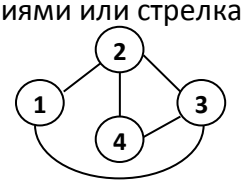
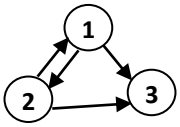
Занятие №10. Графы I. Определения, хранение

Многие, совершенно различные системы реального мира, например хорошо представляются при помощи простой модели. Точками (удобно изображать их кружками) и линиями или стрелками, которые соединяют некоторые из них.

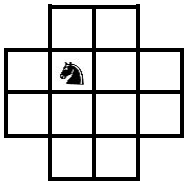




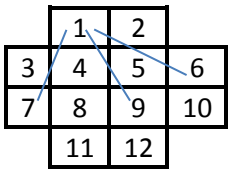
Такая модель называется **графом** (неориентированным или ориентированным).

При помощи графов решаются самые разные задачи. Например:

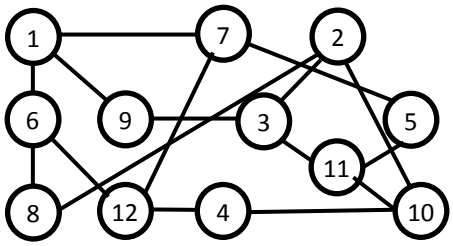
Можно ли доску 4х4 с вырезанными уголками и обойти шахматного коня и вернуться на исходную клетку, побывав на всех клетках ровно по одному разу?



Пронумеруем клетки – это будут вершины графа.

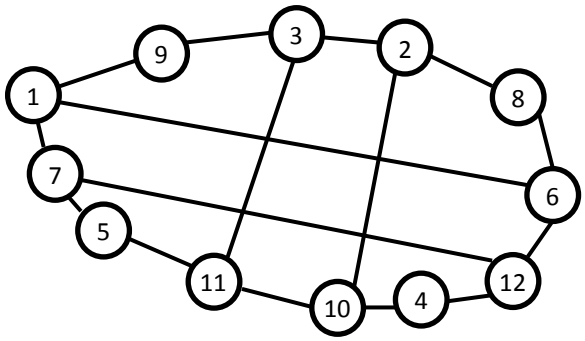


И соединим линиями-ребрами «соседние», в которые конь может попасть за один ход. На рисунке показаны только соседние с первой вершиной.



Во-первых, мы видим, что граф связный из любой клетки можно «дойти» до любой. Но вот можно ли пройти по всем клеткам по одному разу и вернуться? Такой путь называется «*гамильтонов цикл*».

Нам неважно точное месторасположение вершин. В том смысле, что важно, какие именно вершины соединены ребрами, но не где «на плоскости» они находятся. Так же неважны «длина», «форма» и «места крепления» ребер. В этом состоит удобство использования графа для решения. Перерасположим вершины, чтобы было наглядно.



Теперь видно, что и ответ на вопрос задачи: «Да». Достаточно обойти вершины «по кругу», начиная с любой, возможны и другие пути. Решать задачу при помощи графа оказалось значительно удобнее, чем на доске.

Конечно, мы решили эту задачу не алгоритмически, при помощи интуиции. Задача поиска гамильтонова цикла на произвольном графе называется имеет название «задачи коммивояжера» и достаточно сложна.

Немного теории

Дадим формальное определение графов для математиков.

Граф, или неориентированный граф G — это упорядоченная пара (V, E) , для которой выполнены следующие условия:

V — **это непустое множество вершин** или узлов;

E — **это множество пар** (в случае неориентированного графа — неупорядоченных) **вершин**, называемых рёбрами.

Ориентированный граф (сокращенно **орграф**) G — это упорядоченная пара (V, A), для которой выполнены следующие условия:

V — **это непустое множество вершин** или узлов;

A — **это множество упорядоченных пар вершин**, называемых дугами.:

Дуга — это упорядоченная пара вершин (v, u), где вершину v называют началом, а w— концом дуги. Можно сказать, что дуга $v \to w$ ведёт от вершины v к вершине w.

Основные понятия

Перечислим основные понятия, связанные с графами.

(Нарисуйте рядом с каждым определением пример)

Петля это дуга, начальная и конечная вершина которой совпадают.

Простой граф граф без кратных ребер и петель.

Степень вершины это удвоенное количество петель, находящихся у этой вершины плюс количество остальных прилегающих к ней ребер.

Пустым называется граф без ребер. **Полным** называется граф, в котором каждые две вершины смежные.

Путь в ориентированном графе - это последовательность дуг, в которой конечная вершина всякой дуги, отличной от последней, является начальной вершиной следующей.

Конечные вершины в пути называются связанными данным путем (или просто связанными). Если они совпадают, то путь называют **замкнутым**. Количество дуг в пути называется **длиной пути**.

Маршрут в графе - путь, ориентацией дуг которого можно пренебречь.

Цепь - маршрут, в котором все ребра попарно различны.

Цикл - замкнутый маршрут, являющийся цепью.

Маршрут, в котором все вершины попарно различны, называют **простой цепью**. Цикл, в котором все вершины, кроме первой и последней, попарно различны, называются **простым циклом**.

Подграф - это граф, который содержит некоторые вершины исходного графа и некоторые ребра (только те, оба конца которых входят в подграф)

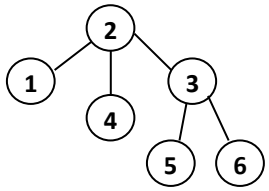
Граф называется связным, если любая пара его вершин связана.

Связными компонентами графа называются подграфы данного графа, вершины которых связаны.

Деревья

Очень важным классом графов являются деревья.

Неформально дерево это граф, похожий на генеалогическое дерево у каждой вершины может быть только один «отец» («родитель») и много «потомков»



Вершины степени 1 в дереве называются **листьями**. На приведенном рисунке листьями являются вершины 1, 4, 5, 6.

Формально дерево определяют несколькими способами.

Дерево – это связанный граф без циклов.

Дерево – это простой граф, в котором N вершин и N-1 ребро.

Дерево - это связный граф но после удаления *любого* ребра становится несвязным.

Они эквиваленты, то есть можно из любого из этих определений вывести (доказать) другие как свойства и признаки.

Как хранить граф в памяти компьютера? Существует несколько способов, каждый из которых имеет свои достоинства и недостатки.

Способы хранения графов

Способ №0. Иногда граф можно вообще не хранить специальным образом.

Например, «с точки зрения» шахматного короля шахматная доска - это граф, у которого вершины которого это клетки, а ребра связывают вершины-клетки с соседними координатами. А в случае ладьи ребра связывают все вершины-клетки, лежащие на одной вертикали и горизонтали. В этом случае достаточно иметь в памяти только описание шахматной доски (например, в виде квадратной таблицы), а хранить информацию о ребрах не нужно. Можно каждый раз определять наличие ребра, просто сравнивая координаты клеток между собой.

Способ №1. Матрица смежности

Граф хранится в виде квадратной таблицы, со стороной, равной количеству вершин. В пересечении строки i и столбца j находится 1, если дуга (ребро) между вершинами i и j есть и 0, если нет. Таким образом, описание графа без петель содержит нули на главной диагонали, а матрица неориентированного графа относительно нее симметрична.

Постройте ориентированный граф по следующему описанию:

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

(Сверьте свой ответ с Подсказкой 10.1.)

Способ №2. Список ребер

Граф хранится парами чисел-номеров вершин. Каждая пара соответствует дуге. Например пара (3, 4) соответствует дуге из третьей вершины в четвертую. Запишите граф из способа №1 списком ребер. _____ Сверьте свой ответ с Подсказкой 10.2.

Способ №3. Списки смежности

Этот способ похож на первый, только вместо прямоугольной таблицы храняться строки с номерами вершин, в которую из данной есть дуга. Для графа, представленного матрицей смежности в способе №1 списки смежности выглядят так:

2	3
1	4
2	
1	

В языке Java этот способ очень удобно реализовывать неровными массивами.

Все описанные способы часто используются.

Третий способ намного экономичней первого, хотя матрицу смежности проще обрабатывать. Поэтому если есть возможность использовать матрицу смежности, видимо, лучше использовать ее, ошибок будет меньше. Если же в задаче предъявляются жесткие требования к объему памяти или для задачи критично время выполнения, а конкретный алгоритм работает гораздо быстрее именно на списках смежности – нужно использовать третий способ хранения графов. Наконец, во входных данных в задачах графы очень часто передаются списками ребер.

Немного о задачах

Практически все задачи этого занятия простые, на технику.

Прежде всего, нужно научиться быстро переводить графы из одного представления в другое и устанавливать их простейшие свойства (является ли граф ориентированным, есть ли в нем петли и так далее). Чаще всего это сводится к простым операциям с двумерным массивом – матрицей смежности, поэтому ко многих задачам приведены лишь краткие указания.

Особо интересна последняя задача «Дерево ли?». Попробуйте написать самостоятельно, не заглядывая в разбор, программу, которая выясняет по матрице смежности графа, является ли он деревом.

Задачи

Задача A. Петли

По заданной матрице смежности неориентированного графа определите, содержит ли он петли.

Формат входных данных

На вход программы поступает число n ($1 \le n \le 100$) – количество вершин графа, а затем n строк по n чисел, каждое из которых равно 0 или 1, – его матрица смежности.

Формат выходных данных

Выведите «YES», если граф содержит петли, и «NO» в противном случае.

Примеры

Входные данные	Выходные данные
3 0 1 1 1 0 1 1 1 0	NO
3 0 1 0 1 1 1 0 1 0	YES

Задача B. Проверка на неориентированность

По заданной квадратной матрице $n \times n$ из нулей и единиц определите, может ли данная матрица быть матрицей смежности простого неориентированного графа.

Формат входных данных

На вход программы поступает число n ($1 \le n \le 100$) – размер матрицы, а затем n строк по n чисел, каждое из которых равно 0 или 1, – сама матрица.

Формат выходных данных

Выведите «YES», если приведенная матрица может быть матрицей смежности простого неориентированного графа, и «NO» в противном случае.

Примеры

Входные данные	Выходные данные
3 0 1 1 1 0 1	YES

1 1 0	
3 0 1 0 1 0 1 1 1 0	NO
3 0 1 0 1 1 1 0 1 0	NO

Задача C. От матрицы смежности к списку ребер, неориентированный вариант

Простой неориентированный граф задан матрицей смежности, выведите его представление в виде списка ребер.

Формат входных данных

Входные данные включают число n ($1 \le n \le 100$) – количество вершин в графе, а затем n строк по n чисел, каждое из которых равно 0 или 1, – его матрицу смежности.

Формат выходных данных

Выведите список ребер заданного графа (в любом порядке).

Примеры

Входные данные	Выходные данные
3 0 1 1 1 0 1 1 1 0	1 2 2 3 1 3

Задача D. От списка ребер к матрице смежности, неориентированный вариант

Простой неориентированный граф задан списком ребер, выведите его представление в виде матрицы смежности.

Формат входных данных

На вход программы поступают число n ($1 \le n \le 100$) – количество вершин в графе и поступает число m ($1 \le m \le \frac{n(n+1)}{2}$) – количество ребер. Затем следует m пар чисел – ребра графа.

Формат выходных данных

Выведите матрицу смежности заданного графа.

Примеры

Входные данные	Выходные данные
3 3 1 2 2 3 1 3	0 1 1 1 0 1 1 1 0

Задача E. Степени вершин

Неориентированный граф задан матрицей смежности. Найдите степени всех вершин графа.

Формат входных данных

Сначала вводится число число n ($1 \le n \le 100$) – количество вершин в графе, а затем n строк по n чисел, каждое из которых равно 0 или 1, – его матрица смежности.

Формат выходных данных

Выведите n чисел – степени вершин графа.

Примеры

Входные данные	Выходные данные
3 0 1 0 1 0 1 0 1 0	1 2 1

Задача F. Истоки и стоки

Напомним, что вершина ориентированного графа называется истоком, если в нее не входит ни одно ребро и стоком, если из нее не выходит ни одного ребра.

Ориентированный граф задан матрицей смежности. Найдите все вершины графа, которые являются истоками, и все его вершины, которые являются стоками.

Формат входных данных

Сначала вводится число число n ($1 \le n \le 100$) – количество вершин в графе, а затем n строк по n чисел, каждое из которых равно 0 или 1, – его матрица смежности.

Формат выходных данных

В первой строке выведите k – число истоков в графе и затем k чисел – номера вершин, которые являются истоками, в возрастающем порядке. Во второй строке выведите информацию о стоках в том же порядке.

Примеры

Входные данные	Выходные данные
4 1 0 0 1 0 0 0 0 1 1 0 1 0 0 0 0	1 3 2 2 4

Задача G. Транзитивность неориентированного графа

Граф называется транзитивным, если всегда из того, что вершины *u* и *v* соединены ребром и вершины *v* и *w* соединены ребром следует, что вершины *u* и *w* соединены ребром.

Проверьте, что заданный неориентированный граф является транзитивным.

Формат входных данных

Сначала вводится число число *n* ($1 \le n \le 100$) – количество вершин в графе и число *m* ($1 \le m \le \frac{n(n+1)}{2}$) – количество ребер. Затем следует m пар чисел – ребра графа.

Формат выходных данных

Выведите «YES», если граф является транзитивным, и «NO» в противном случае.

Примеры

Входные данные	Выходные данные
3 3 1 2 1 3 3 2	YES
3 2 1 2 1 3	NO

Задача J. Дерево ли?

Неориентированный граф без петель и кратных ребер задан матрицей смежности. Определить, является ли этот граф деревом.

Формат входных данных

Сначала вводится число *N* – количество вершин графа (от 1 до 100). Далее записана матрица смежности размером *N***N*, в которой 1 обозначает наличие ребра, 0 – его отсутствие. Матрица симметрична относительно главной диагонали.

Формат выходных данных

Введите сообщение YES, если граф является деревом, и NO в противном случае.

Пример

Входные данные	Выходные данные
3 0 1 0 1 0 1 0 1 0	YES

Подсказки и решения. 10 Занятие.

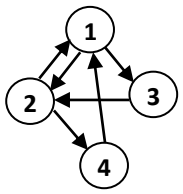
10.1

0 1 1 0

1 0 0 1

0 1 0 0

1 0 0 0



10.2

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

10.2

Действительно так. Если вершины связаны, то кратчайший путь из одной в другую должен содержать не более N-1 ребер. А наша операция каждый раз удлиняет путь на один.

Разбор. Занятие 10

Во всех разборах этого занятия считается, что в переменой N лежит количество вершин графа

10A. Петли

В этой задаче можно считать матрицу смежности в двумерный массив и проверить, чтобы все элементы `a[i][i]` были равны 0. Но можно и не сохранять матрицу, а выяснить требуемое прямо во время считывания. Просто проверяя на 0 каждое N+1-ое считываемое число:

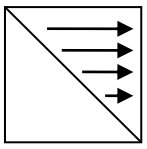
```
boolean loop = false; // петель пока не обнаружено
for (int i = 0; i < N * N; i++)
{
     int x = in.nextInt();
     if (i % (N + 1) == 0 && x != 0)
     {
          loop = true;
          break;
     }
}
```

10B. Проверка на неориентированность

После считывания нужно проверить матрицу смежности на симметричность относительно главной диагонали (как в задаче «Симметрична ли матрица?» из занятия по двумерным массивам) и установить отсутствие петель, так как по условию граф простой.

10C. От матрицы смежности к списку ребер, неориентированный вариант

Главное при решении этой задачи не выводить каждое ребро дважды. Это можно сделать, например, обходя только половину матрицы смежности, например, верхний правый треугольник:



```
for (int i = 0; i < N; i++)
     for (int j = i; j < N; j++)
     {
          if (m[i][j] == 1) …// выводим i+1 j+1
     }
```

10D. От списка ребер к матрице смежности, неориентированный вариант

Считываем пару чисел i, j и так как граф неориентированный, то проставляем единички в ячейках m[i][j] и m[j][i]. Практически полностью код приведен в Справочнике.

10E. Степени вершин

Несложно понять, что нужно посчитать и вывести количество единичек в каждой строке.

10F. Истоки и стоки

Надо посчитать количество строк, состоящих из одних нулей по вертикали и по горизонтали и вывести их номера. При подсчете можно добавлять эти номера в массив, а можно пройтись по матрице смежности дважды, ограничения в условии задачи это позволяют.

10G. Транзитивность неориентированного графа

Переберем все тройки вершин (перебирать будем без повторений – k > j > i) и посчитаем количество ребер их соединяющих.

```
for (int i = 0; i < N; i++)
     for (int j = i + 1; j < N; j++)
          for (int k = 0; k < N; k++)
          {
               int n = m[i][j] + m[i][k] + m[j][k];
               // ...анализируем n
          }
}
```

Если нет ни одного ребра в данной тройке (`n == 0`), то идем дальше, так как ничего проверять не надо (никакие из ребер не соединены).

Если есть одно ребро (`n == 1`), то следовательно соединены только две вершины между собой и третья не затронута, поэтому идем дальше.

Если два ребра, то с одной вершиной соединены две вершины (`n == 2`), а другие две между собой нет, что противоречит транзитивности графа, выводим NO и запоминаем это.

Если три ребра, то получаем полный трехвершинный подграф (`n == 3`), он транзитивный.

После окончания циклов, если не встретили неправильной тройки, выводим YES.

Таким образом, единственная проверка, которую надо делать – это `n == 2`.

10J. Дерево ли?

Для проверки того, что граф является деревом, проще всего проверить, что граф содержит N-1 ребро и является связанным.

Количество ребер считается просто – достаточно посчитать в матрице смежности количество единичек и разделить результат на два.

Связанность же установить сложнее. Можно, например, действовать так. Заведем массив «`c`» на `N` ячеек и заполним его нулями. В нулевую ячейку поставим 1 (нулевая вершина связана сама с собой). Пройдемся по первой строке и, если в `j`-й ячейке первой строки матрицы смежности стоит 1 - поставим в `c[j]` единицу.

То же сделаем для каждой строки с номером j, если в `c[j]` стоит 1. То есть

```
for (int j = 0; j < N; j++)
     if (c[j] == 1)
     {
          for (int i = 0; i < N; i++)
          {
               if (m[j][i] == 1) c[i] = 1;
          }
     }
}
```

Эту операцию нужно проделать максимум N-1 раз. Почему? См. Подсказку 10.2.

Если после этого в массиве с останется хотя бы один ноль – граф не связанный, и надо выводить «NO».

Время работы такого алгоритма *$O(n^3)$. На самом деле это наивная реализация известного алгоритма - поиска в ширину.*

Занятие 10.Справочник

Чтение графа, заданного списком ребер в матрицу смежности

```
// ...перед этим создаем двумерный массив m для матрицы
смежности
N = in.nextInt(); // считываем количество ребер
for (int i = 0; i < N; i++)
{
     int v = in.nextInt();
     int w = in.nextInt();
     m[v - 1][w - 1] = 1
     // если граф неориентированный, то:
     m[w - 1][v - 1] = 1;
}
```