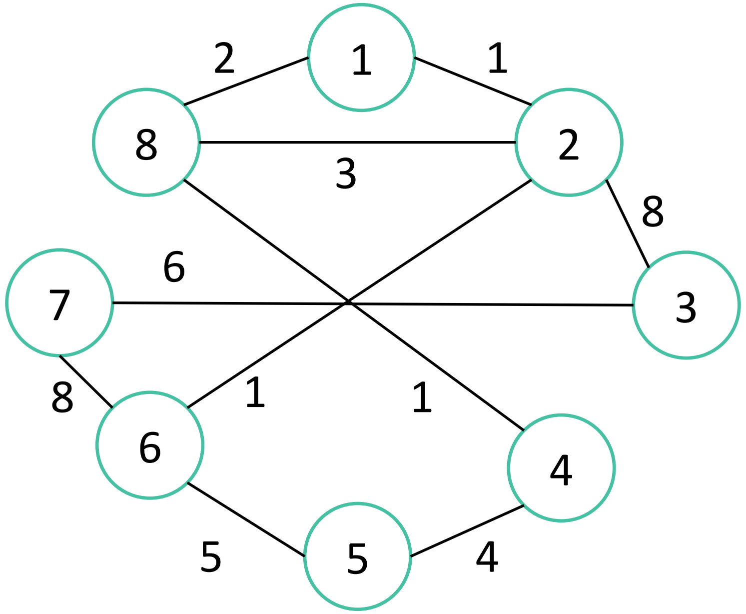
Следующие ребра — AB и BE с весом 7. Произвольно выбирается ребро AB, выделенное на рисунке. Ребро BD выделено красным, так как уже существует путь (зелёный) между A и D, поэтому, если бы это ребро было выбрано, то образовался бы цикл ABD.

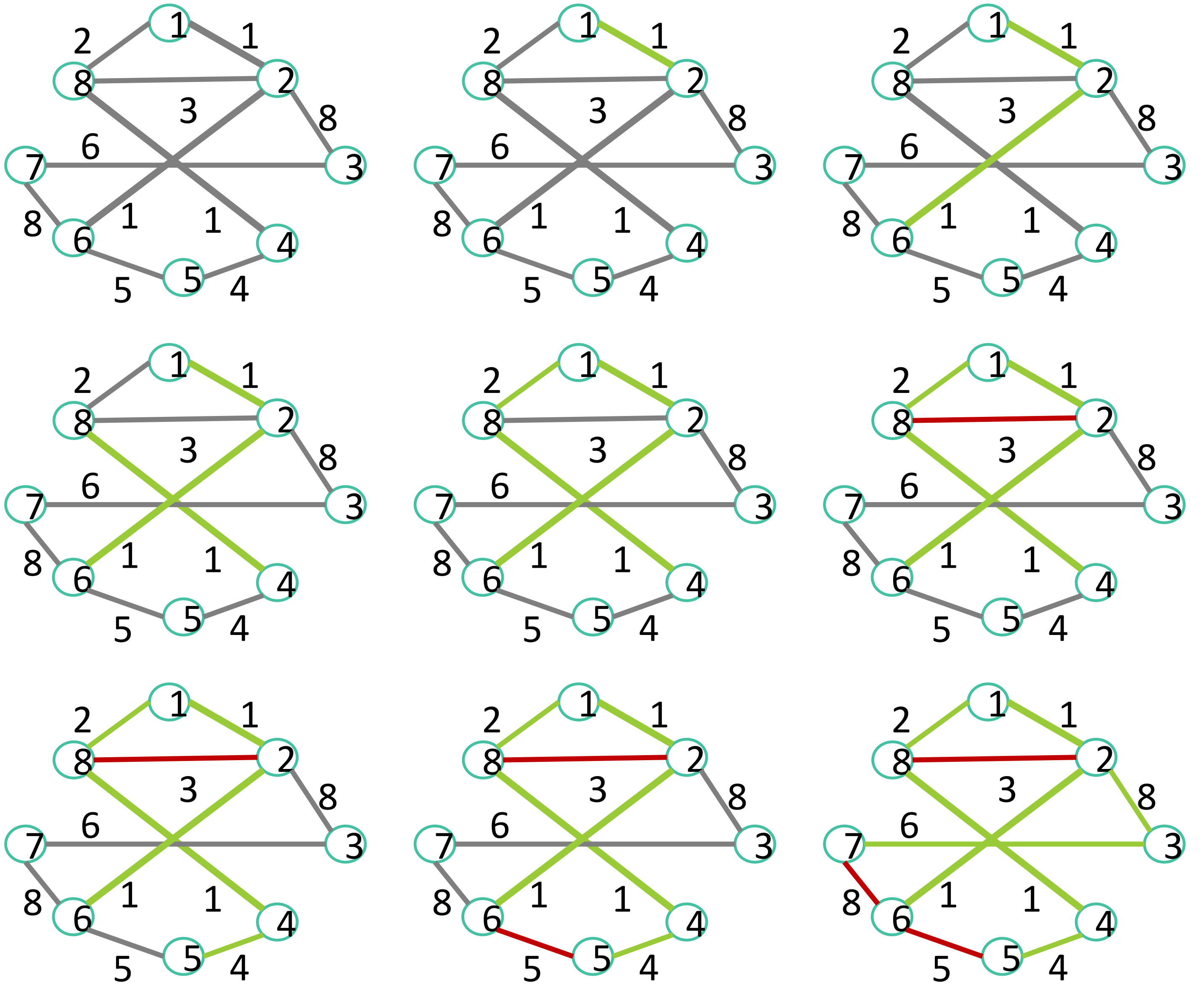


Аналогичным образом выбирается ребро BE, вес которого равен 7. На этом этапе красным выделено гораздо больше ребер: BC, потому что оно создаст цикл BCE, DE, потому что оно создаст цикл DEBA, и FE, потому что оно сформирует цикл FEBAD.



Алгоритм завершается добавлением ребра EG с весом 9.
Минимальное остовное дерево построено построено.


1
2
3
4
5
6
7
8



Поиск минимального пути

Алгоритм Флойда-Уоршелла

Итогом работы алгоритма является матрица расстояний между всеми парами вершин.

Инициализация:

Начальная матрица получается из матрицы смежности, следующим образом:

- Если в графе есть ребро из вершины i в вершину j длиной L, то $d_{ij} = L$.
- Расстояние от любой вершины до самой себя равно 0, т.е. $d_{ij} = 0$, если $i = j$.
- Для всех остальных пар i и j $d_{ij} = \infty$.

$$D = \begin{cases} d_{ii} = 0, \\ d_{ij} = |e_{ij}|, \quad e_{ij} \in E, \\ d_{ij} = \infty, \quad e_{ij} \notin E, \end{cases}$$

Алгоритм Флойда-Уоршелла

Алгоритм состоит из N итераций, где N – количество вершин в графе.

Перед k-ой итерацией в d_{ij} находится:

- длина минимального пути, проходящего через вершины с номерами, меньшими k, либо
- ∞, если пути между данными вершинами не найдено.

Во время k-ой итерации путь между вершинами i и j может либо сохраниться, либо пройти через вершину с номером k, если суммарный путь от вершины i в вершину k и из вершины k в вершину j меньше, чем d_{ij}.

То есть на k-ой итерации $d_{ij} = \min(d_{ij}, d_{ik} + d_{kj})$.

Восстановление пути в алгоритме Флойда-Уоршелла

Заведём дополнительную матрицу, размера $N \times N$, заполненную -1. Когда расстояние между вершинами i и j обновляется, благодаря проходу через вершину k, запомним это ($p_{ij} = k$).

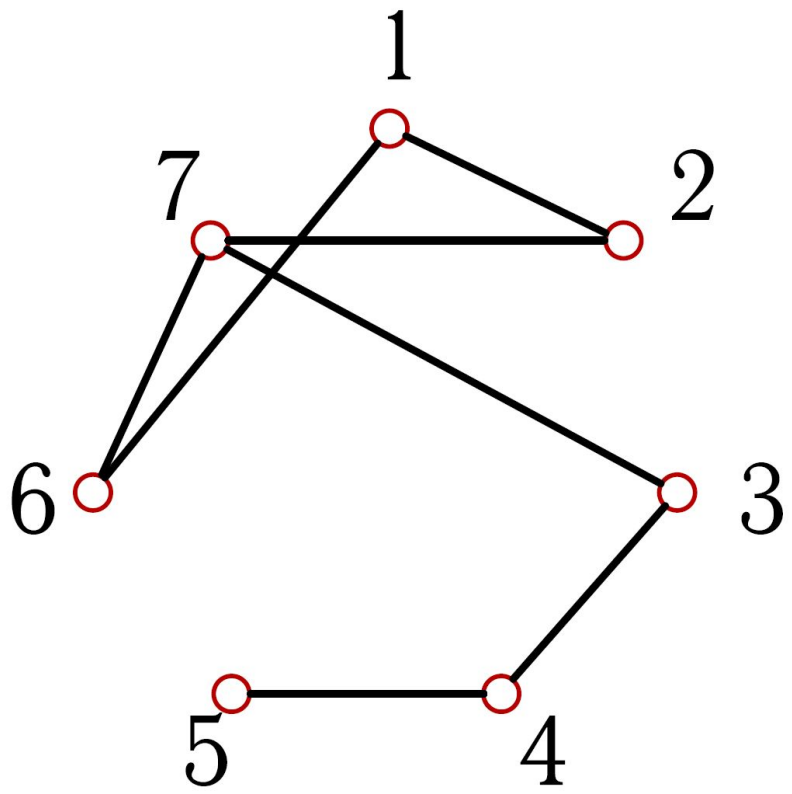
Если $d_{ij} = \infty$, то пути не существует, а значит восстанавливать нечего.

Если существует непосредственный путь из i в j, то $p_{ij} = -1$.

Иначе следует восстановить пути из i в p_{ij} и из p_{ij} в j, объединение которых и будет восстановленным путём из i в j.

Восстановление удобно реализуется рекурсивной функцией.

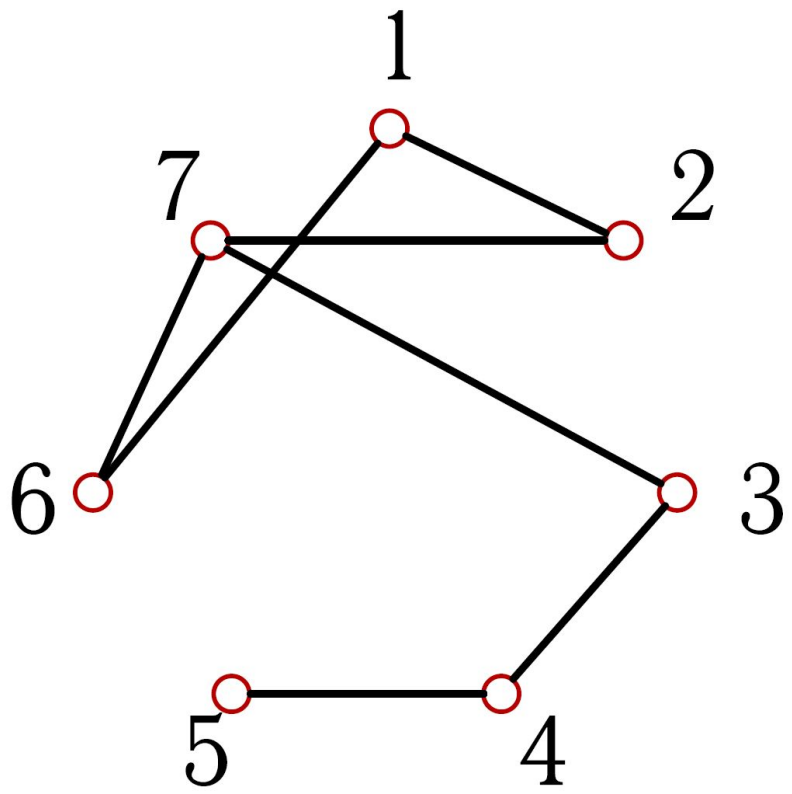
Пример



Матрица смежности:

0	1	0	0	0	1	0
1	0	0	0	0	0	1
0	0	0	1	0	0	1
0	0	1	0	1	0	0
0	0	0	1	0	0	0
1	0	0	0	0	0	1
0	1	1	0	0	1	0

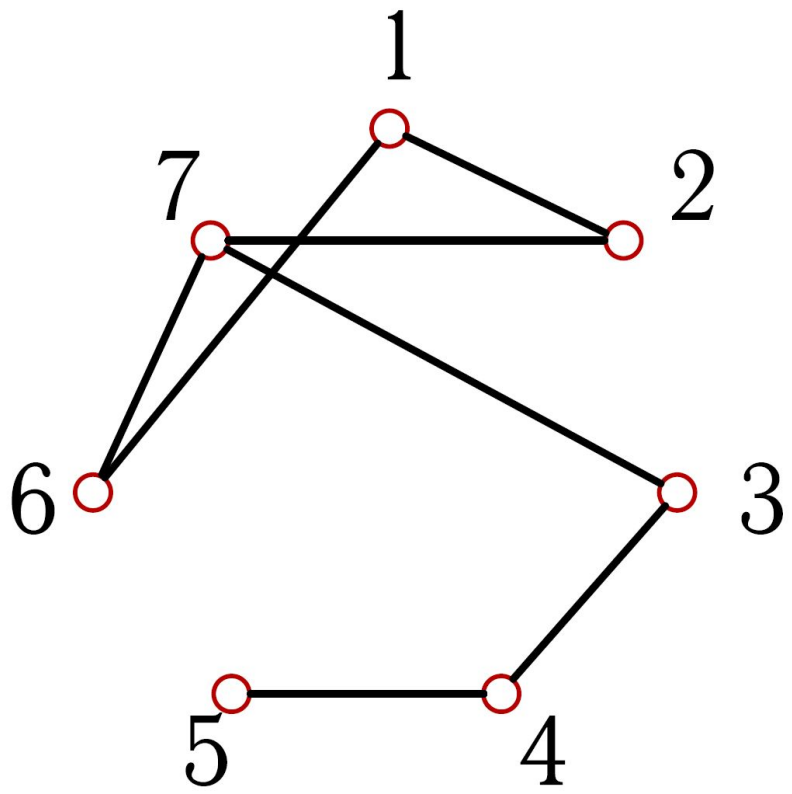
Пример



Матрица 1-расстояний

0	1	∞	∞	∞	1	∞
1	0	∞	∞	∞	∞	1
∞	∞	0	1	∞	∞	1
∞	∞	1	0	1	∞	∞
∞	∞	∞	1	0	∞	∞
1	∞	∞	∞	∞	0	1
∞	1	1	∞	∞	1	0

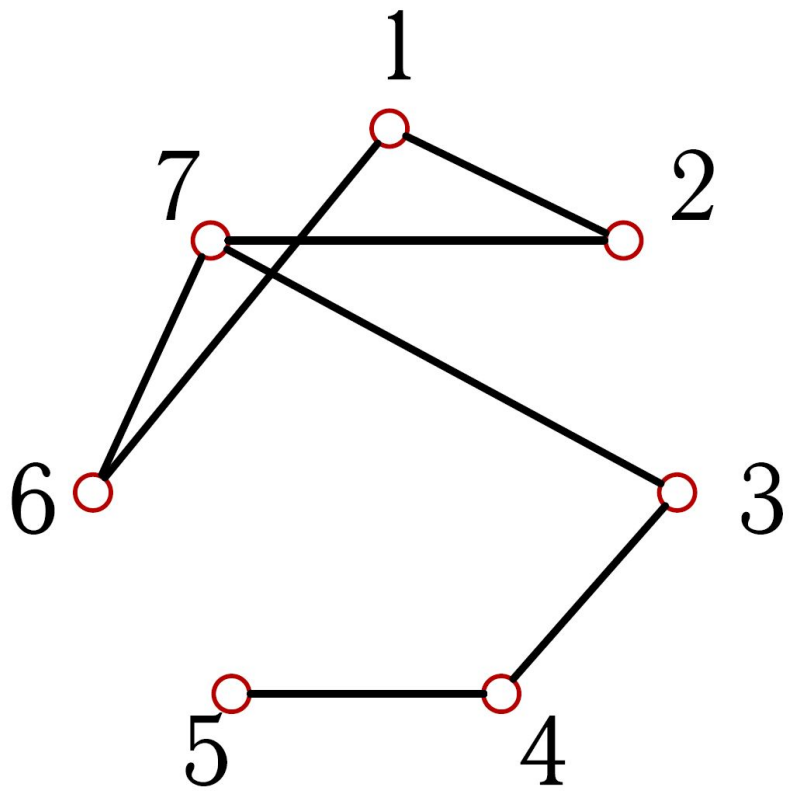
Пример



1-я итерация

0	1	∞	∞	∞	1	∞
1	0	∞	∞	∞	2	1
∞	∞	0	1	∞	∞	1
∞	∞	1	0	1	∞	∞
∞	∞	∞	1	0	∞	∞
1	2	∞	∞	∞	0	1
∞	1	1	∞	∞	1	0

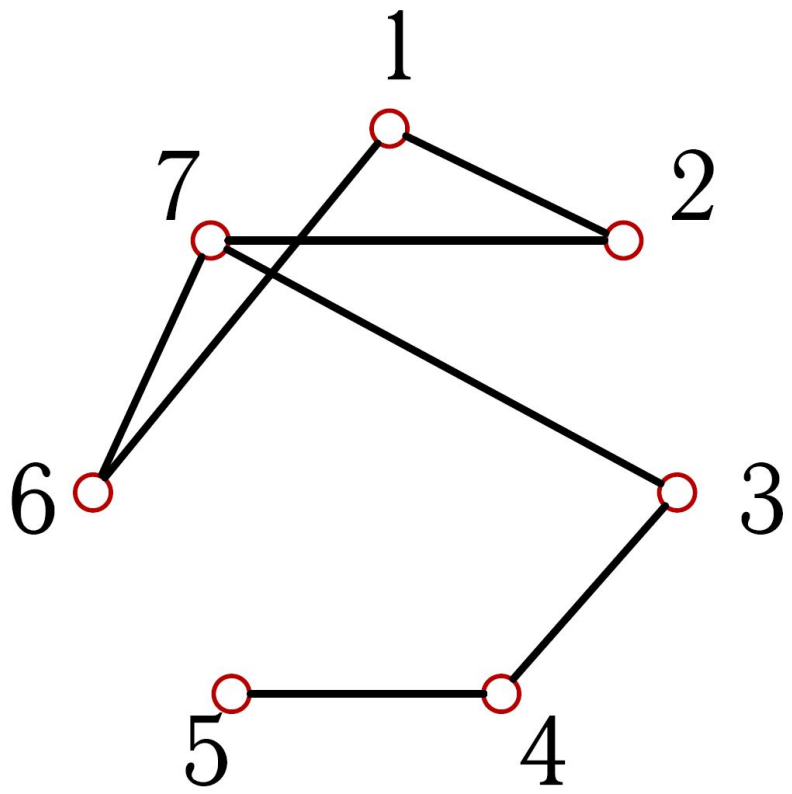
Пример



2-я итерация

0	1	∞	∞	∞	1	2
1	0	∞	∞	∞	2	1
∞	∞	0	1	∞	∞	1
∞	∞	1	0	1	∞	∞
∞	∞	∞	1	0	∞	∞
1	2	∞	∞	∞	0	1
2	1	1	∞	∞	1	0

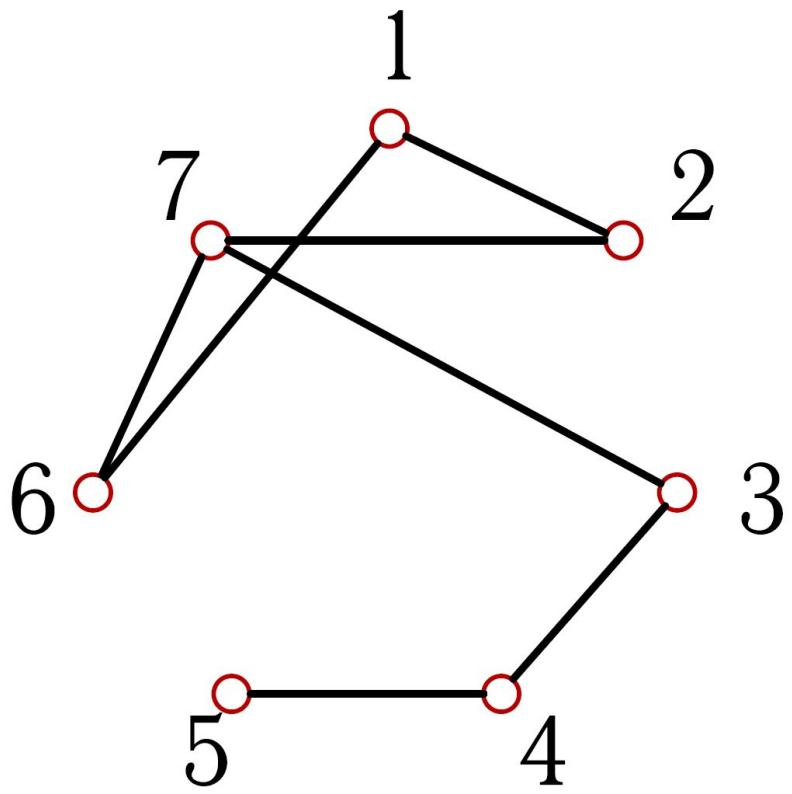
Пример



3-я итерация

0	1	∞	∞	∞	1	2
1	0	∞	∞	∞	2	1
∞	∞	0	1	∞	∞	1
∞	∞	1	0	1	∞	2
∞	∞	∞	1	0	∞	∞
1	2	∞	∞	∞	0	1
2	1	1	2	∞	1	0

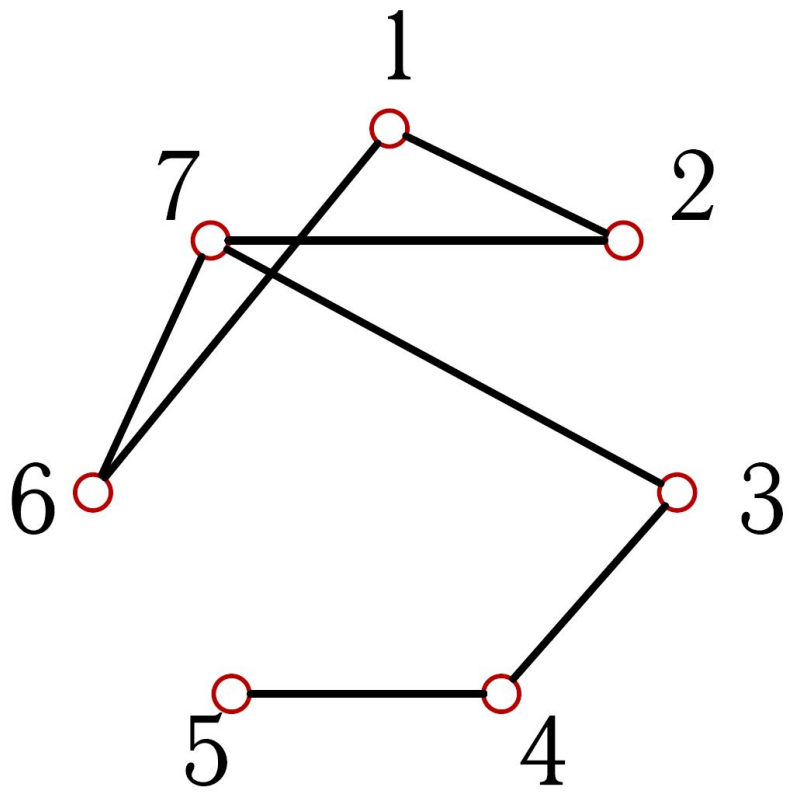
Пример



4, 5, 6 итерации

0	1	∞	∞	∞	1	2
1	0	∞	∞	∞	2	1
∞	∞	0	1	2	∞	1
∞	∞	1	0	1	∞	2
∞	∞	2	1	0	∞	3
1	2	∞	∞	∞	0	1
2	1	1	2	3	1	0

Пример



Матрица кратчайших расстояний:

$$\begin{matrix} 0 & 1 & 3 & 4 & 5 & 1 & 2 \\ 1 & 0 & 2 & 3 & 4 & 2 & 1 \\ 3 & 2 & 0 & 1 & 2 & 2 & 1 \\ 4 & 3 & 1 & 0 & 1 & 3 & 2 \\ 5 & 4 & 2 & 1 & 0 & 4 & 3 \\ 1 & 2 & 2 & 3 & 4 & 0 & 1 \\ 2 & 1 & 1 & 2 & 3 & 1 & 0 \end{matrix}$$

Эксцентриситеты вершин:

$$[5 \quad 4 \quad 3 \quad 4 \quad 5 \quad 4 \quad 3]$$

Алгоритм Дейкстры

Данный алгоритм даёт кратчайшие пути от некоторой вершины графа, до всех остальных.

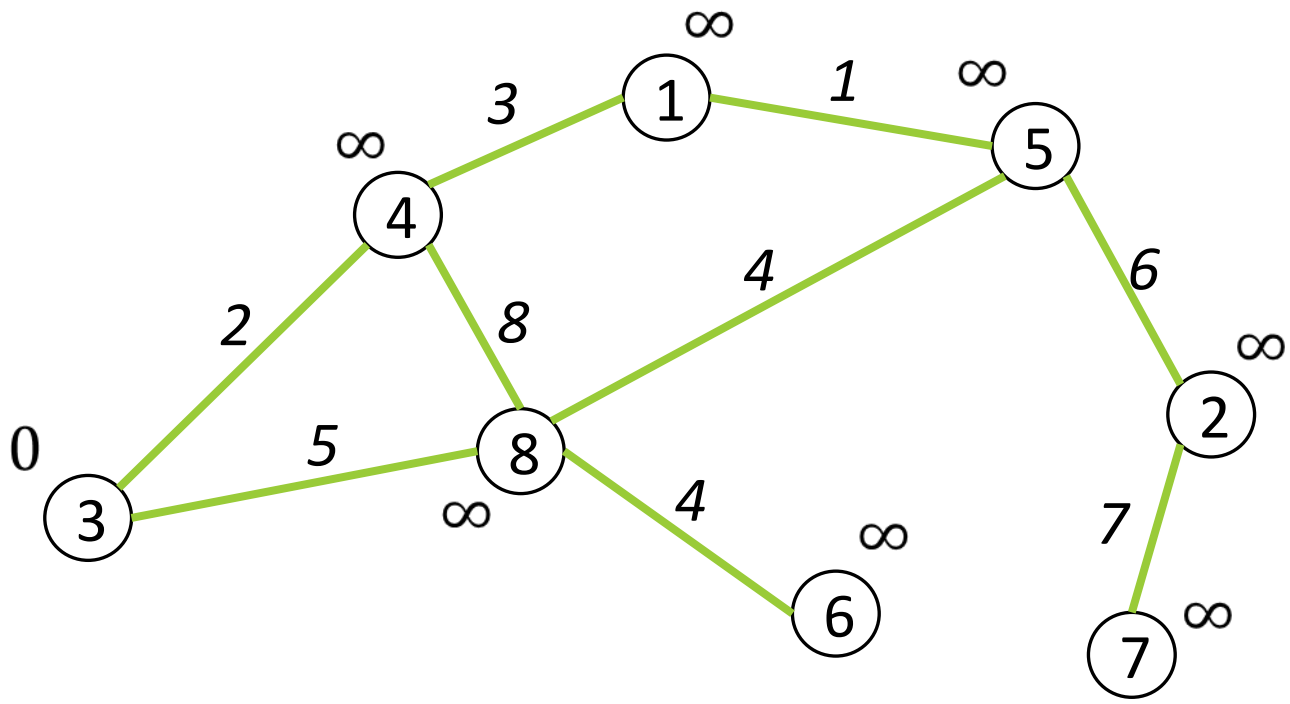
Инициализация:

Для работы алгоритма каждой вершине сопоставляется минимальное известное расстояние до неё. Изначально расстояние до стартовой вершины равно 0, а до всех остальных – ∞. Также все вершины графа помечаются как непосещённые.

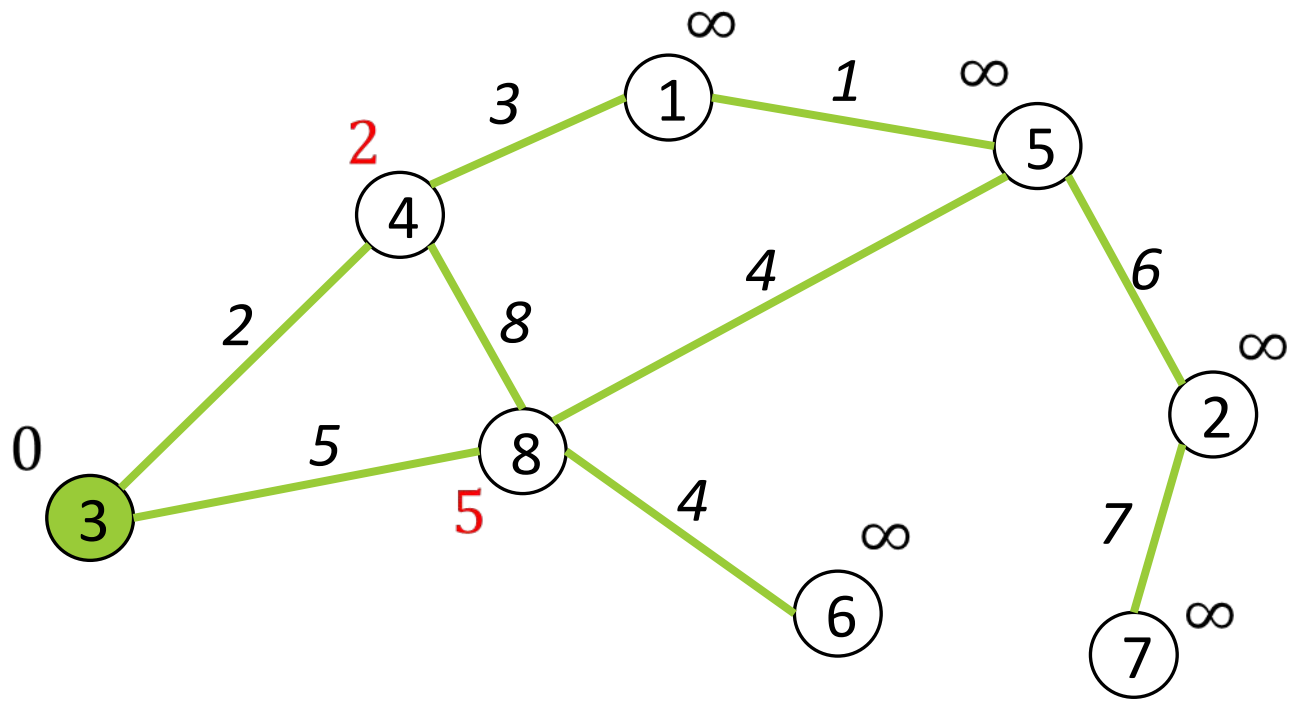
Алгоритм Дейкстры

1. Если все вершины посещены, алгоритм завершается.
2. Из списка непосещённых выбирается вершина v_j с минимальной меткой.
3. Для каждой вершины v_k, смежной с v_j непосещённой вершины:
   a) находится сумма между текущим расстоянием до v_j и длиной ребра, соединяющего v_j и v_k,
   b) если найденная сумма меньше, чем текущее расстояние до v_k, то она его заменяет.
4. После рассмотрения всех смежных вершин текущая помечается как посещённая.

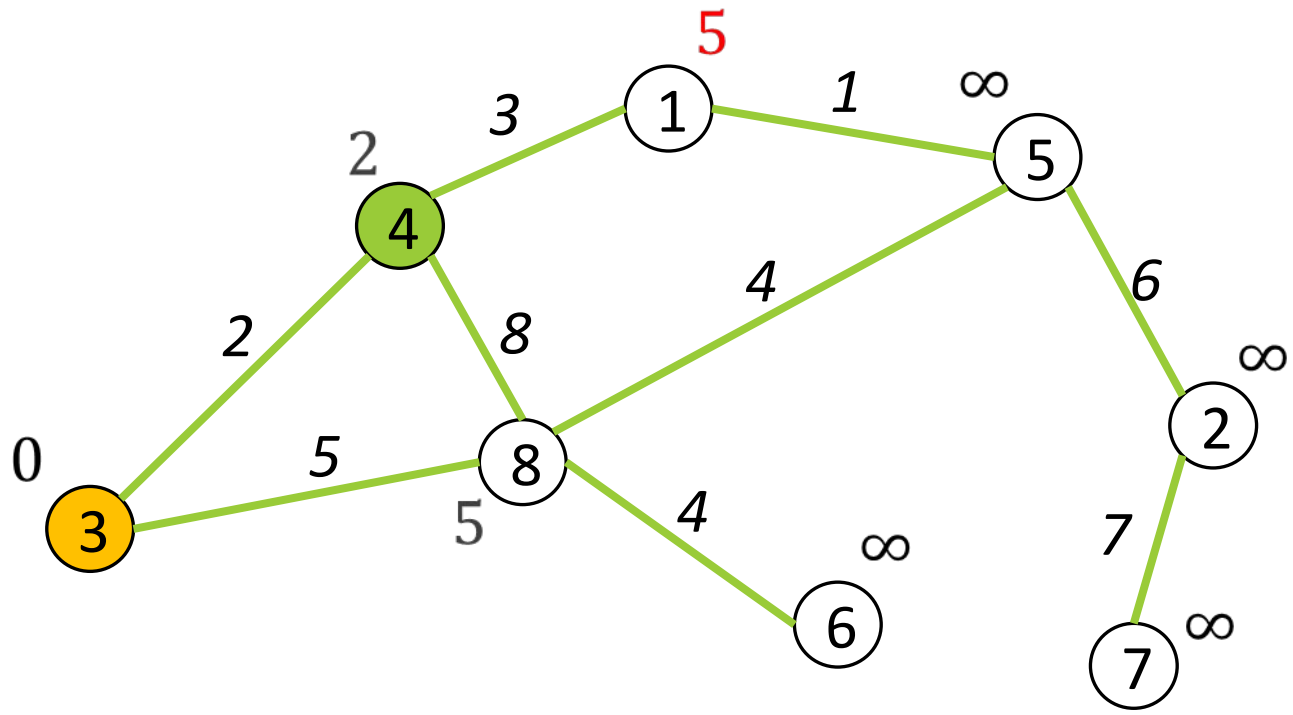
Пример



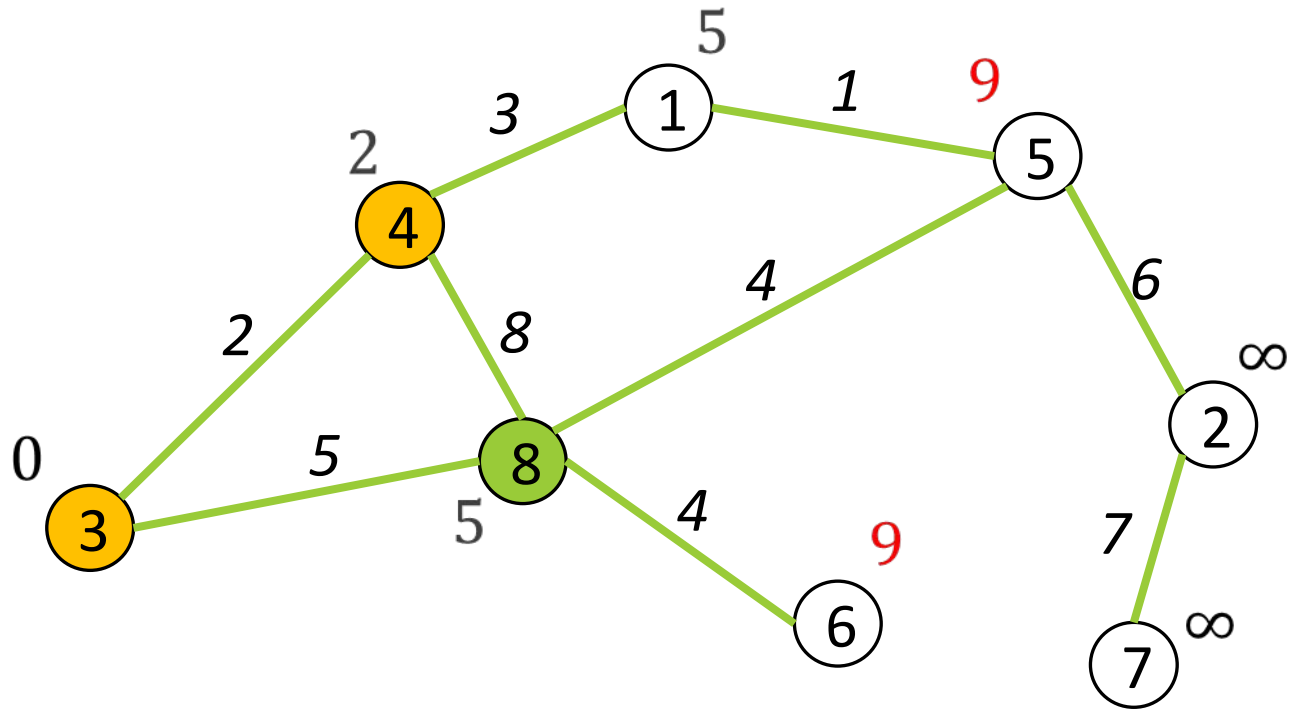
Пример



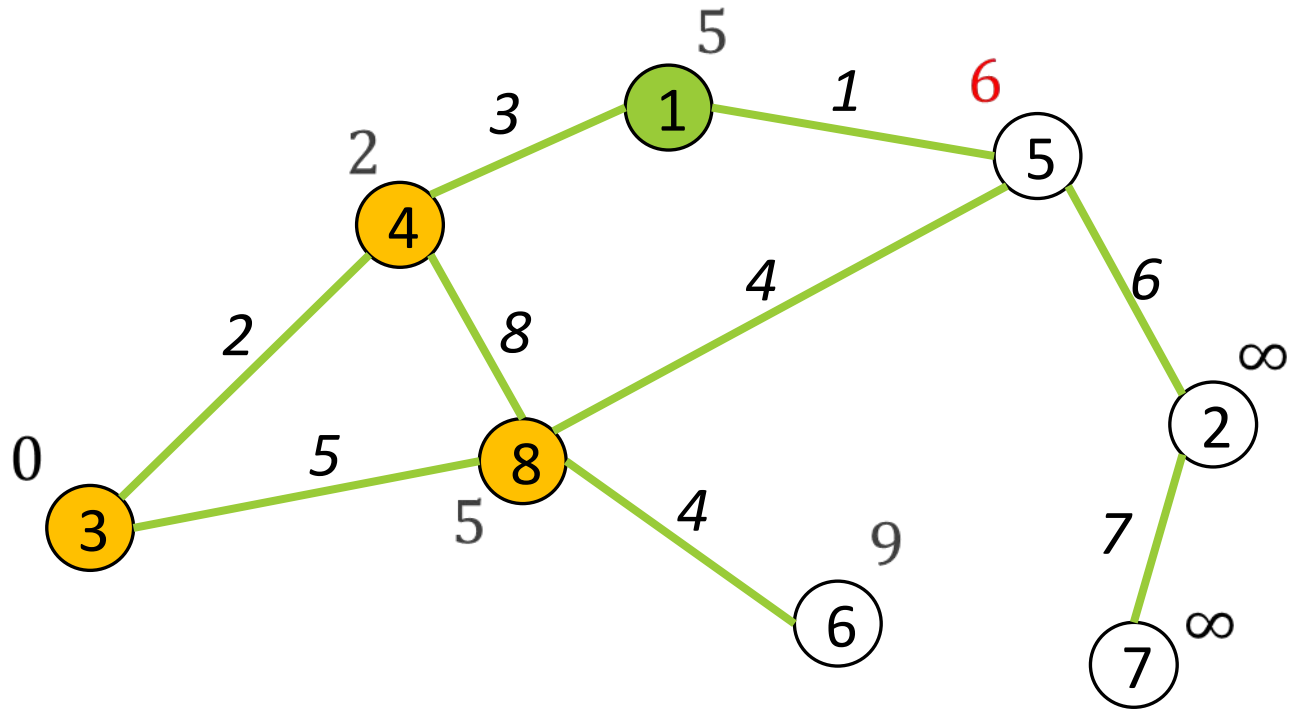
Пример



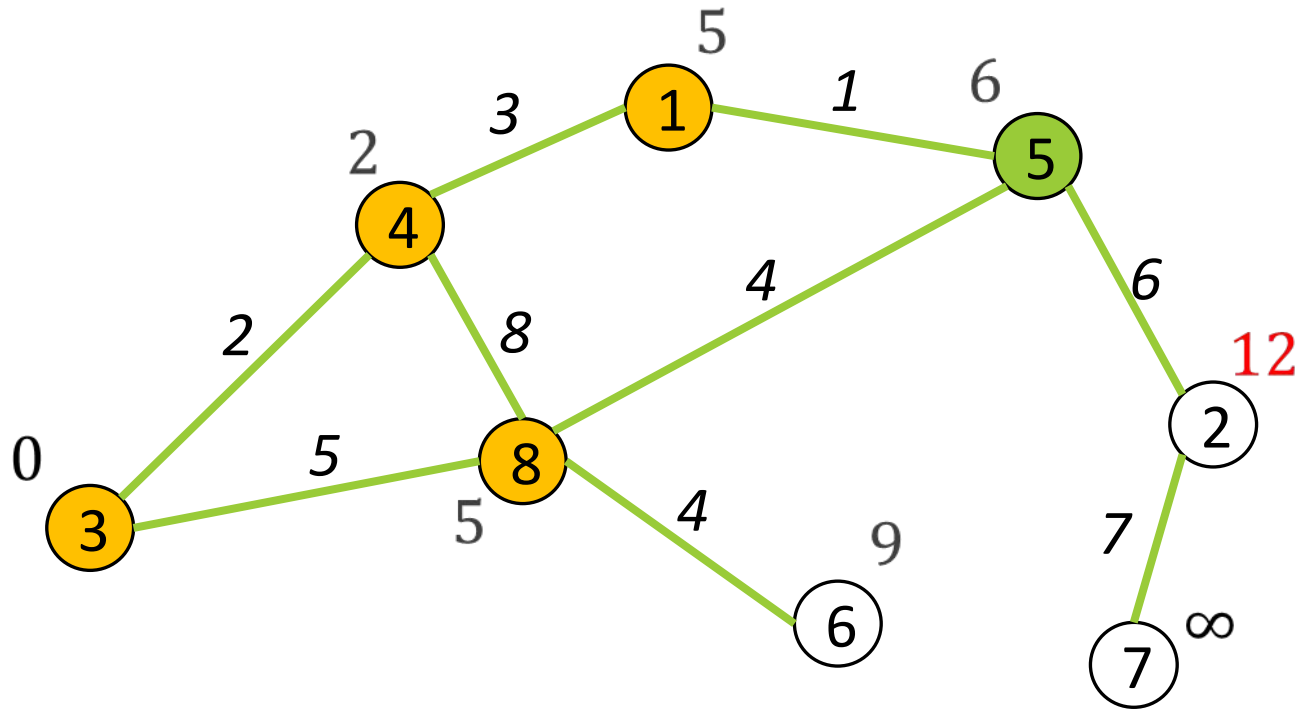
Пример



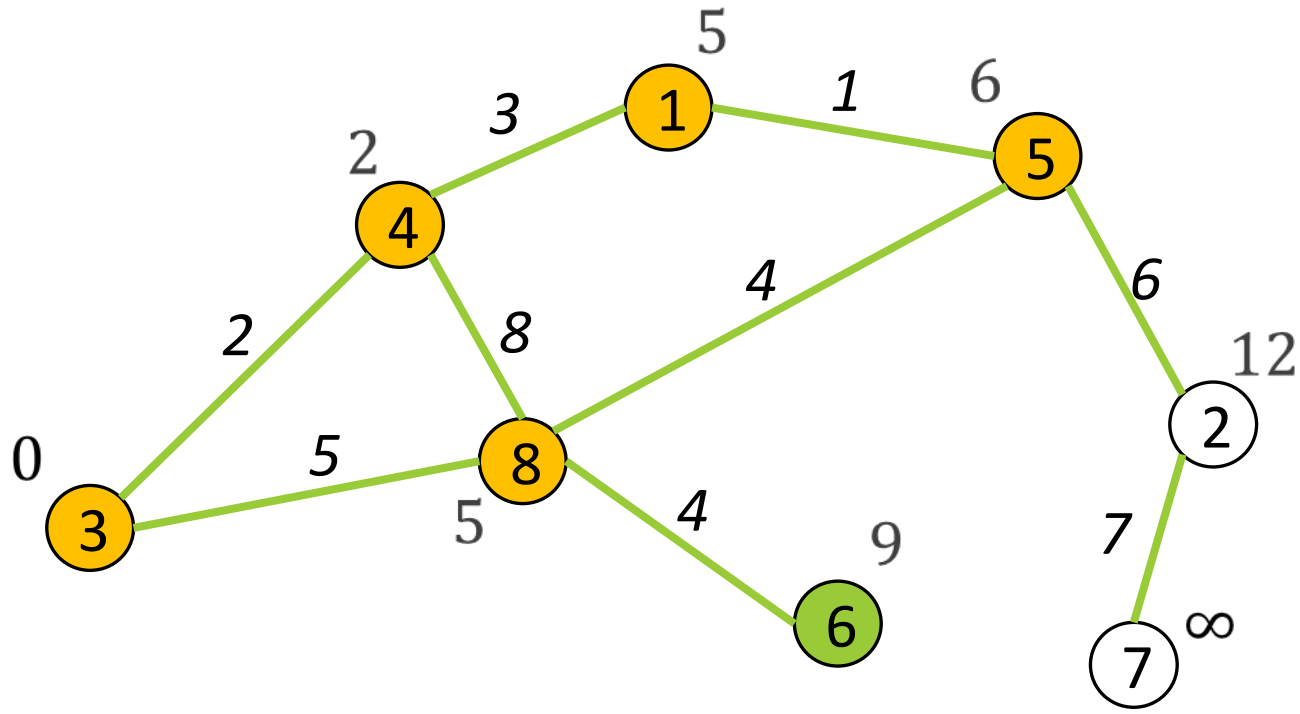
Пример



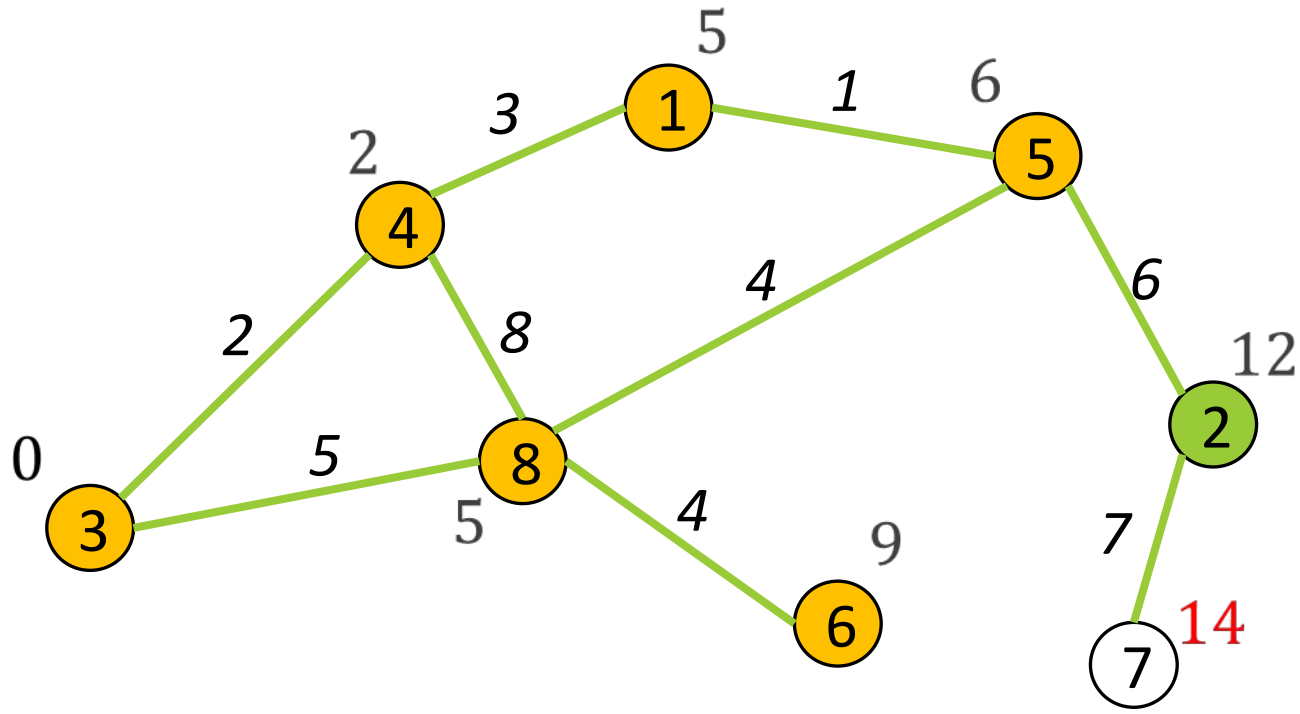
Пример



Пример



Пример



Пример