Модуль 5 «Алгоритмы на графах»

Расстояние в графе.

Расстоянием между вершинами ***a*** и ***b*** называется длина минимальной простой цепи, связывающей их.

Расстояние обозначается ***d(a, b)***.

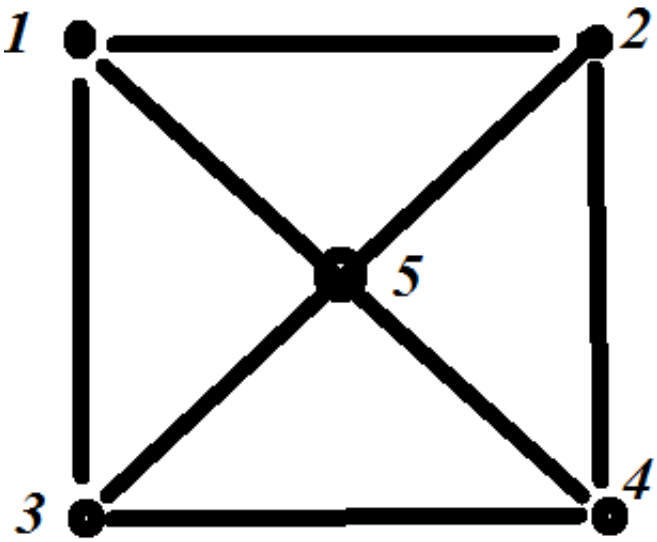
Аксиомы метрики:

1) ***d(a, b) = d(b, a);***

2) *d*(*a*, *b*) ≥ 0, *d*(*a*, *b*) = 0 ↔ *a* = *b*;

3) *d*(*a*, *b*) ≤ *d*(*a*, *c*) + *d*(*c*, *b*)

Пример



	1	2	3	4	5
1	0	1	1	2	1
2	1	0	2	1	1
3	1	2	0	1	1
4	2	1	1	0	1
5	1	1	1	1	0

ri – ***эксцентриситет*** *i*-ой вершины – расстояние от этой вершины до наиболее удаленной от нее вершины.

***ri* = max *d*(*vi*,*vj*)**

по всем *j* от 1 до *n*.

Диаметр графа ***G*** – максимальное расстояние между вершинами графа

***d*(*G*)= max *d*(*vi*,*vj*)**

по всем *i* и *j* от 1 до *n*, или

d*(*G*)=max *ri по всем *i* от 1 до *n*.

Центр графа ***G*** – это вершина, расстояние от которой до наиболее удаленной вершины – минимальное.

Чтобы найти центр, надо сначала найти радиус графа.

Радиус графа *G* – расстояние от центра графа до наиболее удаленной вершины.

r* (*G*) = min *ri

по всем *i* от 1 до *n*.

Центр графа *G* – такая вершина *vi*, для которой

***ri* = *r*(*G*).**

Замечание:

Центр в графе может быть не единственный.

В нашем

примере

центром
является
вершина 5.
Радиус – 1,
диаметр – 2.

	1	2	3	4	5	r_i
1	0	1	1	2	1	2
2	1	0	2	1	1	2
3	1	2	0	1	1	2
4	2	1	1	0	1	2
5	1	1	1	1	0	1

Диаметральные цепи графа ***G*** – простые цепи, длина которых равна ***d*(*G*)**, соединяющие наиболее удаленные вершины графа.

Связанные компоненты графа.

Пусть $G = (V, E)$ – н-граф.

связными называются вершины ***a*** и ***b*** если существуют маршрут, связывающий их.

Н-граф ***G*** называется ***связным,*** если все его вершины связны.

Утверждение: отношение связности является отношением эквивалентности.

Доказательство:

1. Каждая вершина связана сама с собой маршрутом нулевой длины, значит отношение связности ***рефлексивно.***

2. Если вершина ***a*** связна с ***b,*** то и ***b*** связна с ***a.*** Если ***a*** с ***b*** связаны маршрутом ***M(a,b),*** то ***b*** с ***a*** связаны маршрутом ***M(b,a),*** где ребра и вершины идут в обратном порядке. Значит отношение связности ***симметрично.***

3. Если вершина ***a*** связана маршрутом с ***b, b*** связана с ***c,*** то и ***a*** связана маршрутом с ***c.***

Это маршрут, начало которого ***M(a,b),*** окончание – ***M(b,c)***, вершина ***b*** – общая.

Значит отношение связности ***транзитивно.***

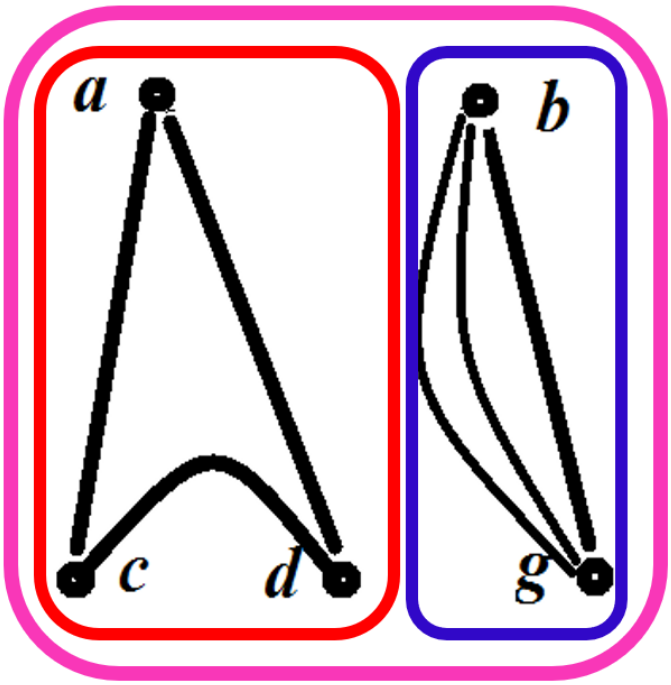
Отношение **рефлексивно, симметрично** и **транзитивно**, значит является отношением ***эквивалентности.***

Множество вершин ***V*** разбивается отношением связности на ***классы эквивалентности –*** подмножества связных вершин.

Связными компонентами графа ***G*** называются подграфы, порожденные ***классами эквивалентности*** по отношению связности.

Замечание: В связном графе одна связная компонента.

$$V=\{a,b,c,d,g\},$$



класс $V_1 = \{ a,c,d \}$

класс $V_2 = \{b,g\}.$

Связные компоненты

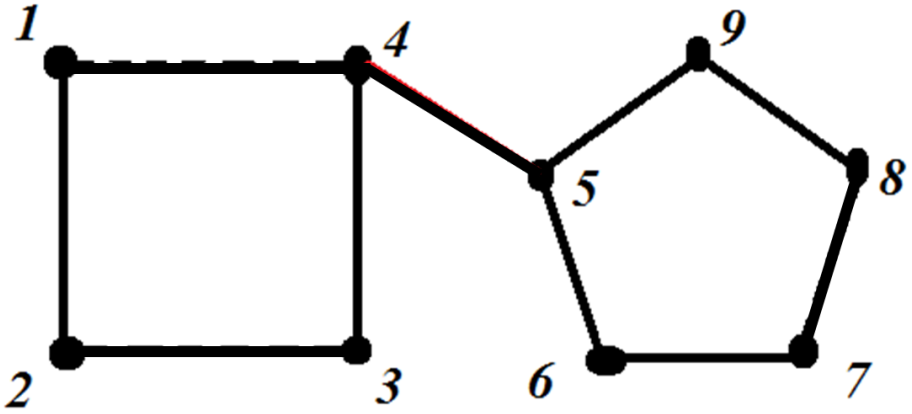
Разделяющим множеством

н-графа ***G* = (*V, E*)** называется множество ребер, при удалении которых число компонент связности графа увеличивается.

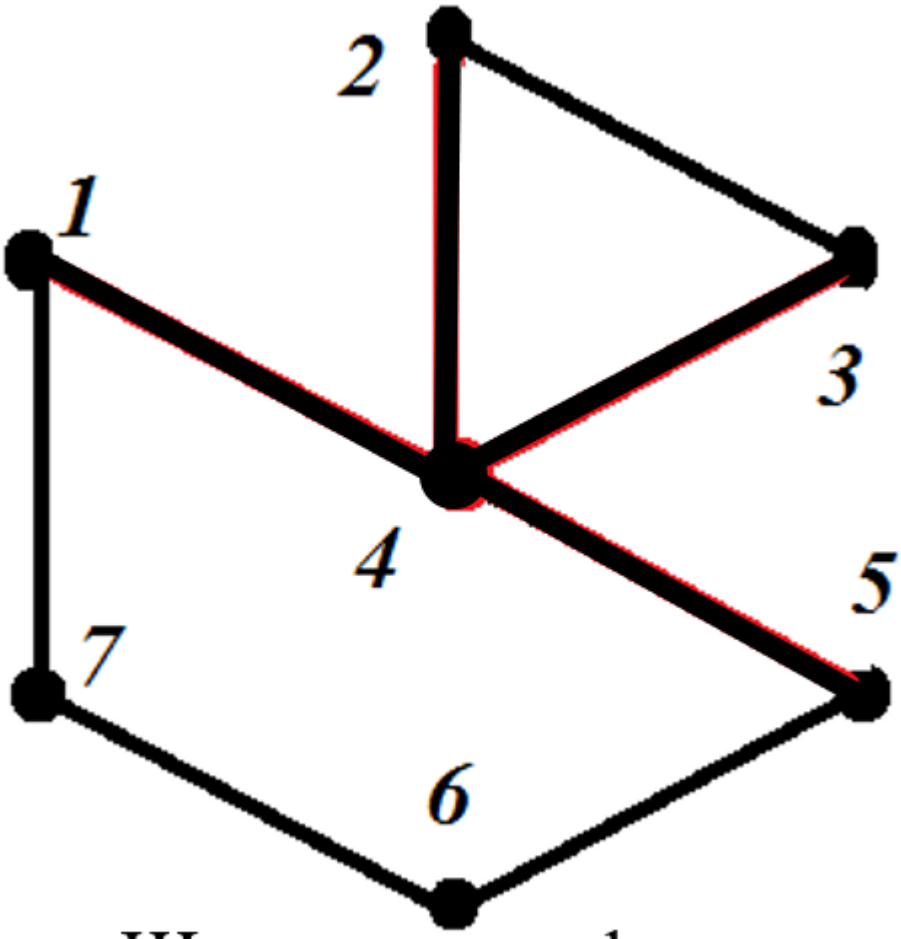
Разрезом в н-графе ***G* =(*V, E*)** называется разделяющее множество в котором нет лишних ребер, то есть ***минимальное*** разделяющее множество.

Мостом или ***перешейком***
в н-графе ***G* = (*V, E*)** называется разрез, состоящий из одного ребра.

***Разделяющее множество*:**
{(1,4), (2,3), (7,8)};
***разрез*:**{(1,4), (2,3)}, (7,8) – лишнее;
***мост* :**{(4,5) }.



Вершина v_0 н-графа ***G* = (*V, E*)** называется ***шарниром,*** если удаление этой вершины и всех инцидентных ей ребер приводит к увеличению числа компонент связности графа.



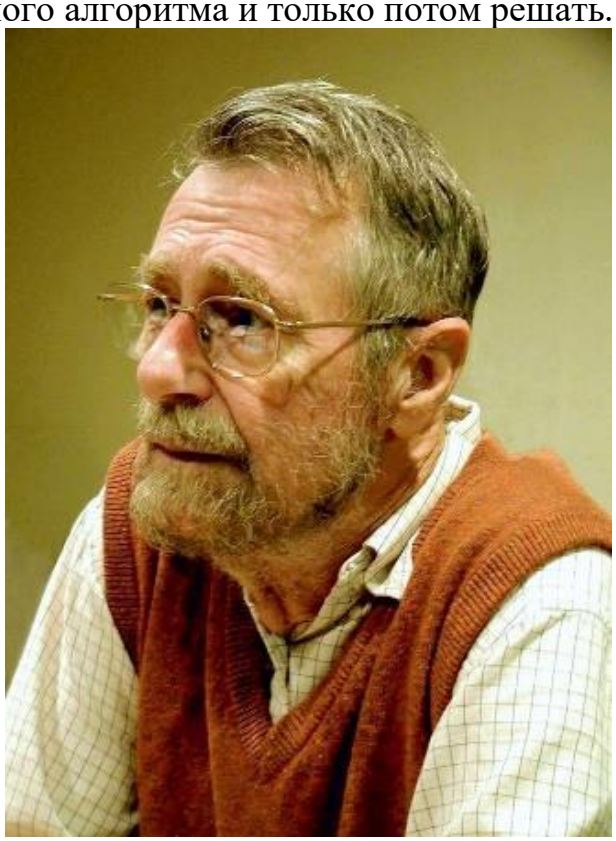
Шарнир в графе

Шарниром является вершина 4.

Эдсгер Дейкстра (Edsger Wybe Dijkstra) – нидерландский ученый (структурное программирование, язык Алгол, семафоры, распределенные вычисления)

Лауреат премии Тьюринга (ACM A.M. Turing Award)

Те, кто начинает в IT, часто интересуются: «Нужна ли математика программисту?» Им обычно отвечают, что да, но не везде, не всегда и вообще можно без неё. А вот Эдсгер Дейкстра считал, что любую программистскую задачу нужно математически описать, доказать правильность выбранного алгоритма и только потом решать.



Алгоритм основан на приписывании вершинам v_i временных меток $d(v_i)$. Метка вершины дает верхнюю границу длины пути от s к этой вершине.

Шаг 1 (начальная установка).

Процесс построения дерева начинается с заданной вершины.

Положить d(s)=0, считать метку постоянной.

$d(v_i)=\infty$, i=1...n, считать метки временными. ***p=s***.

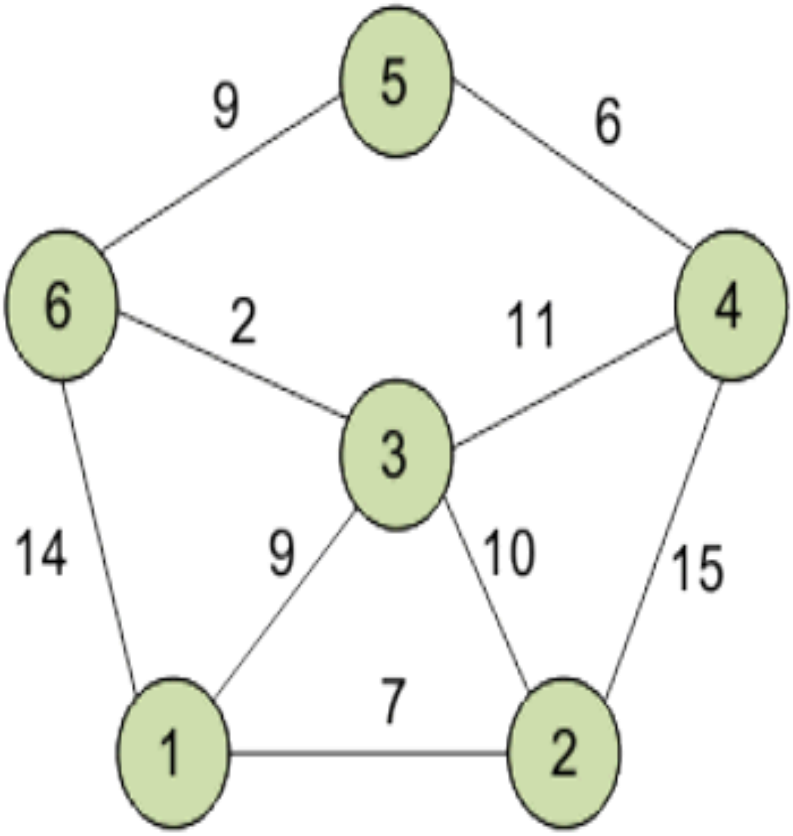
Шаг 2 (общий шаг).

В дальнейшем на каждом шаге к дереву присоединяется одно новое ребро (и одна вершина). Это ребро выбирается из подходящих ребер, причем подходящим считается ребро, соединяющее вершину дерева с вершиной, ему не принадлежащей. Среди подходящих ребер выбирается ребро наименьшего веса.

Повторяется *n* раз, пока не будут упорядочены все вершины.

Рассмотрим пример нахождение кратчайшего пути. Дана сеть автомобильных дорог, соединяющих области города. Некоторые дороги односторонние. **Найти кратчайшие пути от центра города до каждого города области**.

Для решения указанной задачи можно использовать *алгоритм Дейкстры* — алгоритм на графах, изобретённый нидерландским ученым Э. Дейкстрой в 1959 году. Находит кратчайшее расстояние от одной из вершин графа до всех остальных. Работает только для графов без рёбер отрицательного веса.

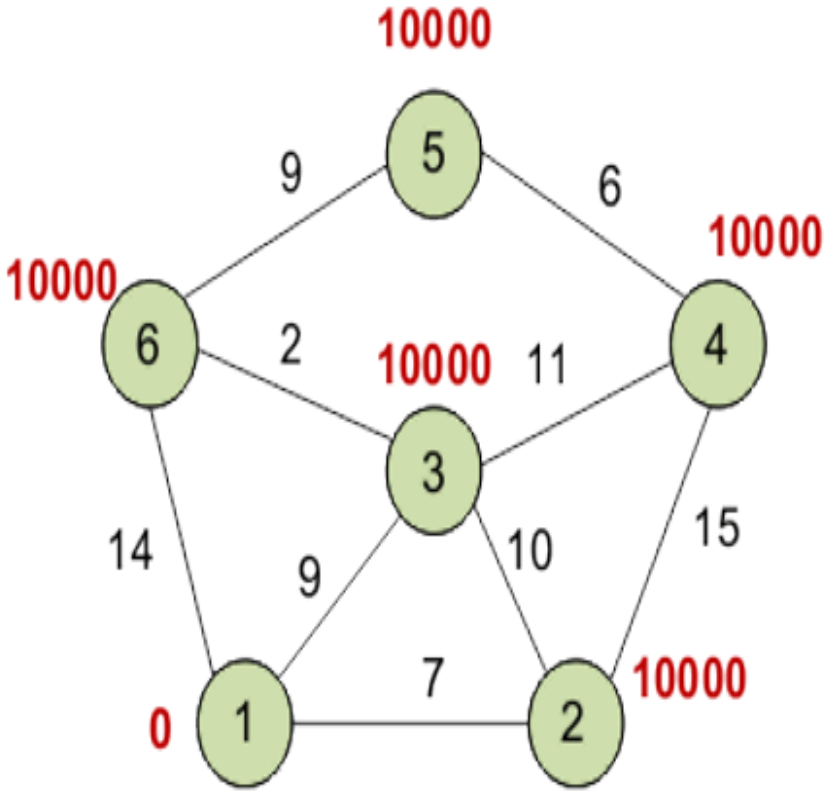


Пусть требуется найти кратчайшие расстояния от 1-й вершины до всех остальных.

Кружками обозначены вершины, линиями – пути между ними (ребра графа). В кружках обозначены номера вершин, над ребрами обозначен их вес – длина пути. Рядом с каждой вершиной красным обозначена метка – длина кратчайшего пути в эту вершину из вершины 1.

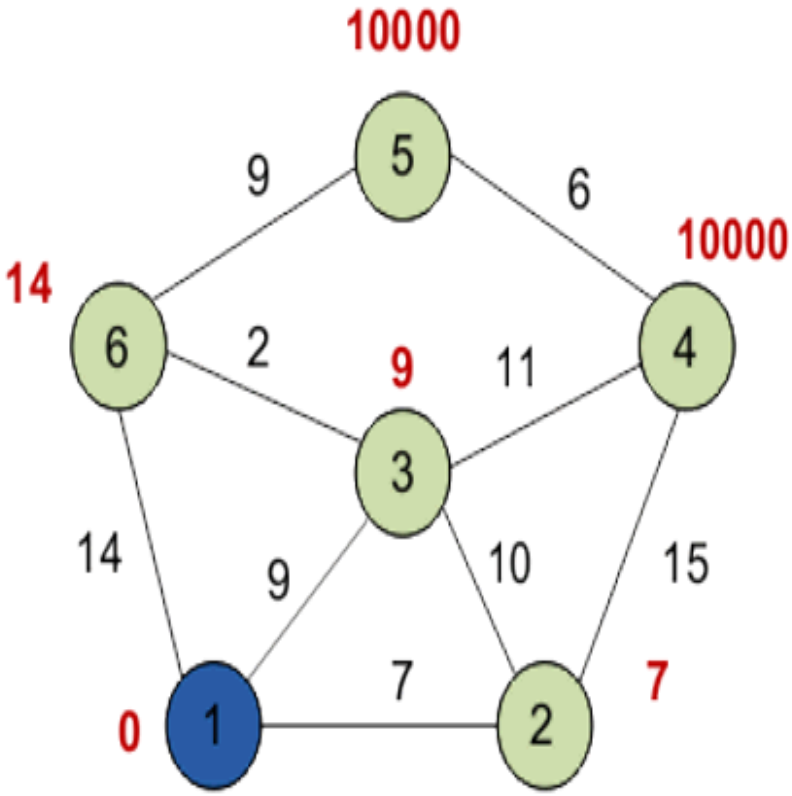
Метка самой вершины 1 полагается равной 0, метки остальных вершин – недостижимо большое число (в идеале — бесконечность).

Это отражает то, что расстояния от вершины 1 до других вершин пока неизвестны. Все вершины графа помечаются как **непосещенные**.



Минимальную метку имеет вершина 1. Её соседями являются вершины 2, 3 и 6. Обходим соседей вершины по очереди.

Первый сосед вершины 1 – вершина 2, потому что длина пути до неё минимальна. Длина пути в неё через вершину 1 равна сумме кратчайшего расстояния до вершины 1, значению её метки, и длины ребра, идущего из 1-й в 2-ю, то есть 0 + 7 = 7. Это меньше текущей метки вершины 2 (10000), поэтому новая метка 2-й вершины равна 7.

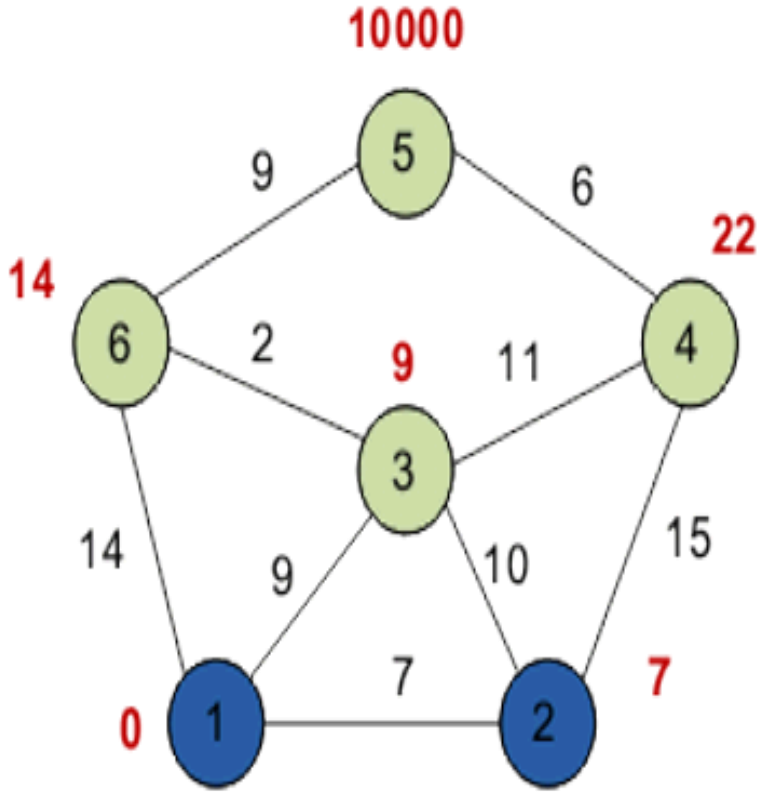


Аналогично находим длины пути для всех других соседей (вершины 3 и 6).

Все соседи вершины 1 проверены. Текущее минимальное расстояние до вершины 1 считается окончательным и пересмотру не подлежит. Вершина 1 отмечается как посещенная.

Шаг 1 алгоритма повторяется. Снова находим «ближайшую» из непосещенных вершин. Это вершина 2 с меткой 7.

Снова пытаемся уменьшить метки соседей выбранной вершины, пытаясь пройти в них через 2-ю вершину. Соседями вершины 2 являются вершины 1, 3 и 4.

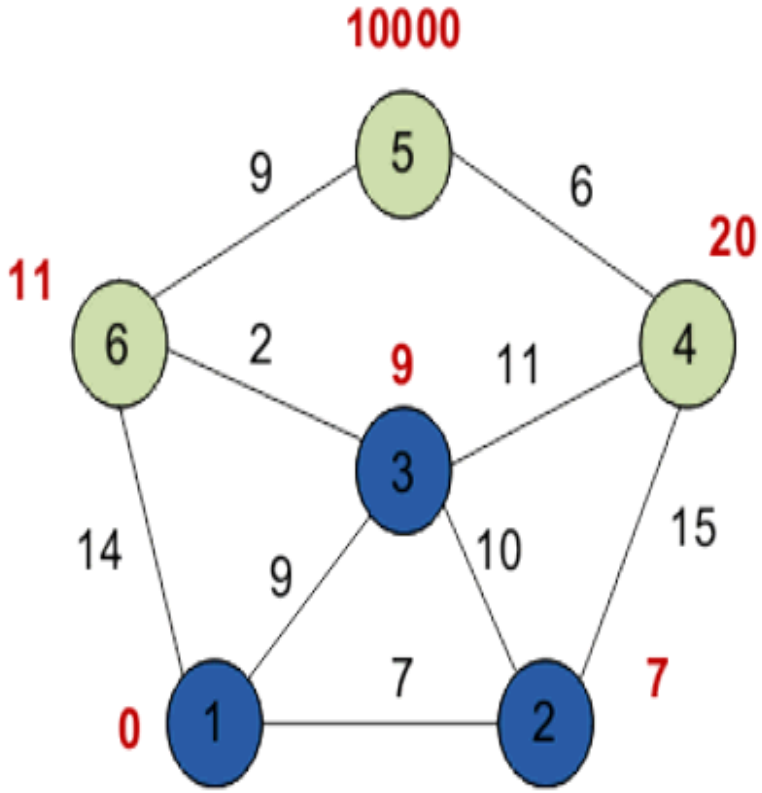


Вершина 1 уже посещена. Следующий сосед вершины 2 — вершина 3, так как имеет минимальную метку из вершин, отмеченных как не посещённые. Если идти в неё через 2, то длина такого пути будет равна 17 (7 + 10 = 17). Но текущая метка третьей вершины равна 9, а 9 < 17, поэтому метка не меняется.

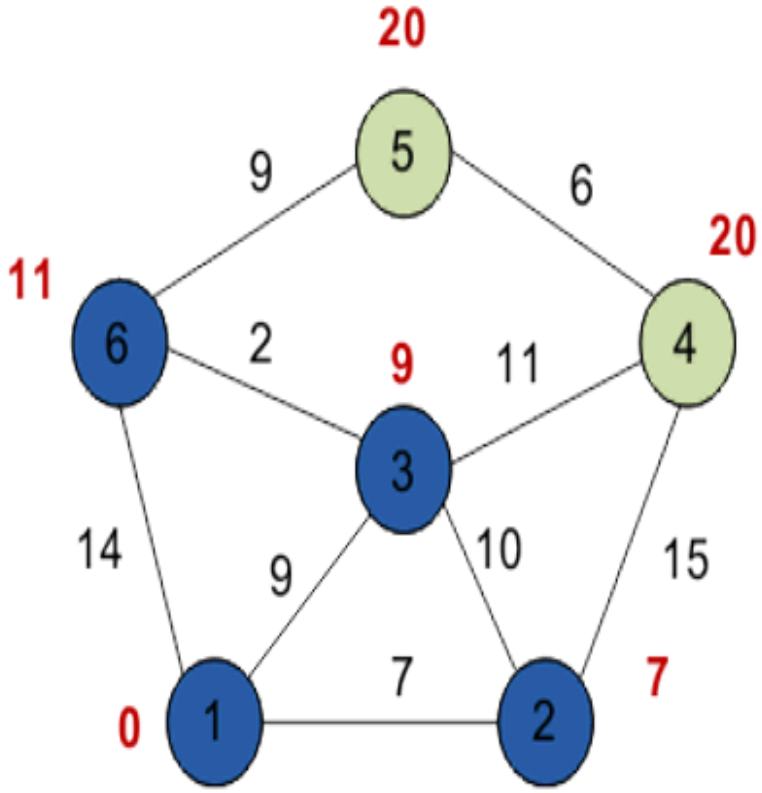
Ещё один сосед вершины 2 — вершина 4. Если идти в неё через 2-ю, то длина такого пути будет равна 22 (7 + 15 = 22). Поскольку 22<10000, устанавливаем метку вершины 4 равной 22.

Все соседи вершины 2 просмотрены, помечаем её как посещенную.

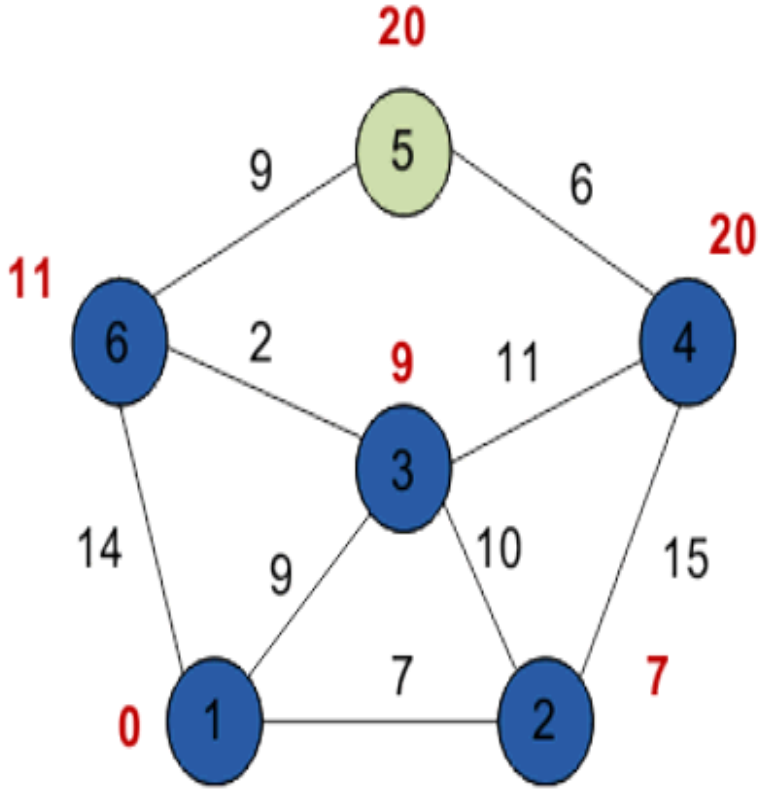
Повторяем шаг алгоритма, выбрав вершину 3. После её «обработки» получим следующие результаты.



Четвертый шаг:

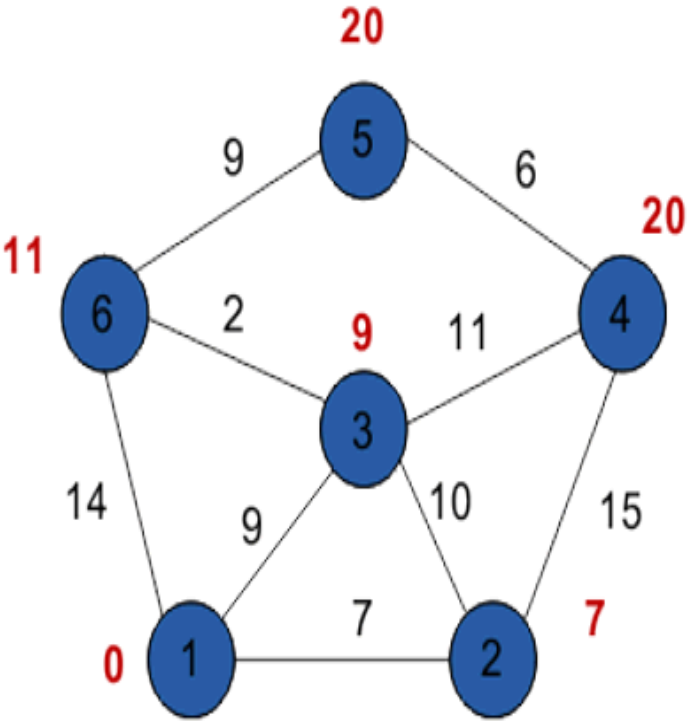


Пятый шаг

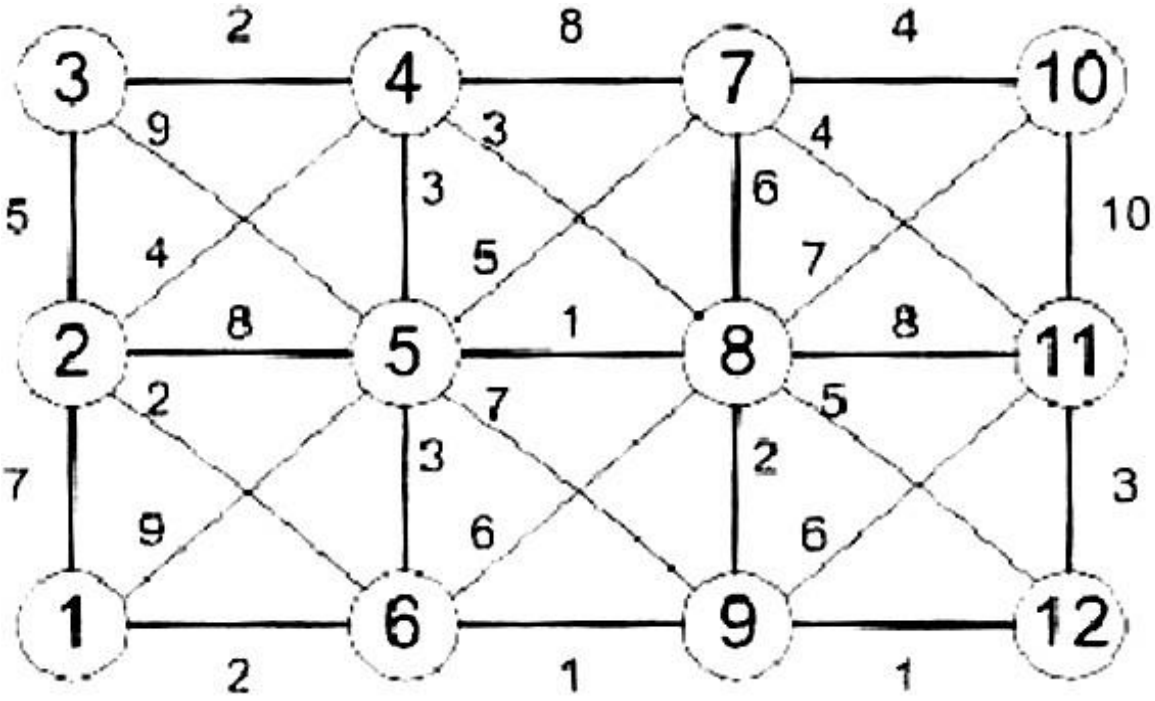


Шестой шаг

Таким образом, **кратчайшим путем из вершины 1 в вершину 5** будет путь через вершины **1 — 3 — 6 — 5**, поскольку таким путем мы набираем минимальный вес, равный 20.



Пример:



Выполним шаг 1 и заполним первую строку таблицы.

вершины	1	2	3	4	5	6	7	8	9	10	11	12	
шаг 1	0+	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	p = 1

Из вершины 1 выходят дуги в вершины 2, 5, 6. Вычисляем метки этих вершин.

d(2) = min (∞; 0 + 7) = 7

d(5) = min (∞; 0 + 9) = 9

d(6) = min (∞; 0 + 2) = 2

вершины	1	2	3	4	5	6	7	8	9	10	11	12	
шаг 1	0+	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	p = 1
шаг 2		7	∞	∞	9	2+	∞	∞	∞	∞	∞	∞	p = 6

Из найденных значений наименьшее – для вершины 6 (2). Помечаем эту вершину и снова пересчитываем метки вершин, в которые можно перейти из вершины 6:

d(2) = min (7; 2 + 2) = 4; d(5) = min (9; 2 + 3) = 5;

d(8) = min (∞; 2 + 6) = 8; d(9) = min (9; 2 + 1) = 3

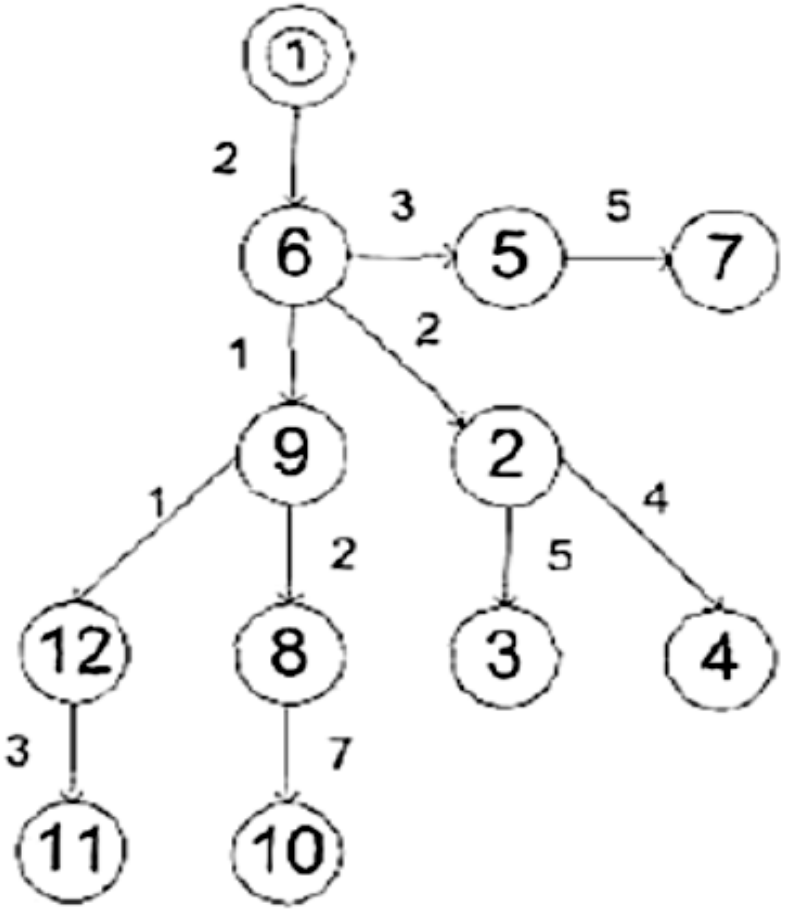
вершины	1	2	3	4	5	6	7	8	9	10	11	12	
шаг 1	0+	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	p = 1
шаг 2		7	∞	∞	9	2+	∞	∞	∞	∞	∞	∞	p = 6
шаг 3		4	∞	∞	5		∞	8	3+	∞	∞	∞	p = 9

Метка вершины 9 становится постоянной. Пересчитываем метки вершин, в которые можно перейти из вершины 9. И так далее заполняем остальные строки таблицы.

вершины	1	2	3	4	5	6	7	8	9	10	11	12	
шаг 1	**0+**	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	∞	p = 1
шаг 2		7	∞	∞	9	**2+**	∞	∞	∞	∞	∞	∞	p = 6
шаг 3		4	∞	∞	5		∞	8	**3+**	∞	∞	∞	p = 9
шаг 4		**4+**	∞	∞	5		∞	5		∞	9	4	p = 2
шаг 5			9	8	5		∞	5		∞	9	**4+**	p = 12
шаг 6			9	8	**5+**		∞	5		∞	7		p = 5
шаг 7			9	8			10	**5+**		∞	7		p = 8
шаг 8			9	8			10			12	**7+**		p = 11
шаг 9			9	**8+**			10			12			p = 4
шаг 10			**9+**				10			12			p = 3
шаг 11							**10+**			12			p = 7
шаг 12										**12+**			p = 10

Дерево кратчайших путей – это ориентированное дерево с корнем в вершине S. Все пути в этом дереве – кратчайшие для данного графа.

Строится по таблице, в него включаются вершины в том порядке, в котором они получали постоянные метки



Алгоритм Флойда-Уоршалла.

Находит кратчайшие расстояния между всеми парами вершин графа
Идея: строится специальная матрица *D*, элементы которой – кратчайшие пути между всевозможными парами вершин графа *G*. При определении кратчайшего пути выбирается минимум из «прямого» расстояния между смежными вершинами v_i и v_j и суммарного расстояния при проходе через дополнительную вершину. Затем – более длинные пути и т.д.

Обозначим через $d_{ij}^{(m)}$ длину кратчайшего пути из v_i в v_j с промежуточными вершинами во множестве $\{v_1,\ldots,v_m\}$.

Алгоритм использует три правила:

1) $d_{ij}^{(0)} = a_{ij}$ - вес дуги, соединяющей вершины v_i и v_j (т.е. первоначально матрица D – это исходная матрица весов).

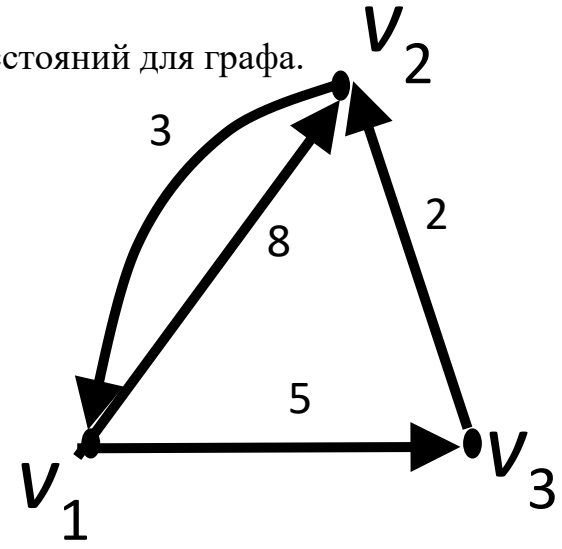
2) $$d_{ij}^{(m+1)} = \min\left(d_{ij}^{(m)}, d_{i,m+1}^{(m)} + d_{m+1,j}^{(m)}\right)$$

3) Длина кратчайшего пути из вершины v_i в вершину v_j:

$$d(v_i, v_j) = d_{ij}^{(n)}$$

Алгоритм строит матрицу за *n* шагов, т.е. строится матрица $D^{(1)}$, …, $D^{(n)}$ =D.

Пример. Найдем матрицу кратчайших расстояний для графа.



$$D^{(0)} = \begin{array}{c} \begin{matrix} v_1 & v_2 & v_3 \end{matrix} \\ \begin{pmatrix} 0 & 8 & 5 \\ 3 & 0 & \infty \\ \infty & 2 & 0 \end{pmatrix} \begin{matrix} v_1 \\ v_2 \\ v_3 \end{matrix} \end{array}$$

Элементы матрицы $D^{(1)}$ находим по правилу:

$$d_{ij}^{(1)} = \min\left(d_{ij}^{(0)}, d_{i1}^{(0)} + d_{1j}^{(0)}\right)$$

$$d_{11}^{(1)} = \min\left(d_{11}^{(0)}, d_{11}^{(0)} + d_{11}^{(0)}\right) = \min(0,0) = 0$$

$$d_{12}^{(1)} = \min\left(d_{12}^{(0)}, d_{11}^{(0)} + d_{12}^{(0)}\right) = \min(8,8) = 8$$

$$d_{13}^{(1)} = \min\left(d_{13}^{(0)}, d_{11}^{(0)} + d_{13}^{(0)}\right) = \min(5,5) = 5$$

$$d_{21}^{(1)} = \min\left(d_{21}^{(0)}, d_{21}^{(0)} + d_{11}^{(0)}\right) = \min(3,3+0) = 3$$

$$d_{22}^{(1)} = \min\left(d_{22}^{(0)}, d_{21}^{(0)} + d_{12}^{(0)}\right) = \min(0,3+8) = 0$$

$$d_{23}^{(1)} = \min\left(d_{23}^{(0)}, d_{21}^{(0)} + d_{13}^{(0)}\right) = \min(\infty,3+5) = 8$$

$$d_{31}^{(1)} = \min\left(d_{31}^{(0)}, d_{31}^{(0)} + d_{11}^{(0)}\right) = \min(\infty, \infty + 0) = \infty$$
$$d_{32}^{(1)} = \min\left(d_{32}^{(0)}, d_{31}^{(0)} + d_{12}^{(0)}\right) = \min(2, \infty + 8) = 2$$
$$d_{33}^{(1)} = \min\left(d_{33}^{(0)}, d_{31}^{(0)} + d_{13}^{(0)}\right) = \min(0, \infty + 5) = 0$$

$$D^{(1)} = \begin{pmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ \infty & 2 & 0 \end{pmatrix}$$

Элементы матрицы $D^{(2)}$ находим по правилу:

$$d_{i\,j}^{(2)} = \min\left(d_{i\,j}^{(1)}, d_{i\,2}^{(1)} + d_{2\,j}^{(1)}\right)$$

$$d_{11}^{(2)} = \mathbf{min}\left(d_{11}^{(1)}, d_{12}^{(1)} + d_{21}^{(1)}\right) = \mathbf{min}(0, 8+3) = 0$$
$$d_{12}^{(2)} = \mathbf{min}\left(d_{12}^{(1)}, d_{12}^{(1)} + d_{22}^{(1)}\right) = \mathbf{min}(8, 8+0) = 8$$
$$d_{13}^{(2)} = \mathbf{min}\left(d_{13}^{(1)}, d_{12}^{(1)} + d_{23}^{(1)}\right) = \mathbf{min}(5, 8+8) = 5$$
$$d_{21}^{(2)} = \mathbf{min}\left(d_{21}^{(1)}, d_{22}^{(1)} + d_{21}^{(1)}\right) = \mathbf{min}(3, 0+3) = 3$$
$$d_{22}^{(2)} = \mathbf{min}\left(d_{22}^{(1)}, d_{22}^{(1)} + d_{22}^{(1)}\right) = \mathbf{min}(0, 0+0) = 0$$
$$d_{23}^{(2)} = \mathbf{min}\left(d_{23}^{(1)}, d_{22}^{(1)} + d_{23}^{(1)}\right) = \mathbf{min}(8, 0+8) = 8$$
$$d_{31}^{(2)} = \mathbf{min}\left(d_{31}^{(1)}, d_{32}^{(1)} + d_{21}^{(1)}\right) = \mathbf{min}(\infty, 2+3) = 5$$
$$d_{32}^{(2)} = \mathbf{min}\left(d_{32}^{(1)}, d_{32}^{(1)} + d_{22}^{(1)}\right) = \mathbf{min}(2, 2+0) = 2$$
$$d_{33}^{(2)} = \mathbf{min}\left(d_{33}^{(1)}, d_{32}^{(1)} + d_{23}^{(1)}\right) = \mathbf{min}(0, 2+8) = 0$$

$$D^{(2)} = \begin{pmatrix} 0 & 8 & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{pmatrix}$$

Элементы матрицы $D^{(3)}$ находим по правилу:

$$d_{ij}^{(3)} = \min\left(d_{ij}^{(2)}, d_{i3}^{(2)} + d_{3j}^{(2)}\right)$$

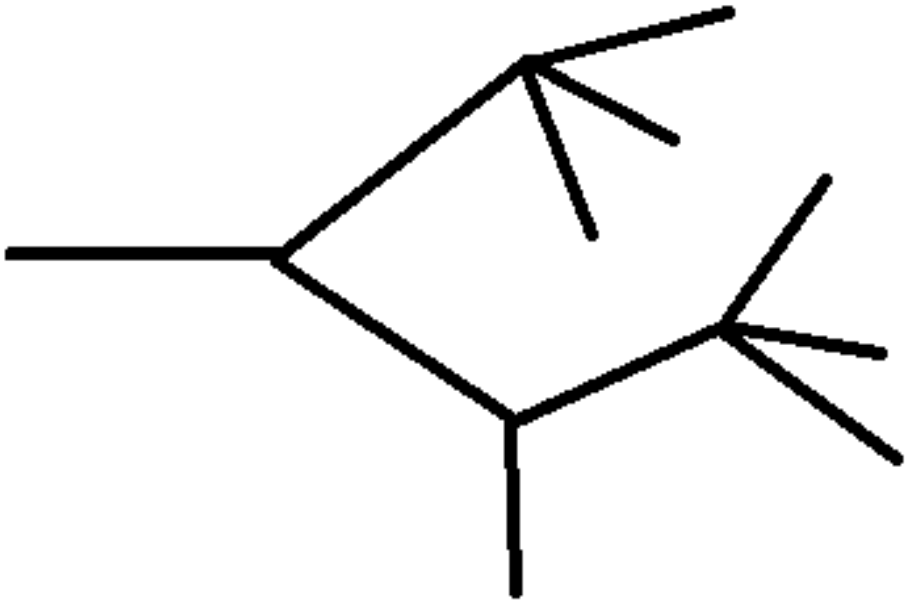
$$d_{11}^{(3)} = \mathbf{min}\left(d_{11}^{(2)}, d_{13}^{(2)} + d_{31}^{(2)}\right) = \mathbf{min}(0,5+5) = 0$$
$$d_{12}^{(3)} = \mathbf{min}\left(d_{12}^{(2)}, d_{13}^{(2)} + d_{32}^{(2)}\right) = \mathbf{min}(8,5+2) = 7$$
$$d_{13}^{(3)} = \mathbf{min}\left(d_{13}^{(2)}, d_{13}^{(2)} + d_{33}^{(2)}\right) = \mathbf{min}(5,5+0) = 5$$
$$d_{21}^{(3)} = \mathbf{min}\left(d_{21}^{(2)}, d_{23}^{(2)} + d_{31}^{(2)}\right) = \mathbf{min}(3,8+5) = 3$$
$$d_{22}^{(3)} = \mathbf{min}\left(d_{22}^{(2)}, d_{23}^{(2)} + d_{32}^{(2)}\right) = \mathbf{min}(0,8+2) = 0$$
$$d_{23}^{(3)} = \mathbf{min}\left(d_{23}^{(2)}, d_{23}^{(2)} + d_{33}^{(2)}\right) = \mathbf{min}(8,8+0) = 8$$
$$d_{31}^{(3)} = \mathbf{min}\left(d_{31}^{(2)}, d_{33}^{(2)} + d_{31}^{(2)}\right) = \mathbf{min}(5,0+5) = 5$$
$$d_{32}^{(3)} = \mathbf{min}\left(d_{32}^{(2)}, d_{33}^{(2)} + d_{32}^{(2)}\right) = \mathbf{min}(2,2+0) = 2$$
$$d_{33}^{(3)} = \mathbf{min}\left(d_{33}^{(2)}, d_{33}^{(2)} + d_{33}^{(2)}\right) = \mathbf{min}(0,0+0) = 0$$

$$D^{(3)} = \begin{pmatrix} 0 & 7 & 5 \\ 3 & 0 & 8 \\ 5 & 2 & 0 \end{pmatrix}, \quad D = D^{(3)}$$

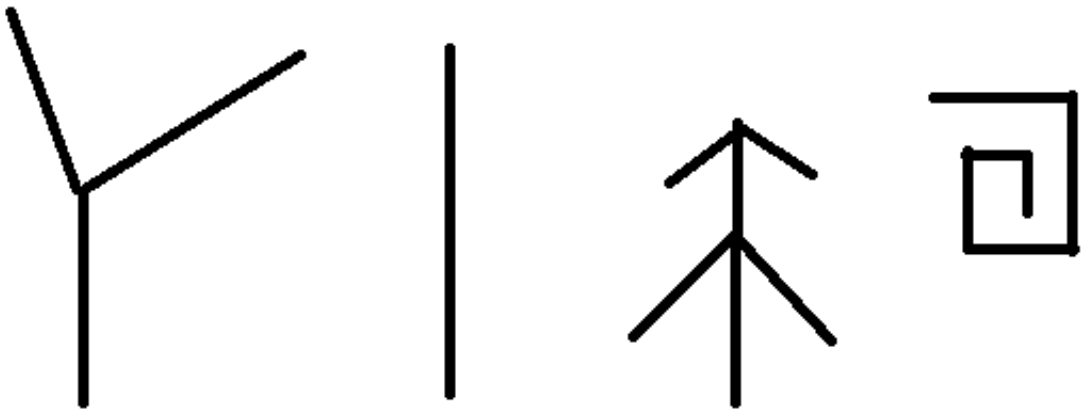
Деревья.

Пусть $G = (V, E)$ – н-граф.

Деревом называется связный ациклический граф.



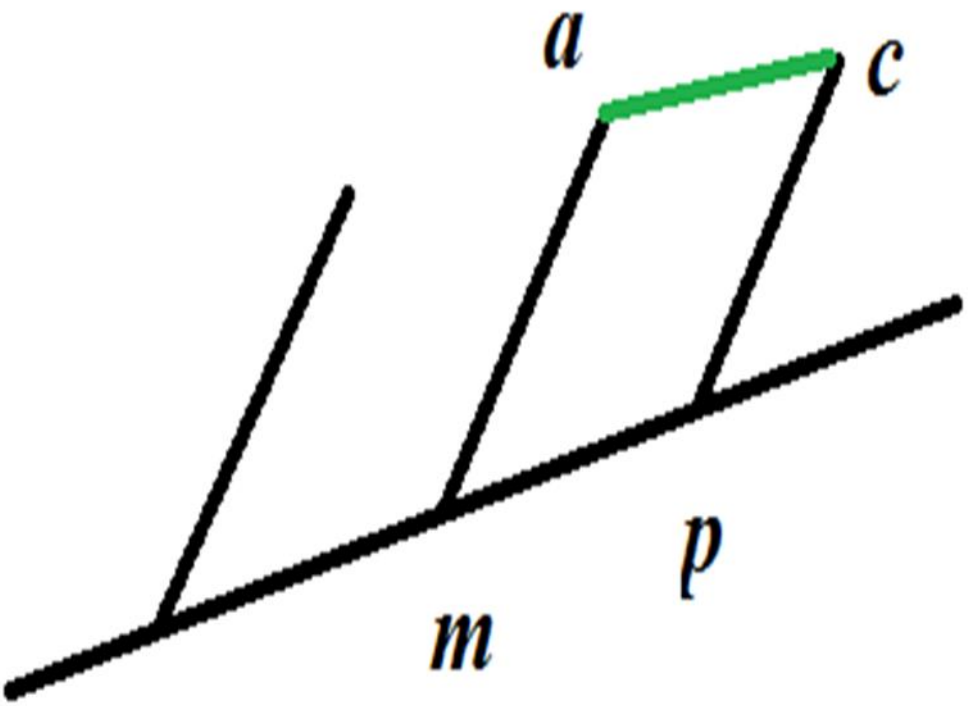
Лесом называется несвязный ациклический граф.



Теорема 1 (о деревьях)

Граф будет дерево тогда и только тогда, когда любые две его вершины связаны единственной простой цепью.

Связность дает наличие такой цепи, ацикличность – ее единственность.

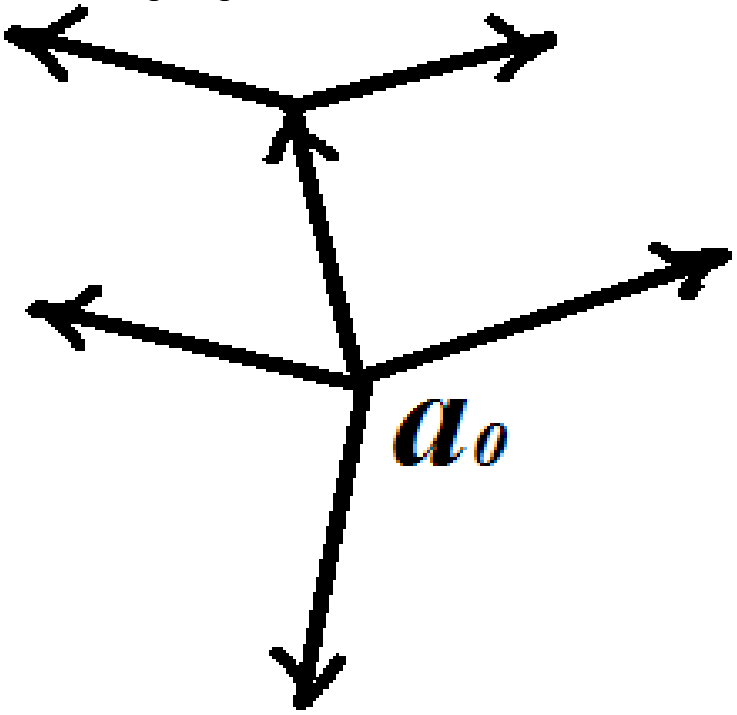


Терема 2 (о деревьях)

Граф с ***n*** вершинами будет деревом тогда и только тогда, в нем ровно ***n*-1** ребро.

Если ориентировать дерево о выбранной вершины, то в каждую вершину будет входить 1 ребро,

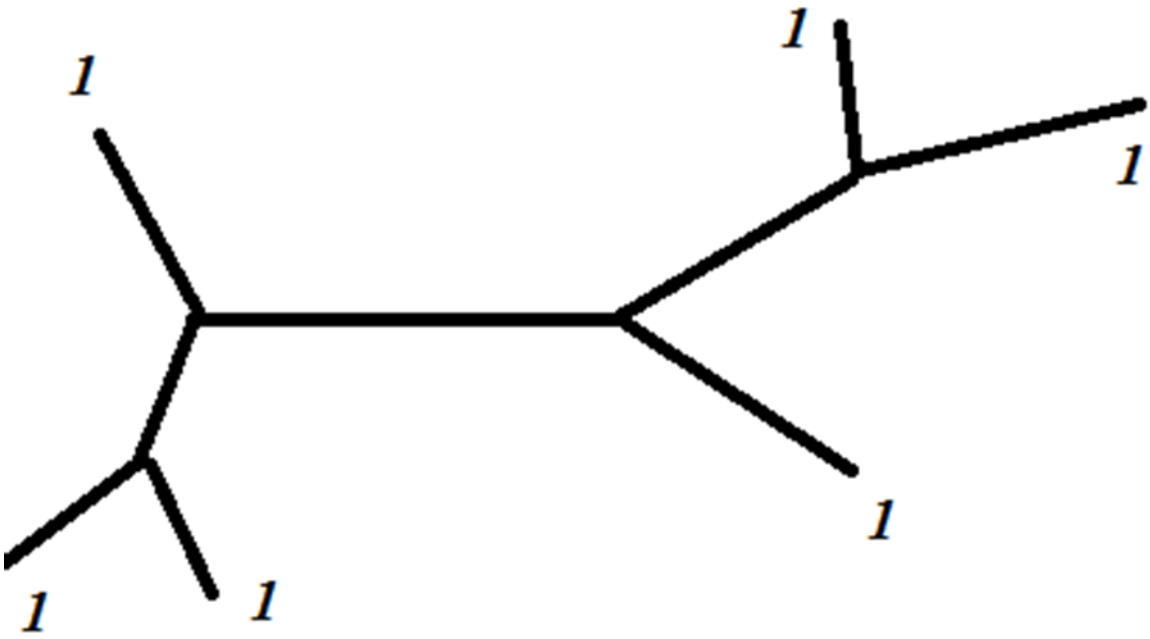
а в a_0 – 0 ребер.



Дано неориентированное дерево *T*.

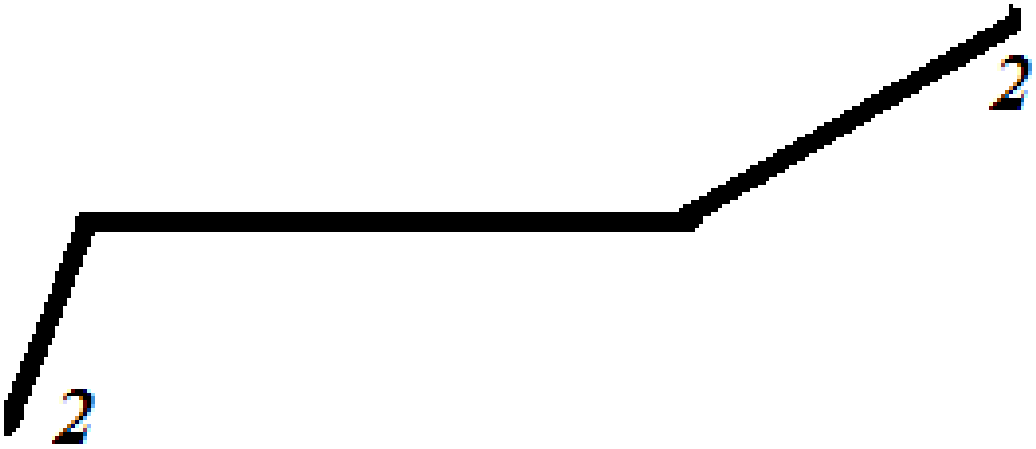
Концевые вершины дерева – вершины, локальная степень которых равна 1.

Назовем их вершинами первого типа дерева *T*.



Удалим из дерева Т ребра, инцидентные концевым вершинам – концевые ребра. Получим дерево Т1.

Концевые вершины дерева Т1 – Вершины типа 2.



Удалим из дерева Т1 концевые ребра. Получим дерево Т2.

Концевые вершины дерева Т2 – Вершины типа 3.

Утверждение 1:

В конечном дереве есть вершины только конечного числа типов.

Утверждение 2:

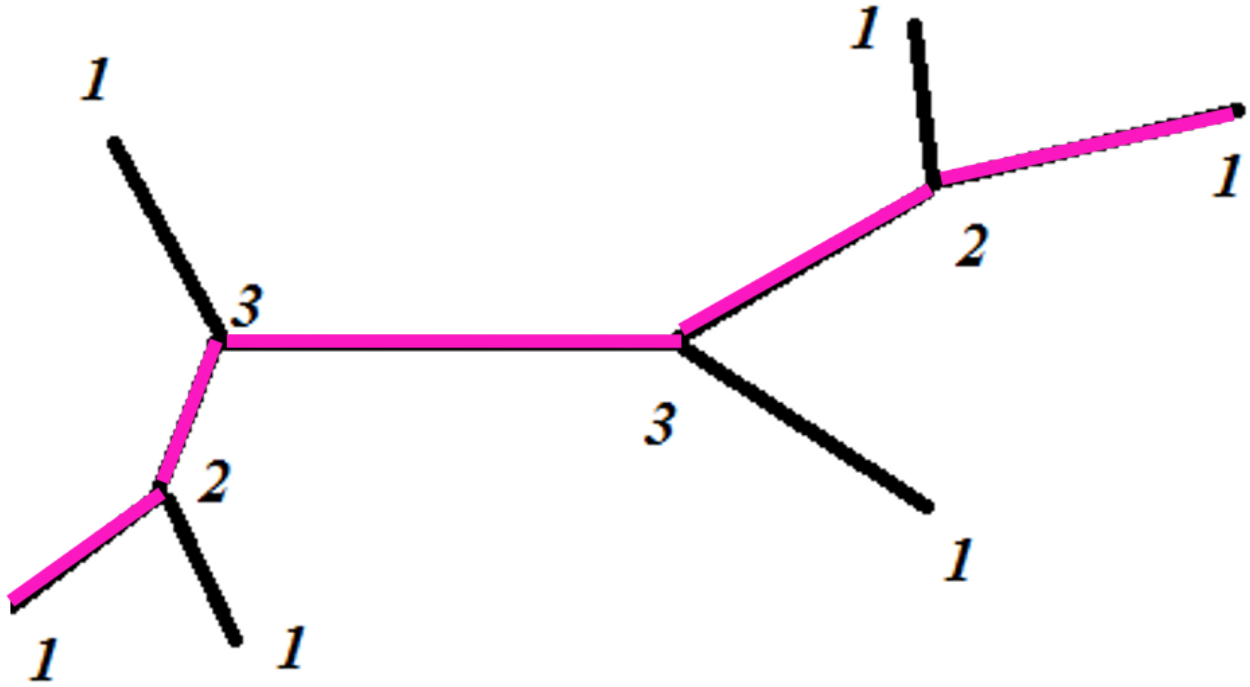
Вершин максимального типа ***k*** одна или две.

Утверждение 3:

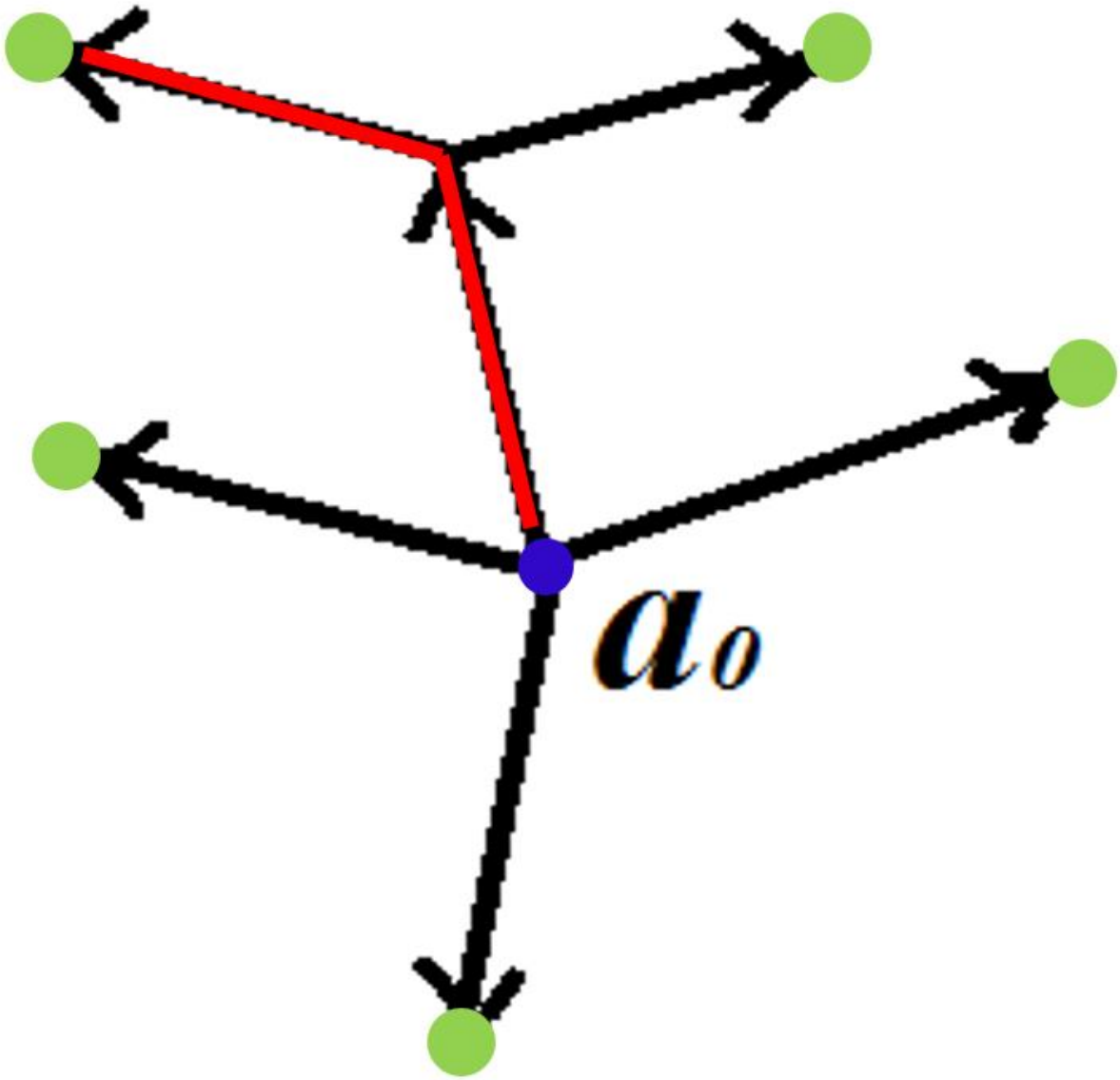
Центрами деревьев являются вершины максимального типа ***k*** и только они. Все диаметральные цепи проходят через центры.

Длина диаметральной цепи равна

2*k*-1, если центра два и **2*k*-2**, если центр один.
***k* = 3** , центров два, длина диаметральной цепи **2*k*-1=5.**



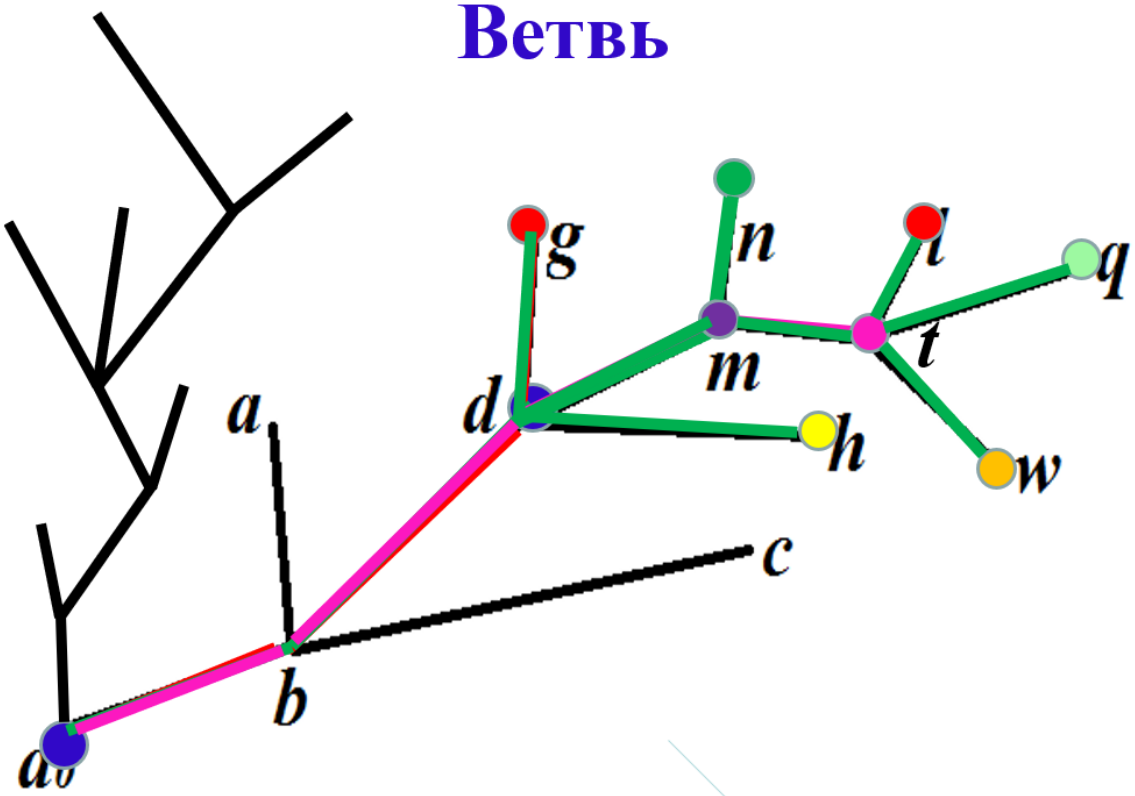
Если дерево не ориентировано, то его можно ориентировать ***от корня***.
Корень – это один из центров дерева.
У всех вершин дерева локальные степени захода равны 1, а у корня – 0.
Вершины, степени исхода которых равны 0 называются ***листьями.***
Высотой дерева называется наибольшее расстояние от корня до листа.



Бинарным деревом называется ориентированное дерево с корнем, где каждая вершина имеет локальную степень исхода, равную 2.

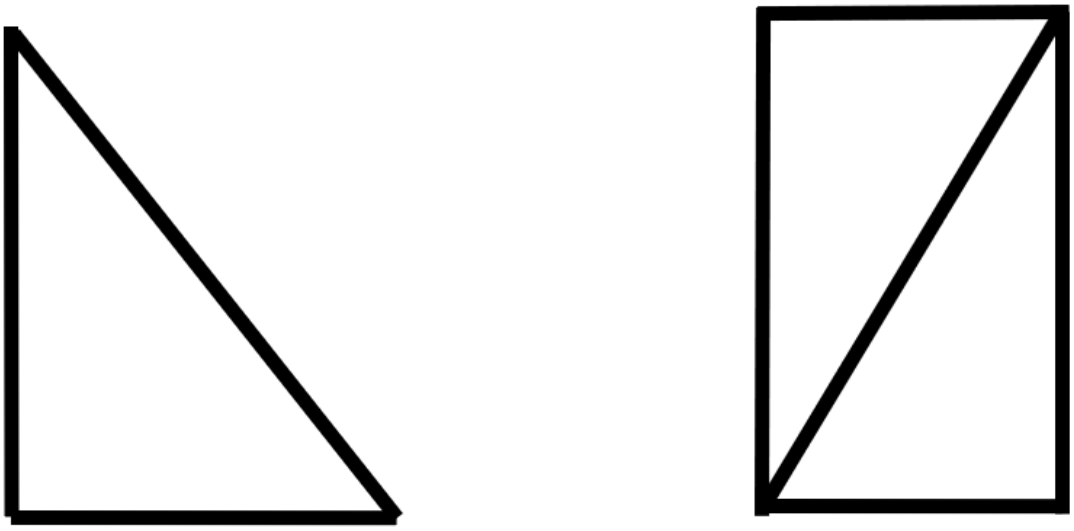
Ветвью вершины a в дереве T с корнем ***a0*** называется подграф, порожденный множеством вершин ***B(a)***, состоящим из вершин, связанных с корнем цепь, проходящей через ***a***.

Ветвь



Цикломатическое число.

Цикломатическим числом графа ***G*** называется число ребер, которые надо убрать, что бы граф стал ацикличным.



1.1 1.2

Цикломатическое число

Цикломатическое число графа ***G*** находится по формуле:

$$\gamma(G) = \mu_e - \mu_v + \mu_c$$

μ_e – число ребер,

μ_v – число вершин,

μ_c – число компонент связности.

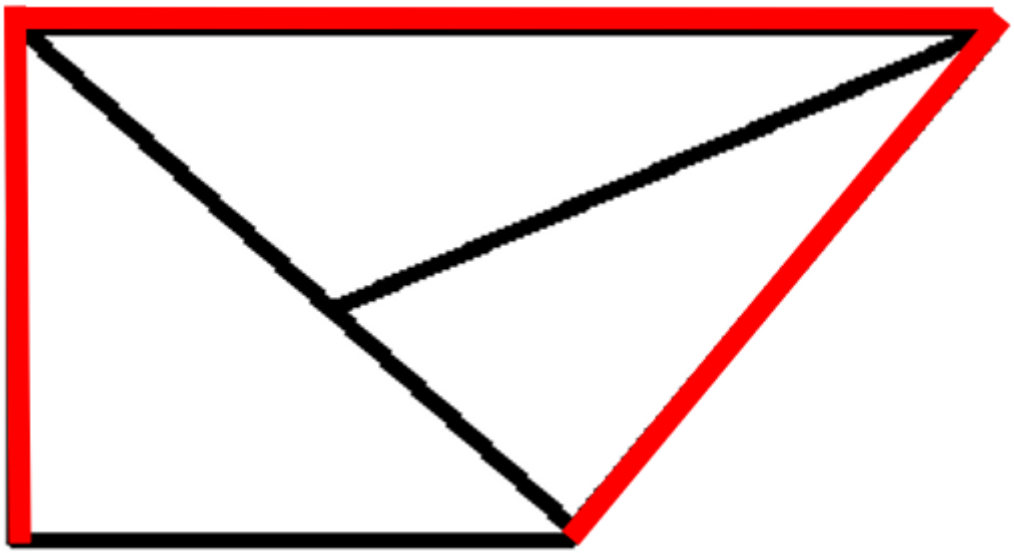
Замечание 1: Цикломатическое число ***дерева равно 0.***

Замечание 2: Цикломатическое число ***леса равно 0.***

Замечание 3: Если цикломатическое число графа равно 1, то в графе ровно 1 цикл.

Пример 1:

Нахождение цикломатического числа



$$\gamma(G) = \mu_e - \mu_v + \mu_c =$$

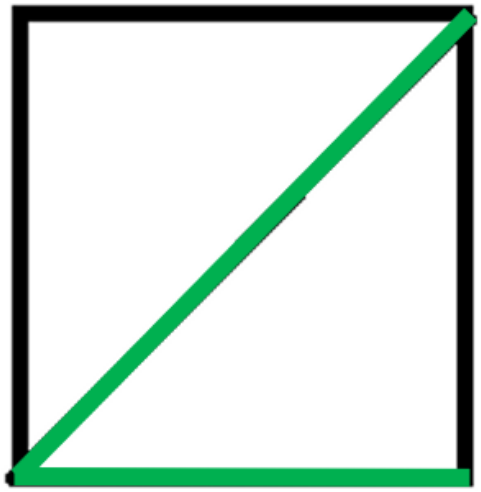
$$= 7 - 5 + 1 = 3.$$

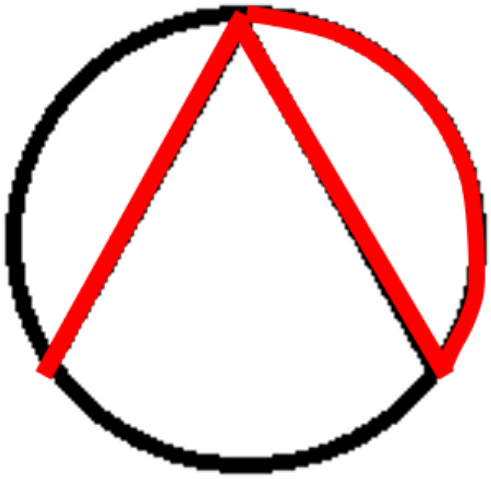
Пример 2:

Цикломатическое число несвязного графа

$$\gamma(G) = \mu_e - \mu_v + \mu_c =$$

$$= 10 - 7 + 2 = 5.$$



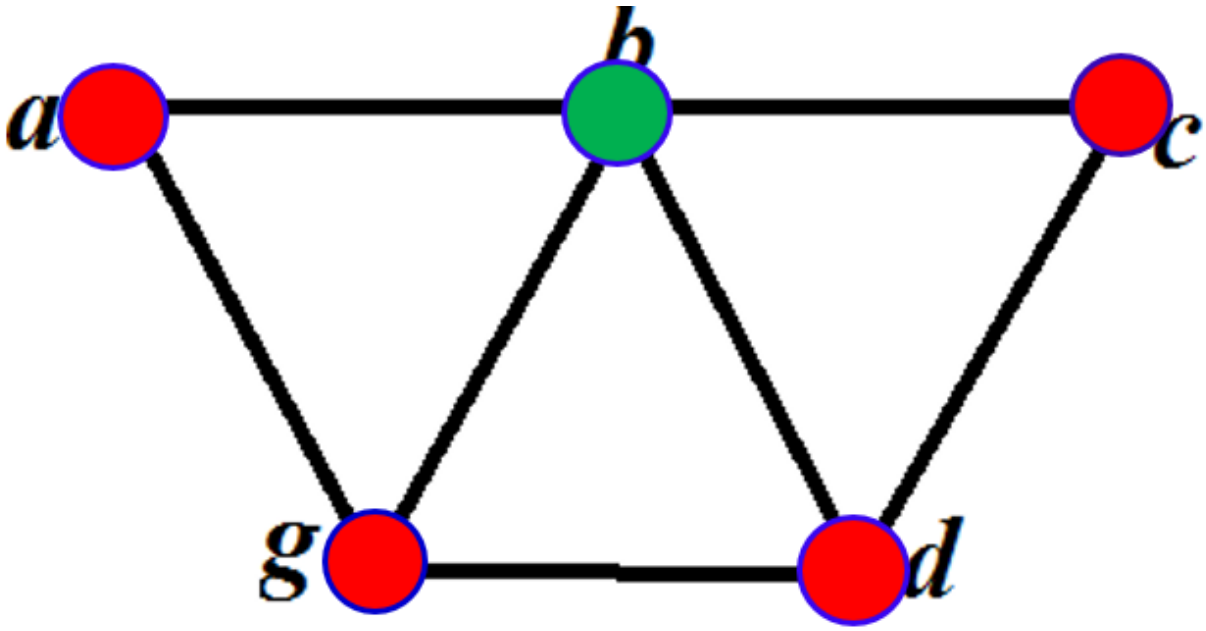


Внутренне устойчивым множеством графа ***G*** называется множество вершин ***S***, все вершины которого попарно несмежны.

Число внутренней устойчивости:

$$\alpha(G) = \max_{S \subseteq V} |S|.$$

Пример 3:



$$S_1 = \{a\}, S_2 = \{a,c\}, S_3 = \{g,c\}, S_4 = \{a,d\}.$$

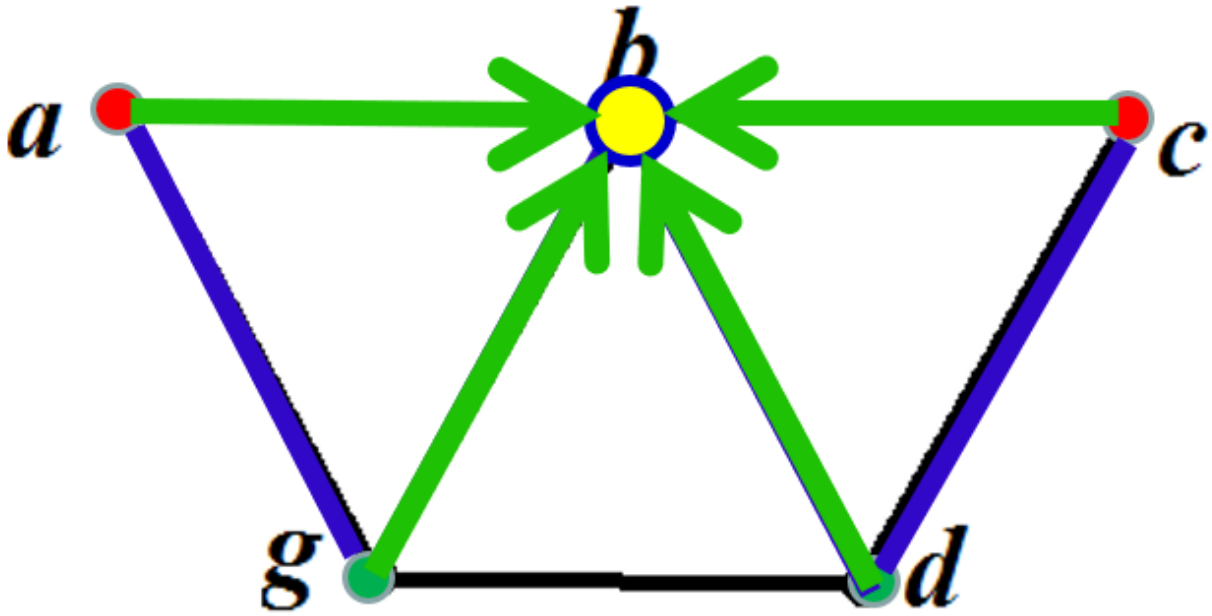
$$\alpha(G) = \max |S_i| = 2.$$

Внешне устойчивым множеством графа ***G*** называется множество вершин ***Q***, таких, что из всех вершин множества ***Q*** ведут ребра в вершины множества ***Q.***

Число внутренней устойчивости:

$$\beta(G) = \min_{Q \subseteq V} |Q|.$$

Пример 4: Число внешней устойчивости



$$Q_1 = \{a, b, c\}, Q_2 = \{g, d\}, Q_2 = \{b\}$$

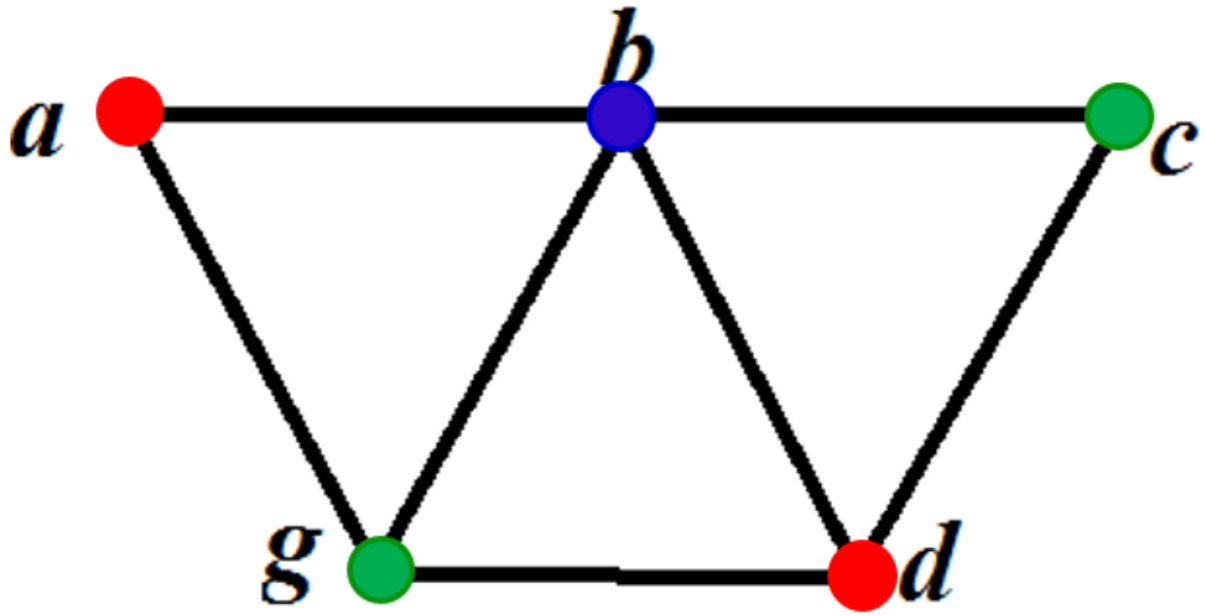
$$\beta(G) = \min |Q_i| = 1.$$

Граф ***G*** называется ***h***- хроматическим, если его вершины можно раскрасить ***h*** различными красками так, чтобы никакие две смежные (различные) вершины не были окрашены в один цвет.

Хроматическое число графа – это наименьшее число красок.

$$\chi(G) = \min h.$$

Пример 5: Хроматическое число



$$H_1 = \{a,d\}, H_2 = \{g,c\}, H_3 = \{b\}.$$

$$\chi(G) = \min h = 3.$$

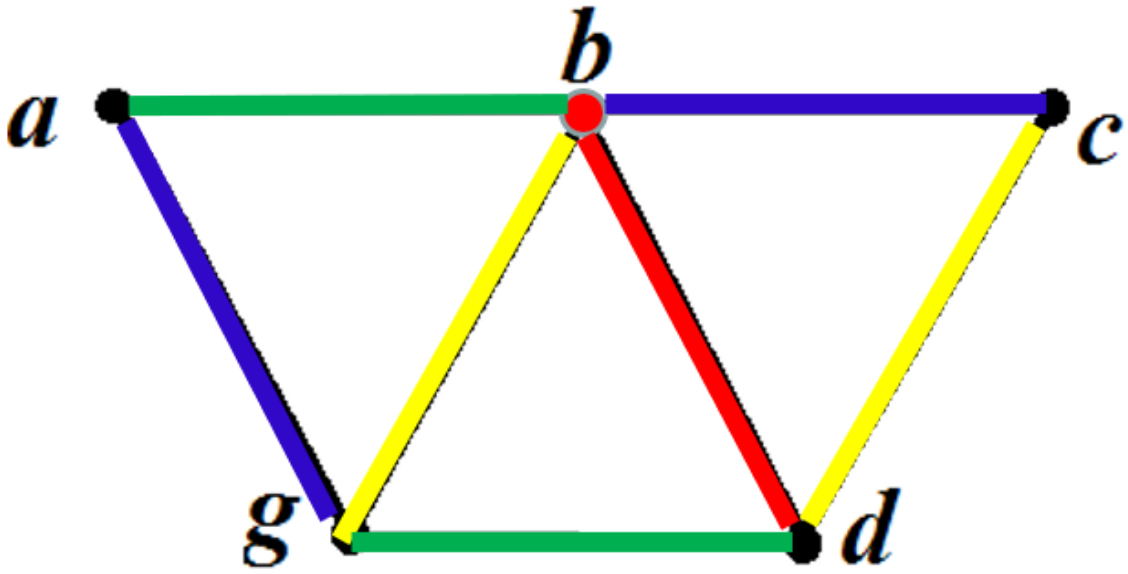
Граф ***G*** называется ***k***-раскрашиваемым, если его ребра можно раскрасить ***k*** различными красками так, чтобы никакие два смежные ребра не были окрашены в один цвет. ***Хроматический индекс графа*** – это наименьшее число красок.

$$\chi'(G) = \min k.$$

Согласно ***теореме Визинга***, если максимальная локальная степень вершины графа равна ***k,*** то хроматический индекс подчиняется условию:

$$k \le \chi'(G) \le k+1.$$

Пример 6:



$$k = \max \rho(v) = \rho(b) = 4;$$

$$4 \le \chi'(G) \le 5; \quad \chi'(G) = 4.$$