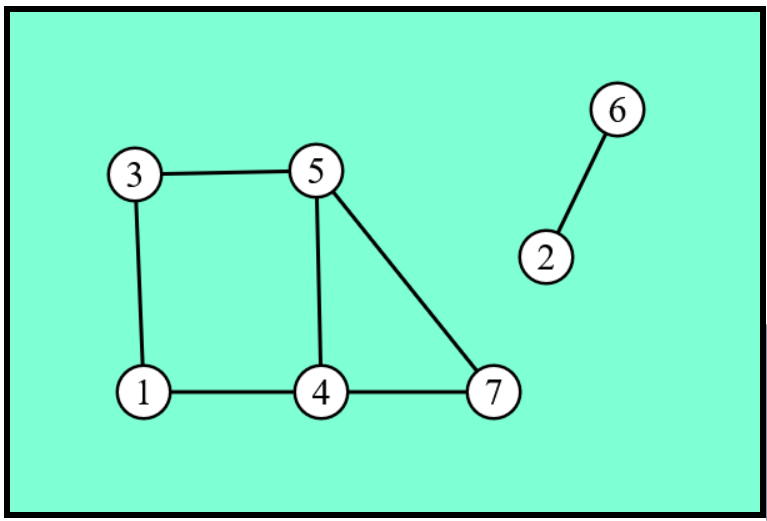
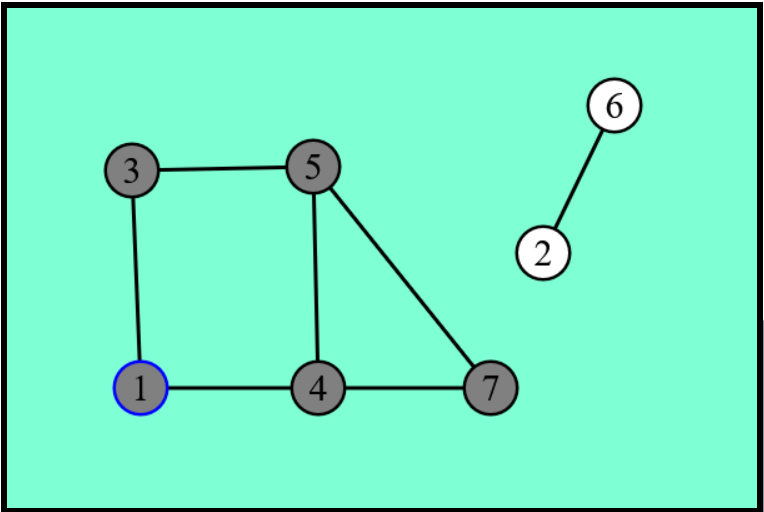
**Обход графа в глубину**

Граф – это структура, состоящая из множества вершин и множества рёбер, соединяющих пары вершин.



**Рассмотрим следующую задачу**: найти в графе путь между двумя заданными вершинами. Если есть несколько путей, можно найти любой. Например, если на рисунке выше нужно найти путь между вершинами 1 и 7, то в качестве ответа подойдёт, например, путь 1-3-5-7 или 1-4-7.

**Идея решения:** запустить из стартовой вершины алгоритм, который будет проходить по всем возможным ребрам, отмечая вершины, до которых он смог добраться из стартовой.



Один из таких алгоритмов обхода – обход в глубину (depth-first search, DFS). Шаги алгоритма:

1. Алгоритм запускается из стартовой вершины, сразу отмечает её как посещённую.

2. Перебираются все рёбра, по которым можно сейчас пройти.

3. Если в этом переборе нашлось ребро в ещё не посещённую вершину, то нужно рекурсивно запустить обход из неё, и только потом продолжить перебор.

Например, в графе из рисунка выше для стартовой вершины 1 алгоритм будет работать следующим образом:

1. Запускаем алгоритм из вершины 1, отмечаем её как посещённую.

2. Перебираем рёбра из вершины 1. Ребро 1-3 ведёт в непосещённую вершину 3, рекурсивно запускаем алгоритм из неё.

3. Отмечаем вершину 3 как посещённую, перебираем рёбра из неё. Ребро 3-1 ведёт в уже посещённую вершину, пропускаем. Ребро 3-5 ведёт в новую вершину, запускаем обход из вершины 5.

4. Отмечаем вершину 5 как посещённую. Перебираем рёбра из вершины 5. Ребро 5-3 ведёт в уже посещённую вершину, пропускаем. Ребро 5-4 ведёт в новую вершину, запускаемся из вершины 4.

5. Отмечаем вершину 4 как посещённую. Перебираем рёбра из вершины 4. Ребро 4-1 ведёт в посещённую вершину, пропускаем. Также пропускаем ребро 4-5. Ребро 4-7 ведёт в новую вершину, запускаем из неё обход.

6. Отмечаем вершину 7 как посещённую. Перебираем рёбра из вершины 7. Рёбра 7-4 и 7-5 пропускаем, потому что вершины 4 и 5 уже посещены. Больше рёбер из вершины 7 нет, завершаем вызов алгоритма из вершины 7.

7. Завершился обход из вершины 7, вызванный проходом по ребру 4-7 на шаге 5. Теперь нужно продолжить перебор рёбер из вершины 4. Но из неё больше нет рёбер (они все уже были рассмотрены на шаге 5). Завершаем вызов алгоритма из вершины 4.

8. Завершился обход из вершины 4, вызванный проходом по ребру 5-4 на шаге 4. Продолжаем шаг 4, т.е. перебор рёбер из вершины 5. Ребро 5-7 ведёт в уже посещённую вершину, пропускаем. Больше рёбер нет, завершаем обход.

9. Завершился обход из вершины 5, вызванный проходом по ребру 3-5 на шаге 3. Продолжаем шаг 3, т.е. перебор рёбер из вершины 3. Но больше из неё рёбер нет – завершаем обход.

10. Завершился обход из вершины 3, вызванный проходом по ребру 1-3 на шаге 2. Продолжаем перебор рёбер из вершины 1. Ребро 1-4 ведёт в посещённую вершину, пропускаем. Больше рёбер нет – завершаем обход из вершины 1.

Таким образом, алгоритм смог найти путь до вершин 1, 3, 4, 5, 7.

**Время работы алгоритма**

Так как алгоритм перебирает все вершины не более чем по 1 разу, а ребра не более чем по 2 раза, то можно оценить время работы алгоритма как O(n + m), где n – число вершин графа, а m – число ребер.

**Как вывести сам путь:**

Будем для каждой вершины графа хранить значение предка – вершины, из которой алгоритм пришёл в данную. В примере выше алгоритм прошел по пути 1-3-5-4-7, поэтому предком вершины 3 будет 1, предком вершины 5 будет 3 и т.д.

Чтобы восстановить сам путь в конечную вершину, нужно из неё пройти в обратную сторону по предкам, пока не встретится стартовая вершина.

**Другие задачи на обход в глубину:**

1. Поиск компонент связности.

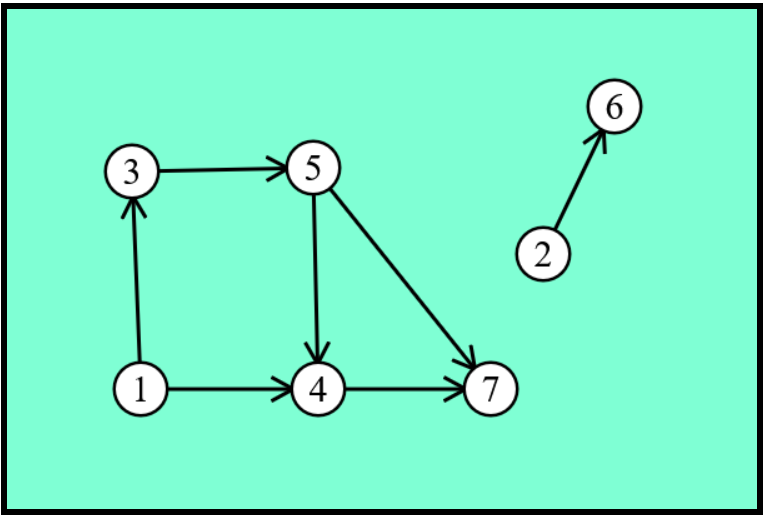
Компонента связности (в неориентированном графе, т.е. рёбра двусторонние) – это множество вершин, достижимых друг из друга, в которое нельзя добавить ещё одну вершину с сохранением этой достижимости. Например, на рисунках выше граф имеет 2 компоненты связности: {1, 3, 4, 5, 7} и {2, 6}.

Чтобы выделить в графе компоненты связности, нужно перебрать все вершины графа, и если вершина ещё не посещена, то запустить из неё обход. Каждый такой обход посетит одну компоненту связности.

2. Топологическая сортировка.

Дан ориентированный граф (т.е. по рёбрам можно двигаться только в одном направлении). В графе нет циклов. Нужно найти такой порядок вершин, что все рёбра ведут от более ранних вершин в этом порядке к более поздним (т.е. слева направо).

Пример: на рисунке ниже подойдёт такой порядок: 1 3 5 4 7 2 6. Но этот ответ не единственный.



**Решение:**

Для удобства будем искать обратный порядок – порядок, в котором все рёбра ведут справа налево. Тогда, чтобы добавить в такой порядок какую-то вершину (например, 1), нужно сначала добавить все вершины, в которые из неё есть рёбра (3 и 4). Этого можно добиться, запустив обход в глубину и записывая вершину в топологическую сортировку при завершении обхода из неё. Это работает, потому что обход завершается, когда не осталось рёбер из текущей вершины.

Так как обход в глубину из одной вершины может посетить не все вершины, а в ответ нужно добавить все, то нужно, как и при поиске компонент связности, перебирать вершину и запускать обход, если она ещё не посещена.

Алгоритм не работает, если в графе есть циклы, потому что цикл невозможно упорядочить так, чтобы все рёбра шли слева направо.

Чтобы алгоритм работал, важно, чтобы обход из текущей вершины завершался после завершения обхода из всех вершин, в которые можно попасть по ребру из текущей. Это верно для всех случаев, когда в графе нет циклов.