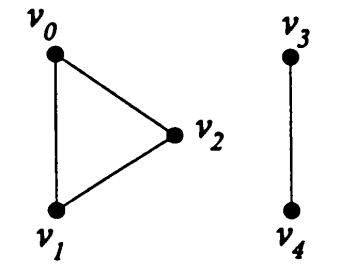
**ЛАБОРАТОРНАЯ РАБОТА. Графы, часть 1.**

***Нахождение компонент связности неориентированного графа.***

Граф  называется *связным*, если имеется путь между *любыми двумя его различным вершинами*. Отношение связанности является отношением *эквивалентности*.



*Рис. 1*.

Граф, приведенный на *рис. 1*, не связанный, например, не существует пути между вершинами **и **.

Рассмотрим неориентированный граф **. Подграф  графа  называется *компонентой связности* графа , если выполнены следующие два условия:

1.  – непустой связный граф.
2.  – максимальный связный подграф графа  (т.е. компонента графа  не является собственным подграфом любого другого *связанного* подграфа графа ).

*Если граф  связен, то он имеет только одну компоненту – самого себя, граф .*

Наш граф на *рис.* *1* имеет *2* компоненты.

*По определению изолированная вершина графа является его компонентой.*

Для нахождения компонент связности неориентированного графа (и работой с любыми алгоритмами на графах), граф должен быть представлен в виде, из которого мы *однозначно* можем сказать, с какими вершинами связана произвольно выбранная нами вершина (например, представлен матрицей смежности или матрицей инцидентности или списками смежности). Выбор представления графа часто диктуется условиями конкретной задачи.

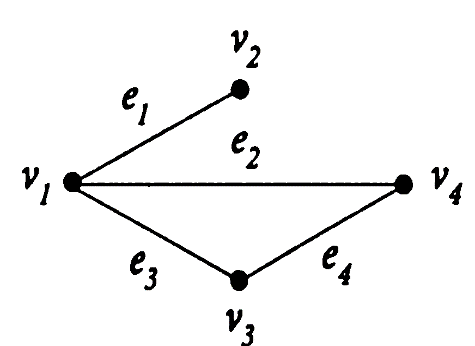
Напомним построение *матрицы смежности* для неориентированного графа. Будем рассматривать *простой* граф.

Напомним, что неориентированный граф, не содержащий петель и кратных ребер, называется *простым*.

Пусть задан граф **. Количество вершин графа, . Количество ребер графа, . Будем считать, что вершины и ребра графа каким либо образом *пронумерованы*.

*Матрицей смежности A* простогографа  называется квадратная матрица размером , строкам и столбцам которой соответствуют вершины графа, записанные *в том же самом порядке*, а элемент *i-*ой строки и *j-*го столбца равен *1*, если имеется ребро из *i*-ой вершины в *j*-ую вершину, и равен нулю в противном случае.

Построим матрицу смежности для простого графа, *рис. 2а*, *рис. 2б*.

*Рис. 2а. Рис. 2б.*

Для решения конкретной задачи граф нужно *«обойти»*. Под обходом графа понимается *систематическое перемещение по ребрам графа, при котором посещаются все его вершины*.

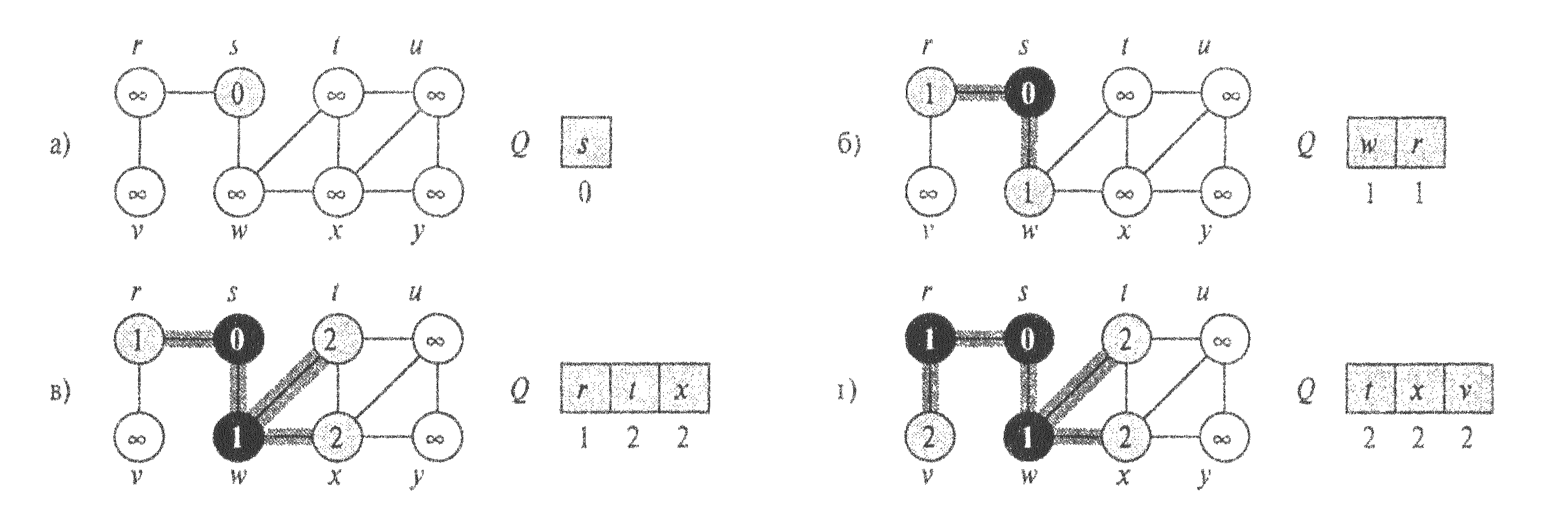
Рассмотрим два базовых обхода графа: *поиск в ширину* и *поиск в глубину*.

*Поиск в ширину* (*breadth-first search*) является основой для многих важных алгоритмов для работы с графами. Например, в алгоритмах *Прима* (*Prim*) поиска минимального остовного дерева или алгоритма *Дейкстры* (*Dijkstra*) поиска кратчайшего пути из одной вершины используется поиск в ширину.

Алгоритм нахождения компонент связности графа стратегией обхода графа методом поиска в ширину заключается в следующих действиях:

1. Числу компонент связности присваиваем начальное значение равное нулю.
2. Выбираем произвольно начальную вершину и помечаем её как пройденную.
3. Все вершины связанные с нашей вершиной помечаем как доступные.
4. Для каждой из доступных (в предыдущем пункте) вершин производим аналогичные действия, т.е. вновь выбранную вершину помечаем как пройденную, а все связанные с ней вершины как доступные (если они еще не были помечены).
5. Продолжаем так выполнять алгоритм для всех вновь доступных вершин, пока все доступные вершины не будут помечены как пройденные.
6. Инкриминируем число компонент связности.
7. Проверяем: если в графе остались не помеченные вершины то для непомеченных вершин выполняем алгоритм со второго пункта, в противном случае алгоритм закончен.

Иногда не пройдённые, доступные и пройденные вершины окрашиваются в белые, серые и черные цвета соответственно. На рис. *3* показаны первые шаги обхода графа в ширину с окрашенными вершинами.



*Рис. 3.*

Поиск в ширину имеет такое название потому, что в процессе обхода мы идем *вширь*, т.е. перед тем как приступить к поиску вершин на расстоянии *k+1* (количество ребер), выполняется обход всех вершин на расстоянии *k* от начальной вершины*.* На *рис 3* уровни вершин при каждом шаге указаны ы строке *Q*.

Поиск в ширину строит *дерево поиска в ширину*, которое изначально состоит из одного корня, которым является начальная вершина *s*.

Алгоритм нахождения компонент связности *неориентированного* графа стратегией обхода графа методом поиска в глубину заключается в следующих действиях:

1. Числу компонент связности присваиваем начальное значение равное нулю.

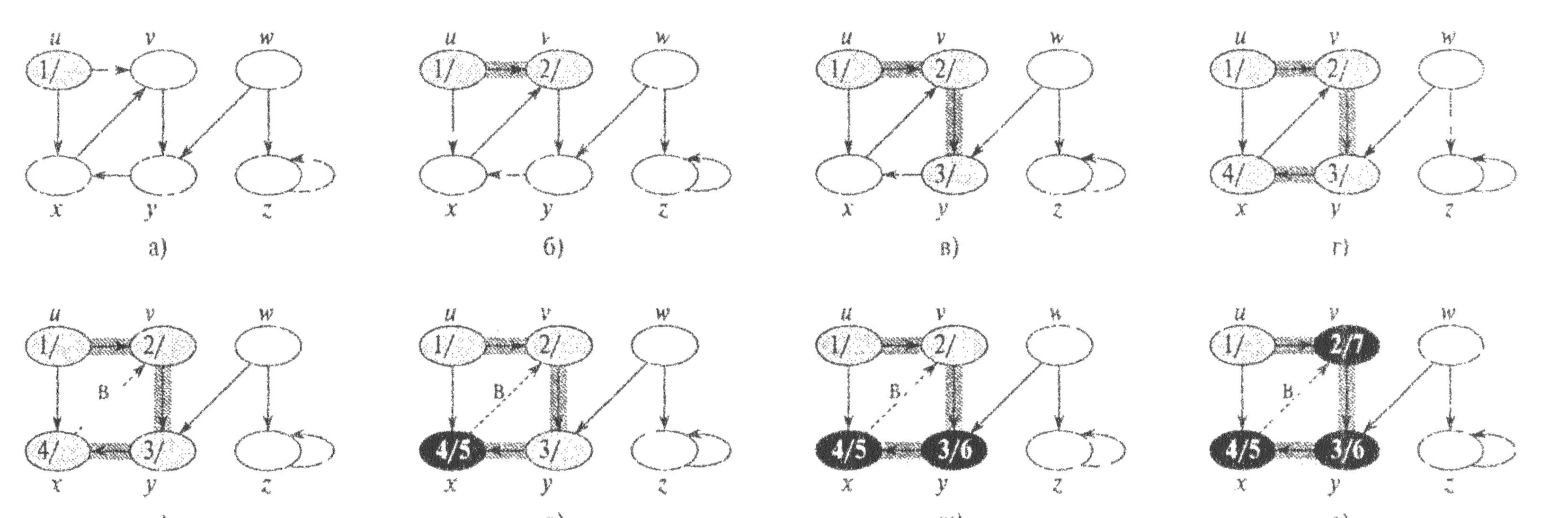
2. Выбираем произвольно начальную вершину и помечаем её как доступную.

3. Проверяем, если пути к другим непомеченным вершинам, если есть, то первую же такую вершину помечаем как доступную и переходим на нее, и далее выполняем пункт *4*. В противном случае:   
 а) начальную вершину помечаем как пройденную.   
 б) инкриминируем число компонент связности;

в) проверяем, остались ли в графе непомеченные вершины и если  
 остались, продолжаем алгоритм со второго пункта для   
 непомеченных вершин, иначе алгоритм закончен.

4. Проверяем, если пути от нее к другим непомеченным вершинам, если есть, то первую же такую вершину помечаем как доступную и переходим на нее, иначе помечаем текущую вершину как пройденную и переходим на предыдущую вершину и снова выполняем этот пункт (если возврат происходит в первую вершину, то переходим к пункту три).

Рассмотрим шаги обхода в глубину для *ориентированного* графа с окрашиваемыми вершинами, *рис.* *4*. Напомним, что от вершины к вершине мы идем *по стрелкам*.



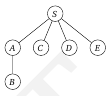
*Рис. 4.*

Поиск в глубину строит *дерево поиска в глубину*.

В заключение рассмотрим неориентированный граф на *рис.* *5* и его дерево поиска в ширину, рисунок *6.а* и дерево поиска в глубину, *рис.* *6.б*.



*Рис. 4.*

  
 *Рис 5а. Дерево поиска в ширину. Рис 5б. Дерево поиска в глубину.*

***Задание.***

1. Построить *матрицу смежности* и *инцидентности* для заданного графа. Изобразить граф.
2. По матрице смежности (инцидентности) для каждой из вершин вычислить ее *степень*.
3. Используя поиск в глубину и поиск в ширину написать программу, определяющую *число компонент связности* графа. Методы представляются в виде отдельных функций (или классов).
4. Построить *деревья поиска в ширину и глубину*.
5. Из заданного неориентированного графа построить произвольным образом ориентированный граф (добавить к каждому ребру стрелку). Для полученного таким образом ориентированного графа построить *матрицу смежности и инцидентности*.
6. Из заданного неориентированного графа построить произвольным образом *псевдограф*.
7. Варианты заданий указаны в *таблице 1*.
8. В таблице граф задан списком ребер, например, запись *(1,2)*означает, что существует ребро, соединяющее вершину *1* с вершиной 2.

***Вопросы к лабораторной работе*** (*отвечать письменно, ответы обосновывать*).

1. В каком из методов обхода графа путь в дереве поиска соответствует кратчайшему (т.е. содержащему наименьшее количество ребер) пути от вершины *s* до вершины *v*.
2. Для какого из обходов строится единственное (с точностью до изоморфизма) дерево поиска, а для какого можно построить их несколько.
3. Какое из утверждений является верным:  
   *а) Полный граф всегда является регулярным графом;  
   б) Регулярный граф всегда является полным графом.*
4. Что показывает сумма строк, столбцов матриц смежности и инцидентности для простого графа?

*Таблица 1.*

|  |  |  |  |
| --- | --- | --- | --- |
| ***№*** | ***Кол.***  ***вершин*** | ***Кол.***  ***ребер*** | ***Задание***  ***графа*** |
| *1.* | *9* | *10* | *(1,2),(1,4),(1,6),(2,3), 2,5),(3,5),*  *(4,5),(4,6),(7,8),(7,9)* |
| *2.* | *6* | *8* | *(1,2),(1,4),(1,5),(1,6),(2,3),(3,4),*  *(4,5),(5,6)* |
| *3.* | *8* | *8* | *(1,2),(1,3),(2,3),(3,4),(3,5)(4,5),*  *(5,6)(7,8)* |
| *4.* | *8* | *6* | *(1,6),(1,8),2,5),(3,6),(3,8),(4,5)* |
| *5.* | *6* | *10* | *(1,2),(1,3),(1,4),(2,3),(2,4),(3,4),*  *(3,5),(4,5),(4,6),(5,6)* |
| *6.* | *9* | *8* | *(1,2),(1,3),(1,4), (2,3),(2,4),(3,4),*  *(5,6),(6,7),(8,9)* |
| *7.* | *8* | *7* | *(1,4),(2,4),(2,6),(3,6),(4,5),(5,6)*  *(7,8)* |
| *8.* | *6* | *10* | *(1,2),(1,3),(2,3),(2,4), (2,5),(3,4),(3,5),*  *(4,5),(4,6),(5,6)* |
| *9.* | *8* | *8* | *(1,2),(1,3),(1,4),(2,3), (2,4),(3,4),*  *(5,7),(6,8)* |
| *10.* | *6* | *8* | *(1,2),(1,4),(2,3), (2,4),(3,4),*  *(3,6),(4,5),(5,6)* |
| *11.* | *8* | *6* | *(1,3),(2,3),(3,4),(4,5),(5,6),(7,8)* |
| *12.* | *7* | *12* | *(1,2),(1,3),(1,4),(1,6),(2,3), (2,4),(2,6),*  *(2,7),(3,4),(4,5),(5,6),(6,7)* |
| *13.* | *9* | *10* | *(1,2),(1,3),(2,3), (3,4),(4,5),*  *(4,6),(3,4),(7,8),(7,9),(8,9)* |
| *14.* | *11* | *10* | *(1,2),(2,4),(2,5),(3,4),(4,7),*  *(5,6),(5,7),(7,8),(9,10),(9,11)* |

***Литература к лабораторной работе.***

1. Т. Кормен и др. Алгоритмы. Построение и анализ.
2. С Дасгупта и др. Алгоритмы.
3. С. Скиена. Алгоритмы. Руководство по разработке.
4. Р Стивенс. Алгоритмы. Теория и практическое применение.