Лекция 5.
Графы.

План лекции

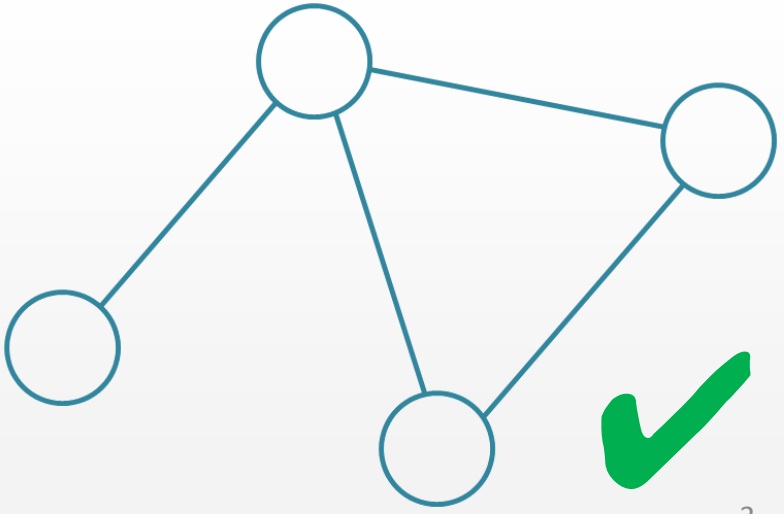
- Определение графа.
- Определения теории графов.
- Проблема семи мостов Кёнигсберга.
- Хранение графа списком рёбер.
- Хранение графа матрицей смежности.
- Хранение графа списком смежности.
- Обход графа в глубину.
- Обход графа в ширину.
- Проверка графа на связность.

Что такое граф?

Граф в математике – это некое множество, элементы которого могут иметь между собой связи.

Строгое определение:

Граф G<V,E> - это некоторое множество вершин V, из которого некоторые пары, возможно, соединены рёбрами из множества E.

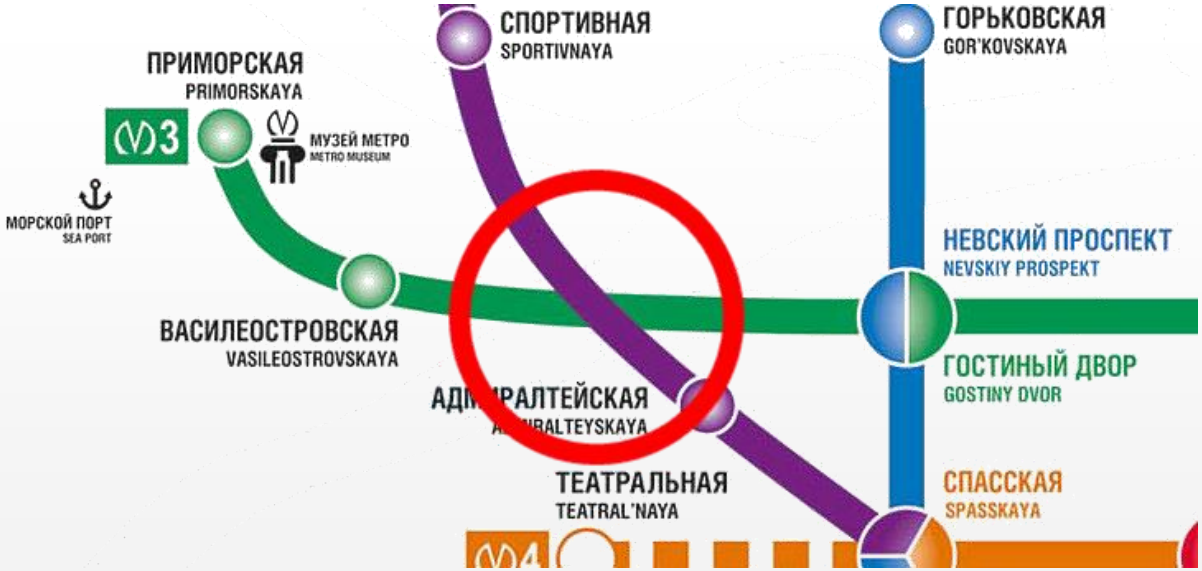


Что такое граф?

Очень хороший пример графа – схема метро, где станции – это вершины графа, а перегоны между станциями – это рёбра.

Что такое граф?

Важно заметить, что изображение графов не всегда понятны и очевидны. Например, такие пересечения рёбер не считаются за вершину.



Определения теории графов

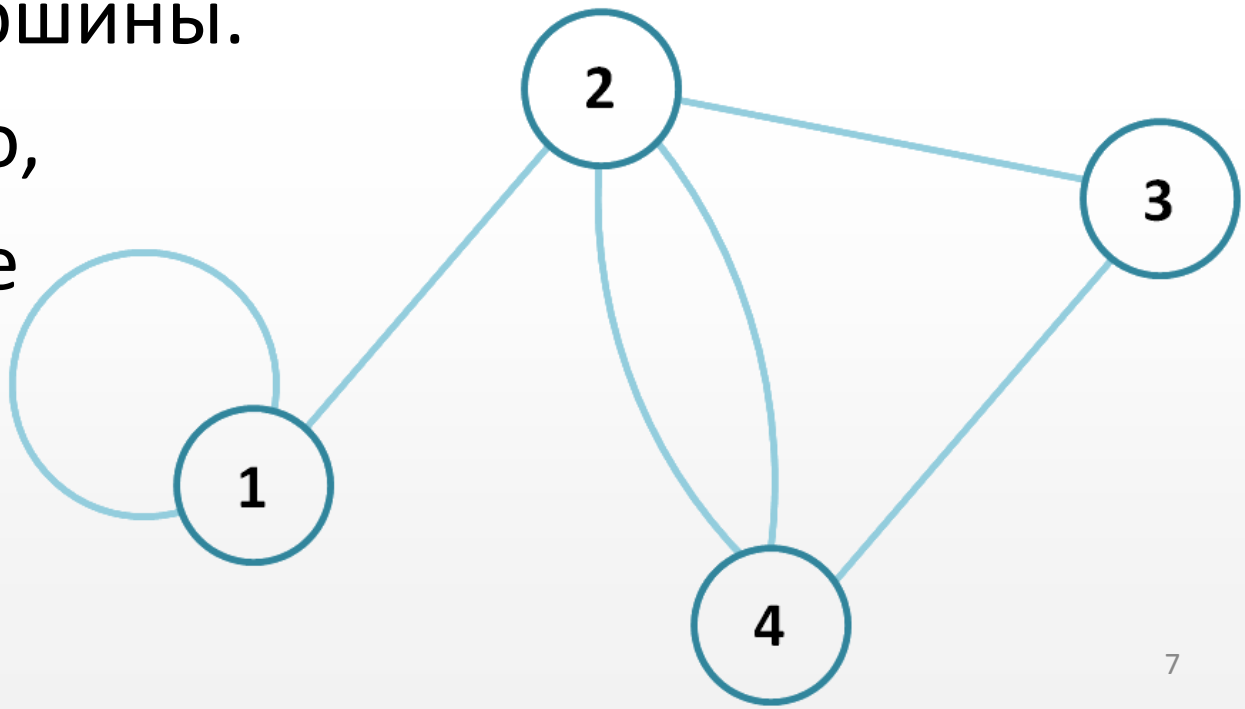
Теория графов – раздел дискретной математики, изучающий свойства графов.

Определения теории графов

Смежные рёбра – рёбра, имеющие общую концевую вершину.

Кратные рёбра – рёбра, имеющие общие концевые вершины.

Петля – ребро, соединяющее вершину с самой собой.



Определения теории графов

Степень вершины – количество рёбер, которые инцидентны ей (причём петля считается дважды).

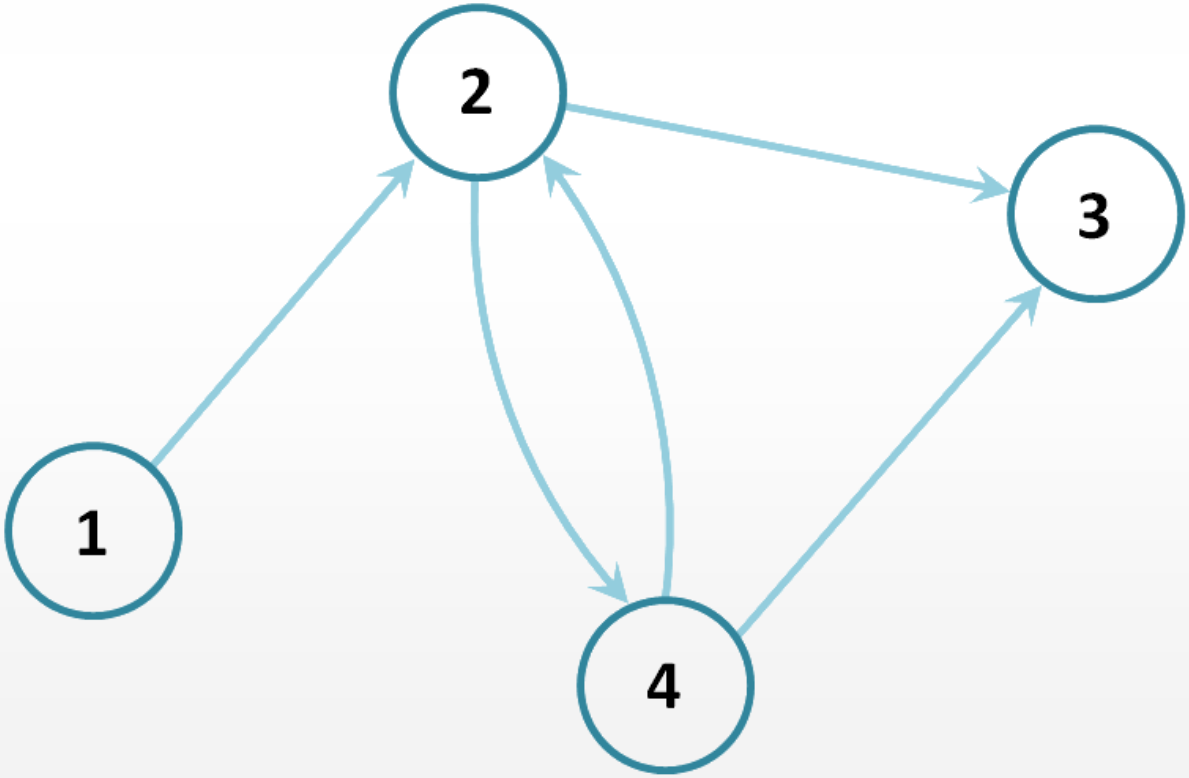
Изолированная вершина – вершина с нулевой степенью (т.е. она не связана ни с чем рёбрами).

Лист – вершина со степенью 1 (т.е. она связана с только одной другой вершиной).

Определения теории графов

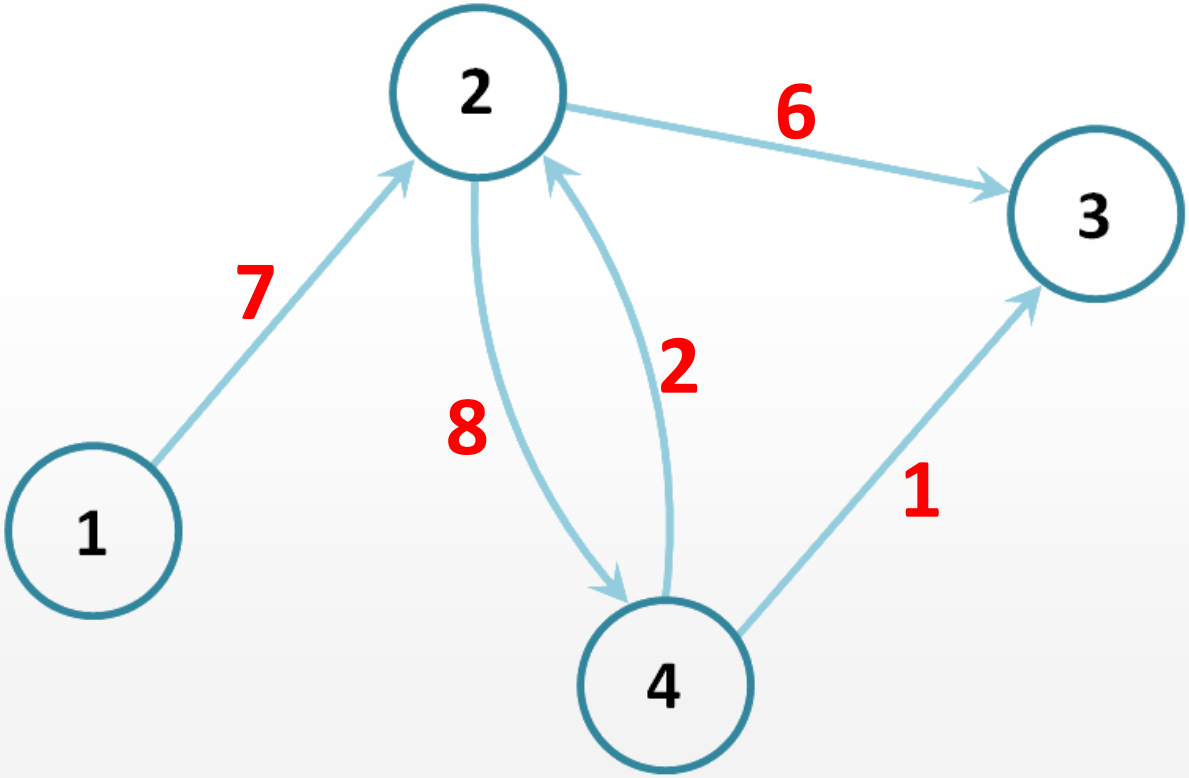
Определения теории графов

Ориентированный граф – это граф с ориентированными рёбрами (дугами).



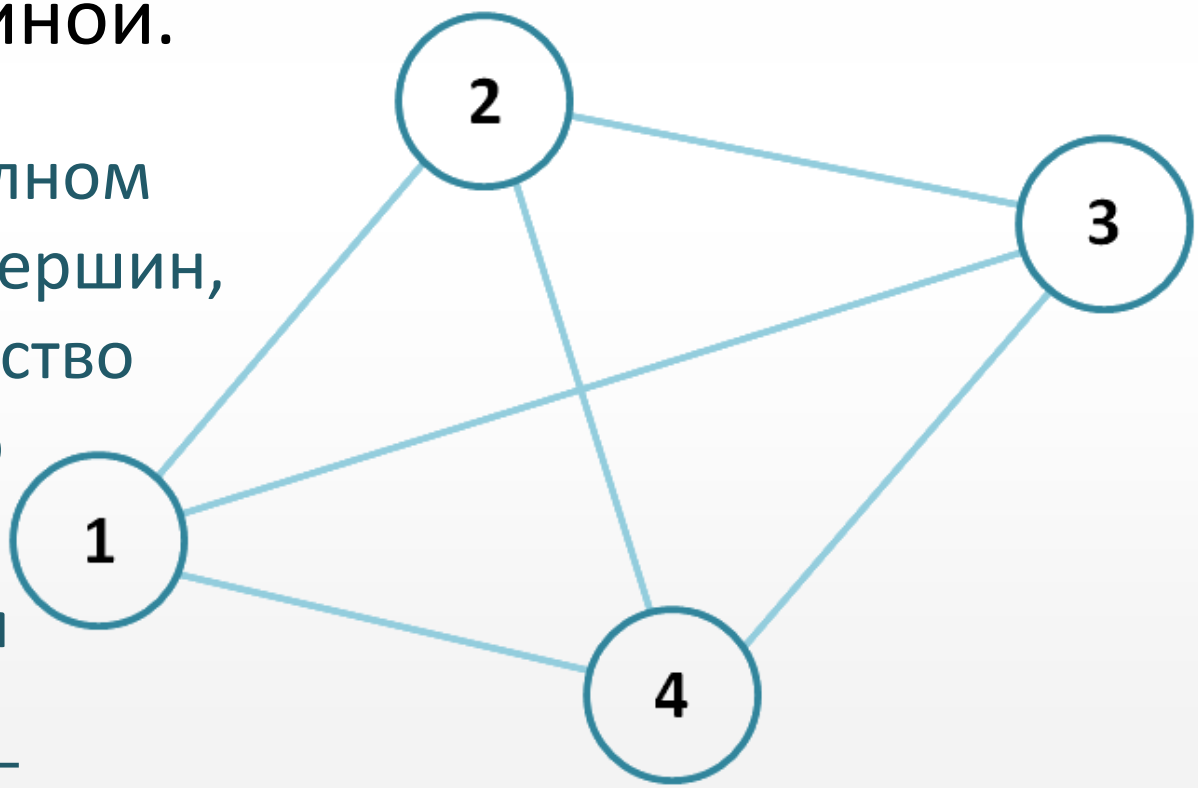
Определения теории графов

Взвешенный граф – это граф, рёбра которого имеют условный вес (длину, стоимость).



Определения теории графов

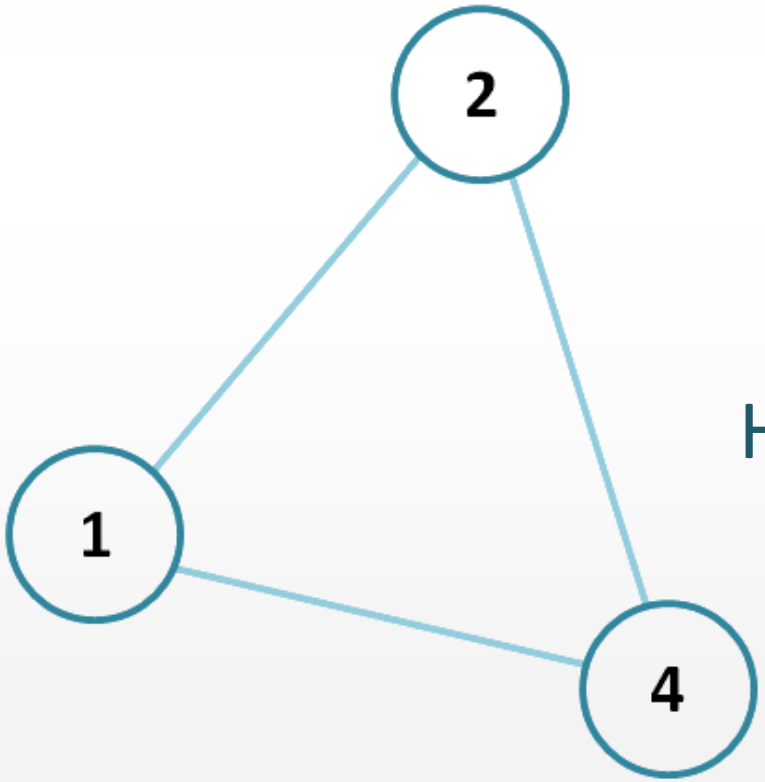
Полный граф – это граф, каждая вершина которого связана ребром с каждой другой вершиной.



Если в полном графе N вершин, то количество его рёбер будет равняться $\frac{N(N-1)}{2}$

Определения теории графов

Связный граф – это граф, в котором каждая пара вершин связана рёбрами, напрямую или через другие вершины.



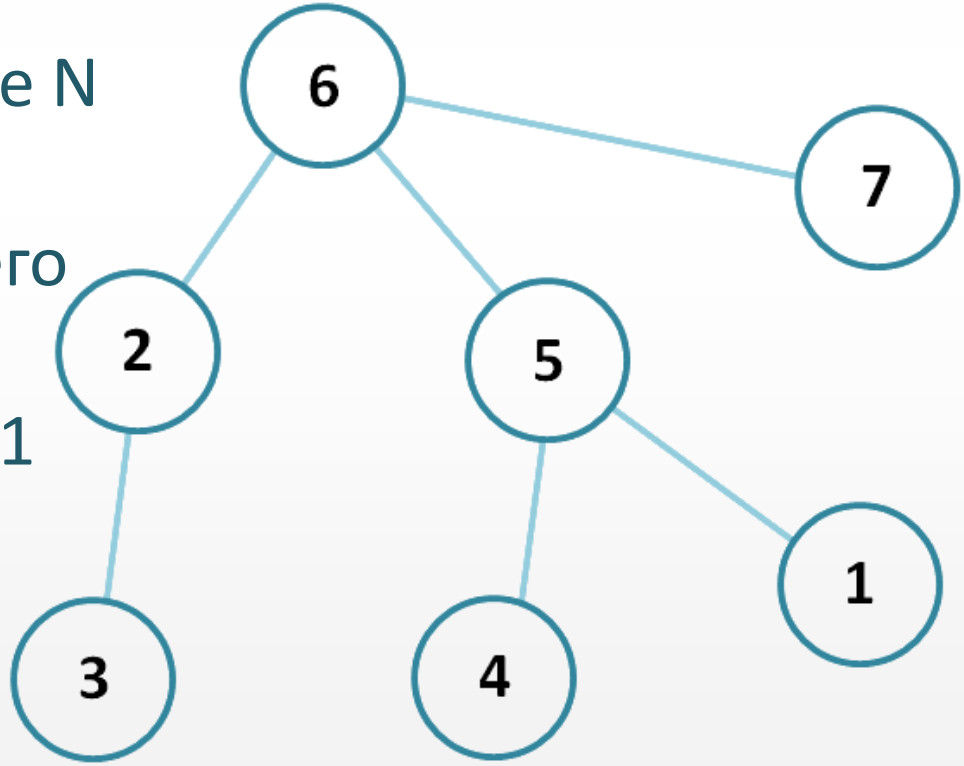
Несвязный граф

Определения теории графов

Цикл – это замкнутый путь в графе.

Дерево – это связный граф без циклов.

Если в дереве N вершин, то количество его рёбер будет равняться N-1



Проблема семи мостов Кёнигсберга

Можно ли обойти все мосты, не проходя по каждому не более одного раза?

Проблема семи мостов Кёнигсберга

Ответ на эту задачу нашёл немецкий и российский математик Леонард Эйлер в 1736 году.

- Не может существовать граф, который имел бы нечётное число нечётных вершин.
- Если все вершины графа чётные, то можно, не отрывая карандаша от бумаги, начертить граф.
- Если ровно две вершины графа нечётные, то можно, не отрывая карандаша от бумаги, начертить граф, при этом можно начинать с любой из нечётных вершин и завершить его в другой нечетной вершине.
- Граф с более чем двумя нечётными вершинами невозможно начертить одним росчерком.

Проблема семи мостов Кёнигсберга

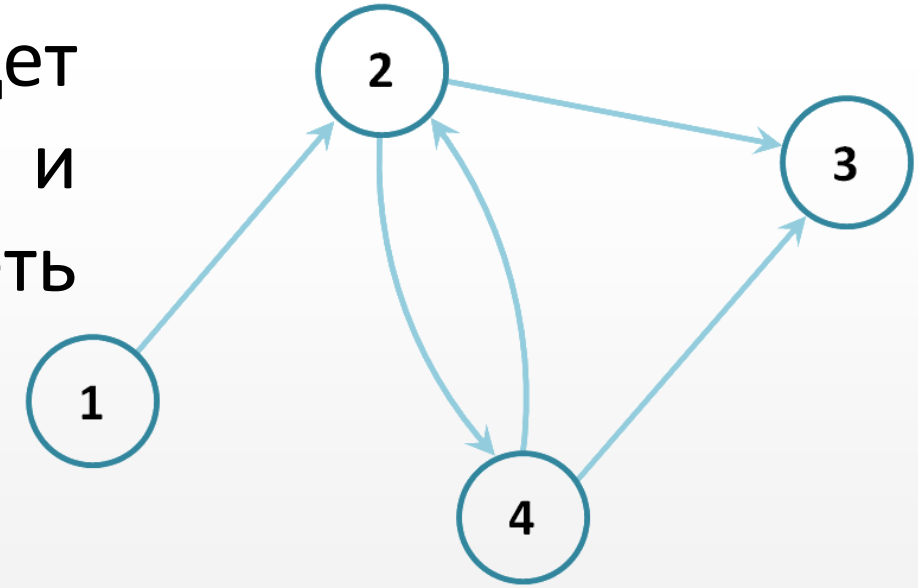
Т.е. ответ -

НЕЛЬЗЯ

Хранение графа

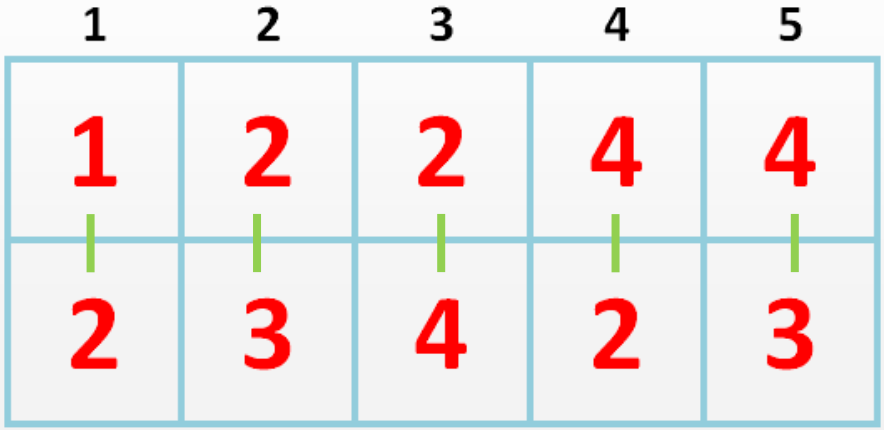
Напоминание: граф – это множество вершин и множество рёбер, связывающее эти вершины.

Пусть наш граф будет ориентированным и невзвешенным, и иметь N вершин и M рёбер:



Список рёбер

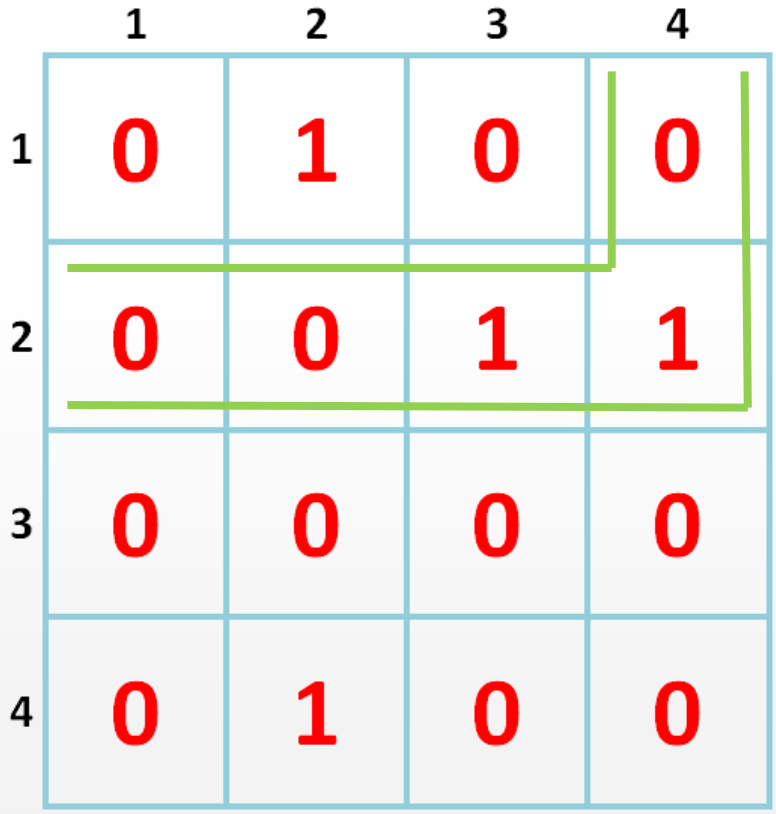
Это массив пар (или пара массивов) размерности M. Каждый элемент списка – это пара чисел, первое из которых обозначает начальную вершину ребра, а второе – конечную. Требует памяти для $2M \leq N(N-1)$ элементов.



Матрица смежности

Это квадратная матрица размерности NxN. На пересечении строки i и столбца j стоит 1, если в графе есть дуга из вершины i в вершину j, иначе – 0.

Требует памяти N^2.



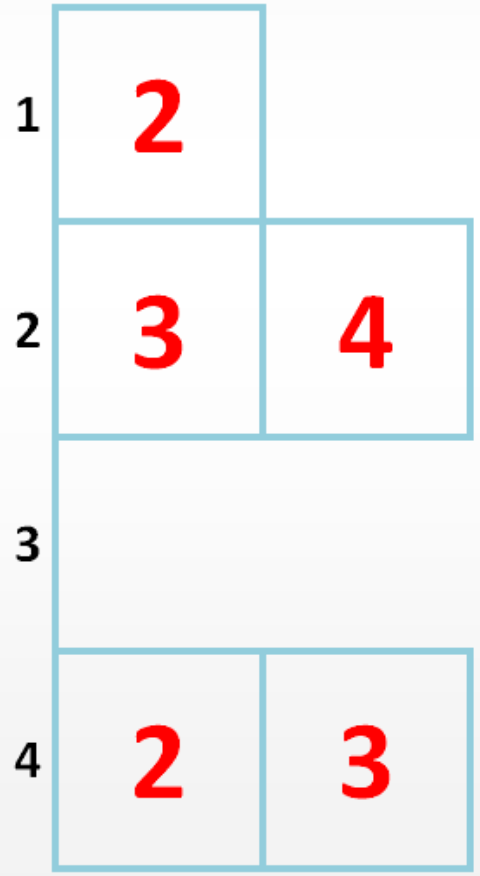
«Существует дуга из вершины 2 в вершину 4»

Список смежности

Этот способ смешивает матрицу смежности и список рёбер.

По сути это – массив векторов, в i-м векторе хранятся все вершины, в которые исходят дуги из вершины i.

Вся конструкция хранит всего M элементов (по одному на каждое ребро).



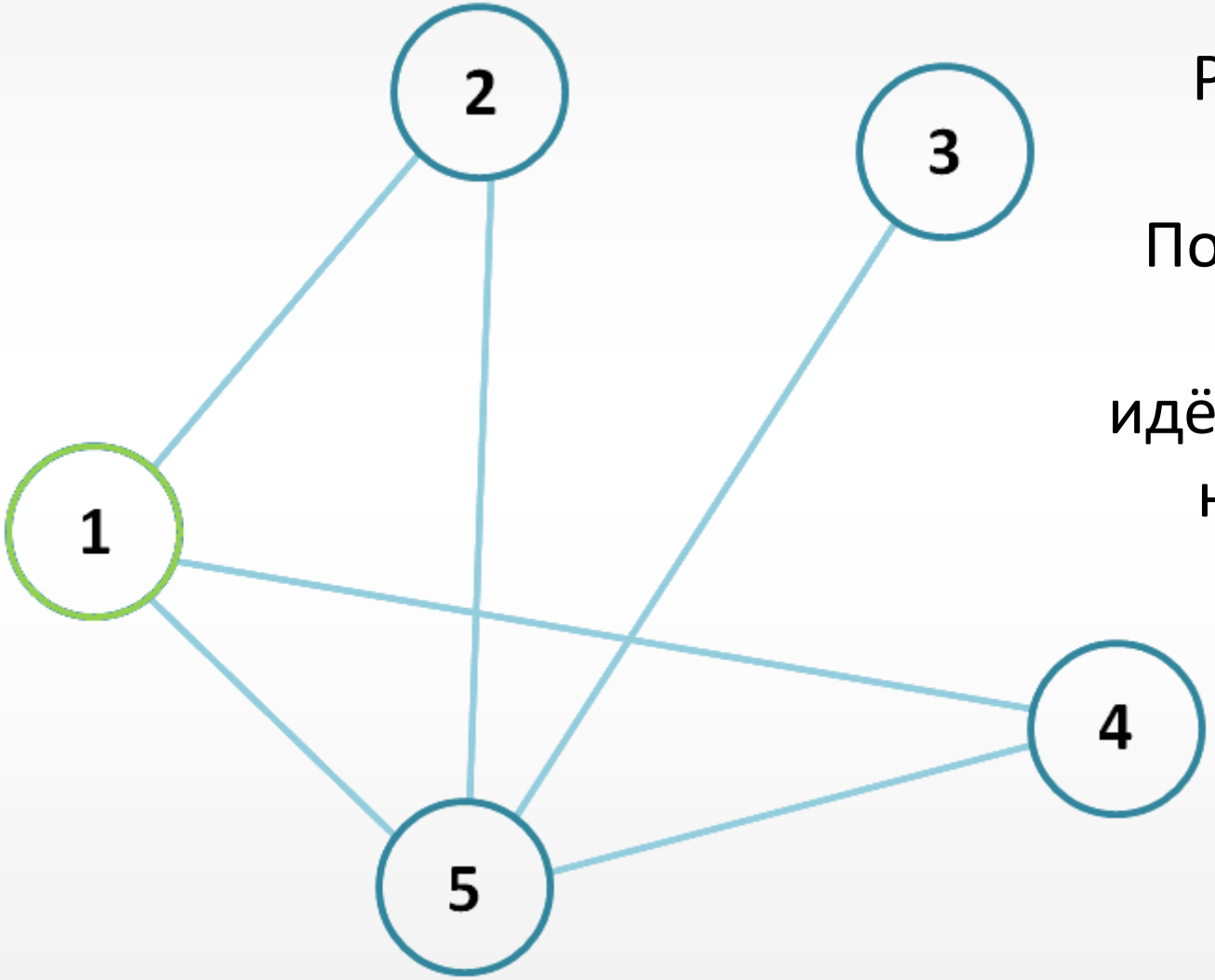
Обход в глубину (DFS)

Алгоритм обхода графа, который стремится уйти как можно глубже в граф.

DFS = **D**epth-**F**irst **S**earch.

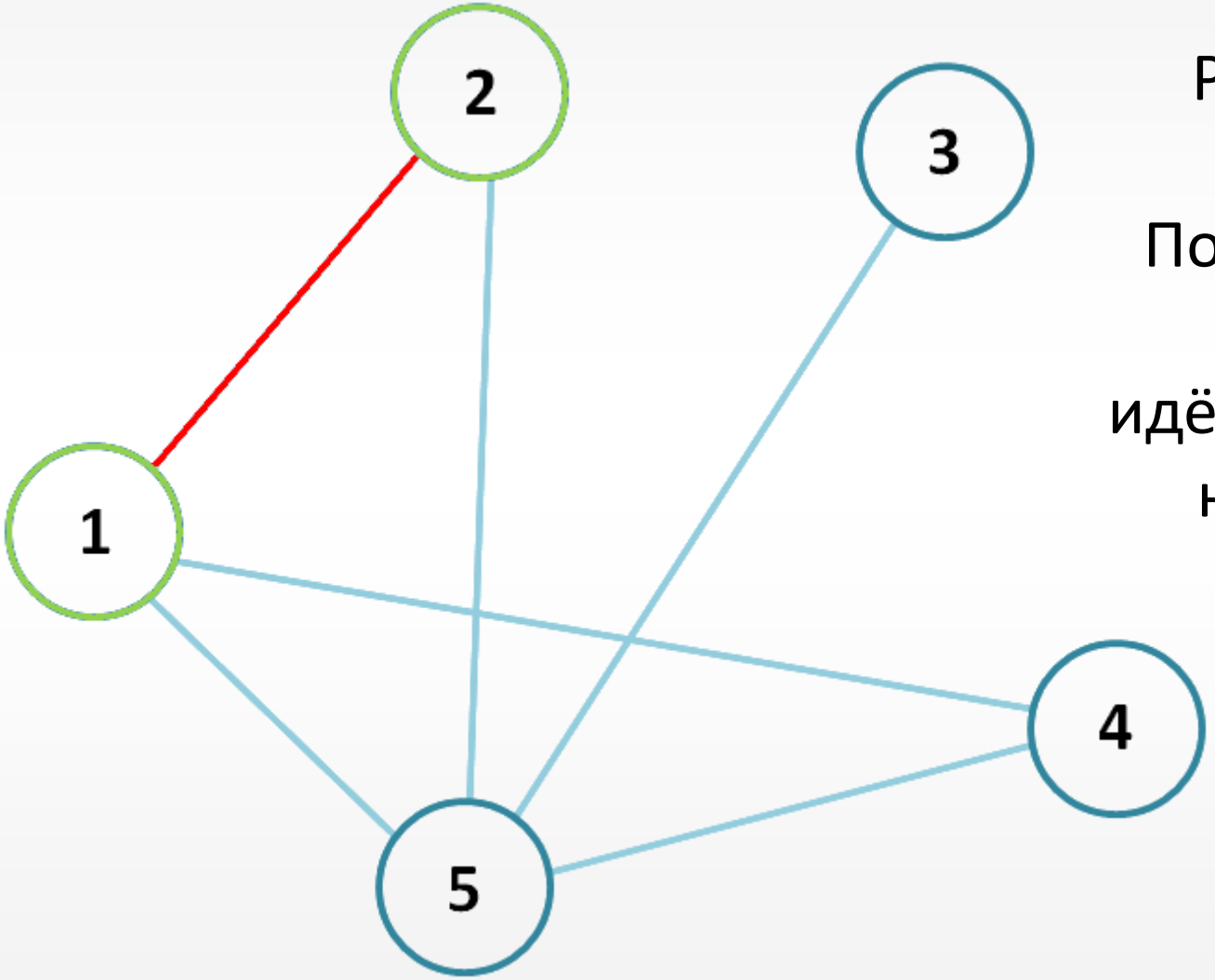
Рассматривая вершину, алгоритм ищет тех её соседей, которых он ещё не посещал, и от каждой рекурсивно запускает обход в глубину.

Обход в глубину (DFS)



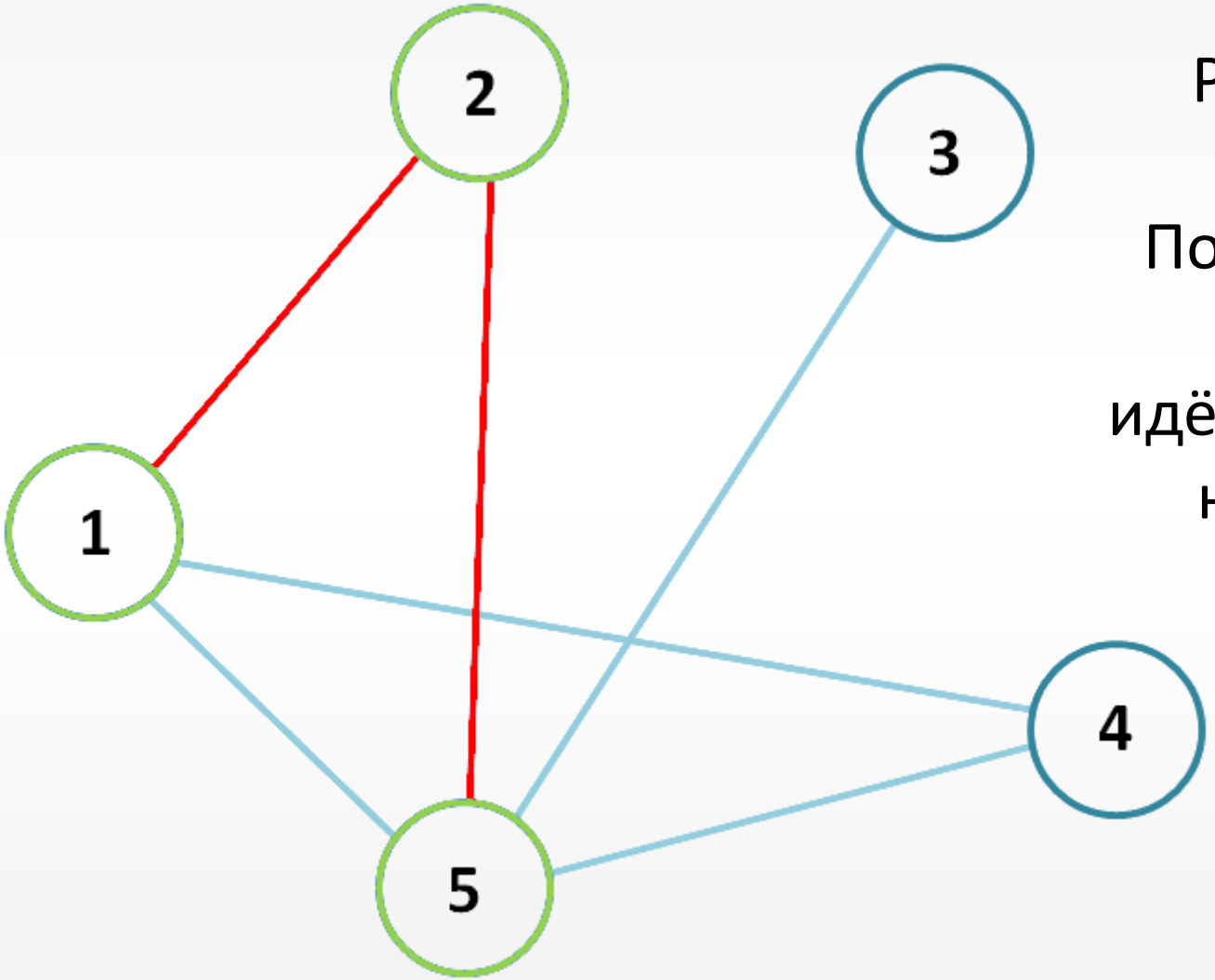
Рассматриваем вершину **1**. Помечаем её как посещённую и идём в первую же непосещённую вершину – **2**.

Обход в глубину (DFS)



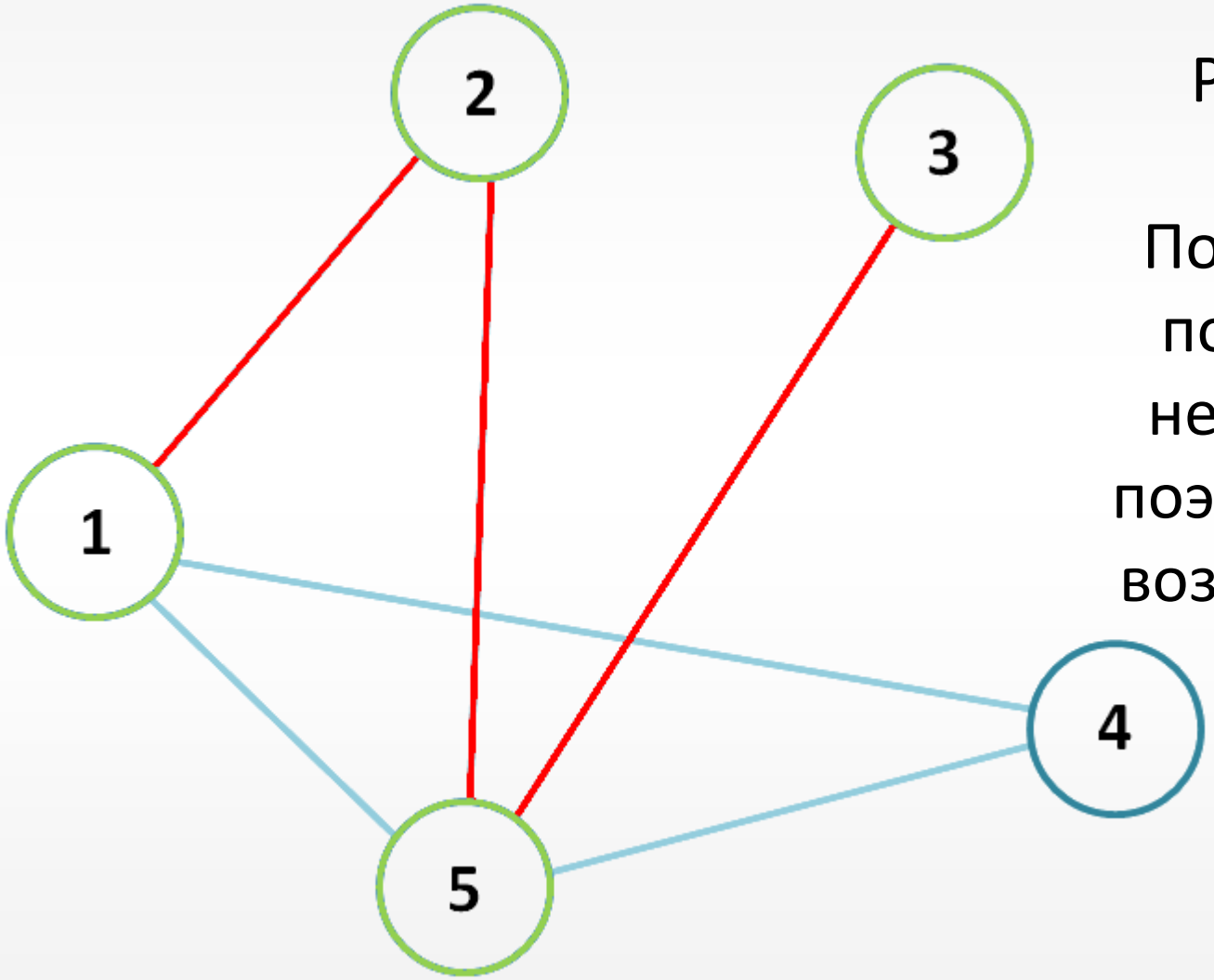
Рассматриваем вершину **2**. Помечаем её как посещённую и идём в первую же непосещённую вершину – **5**.

Обход в глубину (DFS)



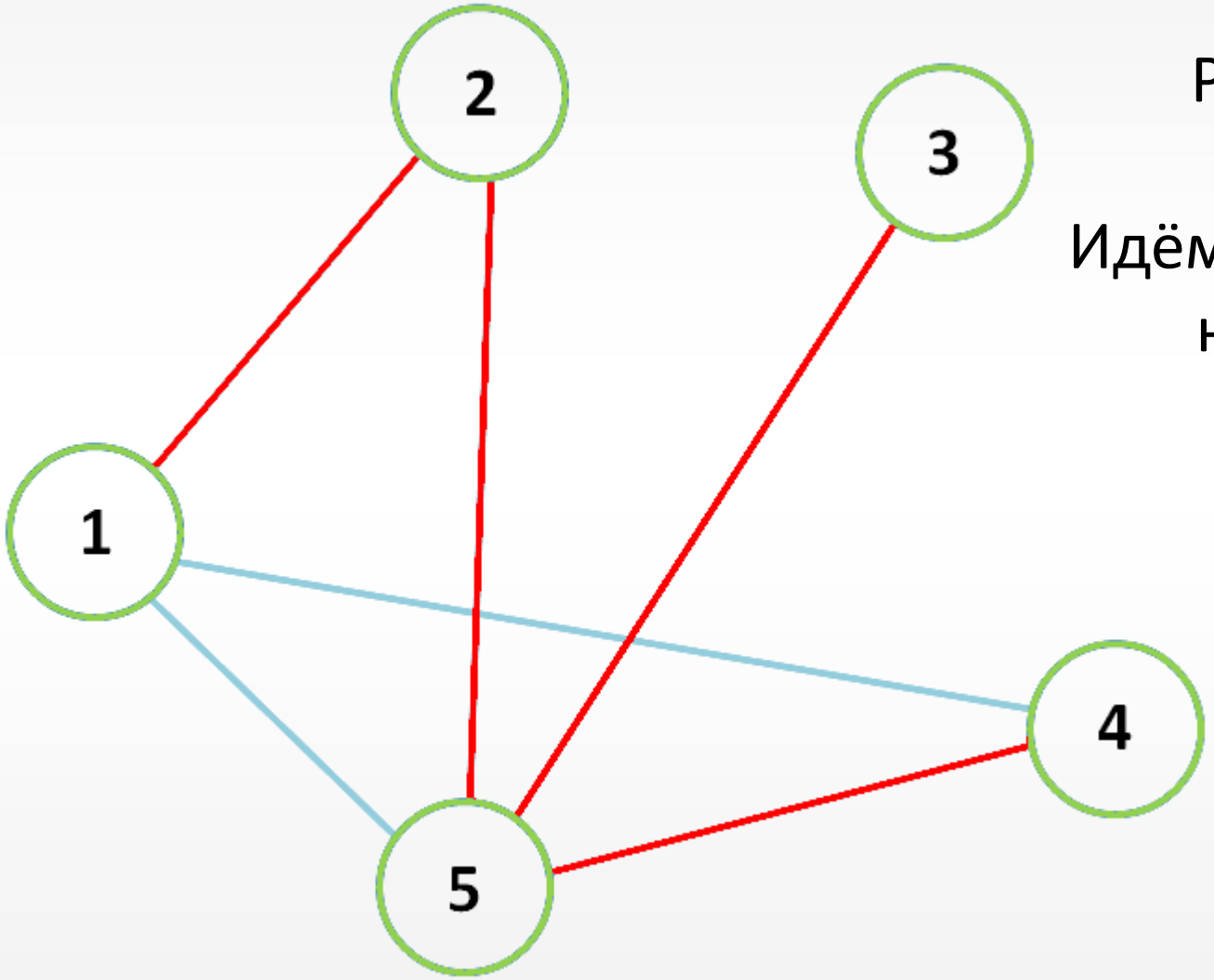
Рассматриваем вершину **5**. Помечаем её как посещённую и идём в первую же непосещённую вершину – **3**.

Обход в глубину (DFS)



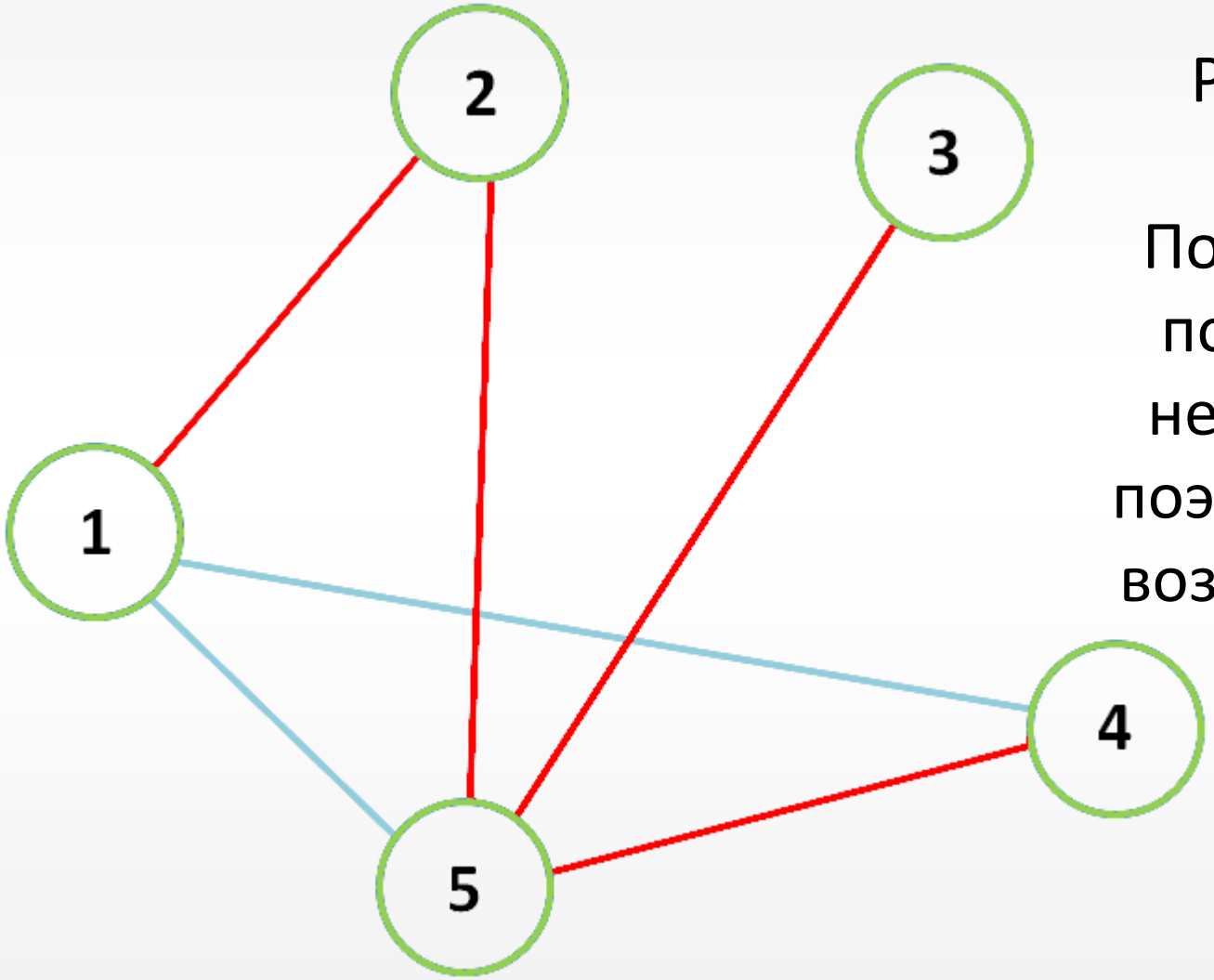
Рассматриваем вершину **3**. Помечаем её как посещённую. Из неё некуда идти, поэтому алгоритм возвращается в **5**.

Обход в глубину (DFS)



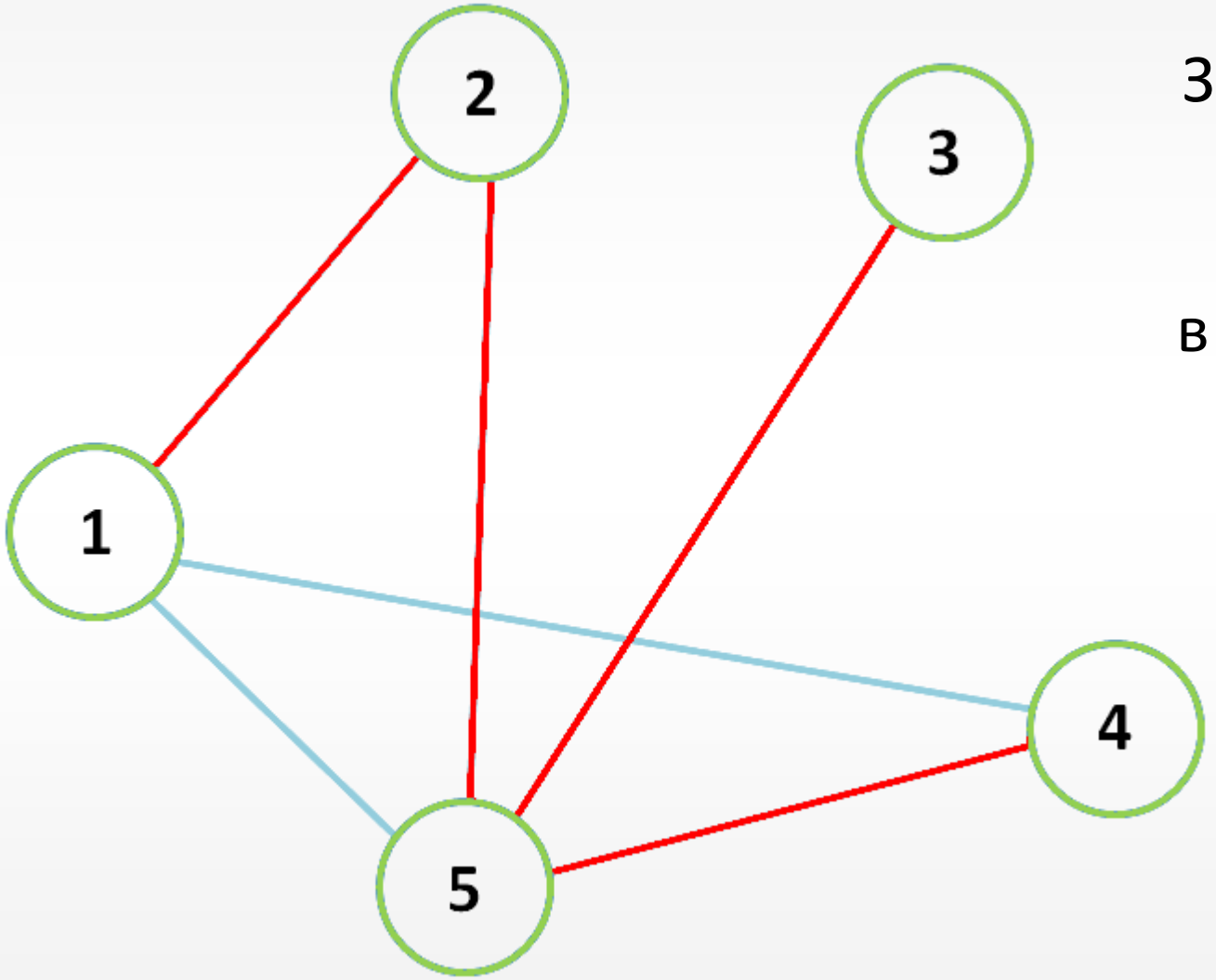
Рассматриваем вершину **5**.
Идём в следующую непосещённую вершину – **4**.

Обход в глубину (DFS)



Рассматриваем вершину **4**. Помечаем её как посещённую. Из неё некуда идти, поэтому алгоритм возвращается в **5**.

Обход в глубину (DFS)



Затем алгоритм возвращается в **3**, в **2** и, в конце концов, в **1**.

Обход в глубину (DFS)

```
vector <int> a[100];
int was[100];

void dfs(int v) {
      was[v]=1;
      for(int i=0;i<a[v].size();i++)
            if(!was[a[v][i]])
                  dfs(a[v][i]);
}
int main() {
      …
      dfs(1);
      …
}
```

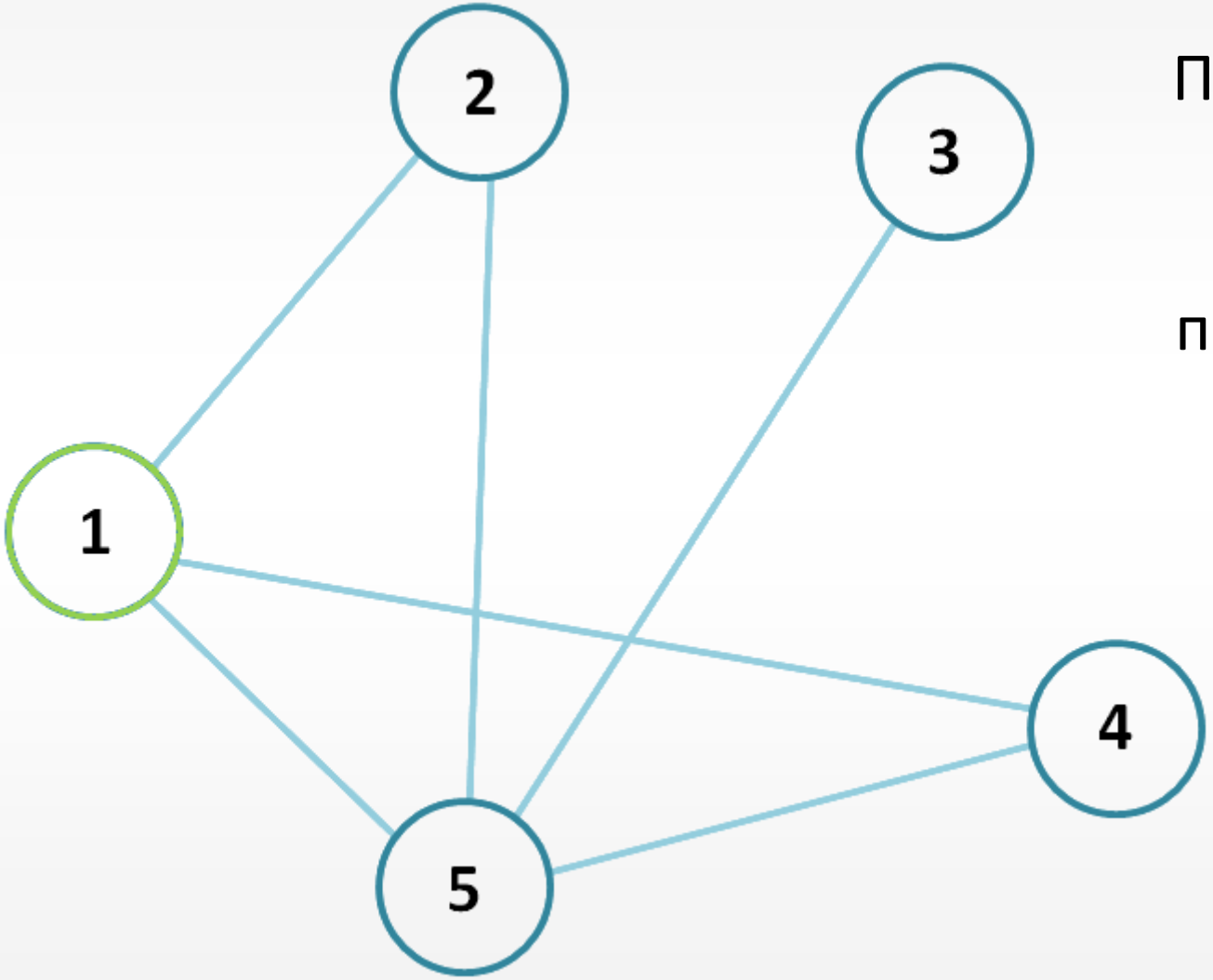
Обход в ширину (BFS)

Алгоритм обхода графа, который обрабатывает вершины в порядке увеличения их расстояния от первой.

BFS = **B**readth-**F**irst **S**earch.

Рассматривая вершину, алгоритм ищет тех её соседей, которых он ещё не посещал, и заносит их всех в очередь, а затем приступает к следующей в очереди вершине.

Обход в ширину (BFS)

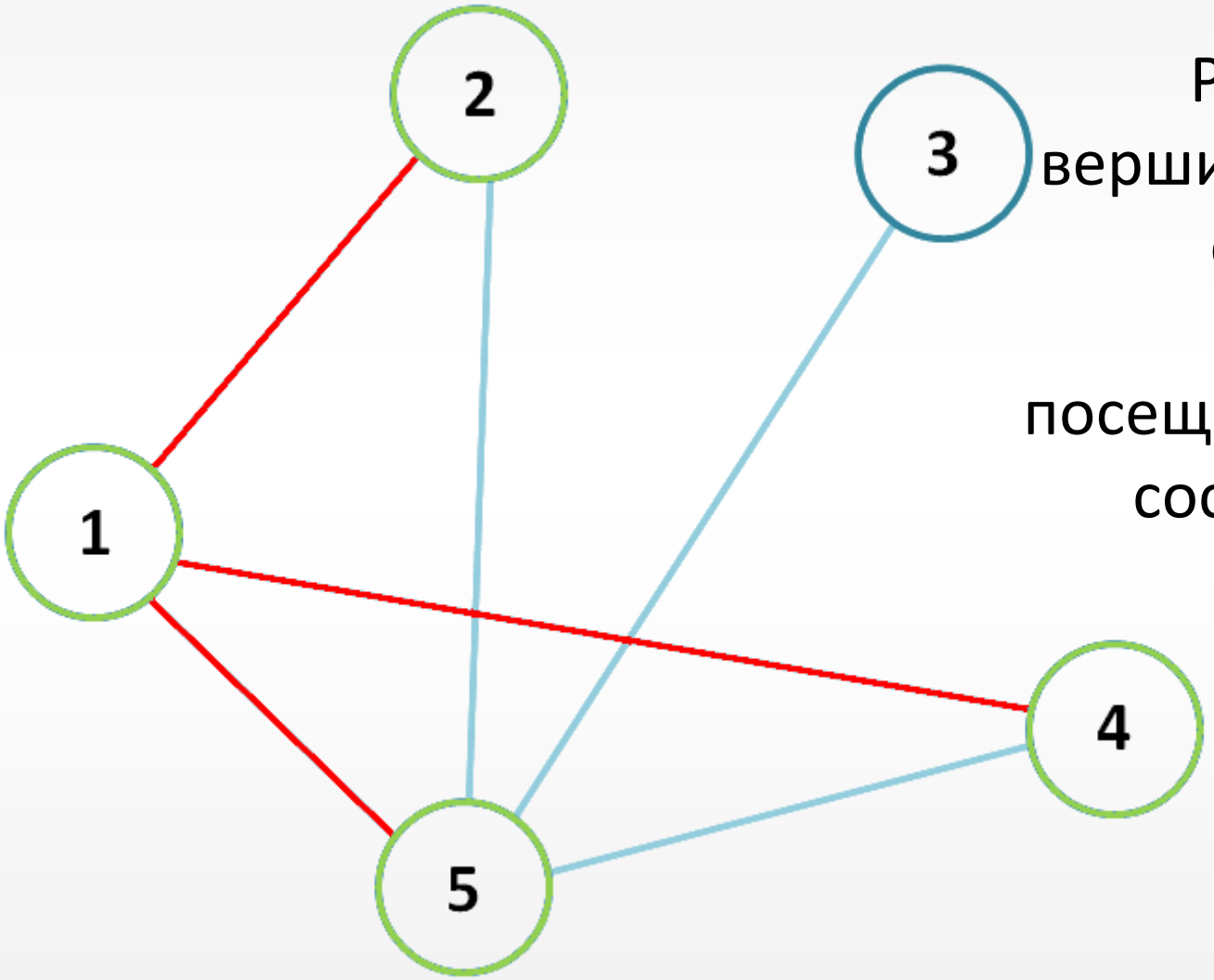


Перед запуском алгоритма вершина **1** помечается, как посещённая.

1

очередь

Обход в ширину (BFS)

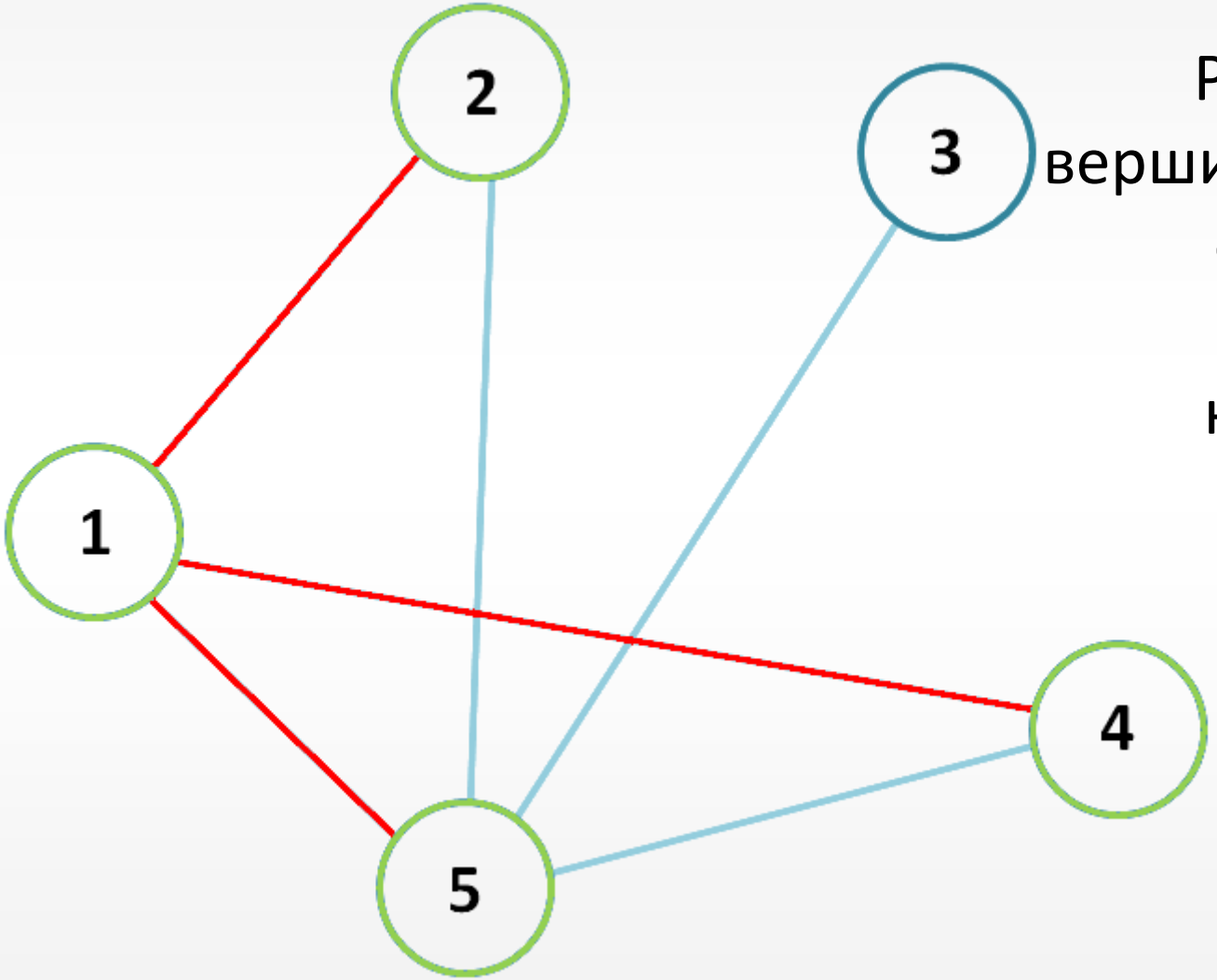


Рассматриваем вершину **1**. Убираем её из очереди. Помечаем, как посещённые, всех её соседей, попутно добавляя их в очередь:

1 2 4 5

очередь

Обход в ширину (BFS)

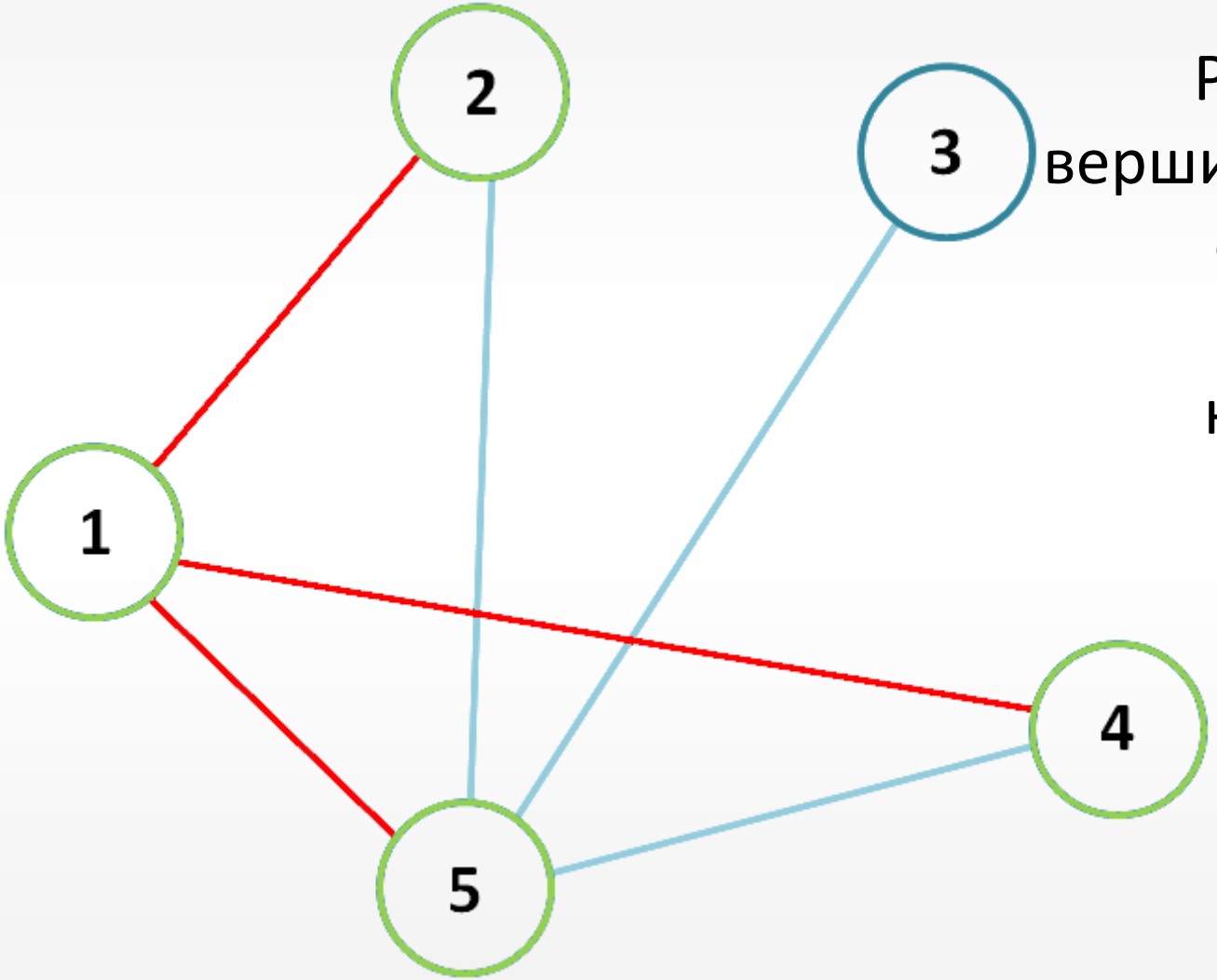


Рассматриваем вершину **2**. Убираем её из очереди. У неё нет непосещённых соседей.

1 2 4 5

очередь

Обход в ширину (BFS)

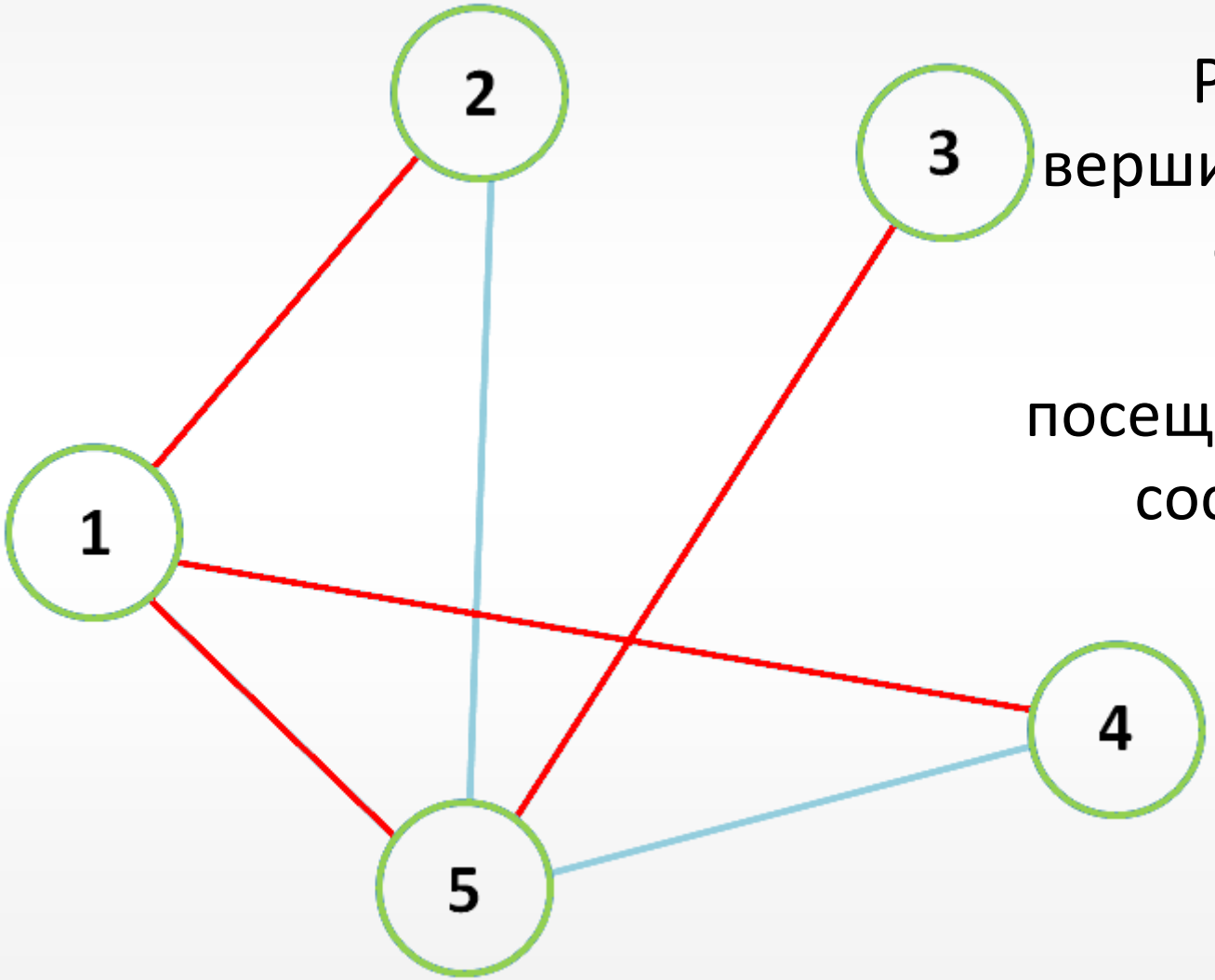


Рассматриваем вершину **4**. Убираем её из очереди. У неё нет непосещённых соседей.

1 2 4 **5**

очередь

Обход в ширину (BFS)

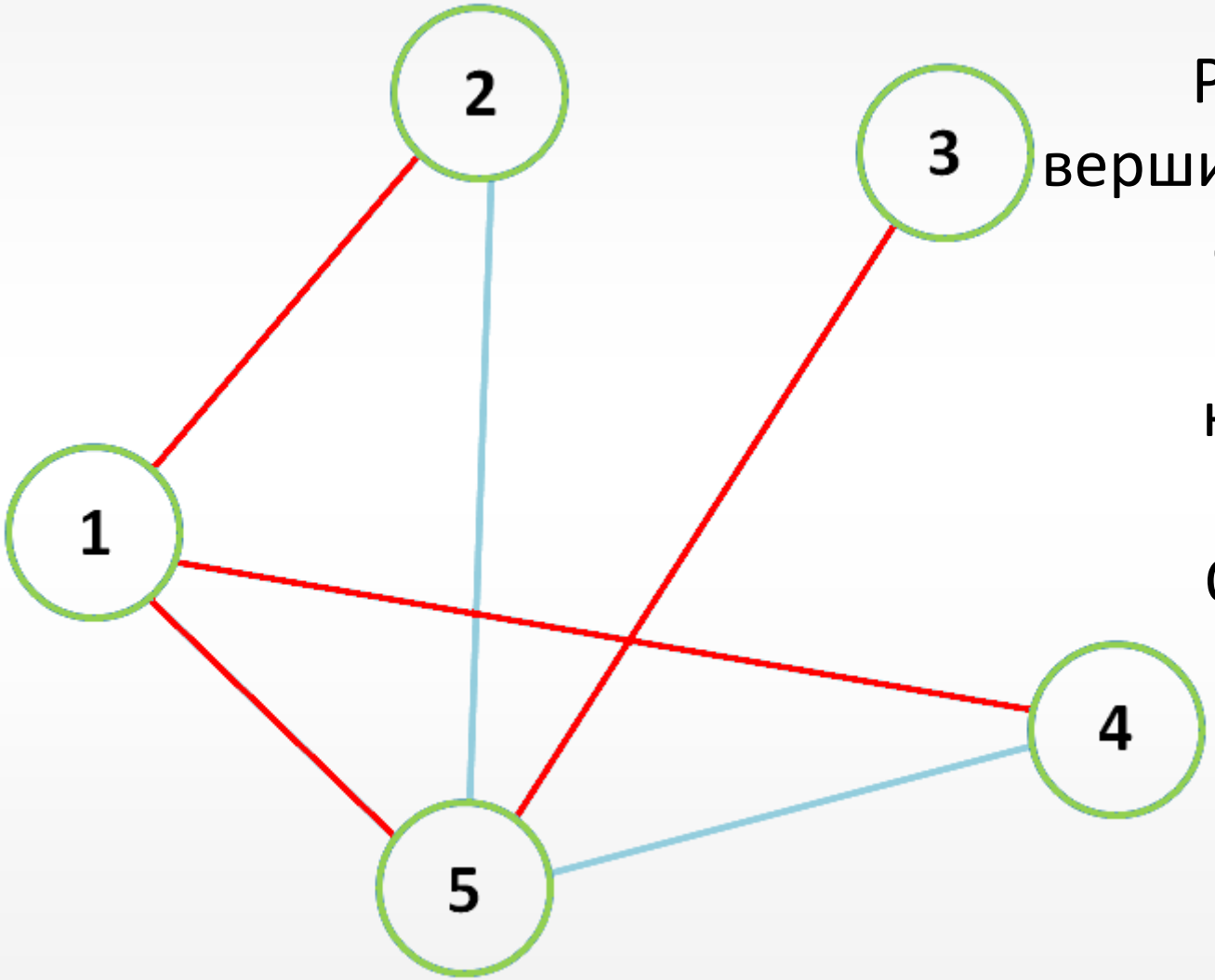


Рассматриваем вершину **5**. Убираем её из очереди. Помечаем, как посещённые, всех её соседей, попутно добавляя их в очередь:

1 2 4 5 **3**

очередь

Обход в ширину (BFS)



Рассматриваем вершину **3**. Убираем её из очереди.
У неё нет непосещённых соседей.
Очередь пуста.

1 2 4 5 3

очередь

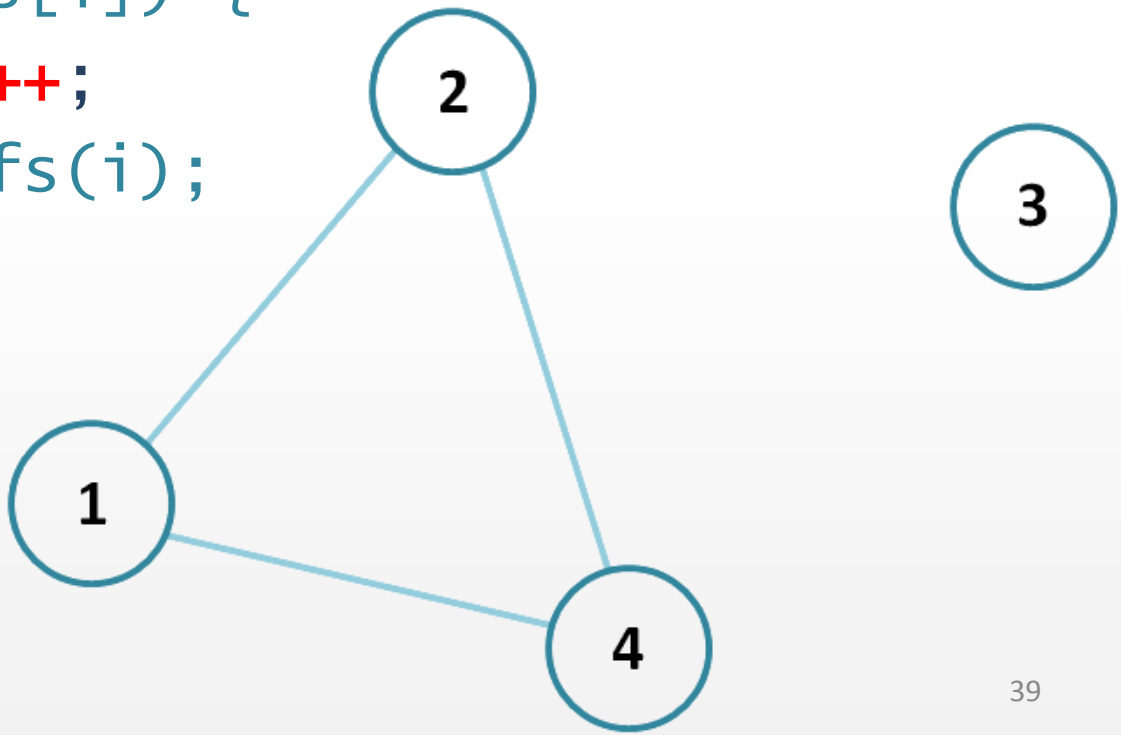
Обход в ширину (BFS)

```
vector <int> a[100];
int was[100];

void bfs() {
        was[1]=1;
        queue <int> q;
        q.push(1);
        while(!q.empty) {
                int v = q.front();
                for(int i=0;i<a[v].size();i++)
                        if(!was[a[v][i]]) {
                                was[a[v][i]]=1;
                                q.push(a[v][i]);
                        }
                q.pop();
        }
}
int main() {
        …
        bfs();
        …
}
```

Поиск компонент связности

```
int main() {
      …
      int k=0;
      for(int i=0;i<n;i++)
            if(!was[i]) {
                  k++;
                  dfs(i);
            }
      …
}
```



Вопросы?