Алгоритмы и структуры данных.

Домашняя работа №1.

X == 22;

Y == 4;

Z == 5;

Задание №1.

Условие:

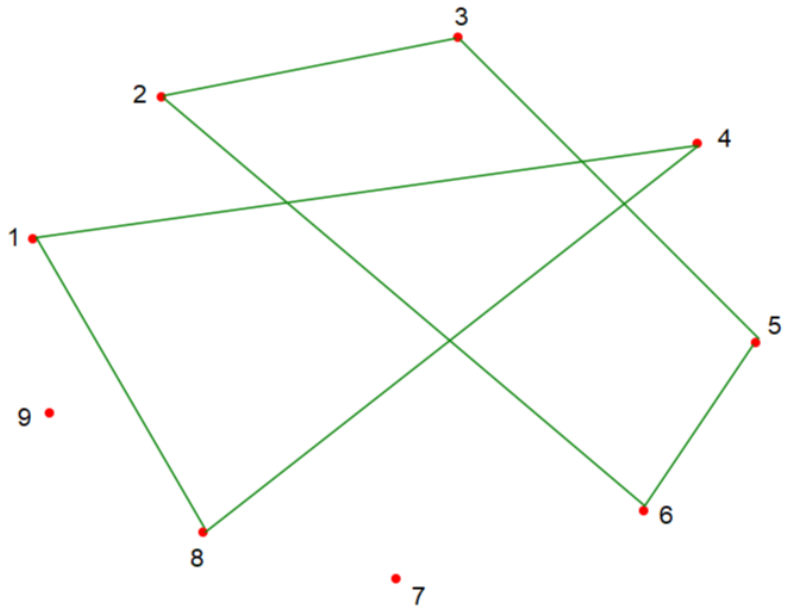
1. Приведите пример графа, в котором 5 + Y вершин, 2 цикла и Z + 2 ребер

Решение:

Количество вершин == 9;

Количество циклов == 2;

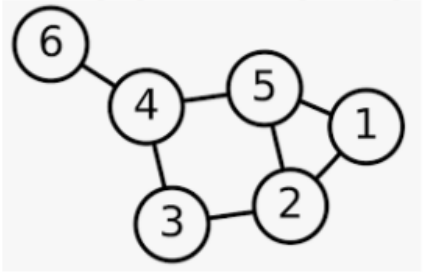
Количество ребер == 7;



Задание №2.

Условие:

2. Для графа с рисунка составьте матрицу смежности и список смежности. Опишите граф: кол-во вершин, ребер, степени вершин, циклы



Решение:

Матрица смежности:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \end{array} \begin{array}{c} \begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\ \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix} \end{array}$$

Список смежности:

1 – 2, 5

2 – 1, 3, 5

3 – 2, 4

4 – 3, 5, 6

5 – 1, 2, 4

6 – 4

Количество вершин = 6;

Количество ребер = 7;

Степени вершин:

У первой вершины степень 2;

У второй вершины степень 3;

У третьей вершины степень 2;

У четвертой вершины степень 3;

У пятой вершины степень 3;

У шестой вершины степень 1.

Циклы:

1) 1, 2, 3, 4, 5, 1
2) 1, 2, 5, 1
3) 2, 3, 4, 5, 2

Задание №3.

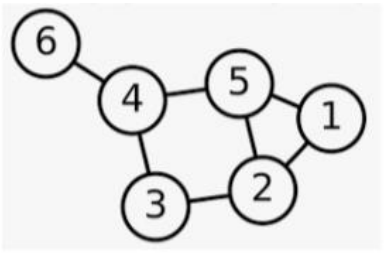
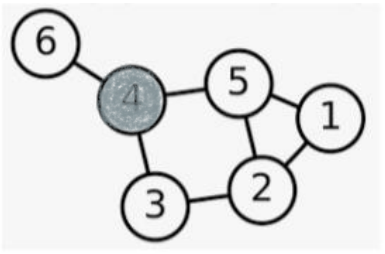
Условие:

3. Для графа с предыдущей картинки продемонстрируйте как будет работать DFS, если стартовать из вершины Y.

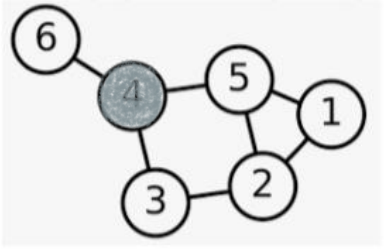
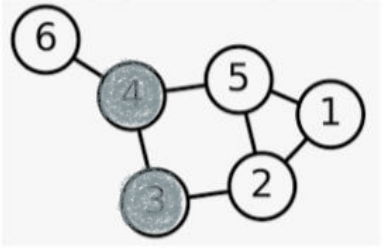
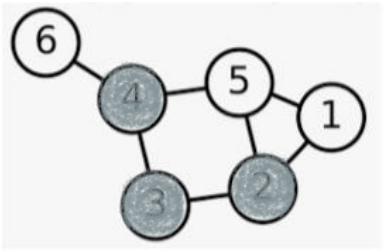
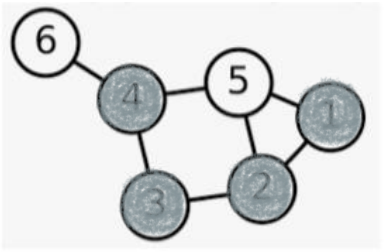
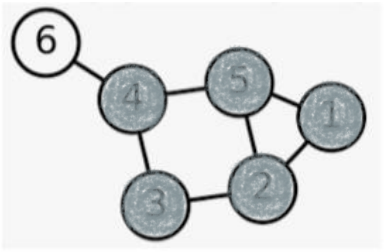
Y == 4;

Решение:

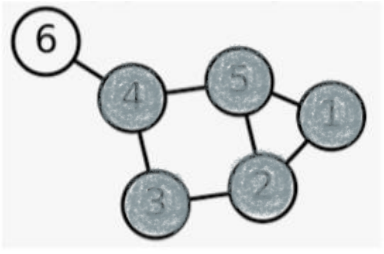
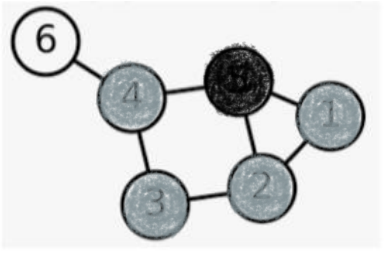
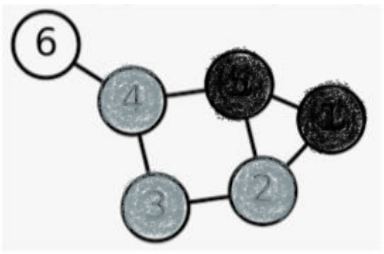
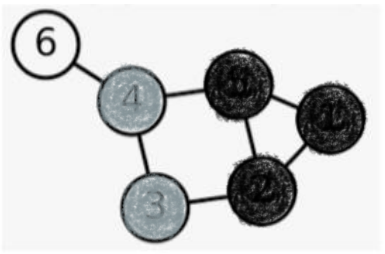
Пометим начальную вершину серым цветом.

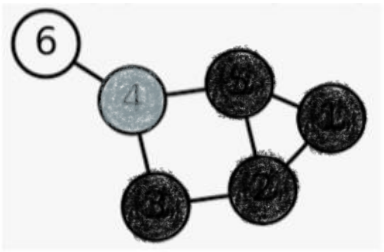



Запустим из нее просмотр остальных вершин смежных с ней. Если нашлась белая вершина, то переходим в нее и помечаем ее серым цветом. После чего повторяем эти действия пока не найдем вершину, которая не будет смежна с белыми вершинами.

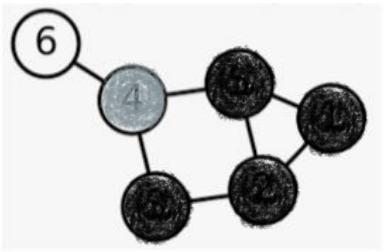






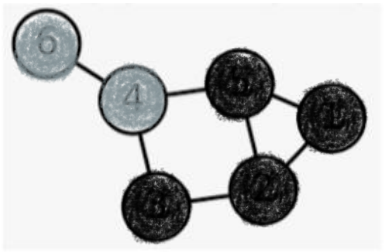
Когда нашли вершину, у которой нет смежных белых вершин, начинаем выкатываться из рекурсии, пока не дойдем до вершины у которой будет связь с белой вершиной, при этом помечаем пройденные вершины черным.

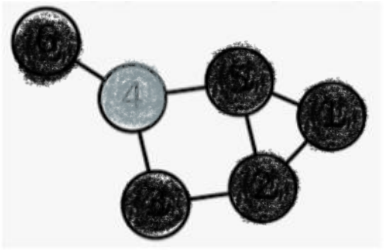





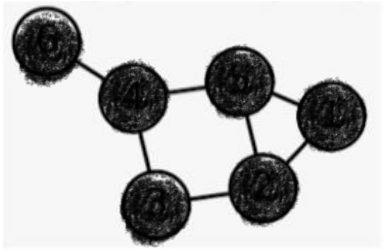


После того как мы пришли в вершину, из которой можно пойти в белую, начинаем повторять действия из предыдущих пунктов, а именно переходим в белую вершину, красим ее в серый цвет, если у нее есть смежные белые вершины, то повторяем эти действия, иначе начинаем выкатываться из рекурсии и помечать черным пройденные вершины.









После того как все вершины помечены черным обход в глубину завершен.

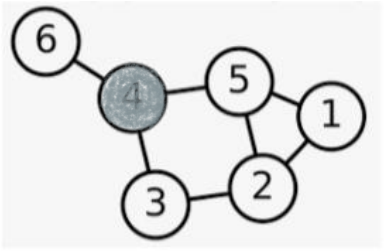
Задание №4.

Условие:

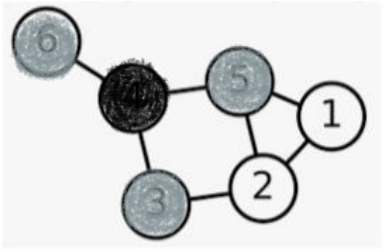
4. Для графа с предыдущей картинки продемонстрируйте, как будет работать BFS, если стартовать из вершины Y.

Y == 4;

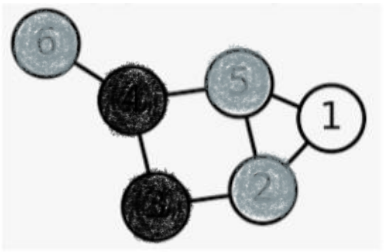
Решение:
Положим стартовую вершину в очередь и покрасим ее в серый цвет. Queue: [4]
Теперь достанем вершину из очереди. Затем все вершины смежные с ней, цвет которых белый покрасим в серый и положим в очередь. Покрасим вершину в черный цвет. Эти действия будут повторяться пока очередь не является пустой.



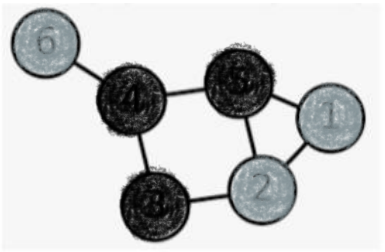
Queue: [4]



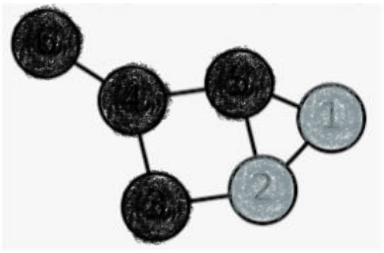
Queue: [3, 5, 6]



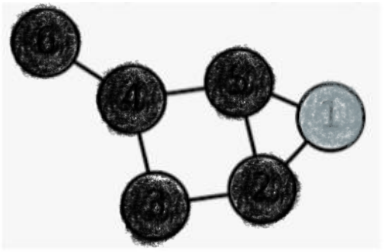
Queue: [5, 6, 2]



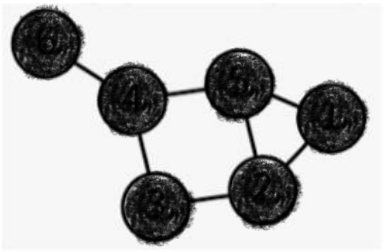
Queue: [6, 2, 1]



Queue: [2, 1]



Queue: [1]



Так как очередь пустая алгоритм завершается.

Задание №5.

Условие:

5. Инспектору нужно проверить состояние дорог в городе, для этого он хочет проехать по каждой дороге в каждую сторону (все дороги двусторонние и одинаковой длины). Постройте кратчайший путь. O(n + m).

Решение:

Данную задачу можно решить через обход в глубину. Мы запускаем обход в глубину из начальной точки и идем по вершинам записывая номера вершин в отдельную строку, никак не привязанную к вершинам, а еще увеличиваем счетчик на единицу, чтоб вывести длину пути. Когда закончится обход вершин у нас будет кратчайший путь, проходящий по всем вершинам, и его длина.

Задание №6.

Условие:

6. Инспектор живет в Москве и хочет приехать в с. Иваново. Для этого ему нужно понять, сколько бензина ему взять с собой. Представьте, что у вас есть карта дорог, на которой есть населенные пункты. Расстояния между всеми пунктами одинаковы. Помогите инспектору добраться до Иваново, взяв меньше всего бензина.

Решение:

Если расстояния между населенными пунктами одинаковые, то инспектор потратит меньше всего бензина если проедет по кратчайшему пути. Кратчайший путь можно найти,

с помощью обходов графов. Рассмотрим алгоритмы поиска кратчайшего пути с помощью обходов в ширину и в глубину.

1) DFS
   Мы запускаем обход графа в глубину и при этом добавляем счетчик в каждую вершину. Но если мы приходим в точку из которой достижимы серые вершины мы тоже в них заходим, чтобы сравнить счетчики и поставить в точке наименьший.
2) BFS
   Мы запускаем обход графа в ширину и при этом так же, как и в первом случае добавляем в каждую вершину счетчик. Но работать с счетчиками мы будем по другому, когда мы будем красить вершины в серый цвет мы будем увеличивать в них счетчик на единицу относительно значения точки из которой пришли.

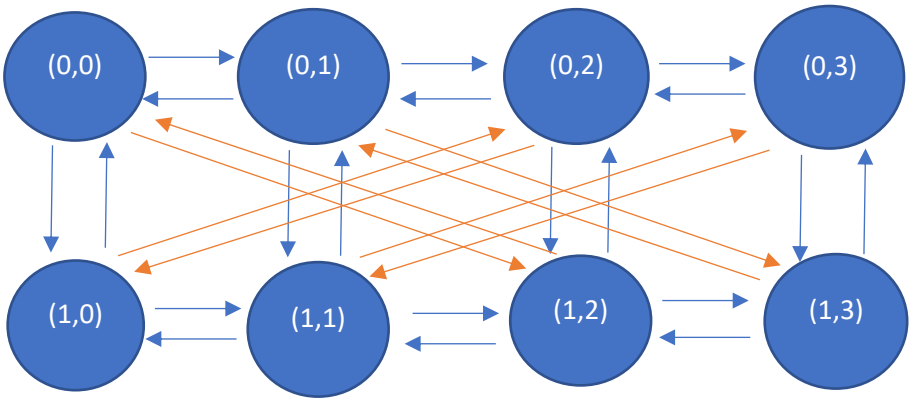
Задание №7.

Условие:

7. Пусть вы играете в шахматы на доске N*N. Вам нужно попасть конем из одной точки поля в другую за минимальное количество шагов. Предложите идею решения такой задачи.

Решение:

1) Идея первая
   Можно представить доску как список, но не обычный.



По факту мы имеем список, в котором у нас каждый элемент является вершиной графа. Синие связи показывают связь клеток, а каждая клетка поля является вершиной графа. Оранжевые связи показывают связь клеток, по которым ходит конь. Еще в каждой вершине мы будем хранить состояние клетки, то есть посещена клетка или нет. Зная, начальную и конечную клетку мы можем запустить обход в ширину с двух сторон. Который будет идти поэтапно. В каждой клетке мы будем хранить ее расстояние от начальной(той из которой был запущен обход). Когда мы придем в клетку которая уже была пройдена из другого обхода(запущенного с другой стороны), кратчайшее расстояние будет найдено, и обход завершится. Чтобы найти путь можно хранить в каждой последующей точке, точку из которой пришли.

2) Идея вторая
   Сделать двумерный массив точек NxN. Точки будут хранить цвет и расстояние.

И уже на этом массиве запустить обход в ширину или в глубину. Так же чтобы найти путь можно добавить поле в точку, в котором будем указывать откуда пришли в эту точку.

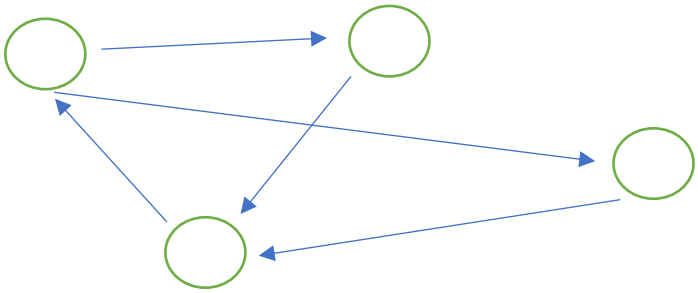
Задание №8.

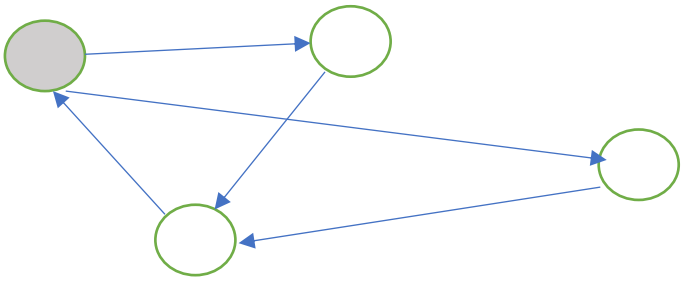
Условие:

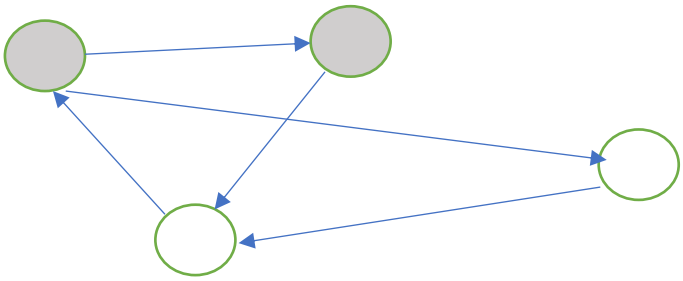
8. Препложите алгоритм, который позволил бы удалить из орграфа минимальное число ребер так, чтобы он стал ациклическим.

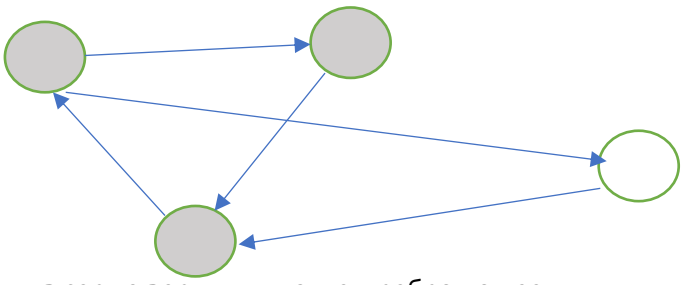
Решение:
Можно запустить обычный обход в глубину и когда будем приходить в серую вершину то будем удалять ребро до нее.

Пример:

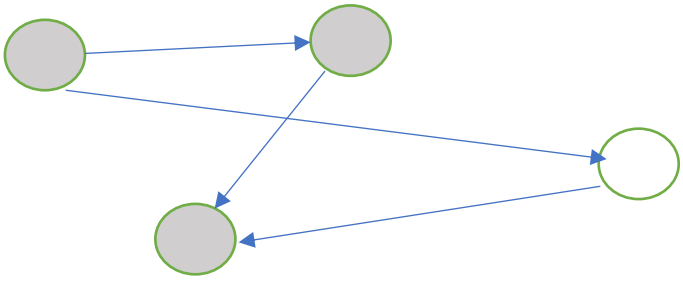


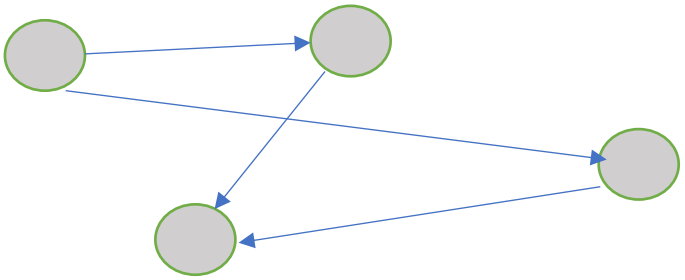


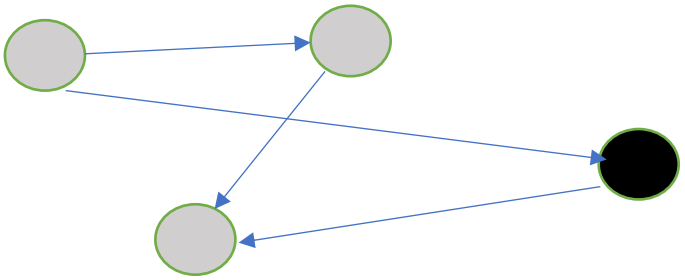


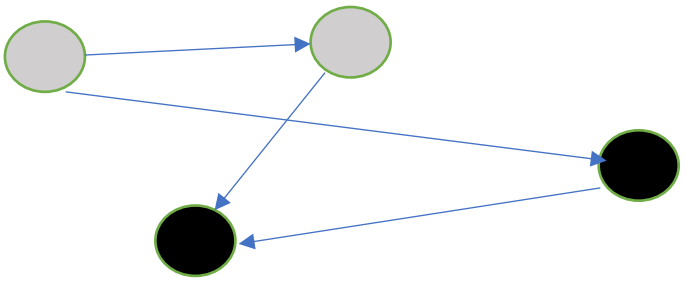


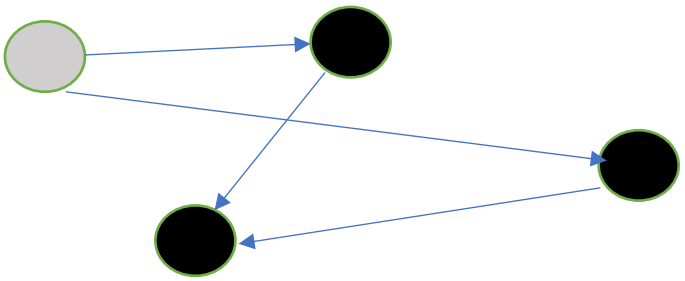
Так как есть путь в серую вершину удаляем ребро до нее.

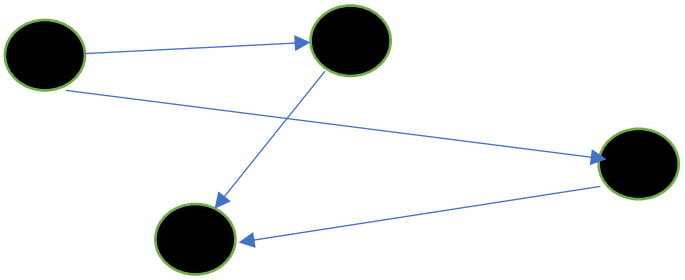








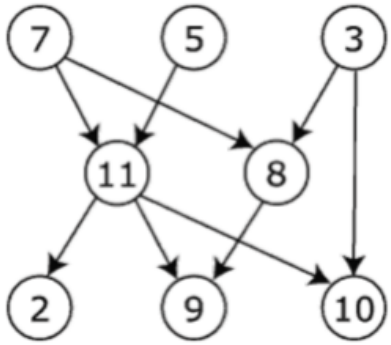




Задание №9.

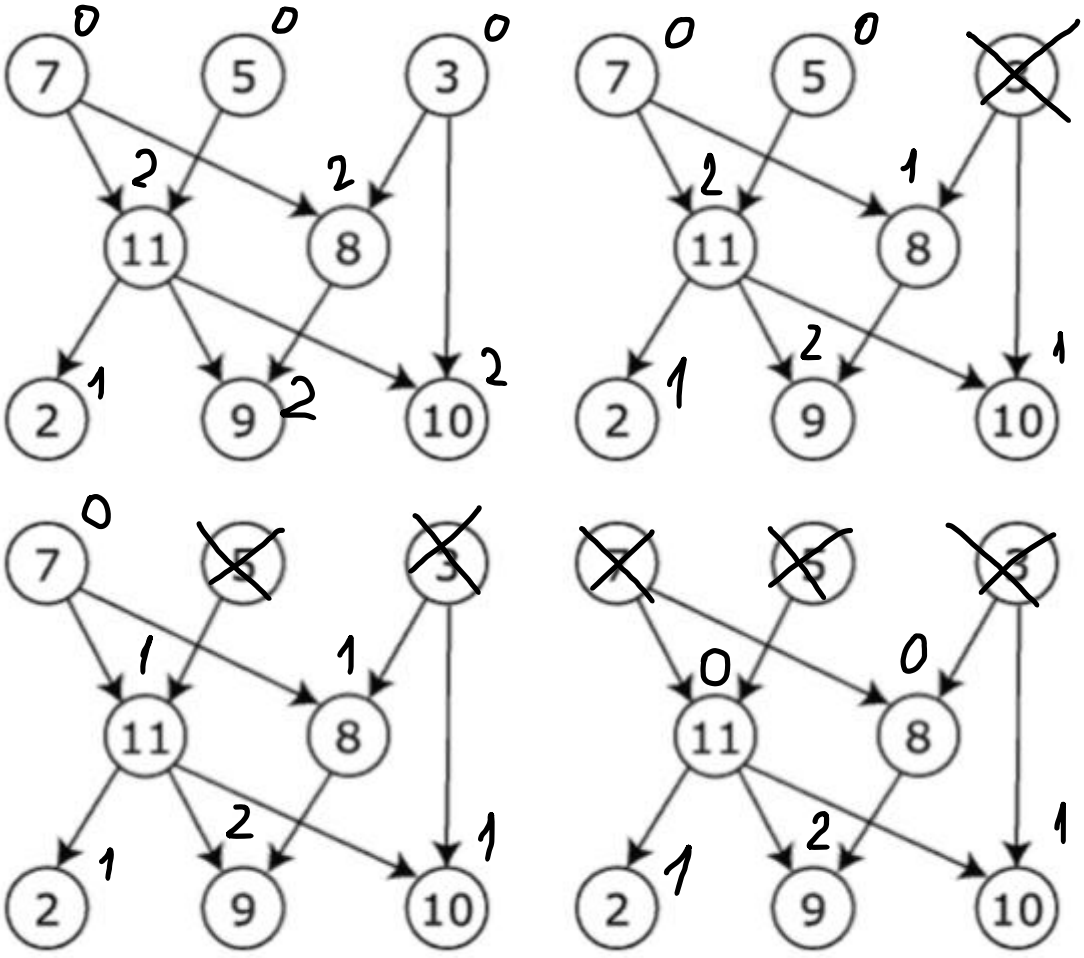
Условие:

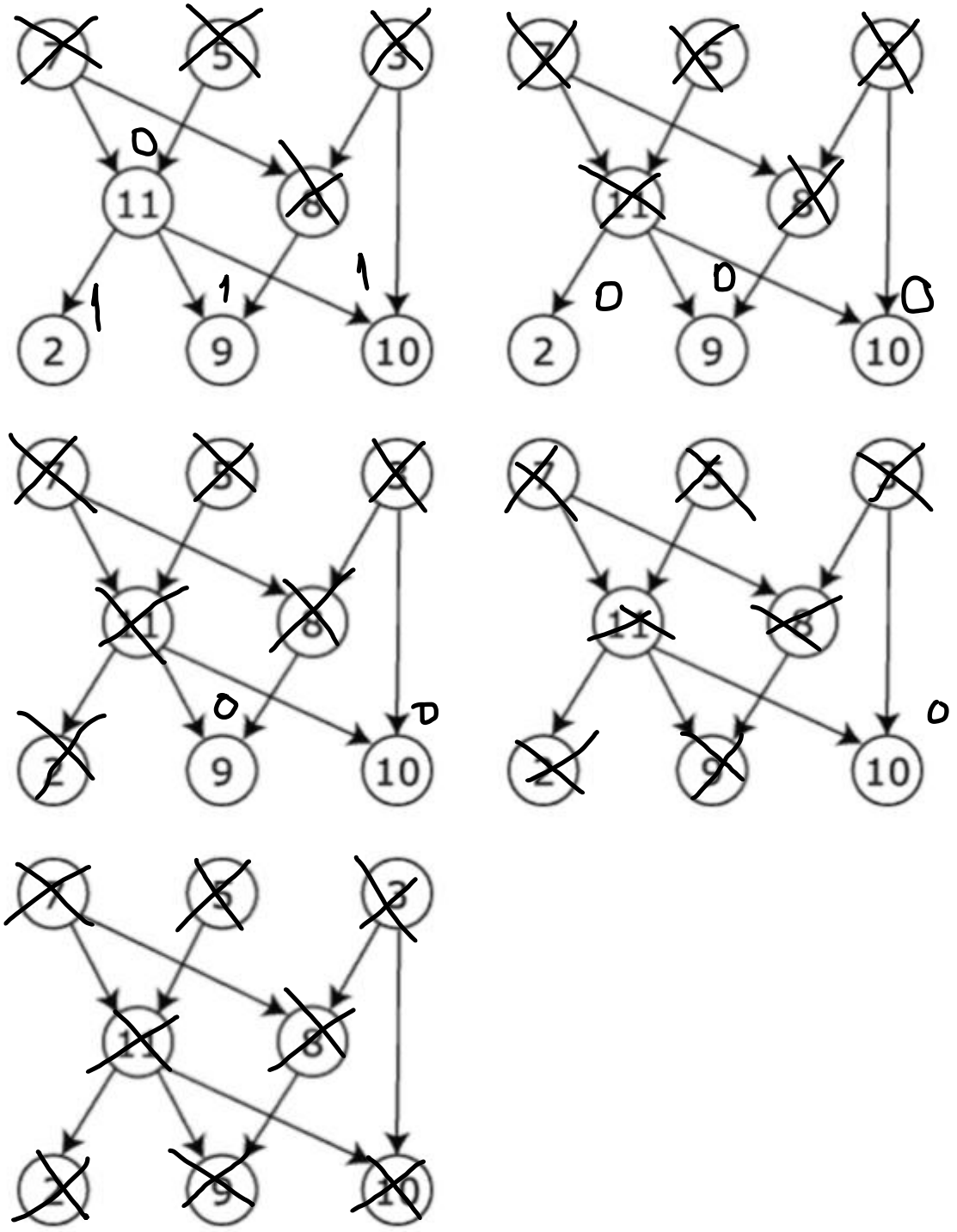
9. Продемонстрируйте работу топологической сортировки на следующем графе. Какие вершины могут быть первыми в топологической сортировке этого графа?



Решение:

Для начала надо пометить степени вершин графа. Для алгоритма топологической сортировки можно использовать только степень, отвечающую за количество ребер, ведущих в вершину.





При этом стоит отметить, что все вершины при их удалении складываются в очередь, из которой потом достаются. В этом примере последовательность будет: 3, 5, 7, 8, 11, 2, 9, 10. Первыми будут вершины, у которых степень 0.

Как работает алгоритм.

Мы берем все вершины, у которых степень ноль и складываем их в отдельную очередь. После чего достаем вершины из очереди и складываем их в конечную очередь при этом уменьшая степень у остальных вершин, до которых мы можем добраться из вершины, которую достали из очереди и если степень вершины, до которой мы можем добраться становится равной нулю то мы складываем ее в очередь. Конечная очередь здесь != очередь. Потом выводим вершины из конечной очереди.

Задание №10.

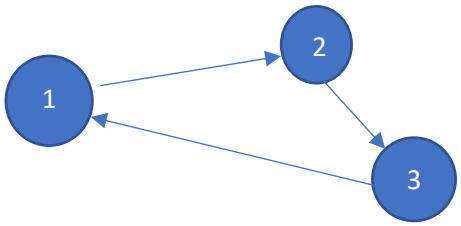
Условие:

10. Запустим алгоритм построения топологической сортировки на орграфе с циклами. Правда ли, что он минимизирует число ребер, ведущих справа налево?

Решение:

Во-первых, зависит от реализации, так как если использовать реализацию не через обход в глубину, то алгоритм может вовсе не запуститься.

Пример:



Если же использовать топологическую сортировку через обход в глубину и нам нужен порядок вершин, при котором, количество ребер ведущих справа на лево будет минимальным то это правда, так как мы пройдемся по всем белым вершинам и потом в стек будем класть уже обработанные. В результате будут развернуты все ребра, которые можно развернуть –> количество ребер, идущих справа на лево будет минимально.

Рассмотрим пример:

Допустим мы запустимся из вершины 2 и покрасим ее в серый цвет, тогда следующая вершина в которую мы пойдем будет вершина 3. Ее мы тоже покрасим в серый цвет и пойдем в вершину 1, и тоже покрасим ее в серый цвет. После чего путей в белые вершины уже не будет. Теперь останется покрасить вершину 1 в черный цвет и положить ее в стек, затем вершину 3 покрасить в черный цвет и положить ее в стек, и наконец покрасить вершину 2 и положить ее в стек. Теперь можно достать вершины из стека по порядку и вывести их. Мы получим последовательность 2 3 1. Если посмотреть, то в такой последовательности число ребер, идущих справа на лево минимально. Если бы нам дали список вершин 1 3 2 и ребер, то топологическая сортировка минимизировала количество ребер ведущих справа на лево.

Задание №11.

Условие:

11. Дан ориентированный граф. Какое минимальное число ребер нужно добавить в него, чтобы он стал сильно связным?

Решение:

Ориентированный граф называют сильно связным если для любых двух различных его вершин существует по крайней мере один путь, соединяющий эти вершины. Это значит, что любые две вершины сильно связного графа взаимно достижимы. Исходя из этого определения, можно придумать алгоритм. Для начала надо найти компоненты сильной связности графа. После чего последовательно соединить эти компоненты связности, т.е. 1 со 2, 2 с 3, 3 с 4 и т.д, если они не

соединены. когда компоненты закончатся надо из последней провести ориентированное ребро к первой. Алгоритм завершен.

Задание №12.

Условие:

12. Дан неориентированный граф. Сколько ребер нужно добавить в него, чтобы в нем не осталось мостов?

Решение:

(рассматривается относительно, того, что граф состоит из одной компоненты связности)

Найдем висячую вершину (или мост при отсутствии вершины), и соединим эту вершину с самой дальней находящийся от нее. Таким образом мы сможем избавиться от всех мостов, находящихся “внутри” графа(стоит отметить, что в начале стоит сравнить количество точек сочленения и мостов чтобы понять есть ли мосты которые не являются висячими ребрами, если таких нет то сразу начинаем соединять висячие вершины) . Если нет висячих вершин, но в графе есть ребра то надо найти две вершины которые находятся дальше всего друг от друга и соединить их ребром. Теперь подумаем об оставшихся висячих вершинах, которые тоже надо убрать. Так в графе уже есть цикл то теперь надо соединить ребрами висячие вершины последовательно. Так мы сможем создать второй путь на висячих вершинах и при этом в вершину всегда будет другой путь.

Вторая идея, это построить граф конденсации и там уже сделать циклы. Но делать это можно примерно тем же способом. То есть создать циклы.

Задание №13.

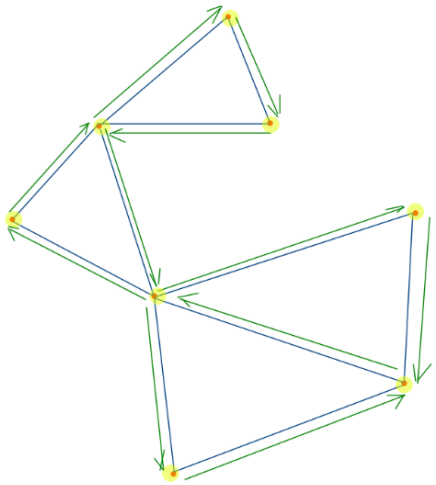
Условие:

13. Дан неориентированный связный граф. Ориентировать его ребра так, чтобы получился сильно связный граф.

Решение:

Можно запустить обход в глубину. Когда мы будем идти в белую вершину, то будем ориентировать ребро в нее, когда будет путь в серую то в нее. Еще мы будем хранить вершину, из которой пришли в эту вершину и кода будем ориентировать ребра в серые вершины, в вершину из которой пришли не будем ориентировать ребро, но также стоит помнить и про висячие ребра и если ребро является висячим, то ребро будет ориентированно в предыдущую вершину.

Пример:



Задание №14.

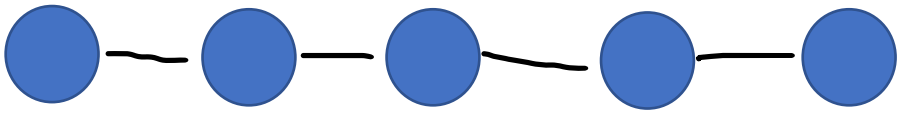
Условие:

14. Какое максимальное число точек сочленения/мостов может быть в графе с n вершинами?

Решение:

Максимальное точек сочленения n – 2,

Максимальное число мостов n – 1

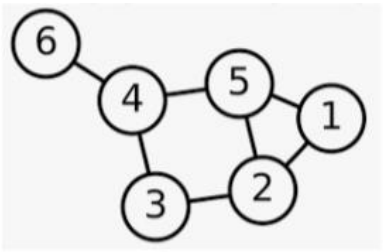


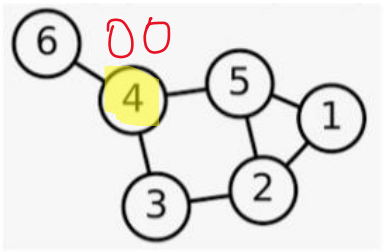
Задание № 15.

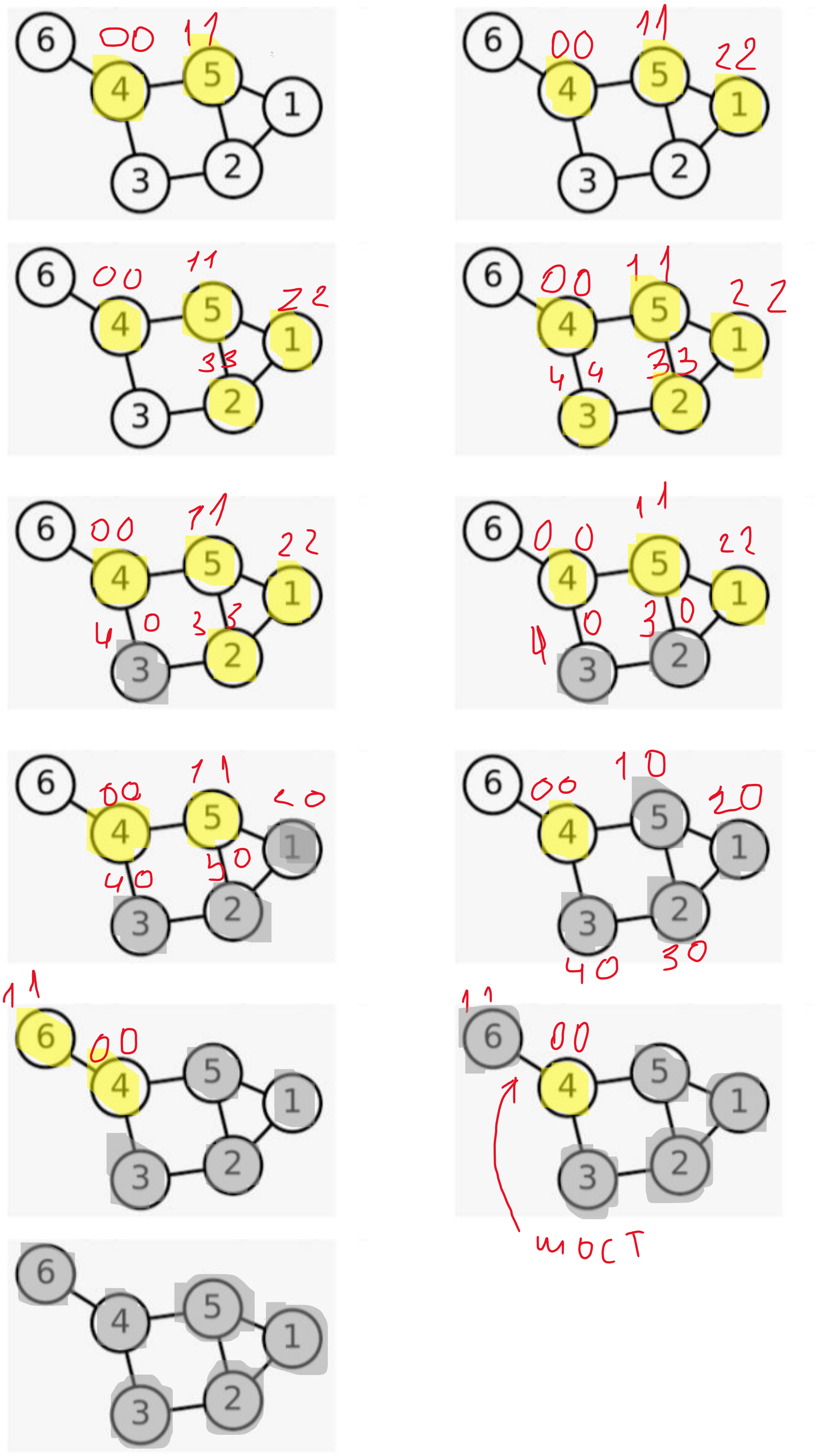
Условие:

15. Продемонстрируйте работу алгоритма поиска мостов на примере из задачи 2.

Решение:





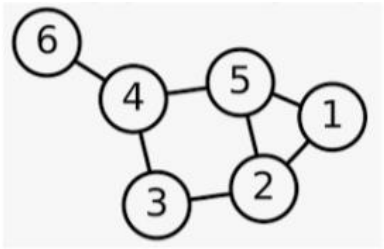

мост

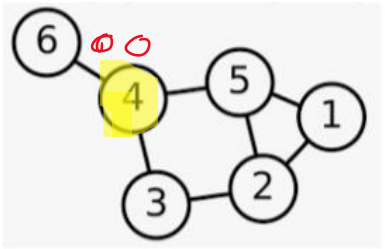
Задание №16.

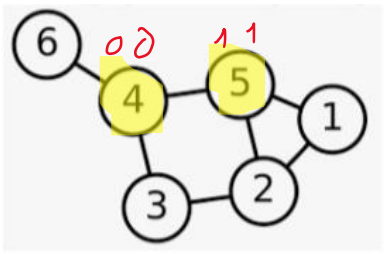
Условие:

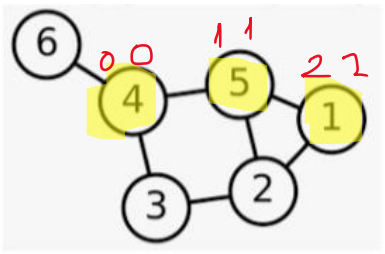
16. Продемонстрируйте работу алгоритма поиска точек сочленения на примере из задачи 2.

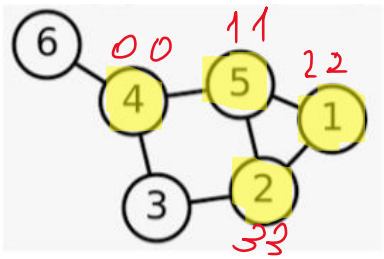
Решение:

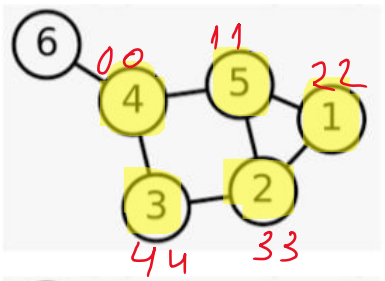


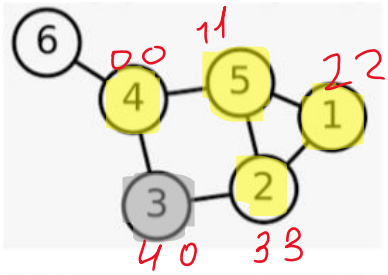


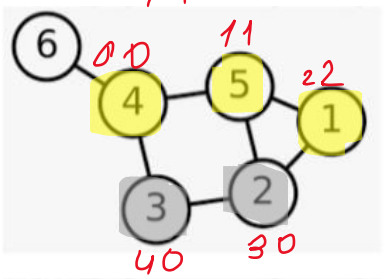


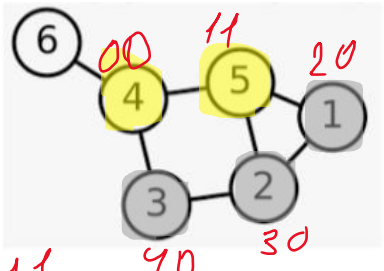


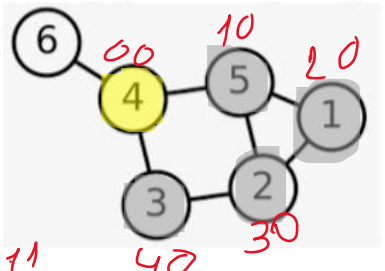


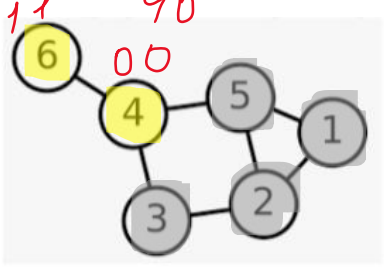


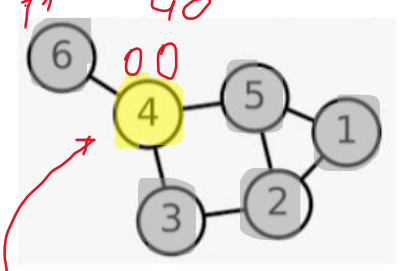




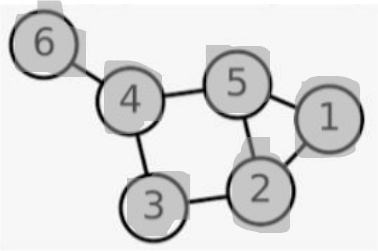








точки сочленения



4 является точкой сочленения, но в данном случае про нее нельзя сразу сказать, что это точка сочленения так как в представлении этого графа в виде дерева обхода в глубину эта вершина является корнем, и чтобы определить является ли она точкой сочленения она должна иметь в дереве более одного потомка.