Графы

Основное определение

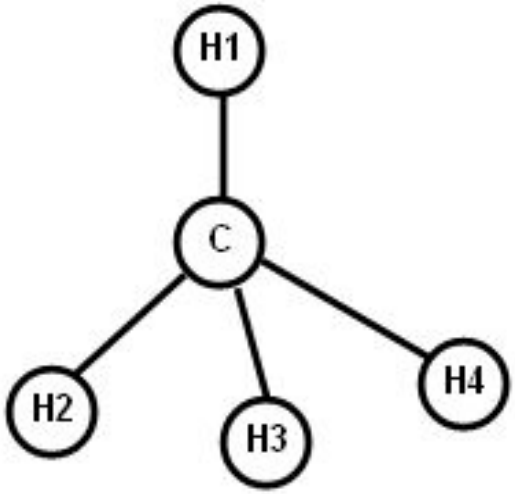
Неориентированный ГРАФ – это упорядоченная пара (V,E), где:

V – множество вершин (конечное);

E – множество пар вершин (множество ребер).

При этом природа множества V может быть любой.

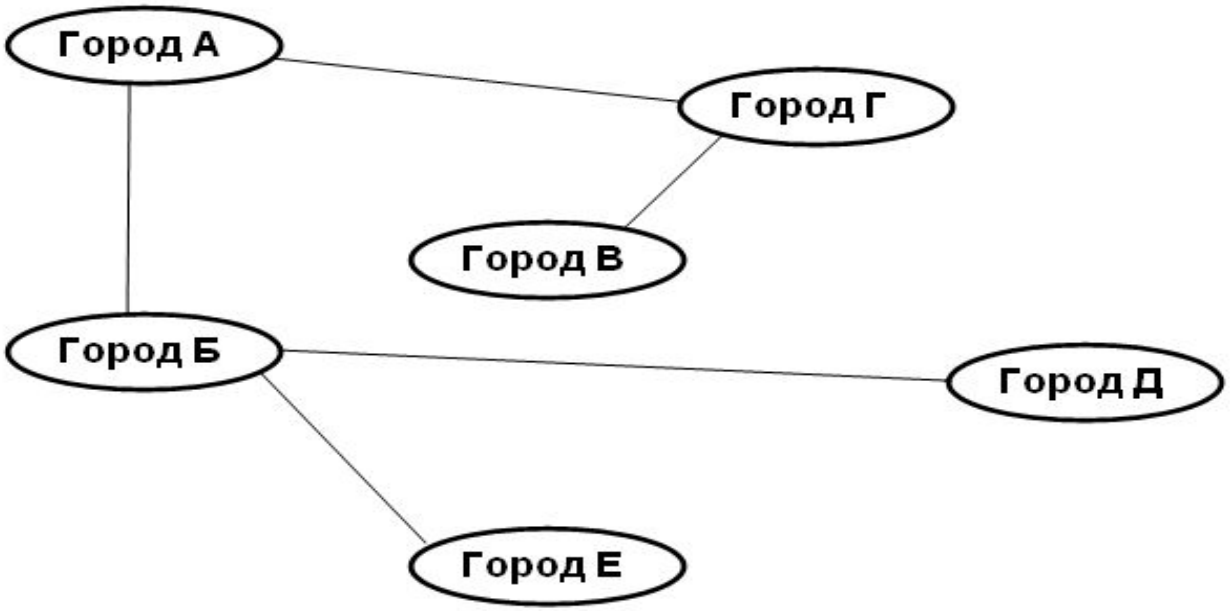
Пример графа-1



$$V=\{C,H1,H2,H3,H4\}$$

$$E=\{(C,H1),(C,H2),(C,H3),(C,H4)\}$$

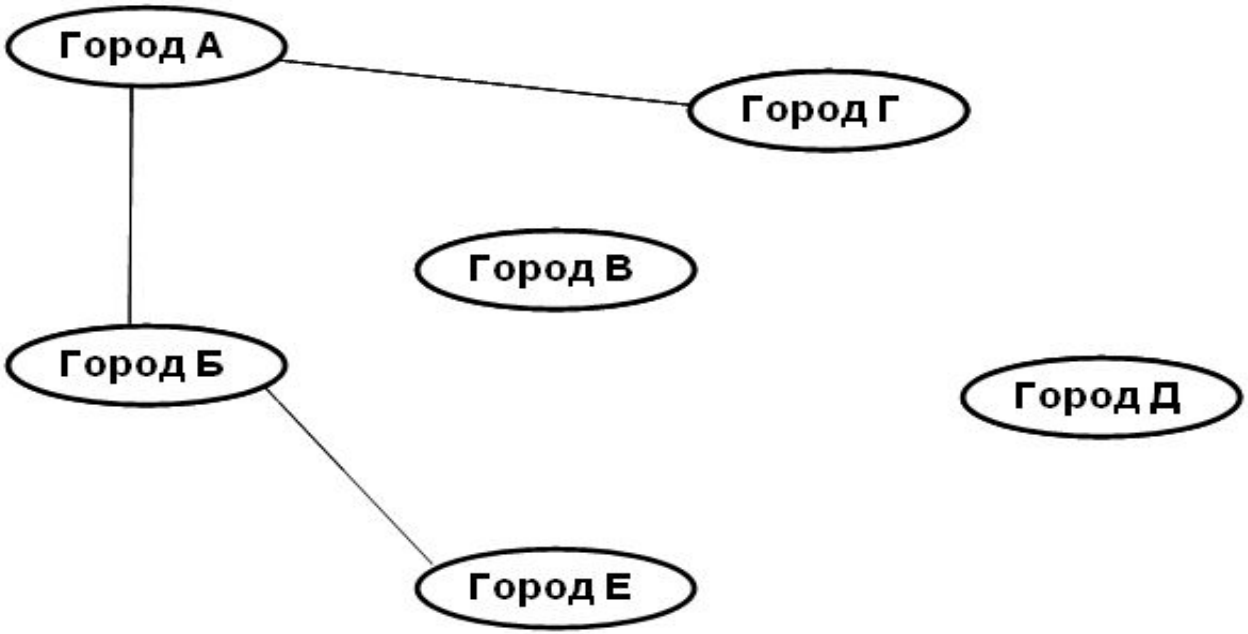
Пример графа-2



V={А,Б,В,Г,Д,Е}

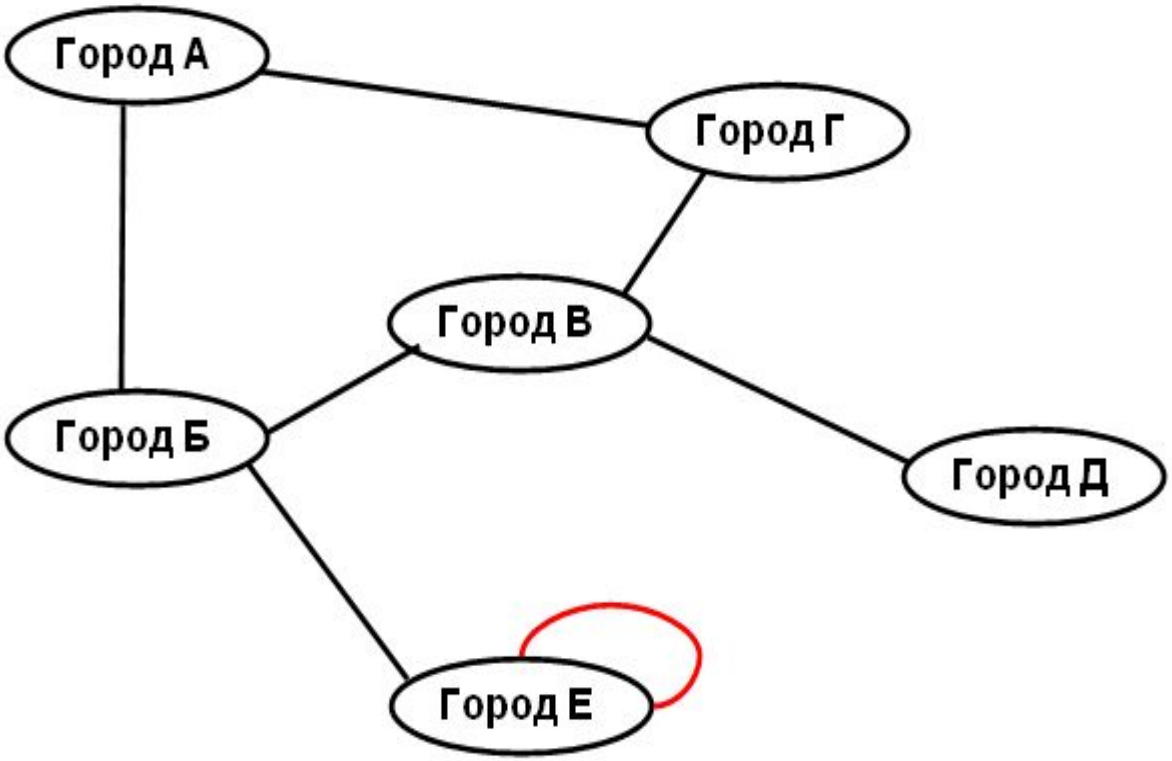
E={(А,Б), (А,Г), (Б,Д), (Б,Е), (Г,В)}

Будет ли ЭТО графом?



Да!

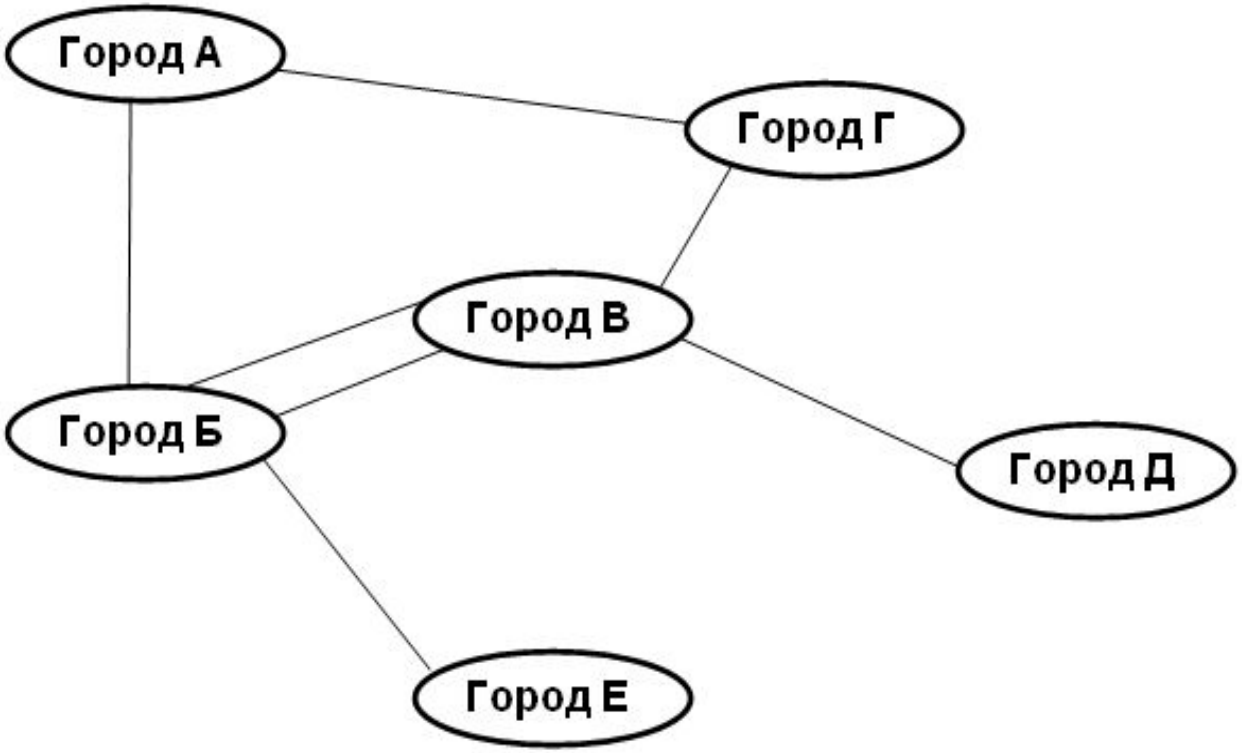
А ЭТО?



Да!

E={… (E,E)…}

А ЭТО?



Да!

E={… (Б,В),(Б,В)…}

Порядок и размер графа

Количество вершин называется порядком графа.

Количество ребер называется размером графа.

Некоторые термины-1

- Пусть R=(a,b) – одно из ребер графа. Тогда вершины a и b называются концевыми вершинами ребра;

- Концевые вершины одного и того же ребра называют соседними;

- Два ребра называют смежными, если они имеют общую концевую вершину;

- Два ребра называются кратными, если множества их концевых вершин совпадают;

- Ребро называется петлей, если его концы совпадают.

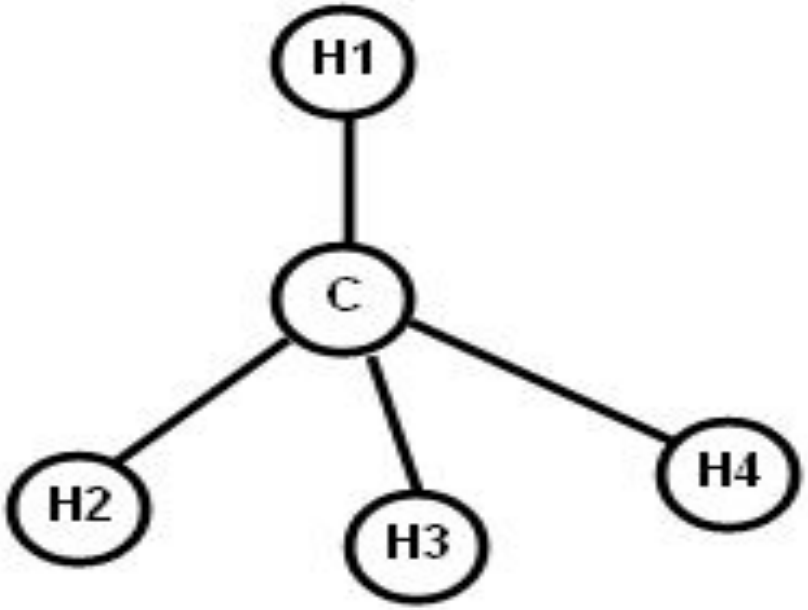
Некоторые термины-2

- **Степенью вершины V обозначается deg(V) называется количество ребер, для которых эта вершина является концевой;**

- **Вершина называется изолированной, если она не является концевой ни для одного ребра;**

- **Вершина называется листом, если она является концевой ровно для одного ребра. Для листа q очевидно deg(q)=1.**

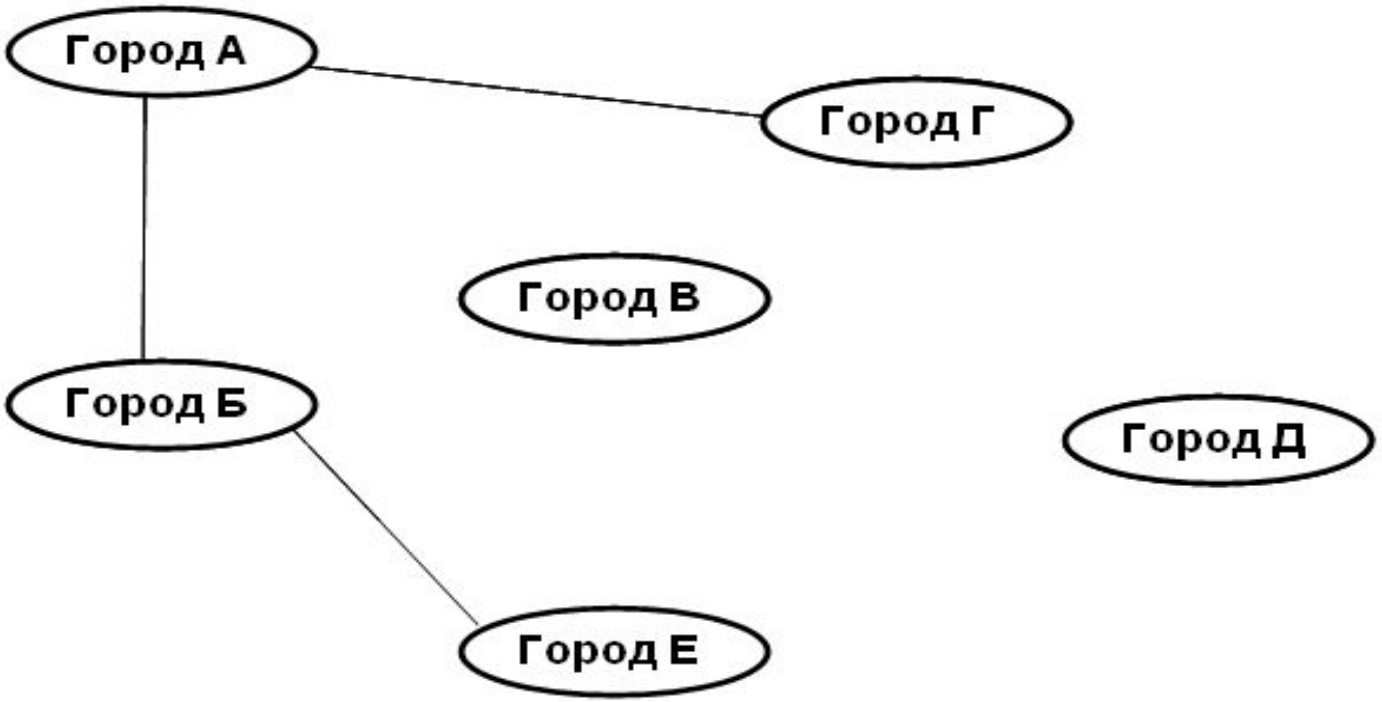
Пример:



deg(C)=4

H1,…H4 - Листья

Еще пример:



Города В и Д – изолированные вершины; Города Г и Е – листья.

Полный граф

Граф называется полным, если любые две вершины соединены ребром.

Сколько ребер у полного графа порядка n?

У полного графа порядка n число ребер равно $C_n^2 = n!/(2*(n-2)!) = n*(n-1)/2$

Давайте это докажем…

Полный граф с двумя вершинами содержит одно ребро – это очевидно.

Подставим n=2 в формулу n*(n-1)/2

Получим:

$$n*(n-1)/2=1$$

Формула верна при n=2

Предположение индукции

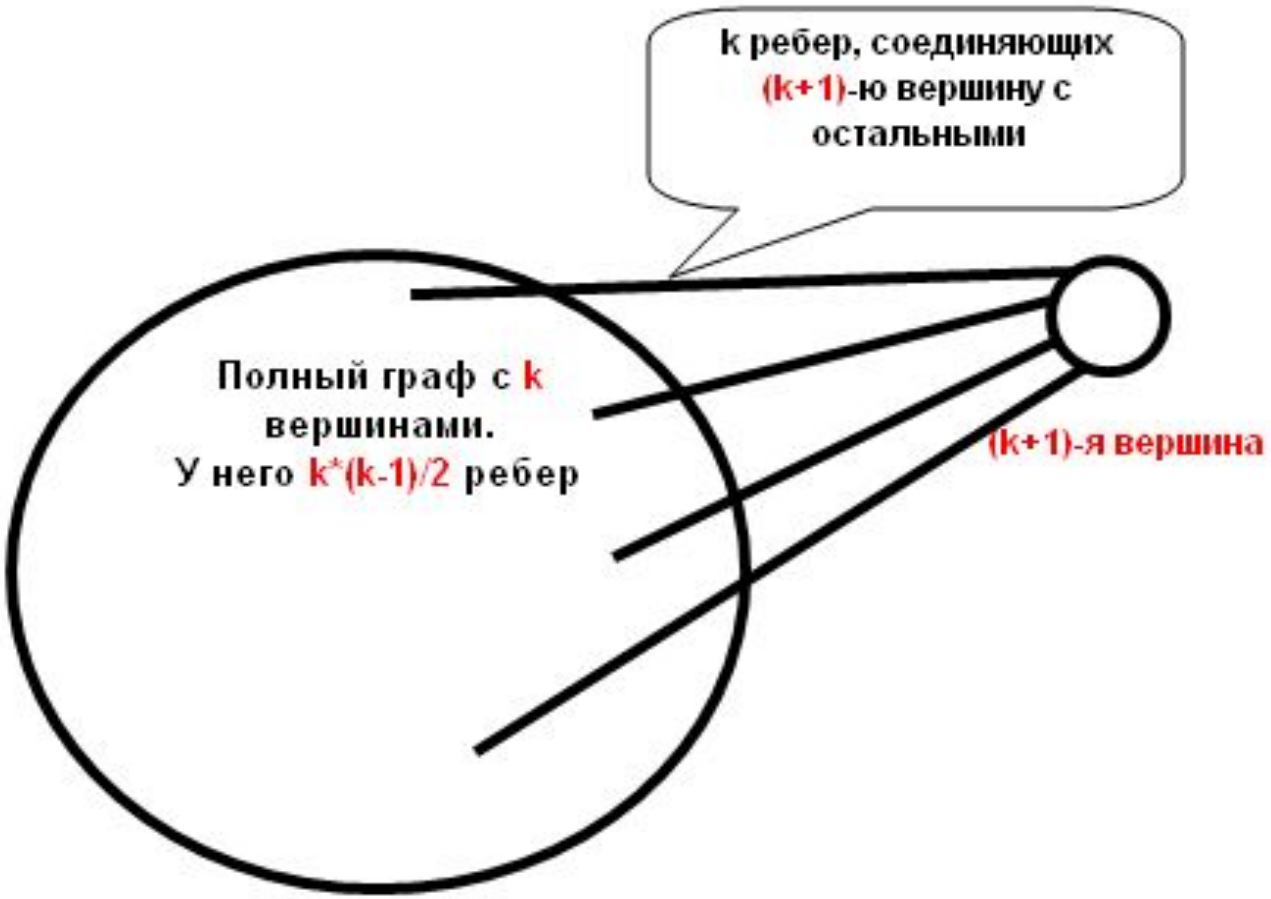
Предположим, что формула верна для графа с k вершинами.

Докажем, что отсюда следует справедливость формулы для графа с (k+1) вершиной.

Добавим к полному графу с K вершинами еще одну вершину.

И соединим ее с первыми K вершинами…

Получим:


k ребер, соединяющих (k+1)-ю вершину с остальными
Полный граф с k вершинами.
У него k*(k-1)/2 ребер
(k+1)-я вершина

Считаем, сколько получилось ребер…

$$K*(K-1)/2 + K$$

$$=$$

$$K*(K+1)/2$$

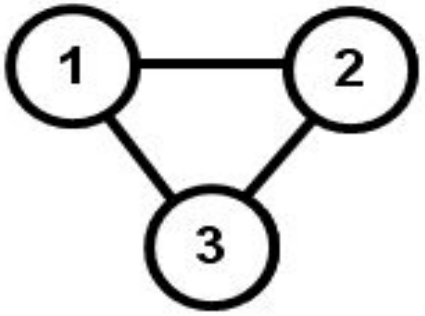
Последнее выражение получается, если в формулу $n*(n-1)/2$ вместо n подставить $K+1$.

Из предположения справедливости утверждения при $n=k$ следует справедливость утверждения при $n=k+1$.

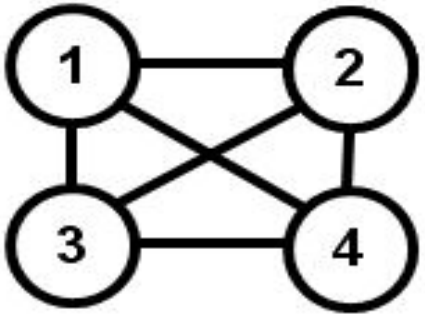
Теорема доказана.

Примеры полных графов

n=2; K=1



n=3; K=3



n=4; K=6

Важное уточнение

Пары, задающие ребра в неориенти-рованном графе, неупорядочены (т.е. пары (a,b) и (b,a) не различают-ся)

Ориентированный граф

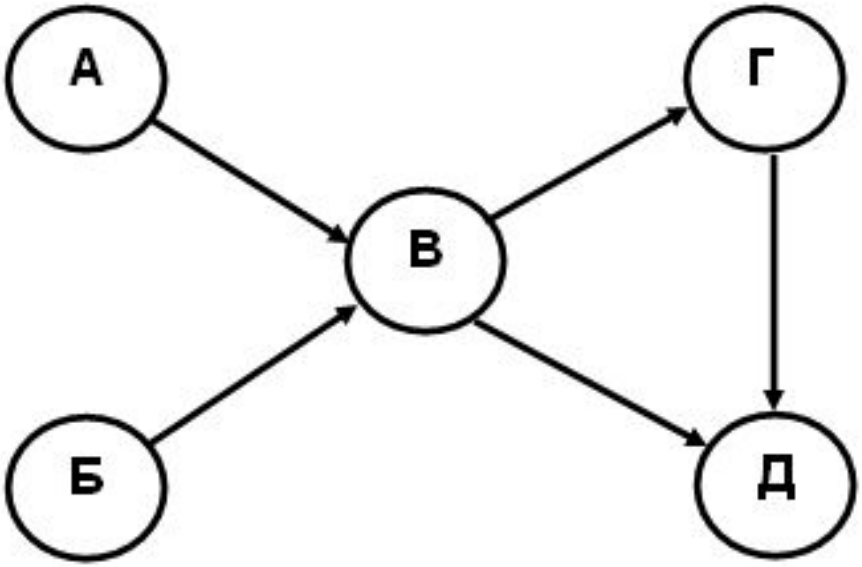
Если ребра графа есть множество упорядоченных пар (т.е. (a,b) ≠ (b,a)),

То граф называется ориентированным (или орграфом)

Как придать понятию ориентации наглядный смысл?

Очень просто – ребра снабжаются стрелками (от начала к концу)!

Пример орграфа



V={А, Б, В, Г, Д}

E={(А, В), (Б, В), (В, Г), (В, Д), (Г, Д)}

Смешанный граф

Смешанный граф – это тройка (V, E, A).

V – множество вершин;

E – множество неориентированных ребер;

A- множество ориентированных ребер.

Кстати, ориентированные ребра называются дугами.

Изоморфизм графов

Пусть имеется два графа G_1 и G_2

Если имеется взаимно-однозначное соответствие F между вершинами графов G_1 и G_2 , такое что:

- если в графе G_1 есть ребро (a,b), то и в графе G_2 есть ребро (F(a),F(b))
- если в графе G_2 есть ребро (p,q), то и в графе G_1 есть ребро $(F^{-1}(p),F^{-1}(q))$

то графы G_1 и G_2 называются изоморфными, а соответствие F – изоморфизмом.

Уточнение

Для орграфов и смешанных графов соответствие F должно сохранять ориентацию дуг.

Необходимое условия изоморфизма

При каких условиях между элементами двух конечных множеств можно установить взаимно-однозначное соответствие?

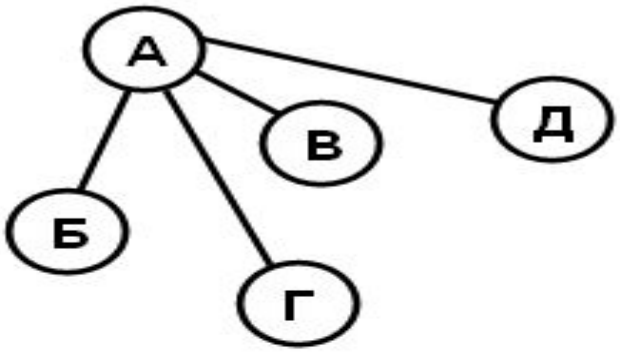
Тогда и только тогда, число их элементов одинаково.

Необходимым условием изоморфизма графов является одинаковой число вершин.

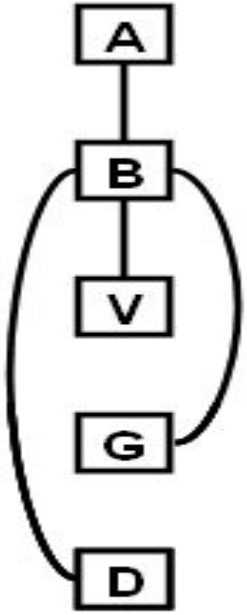
Достаточно ли это условие?

Нет, поскольку вершины могут быть соединены по-разному.

Изоморфны ли эти графы?



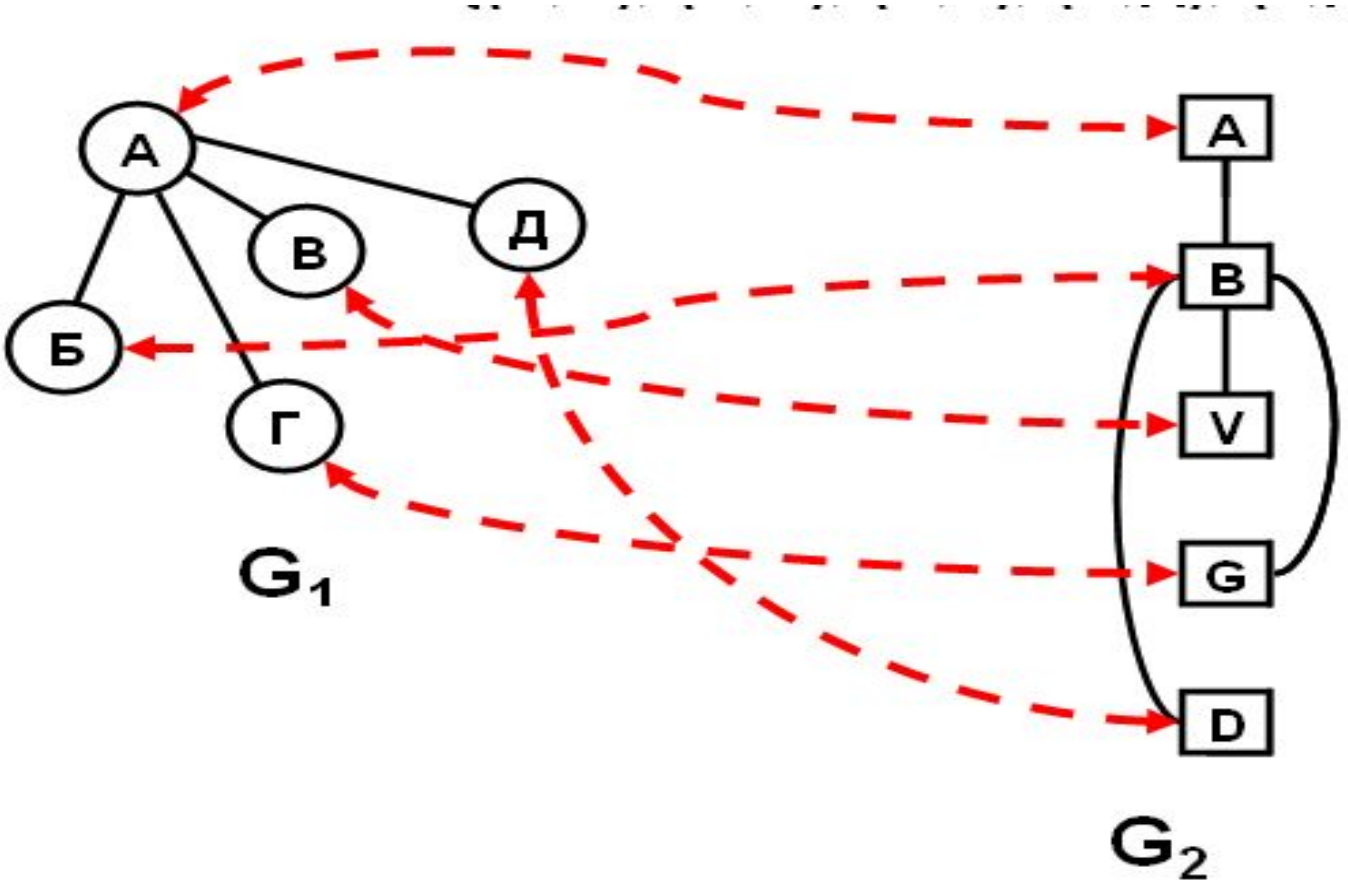
G_1



G_2

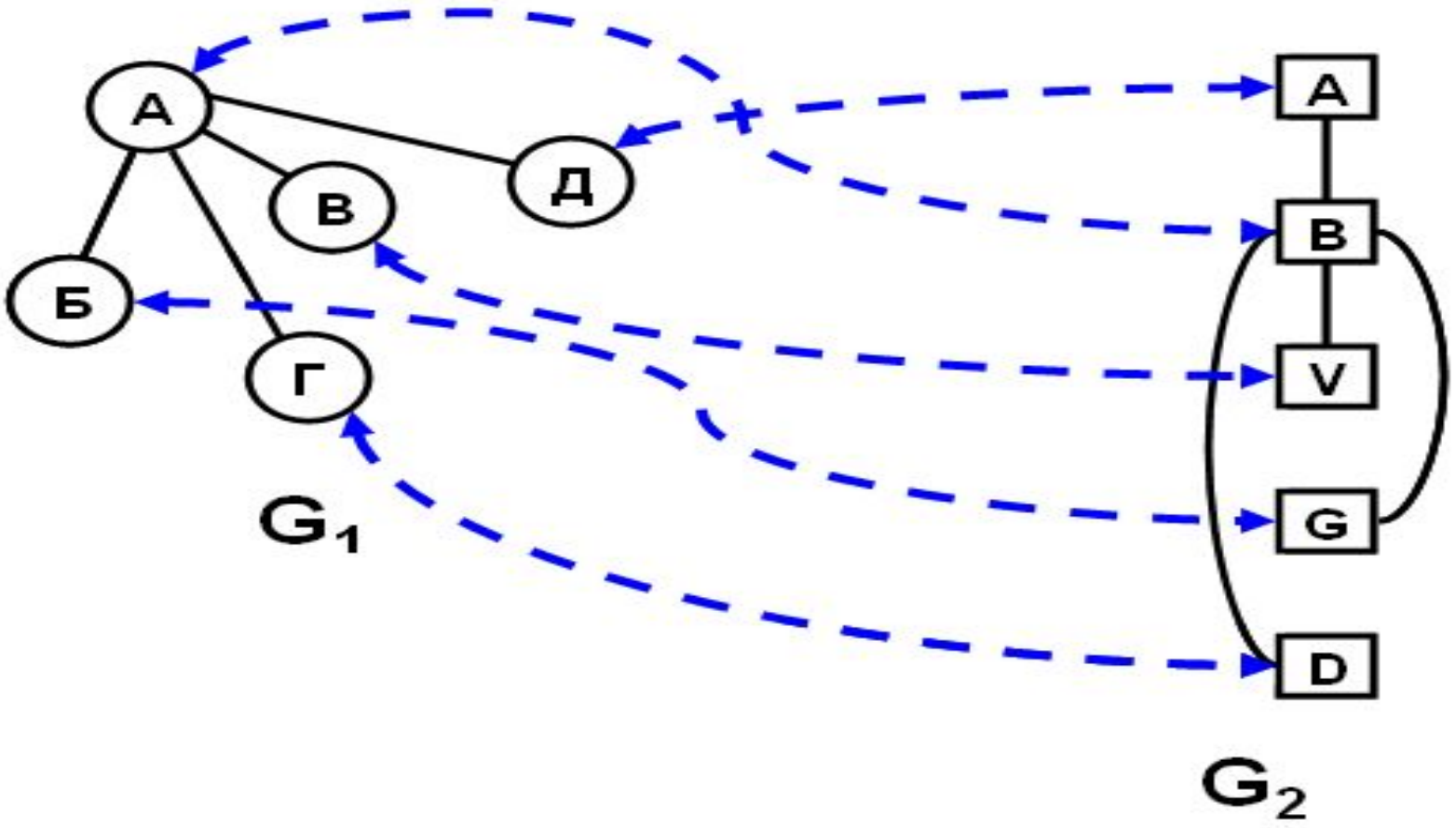
Число вершин одинаково – необходимое условие соблюдено…

Пробуем построить соответствие F…



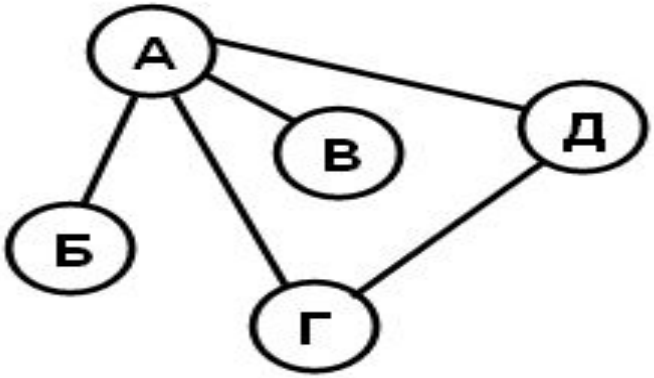
Это – не изоморфизм: в G_1 есть ребро (А,Д), а образы этих ребер в G_2 не соединены.

Другая попытка…

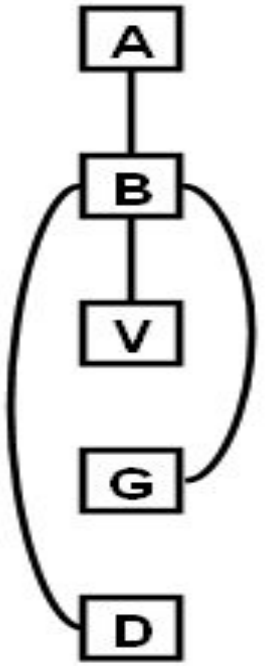


А это изоморфизм!

А эти графы изоморфны?



G_1



G_2

Увы, нет…

С точки зрения теории два изоморфных графа – это один и тот же объект (только, может быть, по-разному изображенный…)

Пути (цепи):

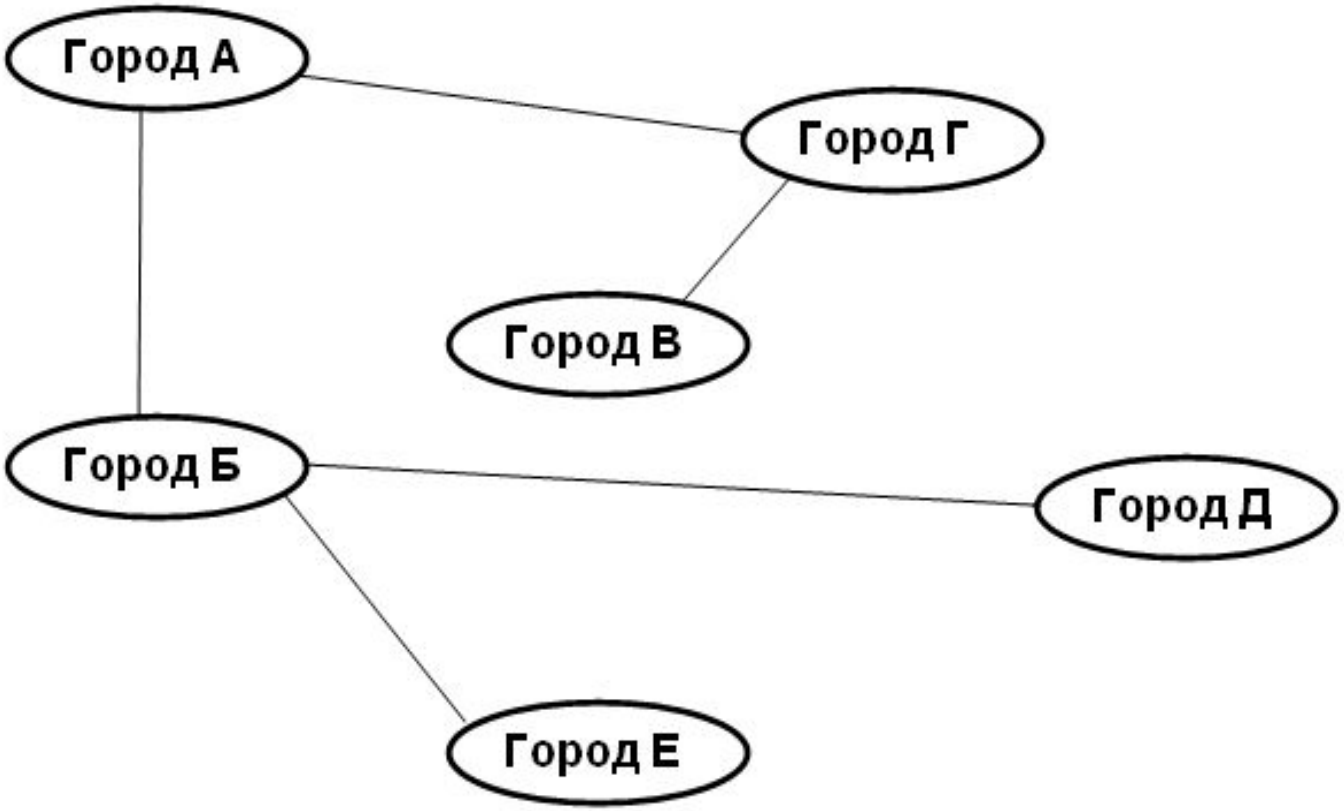
Путь (цепь) это последовательность вершин:

$a_1, a_2, \ldots, a_n$

в которой соседние вершины a_i и a_{i+1} соединены ребрами.

Длина пути есть число составляющих его ребер

Примеры путей:



(А, Г, В) и (А, Б, Д) – пути. (А, Б, В) – не путь.

Понятие пути для орграфа сохраняет силу, но нуждается в дополнении – соседние вершины в последовательности

$$a_1, a_2, \ldots, a_n$$

должны соединяться дугами.

Циклы

Цикл – это путь, у которого начальная и конечная вершина совпадают.

Длина цикла есть число составляющих его ребер.

Цикл называется простым, если ребра в нем не повторяются.

Цикл называется элементарным, если он простой и вершины в нем не повторяются.

Компоненты связности

Вершины произвольного графа можно разбить на классы, такие, что для любых двух вершин одного класса v_1 и v_2 существует путь из v_1 в v_2

Эти классы называются компонентами связности.

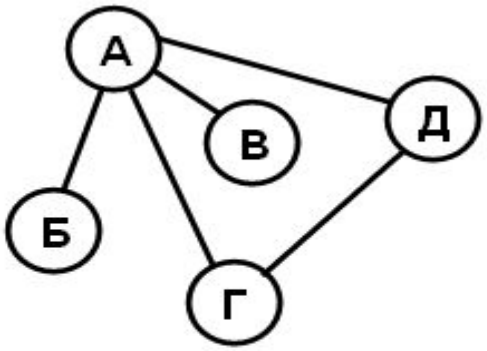
Если у графа ровно одна компонента связности, то граф называется связным.

Машинное представление графов.

Матрица смежности

- Занумеруем вершины графа G последовательными целыми от 1 до n;
- Построим квадратную таблицу n×n и заполним ее нулями;
- Если имеется ребро, соединяющее вершины i и j, то в позициях (i,j) и (j,i) поставим единицы;
- Полученная таблица называется матрицей смежности графа G.

Пример



G_1

	А(1)	Б(2)	В(3)	Г(4)	Д(5)
А(1)	0	1	1	1	1
Б(2)	1	0	0	0	0
В(3)	1	0	0	0	0
Г(4)	1	0	0	0	1
Д(5)	1	0	0	1	0

Некоторые очевидные свойства матрицы смежности

- Если вершина изолирована, то ее строка и столбец будут полностью нулевые;
- Количество единиц в строке (столбце) равно степени соответствующей вершины;
- Для неориентированного графа матрица смежности симметрична относительно главной диагонали;
- Петле соответствует единица, стоящая на главной диагонали.

Обобщение для орграфа

Матрицу смежности для орграфа можно строить аналогичным образом, но, чтобы учесть порядок вершин, можно поступить так:

Если дуга исходит из вершины j и входит в вершину k, то в позиции (j,k) матрицы смежности ставить 1, а в позиции (k,j) ставить -1.

Матрица инцидентности

- Занумеруем вершины графа G последовательными целыми от 1 до n;
- Построим прямоугольную таблицу с n строками и m столбцами (столбцы соответствуют ребрам графа);
- Если j-e ребро имеет концевой вершиной вершину k, то в позиции (k,j) ставится единица. Во всех остальных случаях ставится нуль.

Матрица инцидентности для орграфа

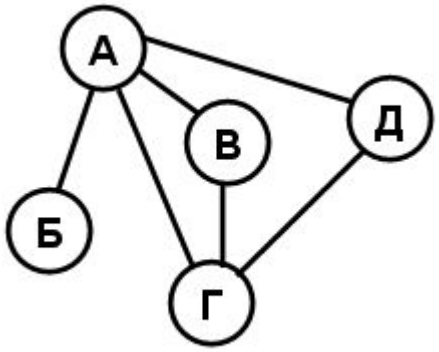
- Если j-я дуга исходит из вершины k, то в позиции (k,j) ставится 1;

- Если j-я дуга входит в вершину k, то в позиции (k,j) ставится -1.

- В остальных случаях в позиции (k,j) остается нуль.

Поскольку столбцы матрицы инцидентности описывают ребра, то в каждом столбце может быть не более двух ненулевых элементов

Пример матрицы инцидентности

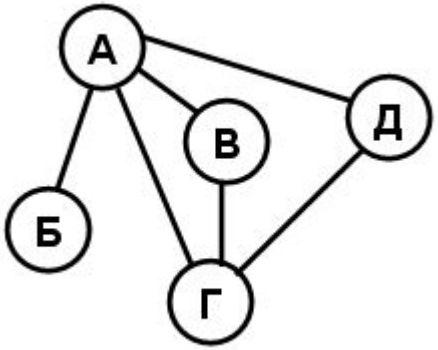


	А-Б	А-В	А-Г	А-Д	В-Г	Г-Д
А(1)	1	1	1	0	0	0
Б(2)	1	0	0	0	0	0
В(3)	0	1	0	0	1	0
Г(4)	0	0	1	0	1	1
Д(5)	0	0	0	1	0	1

Список ребер

Еще один способ представления графа – двумерный массив (список пар). Количество пар равно числу ребер (или дуг).

Пример списка ребер



А(1)
Б(2)
В(3)
Г(4)
Д(5)

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

Сравнение разных способов представления

- Список ребер самый компактный, а матрица инцидентности наименее компактна;

- Матрица инцидентности удобна при поиске циклов;

- Матрица смежности проще остальных в использовании.

Обход графа

Обходом графа называется перебор его вершин, такой, что каждая вершина просматривается один раз.

Соглашение-1

Перед выполнением поиска для графа с n вершинами заведем массив Chk из n элементов и заполним его нулями.

Если Chk[i] = 0, значит i-я вершина еще не просмотрена.

Соглашение-2

Заведем структуру данных (хранилище), в котором будем запоминать вершины в процессе обхода. Интерфейс хранилища должен обеспечивать три функции:

- **Занести вершину;**
- **Извлечь вершину;**
- **Проверить не пусто ли хранилище;**

Соглашение-3

Когда вершина j помещается в хранилище, она отмечается как просмотренная (т.е. устанавливается Chk[j]=1)

Алгоритм обхода-1

1) Берем произвольную начальную вершину, печатаем и заносим ее в хранилище;

2) Хранилище пусто? Если ДА – конец;

3) Берем вершину Z из хранилища;

4) Если есть вершина Q, связанная с Z и не отмеченная, то возвращаем Z в хранилище, заносим в хранилище Q, печатаем Q;

5) Переходим к п.2

Алгоритм обхода-2

1) Берем произвольную начальную вершину и заносим ее в хранилище;

2) Хранилище пусто? Если ДА – конец;

3) Берем вершину Z из хранилища, печатаем и удаляем из хранилища;

4) Помещаем в хранилище все вершины, связанные с Z и еще не отмеченные;

5) Переходим к п.2

Какие структуры данных подходят в качестве хранилища?

- Стек (PUSH – занести; POP – извлечь)

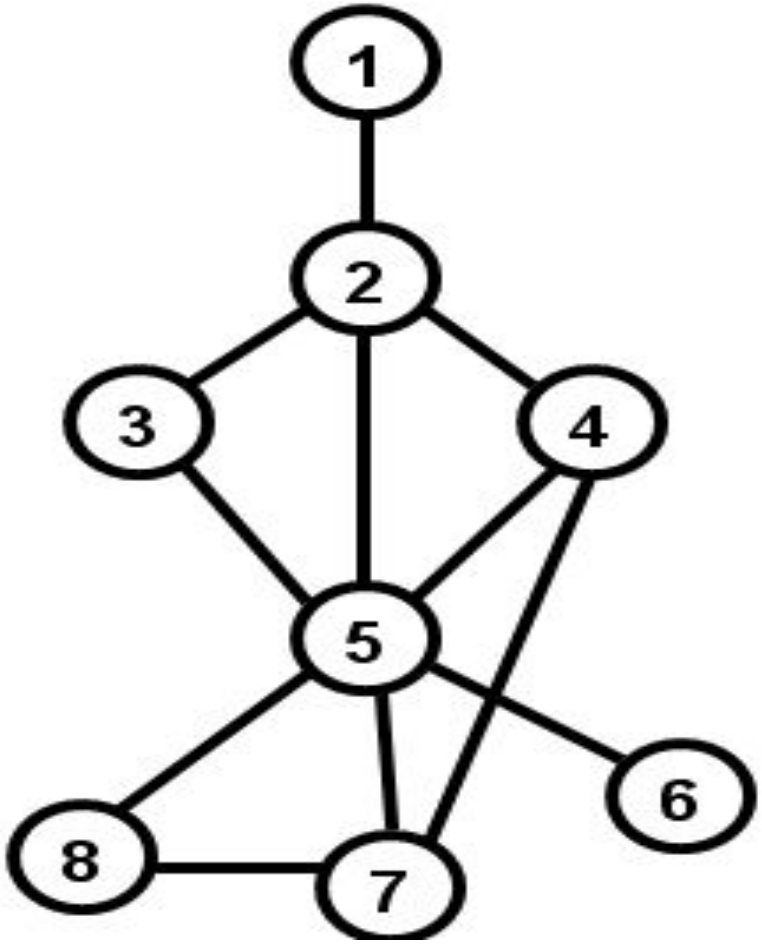
- Очередь (ENQUE – занести; DEQUE – извлечь)

Обе структуры позволяют проверить наличие данных.

Алгоритм-1 в сочетании со **стеком** называется обходом **в глубину**

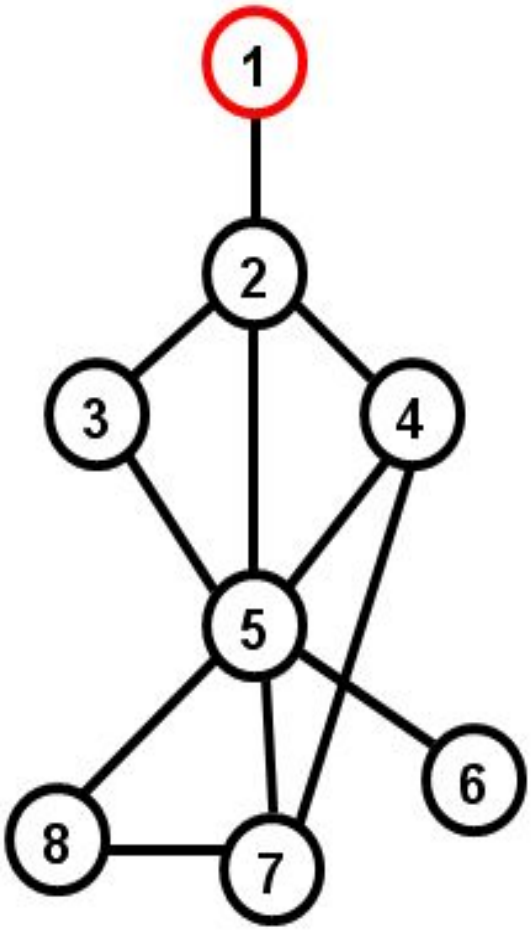
Алгоритм-2 в сочетании с **очередью** называется обходом в **ширину**

Возьмем граф:

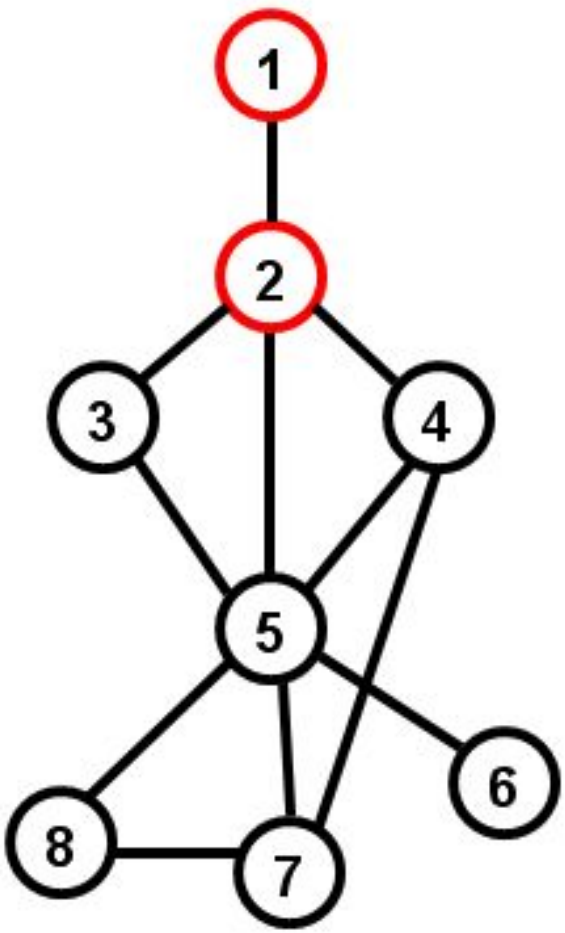


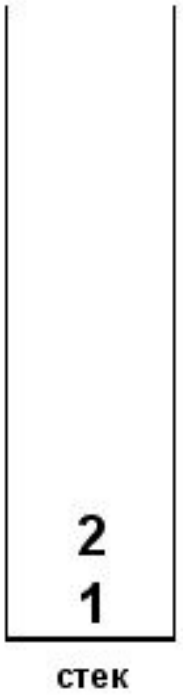
В качестве хранилища возьмем СТЕК.
Используем Алгоритм-1.
Обход начнем с вершины 1

Первый шаг:

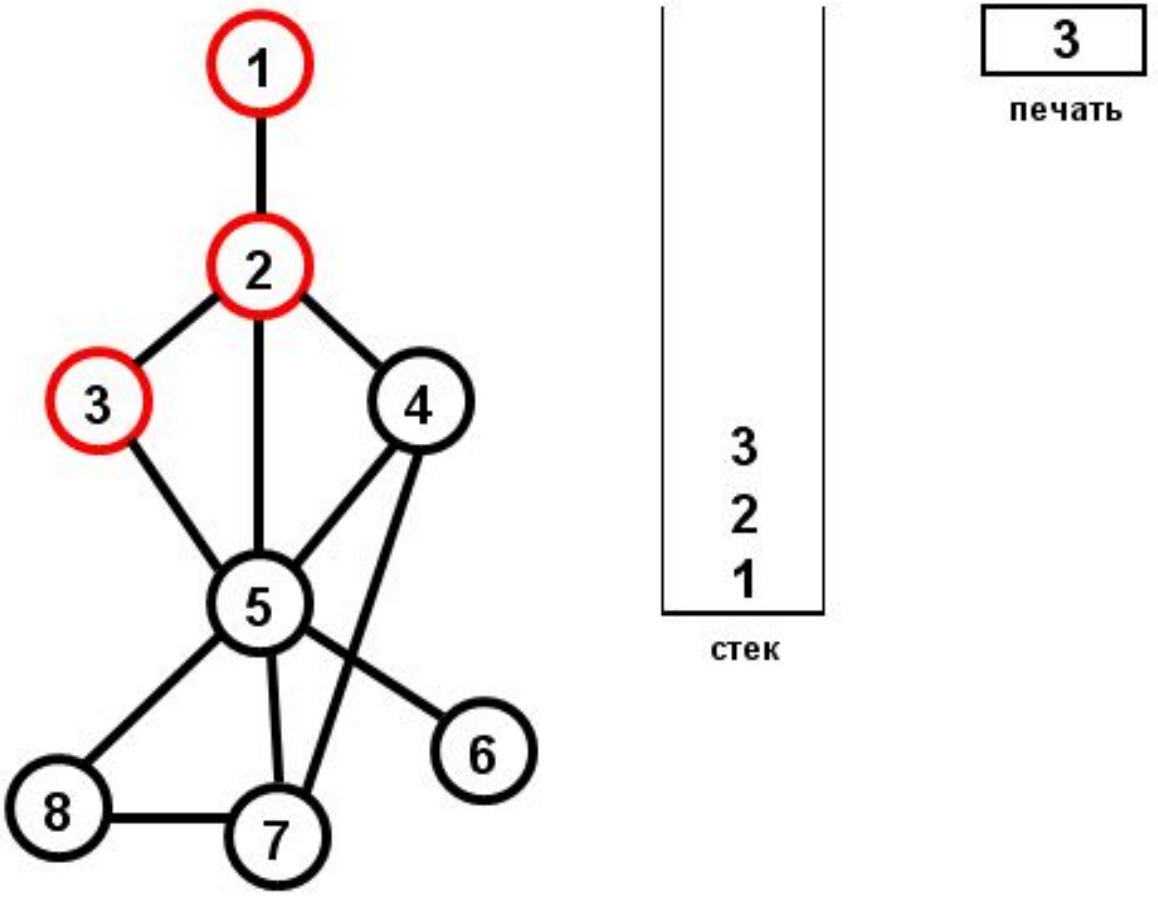


Второй шаг:

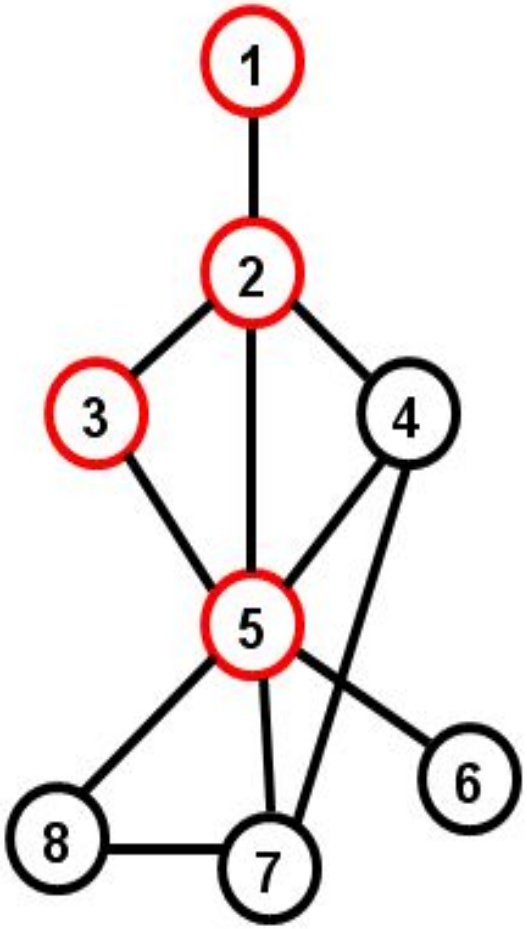




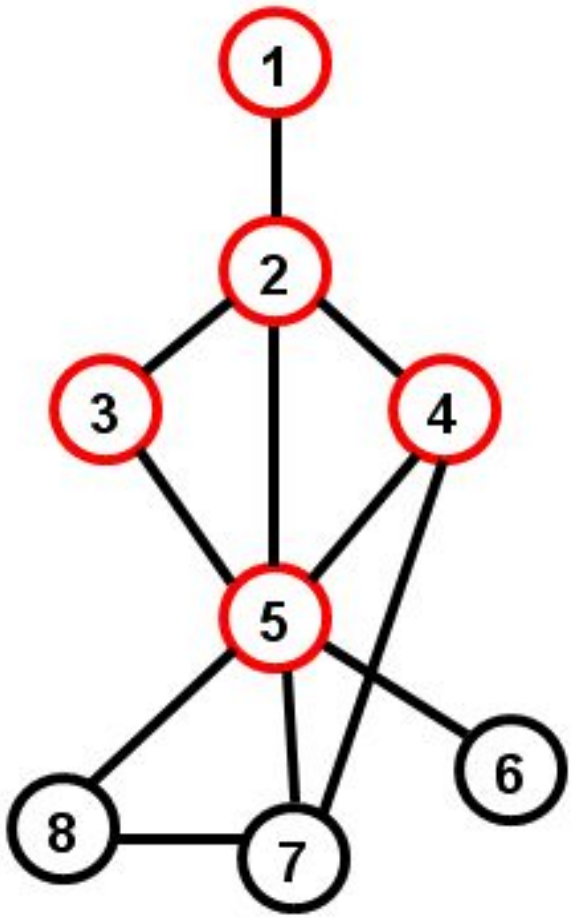
Третий шаг:

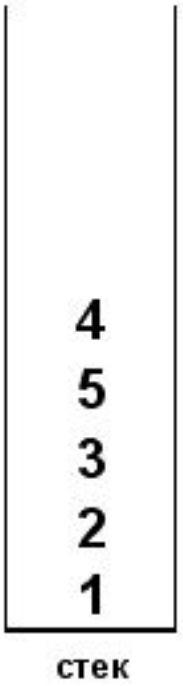


Четвертый шаг:

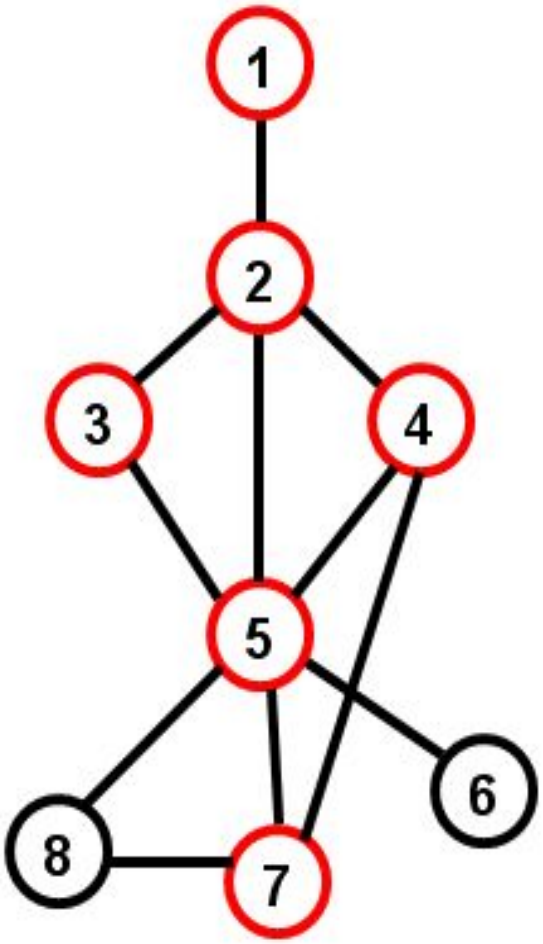


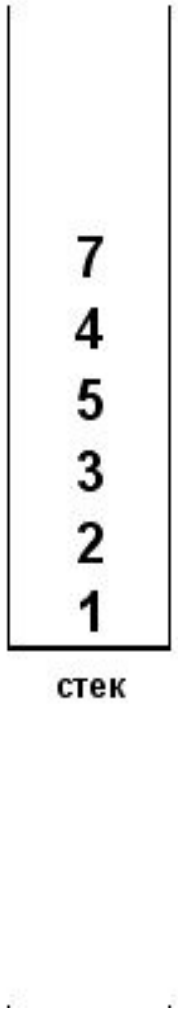
Пятый шаг:



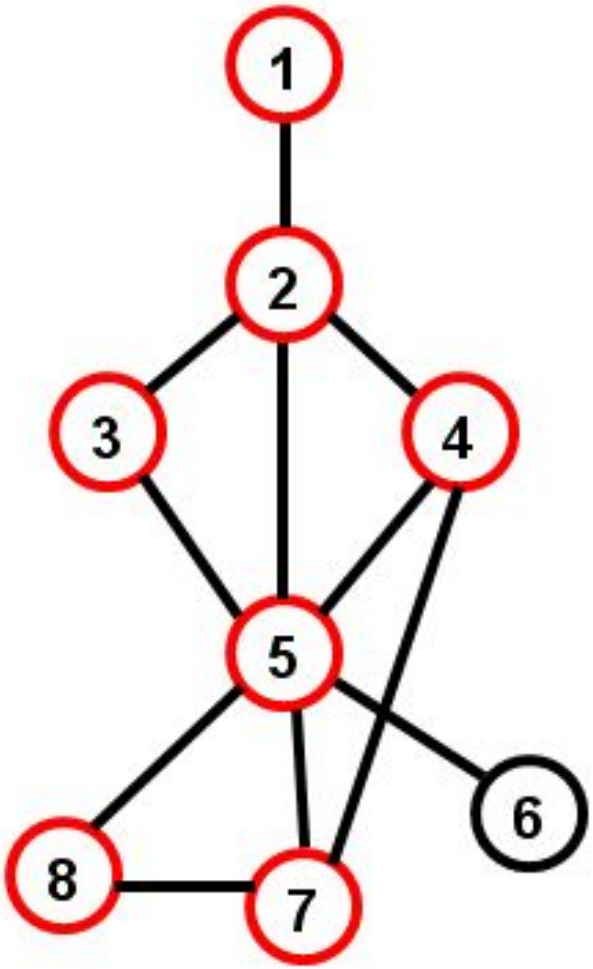


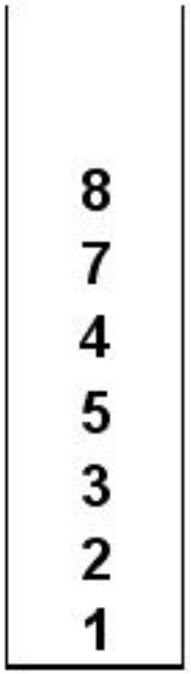
Шестой шаг:



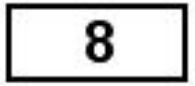


Седьмой шаг:



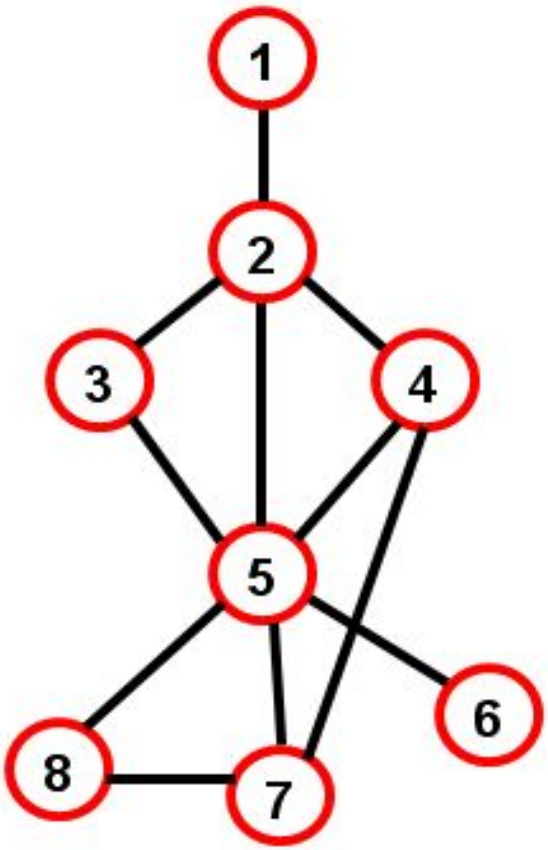


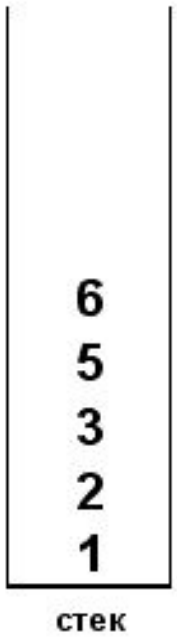
стек



печать

Восьмой шаг:





Вершины 8, 7, 4 выталкиваются из стека, т.к. их связи уже обработаны

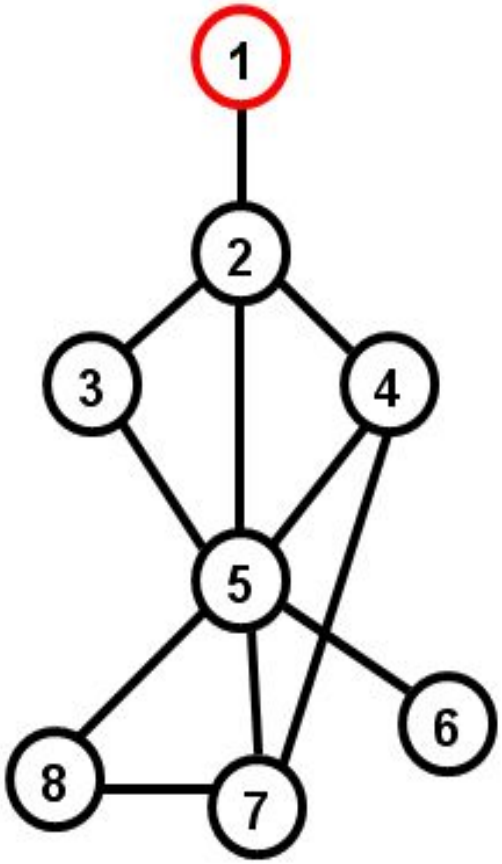
Далее все вершины будут вытолкнуты из стека.

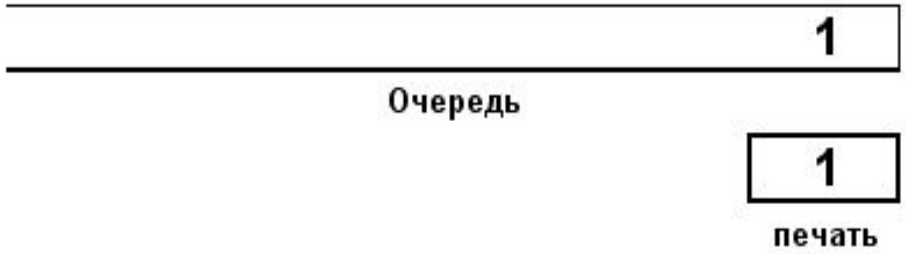
Получился следующий порядок обхода:

1,2,3,5,4,7,8,6

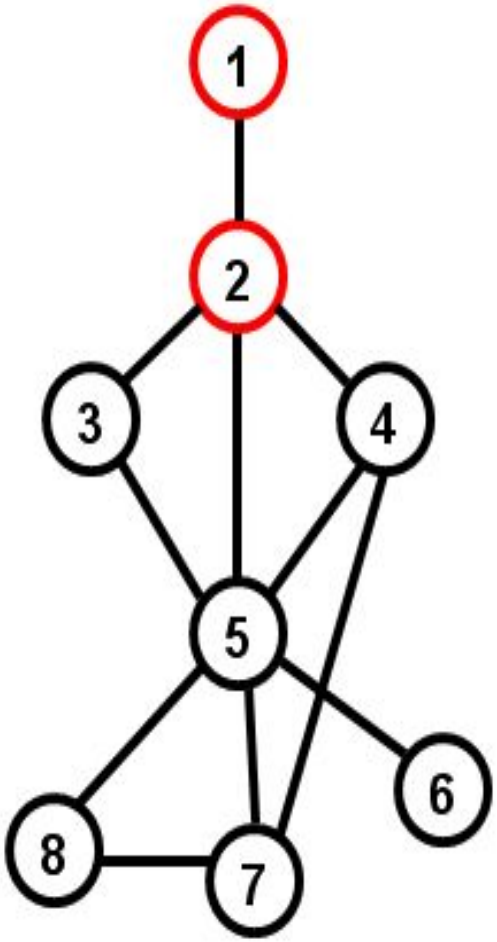
Теперь возьмем в качестве хранилища очередь. Будем использовать Алгоритм-2. Обход снова начнем с вершины 1.

Шаг первый:

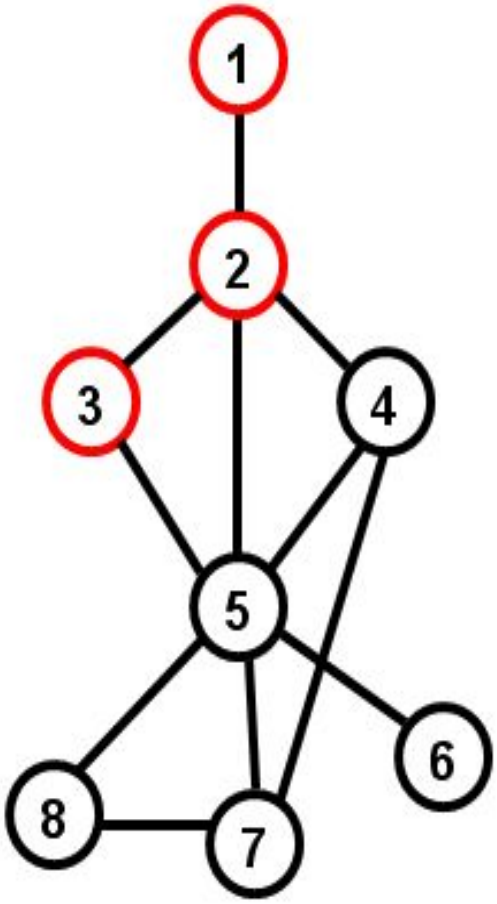




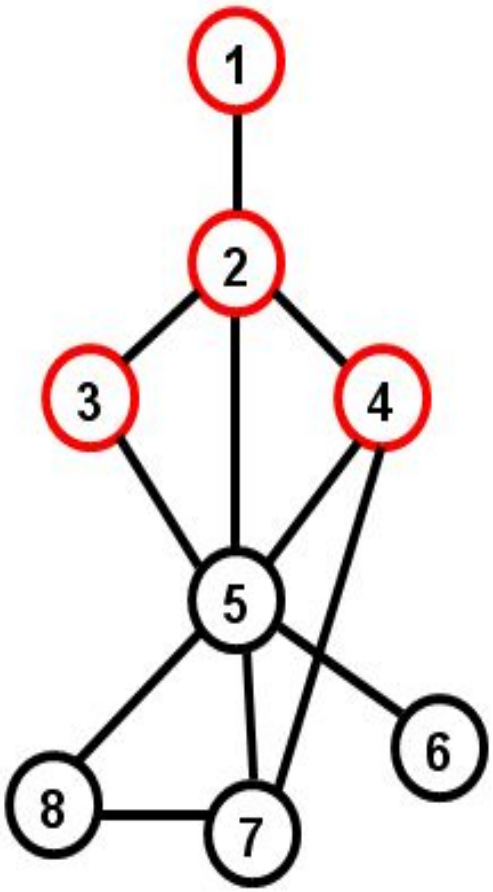
Шаг второй:



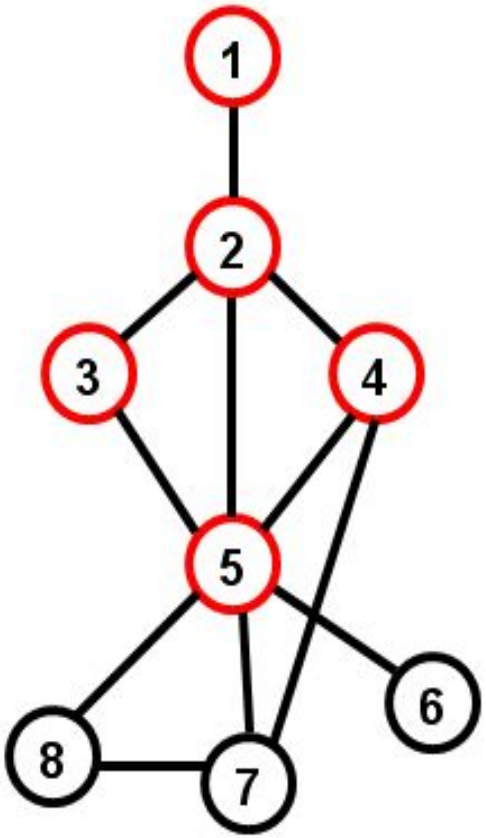
Шаг третий:



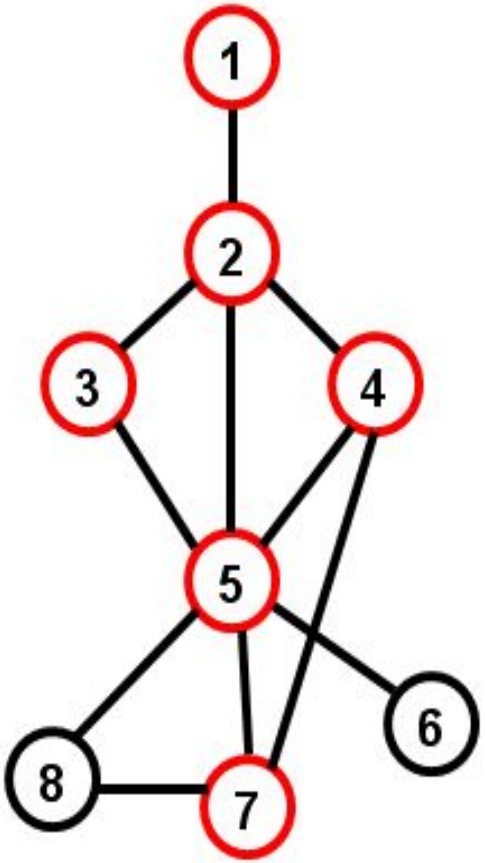
Шаг четвертый:



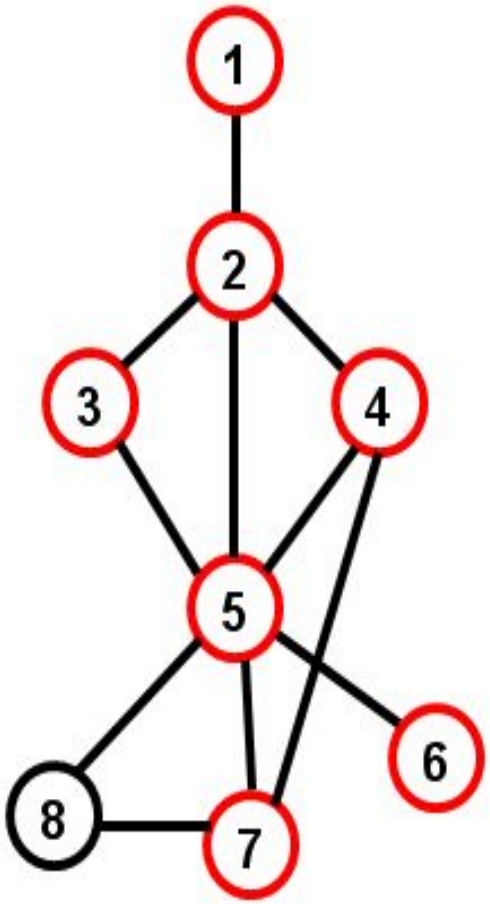
Шаг пятый:



Шаг шестой:



Шаг седьмой:



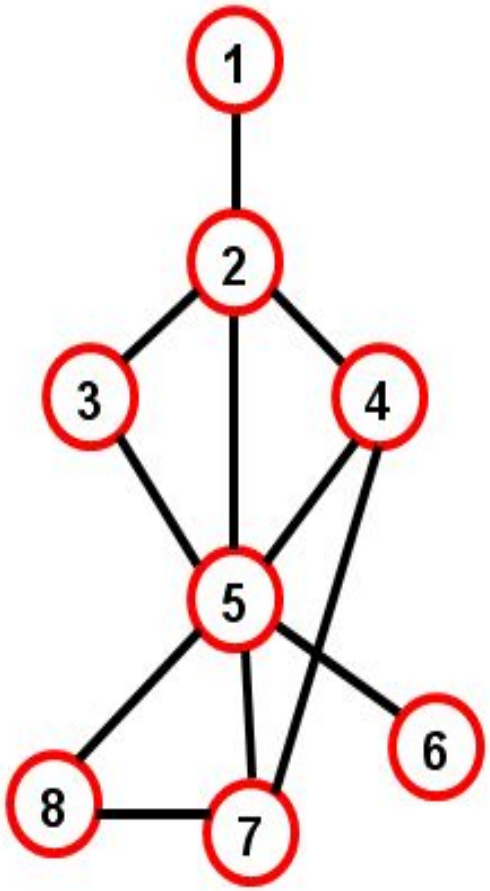
			6	7

Очередь

6

печать

Шаг восьмой:



	8	6	7

Очередь

8

печать

Получился следующий порядок обхода:

1,2,3,4,5,7,6,8

Замечание

Оба алгоритма потребовали одинаковое число шагов. Почему?

Потому, что при обходе каждая вершина печатается один раз.

Поиск в глубину

Перед выполнением поиска в глубину для графа с n вершинами заведем массив Chk из n элементов и заполним его нулями.

Если Chk[i] = 0, значит i-я вершина еще не просмотрена.

Алгоритм поиска в глубину с вершины p.

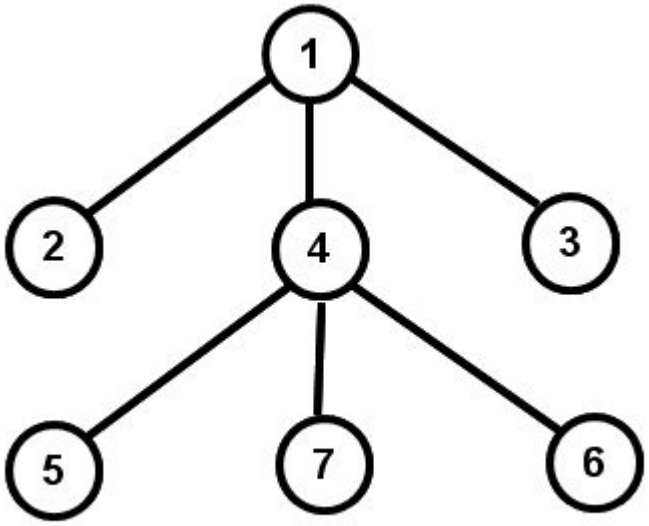
- Если Chk[p]=1 – выходим;

- Устанавливаем Chk[p]=1

- Берем по очереди все вершины k, смежные с p;

- Применяем к каждой из них указанный алгоритм.

Пример-1



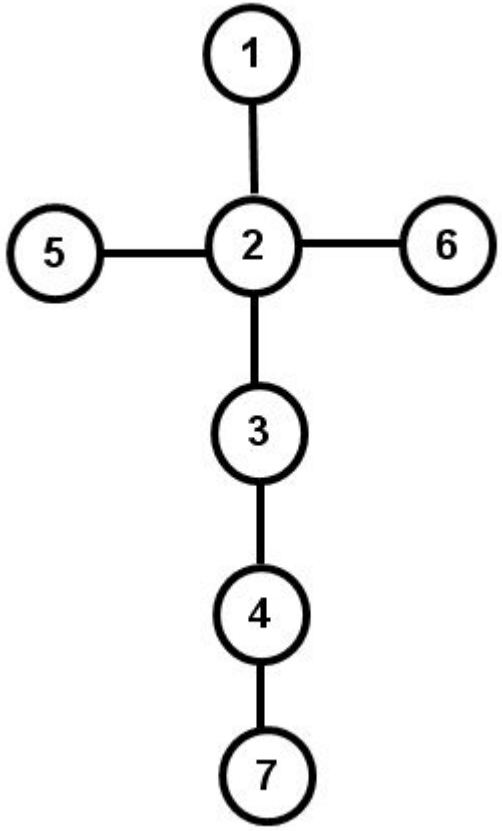
	1	2	3	4	5	6	7
1		1	1	1			
2	1						
3	1						
4	1				1	1	1
5				1			
6				1			
7				1			

Порядок обхода при поиске в глубину

4	1	2	3	5	6	7

1	2	3	4	5	6	7

Пример-2



	1	2	3	4	5	6	7
1		1					
2	1		1		1	1	
3		1		1			
4			1				1
5		1					
6		1					
7				1			

Порядок обхода при поиске в глубину

1	2	3	4	7	5	6

4	3	2	1	5	6	7

Если граф несвязный

В этом случае после обхода останутся непросмотренные вершины.

Можно повторить просмотр, начав с любой из непросмотренных вершин.

Количество таких итераций будет равно числу связных компонент графа.

Сложность алгоритма

Вычислительная сложность алгоритма

$$O(n+m),$$

где n – число вершин, а m – число ребер графа.

http://catstail.narod.ru/lec/lec-06.zip