**Задачи DFS в неориентированном невзвешенном графе.**

**Замечание:** невзвешенные графы определяются матрицей смежности, содержащей только 0 или 1. Поэтому матрицу весов, с которой мы работали надо сначала преобразовать к матрице смежности.

Для неориентированных графов матрица смежности симметричная относительно главной диагонали, т.к. ребро (v,w) равно ребру (w,v).

1. Дан лабиринт, который можно представить в виде графа. (можно взять любой другой лабиринт).

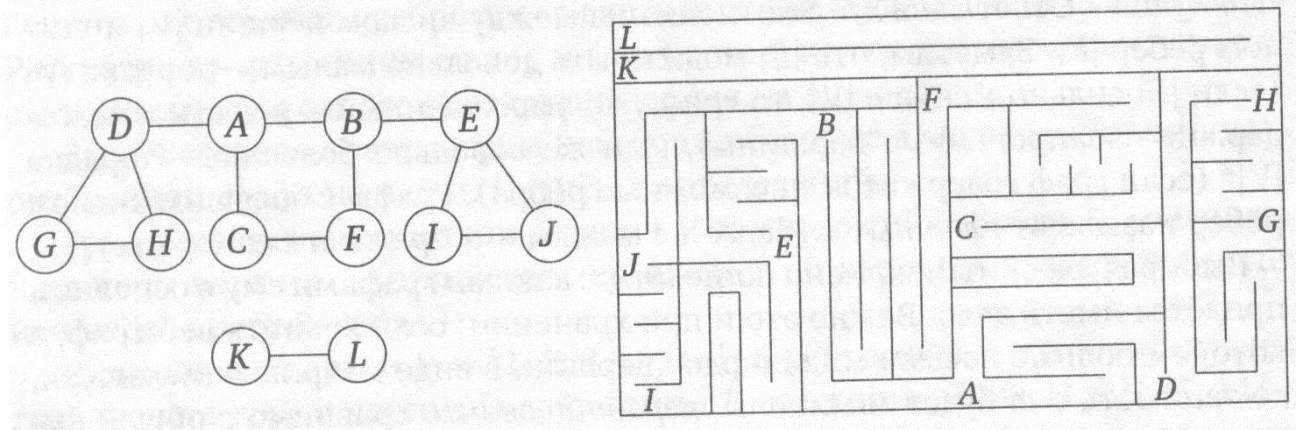


Рис. Лабиринт и соответствующие ему граф.

Определить для вершины X **любой** путь до всех вершин графа (кроме X). Если путь существует вывести длину пути и список вершин, содержащихся в найденном пути. Если таких путей несколько, то вывести только один из них. Если путь не существует вывести только его длину, т.е. 0.

Пояснение: Т.к. граф не взвешенный, то наличие ребра определяем 1, а отсутствие 0. Например, длина минимального пути от вершины A до вершины B равна 1, от A до J равна 3, от A до K длина пути равна 0. Из вершины А до вершины В есть еще один путь АСFB, длиной 3.

1. Решить задачу построения леса поиском в глубину.

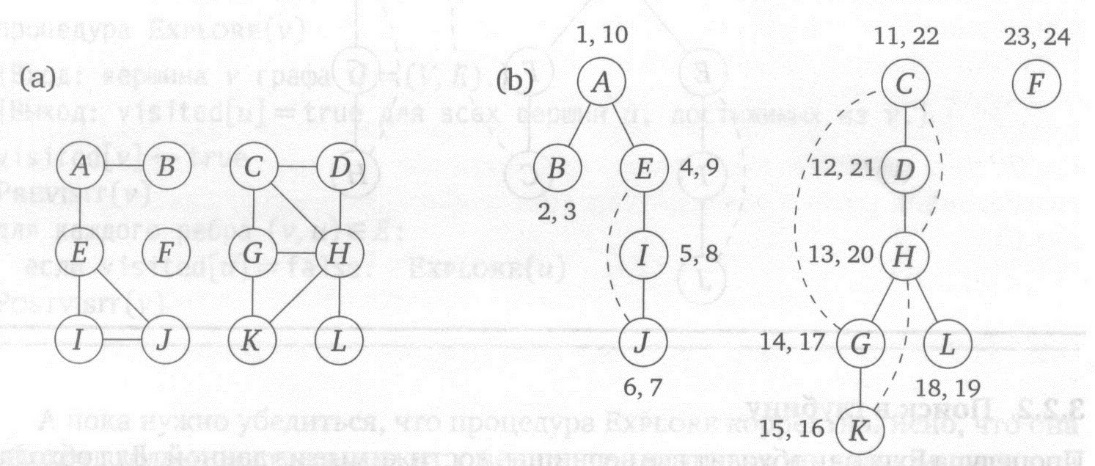


Рис. а) Граф на двенадцати вершинах b) Лес, построенный поиском в глубину.

Каждую компоненту связности вывести с новой строки. Для каждой вершины указать два числа: время входа в вершину и время выхода из вершины. Для этого используйте счетчик clock, массив pre[v] и post[v].

1. Дан невзвешенный граф и две вершины X и Y. Определить длину минимального пути из вершины X до вершины Y. Вывести все вершины, содержащиеся в найденном минимальном пути.
2. Дан невзвешенный граф. Определить для каждой вершины число доступных вершин. Например, для вершины A (рис.2) число доступных вершин равно 4, для вершины F – 0, для вершины C – 5.
3. Неориентированный граф, состоящий из двух компонент связности, всегда можно сделать связанным, добавив одно ребро. Составить программу, которая при наличие двух компонент связности графа добавляет такое ребро. Такое ребро называется мостом.
4. Определить все мосты графа

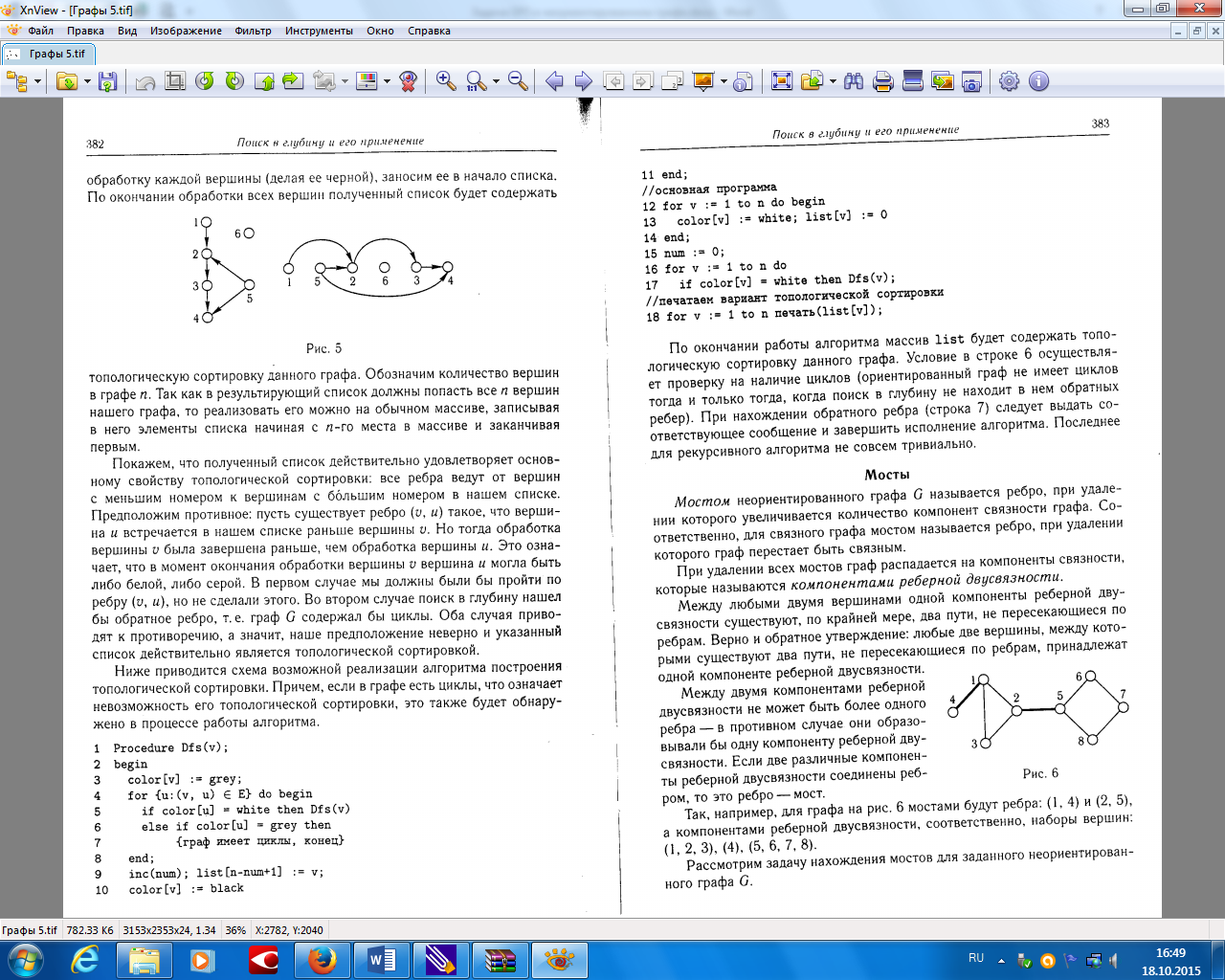


Рис. . Мосты в графе - (1,4) и (2,5)

1. *Степенью* (degree) вершины *d(v)* вершины *v* неориентированного графа *G* называется количество соседей *v* или, как говорят, инцидентных *v* рёбер. Для неориентированного графа выполнено равенство , где *V* – множество вершин, *E* –множество ребер.

Дан связанный граф G (количество компонент связности равно 1). В связном графе из любой вершины доступны все вершины графа, т.е. за один обход из вершины X мы пройдем через все вершины. Пусть такой обход завершается в вершине Y (может быть X=Y).

Пусть в нашем связном графе G есть вершины четной и нечетной степени.

Написать программу, которая запустит DFS из всех вершин X нечетной степени. Определить степень вершины Y. Вывести X, Y, степень Y.

1. Составить программу, доказывающую, что в каждом связном неориентированном графе найдется вершина, удаление которой оставляет граф связанным.
2. Составить программу, которая определит цикл максимальной длины в графе *G*, содержащий вершину *v*.
3. Составить программу, которая определит цикл максимальной длины в графе *G*, содержащий ребро *e*.
4. Составить программу, которая в данном графе G, найдет все циклы, содержащие три вершины (три ребра), т.е. циклы длиной 3. Цикл ABC и цикл BCA, ACB и т.д. считать одинаковыми.
5. *Переливание воды*. Есть три сосуда ёмкостью 10, 7 и 4 литров. Изначально первый сосуд пуст, а оставшиеся два полностью наполнены водой. За один ход разрешается переливать из одного сосуда воду в другой до тех пор, пока первый не станет пустым или же второй не заполнится до верха. Мы хотим проверить, существует ли последовательность ходов, в результате которой в одном из последних двух сосудов останется ровна два литра воды.

Сформулируйте данную задачу как задачу о графах: определите соответствующий граф и переформулируйте вопрос в его терминах. Найдите ответ и последовательность ходов, применив алгоритм на графах. Ваша программа должна решать задачу для любых емкостей. Если переливание невозможно надо вывести, например, 0.

1. *Эйлеровым циклом* неориентированного графа называется замкнутый путь, проходящий по всем ребрам графа ровно по одному разу. В 1736 году Леонард Эйлер решил знаменитую задачу о кёнигсбергских мостах, положившую начало теории графов. В Кёнигсберге (в настоящее время Калининград) два острова и два берега реки были связаны семью мостами, и возник вопрос: можно ли обойти все эти семь мостов и вернуться в исходную точку, пройдя по каждому из мостов *ровно один раз*?

Соответствующий граф приведён ниже. Отметим, что в нём есть кратные рёбра (несколько рёбер между парой вершин; мы такого не допускали).

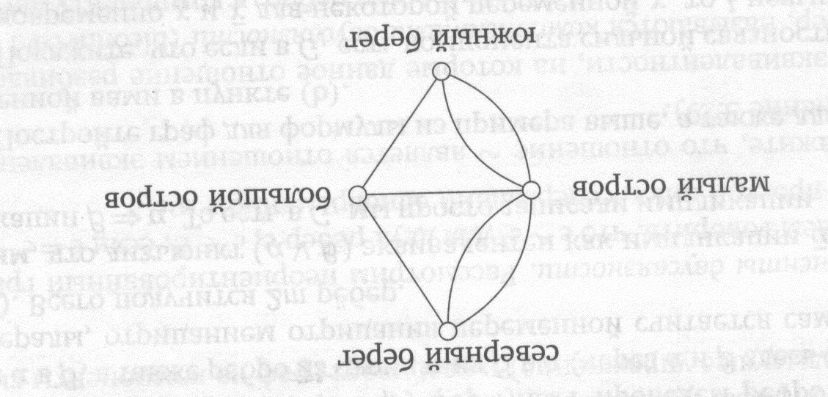


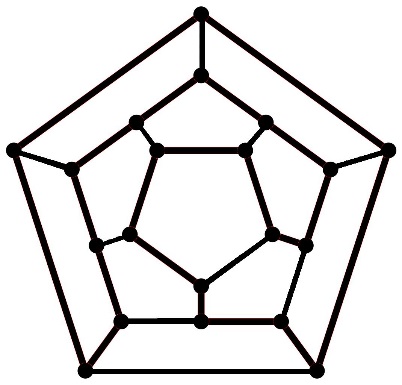
Рис. Мосты Кёнигсберга

Составить программу, которая проверяет утверждение, что в неориентированном графе *G* есть эйлеров цикл тогда и только тогда, когда он связан и степени всех его вершин чётны.

Примечание: кратность ребра можно хранить в матрице смежности, записав в соответствующие элементы матрицы значение равное 2 (для данного примера). В общем случае целое число равное кратности ребер.

1. *Эйлеровым путем* называется путь в графе, проходящий по каждому ребру графа ровно один раз. Составить программу, которая проверяет утверждение, что такой путь возможен в связном графе, если все вершины имеют четную степень или граф содержит ровно две вершины нечётной степени.
2. Пусть имеем так называемый граф дружбы или граф дружеских отношений. Каждая вершина – это студент *Zi*, а соседние вершины – друзья *Z*i. Составьте программу, которая выведет список студентов таким образом, чтобы рядом (справа или слева) стояли только друзья.
3. Разработайте алгоритм с линейным временем исполнения для удаления всех вершин второй степени из графа путем замены ребер *(u,v)* и *(v,w)* ребром *(u,w)*. Также нужно удалить множественные копии ребер, оставив только одно ребро. Обратите внимание, что удаление копий ребра может создать новую вершину второй степени, которую нужно будет удалить, а удаление вершины второй степени может создать кратные ребра, которые также нужно будет удалить.
4. Гамильто́нов граф — математический объект [теории графов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2). Представляет собой [граф](https://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D1%84_%28%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0%29), который содержит гамильтонов [цикл](https://ru.wikipedia.org/wiki/%D0%A6%D0%B8%D0%BA%D0%BB_%28%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2%29). При этом гамильтоновым циклом является такой цикл (замкнутый путь), который содержит все вершины данного графа.

Также с гамильтоновым графом тесно связано понятие гамильтонова пути, который является простым [путём](https://ru.wikipedia.org/wiki/%D0%A1%D0%BB%D0%BE%D0%B2%D0%B0%D1%80%D1%8C_%D1%82%D0%B5%D1%80%D0%BC%D0%B8%D0%BD%D0%BE%D0%B2_%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D0%B8_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2#.D0.9F) (путём без петель), проходящим через каждую вершину графа ровно один раз. Гамильтонов путь отличается от цикла тем, что у пути начальные и конечные вершины могут не совпадать, в отличие от цикла. Гамильтонов цикл является гамильтоновым путём.

В 1952 году было сформулировано условие Дирака существования гамильтонова пути: пусть *p* — число вершин в данном графе и ; если [степень](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B5%D0%BF%D0%B5%D0%BD%D1%8C_%D0%B2%D0%B5%D1%80%D1%88%D0%B8%D0%BD%D1%8B_%28%D1%82%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D0%B3%D1%80%D0%B0%D1%84%D0%BE%D0%B2%29) каждой вершины не меньше, чем , то данный граф — гамильтонов.

Составить программу, которая построит гамильтонов путь в неориентированном графе G. Условие существования гамильтонова пути необходимо проверить в начале алгоритма, чтобы не запускать поиск в холостую. Используйте граф G, содержащий, не более 20 вершин.

Постройте гамильтонов путь (или цикл), например, для додекаэдра.

Рис. 5 Додекаэдр

1. Решить задачу обхода шахматной доски конем. Фигура начинает движение с клетки (x,y) и возвращается в туже клетку (x,y), побывав в любой клетки только один раз и покрыв своими ходами всю шахматную доску. Таких гамильтоновых циклов несколько. Можно визуализировать получившийся цикл(ы). Очень красивая задача!! (на 10 баллов )

**Задачи DFS для ориентированных невзвешенных графов.**

Ориентированные невзвешенные графы отличаются от ориентированных тем, что матрица смежности будет не симметричная относительно главной диагонали, т.к. ребро (v,w) не равно ребру (w,v).

1. Составить алгоритм, который определяет, есть ли в данном ориентированном графе вершина, из которой достижимы все вершины графа.
2. Постройте алгоритм, который по двум вершинам ориентированного графа находит количество различных путей из *s* в *t*. (*V* – множество вершин, *E* – множество ребер)
3. Составить алгоритм, который по данному ориентированному графу G и его ребру e определит, есть ли в *G* цикл, содержащий *e*.
4. Построить алгоритм нахождения цикла нечётной длины в ориентированном графе.
5. Дан ориентированный граф, каждой вершине которого приписана цена (price) , представляющая собой целое положительное число. Определим массив cost следующим образом:

Cost[u] = минимальная цена вершин, достижимых из u (включая u).

Например, значения массива cost для вершин A,B,C,D,E,F графа на рисунке равны 2, 1, 4, 1, 4, 5. (Цены указаны около вершин.)

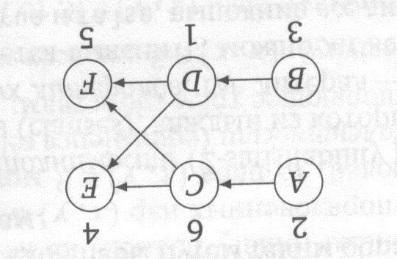


Рис. Граф к задаче про стоимость вершин

Необходимо заполнить массив cost для всех вершин графа.

1. Власти сделали все дороги города односторонними и утверждают, что от любого перекрестка по-прежнему можно добраться до любого другого, не нарушая правила. Как быстро их разоблачить? Сформулируйте данную задачу на языке графов и покажите, как её можно решить.

В ответ на претензии власти заявили, что имели в виду другое: куда бы ни поехать от мэрии по правилам, можно будет вернуться, не нарушая правил. Как проверить это утверждение?

1. Магистерская программа по информатике состоит из n семестровых курсов. Граф G отображает зависимости: вершины графа соответствуют курсам, из v идет ориентированное ребро в w, если w можно изучать только после v. Постройте алгоритм, который по G определит минимальное количество семестров, необходимое для изучения всей программы (в одном семестре может быть сколько угодно курсов).
2. Перед вами стоит задача выстроить n непослушных детей друг за другом, причем у вас имеется список из m утверждений типа “*i* не дружит с *j*”. Если *i* не дружит с *j*, то будет разумно не ставить *i* позади *j*, т.к. *i* может что-нибудь кинуть в *j*. Если такое упорядочивание невозможно, алгоритм должен сообщить об этом.