Графы

Филипп Воронов

Филипп Воронов

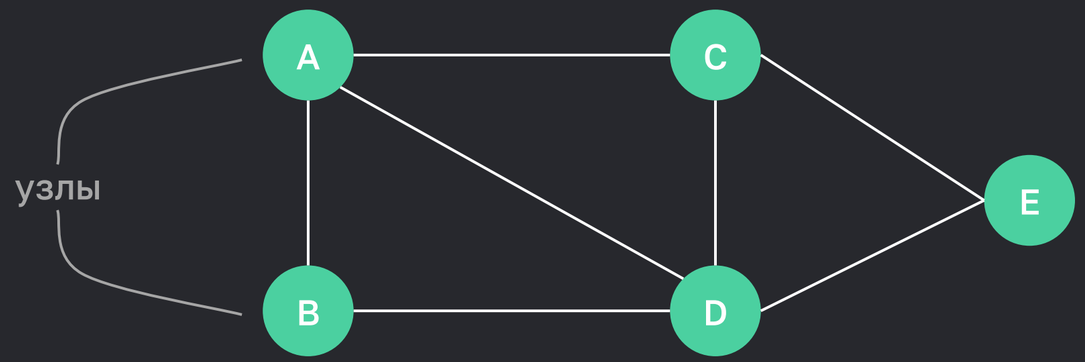
Teamlead, Поиск Mail.ru

Аккаунты в соц.сетях

@Филипп Воронов

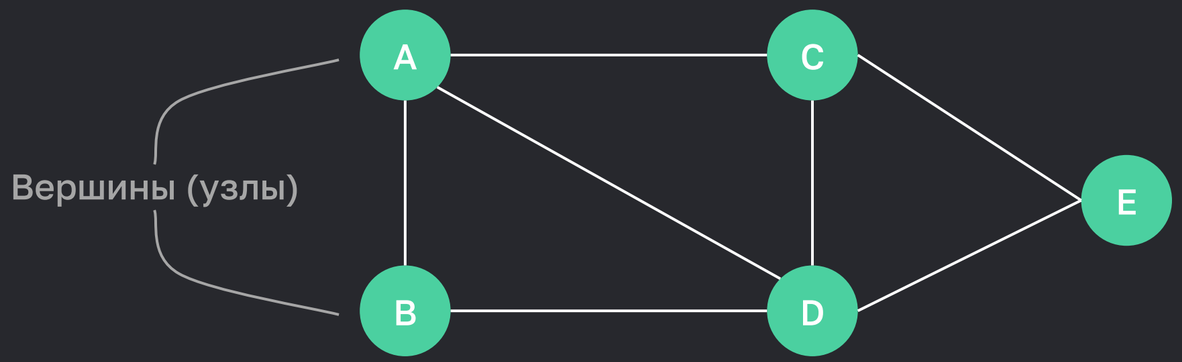
Что такое граф?

Узел. Как и в случаях со связным списком, пирамидой и другими структурами данных, обратимся к понятию узла: обёртки над элементом, которая может быть связана с другими узлами



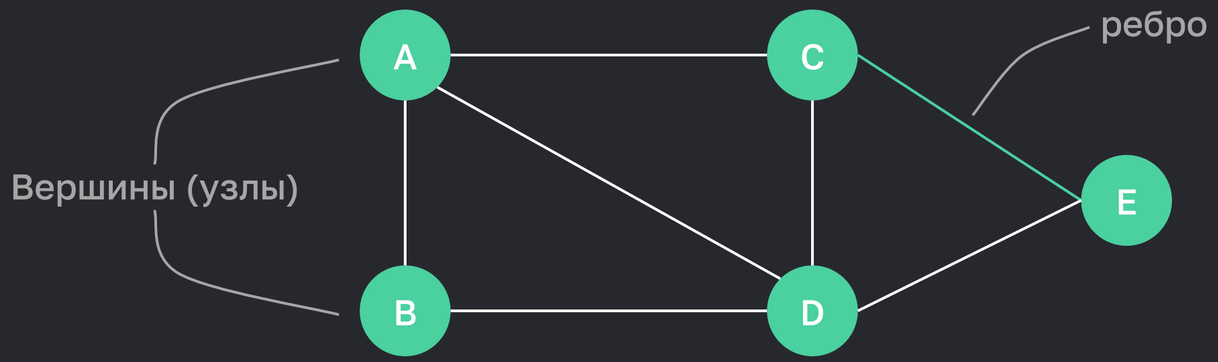
Что такое граф?

В теории графов такие узлы называются вершинами,
а их количество в графе обозначается как V (англ. vertex — вершина)



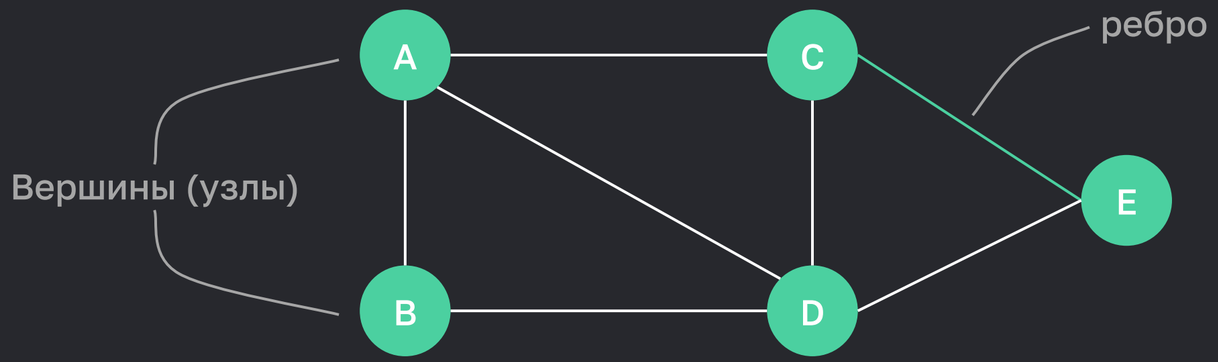
Что такое граф?

Рёбра. В отличие от тех же двоичных деревьев, в графах любая вершина может иметь переход в любую другую вершину. Такой переход называется ребром



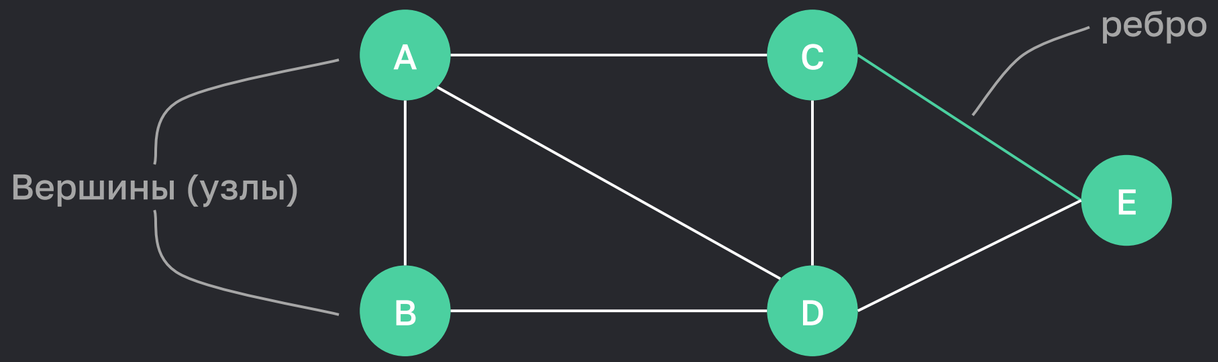
Что такое граф?

Важно, что переход не имеет направления, поэтому если из вершины A есть переход в вершину B, то считается, что и из B есть переход в вершину A. Такие вершины называются смежными



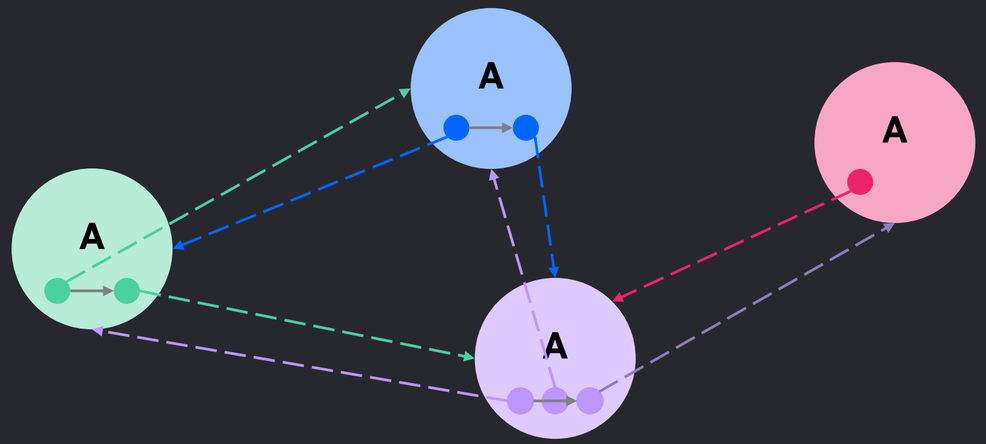
Что такое граф?

Количество рёбер в графе обозначается как E (англ. edge — ребро).
Максимальное значение для E — это V^2



Представление графа

Представление ссылками. В каждой вершине мы храним элемент и список ссылок на смежные ей вершины. Граф будет хранить список всех вершин графа




```
Vertex {
  e: элемент,
  adjacent: [смежные вершины]
}
Graph {
  vertices: [вершины графа]
}
```

Занимаемая память O(V + E).

Представление графа

Представление матрицей смежности. Иногда удобнее отказаться от ссылок и завести двумерный массив, где в каждой ячейке хранится информация о том, есть ли между i-й и j-й вершинами ребро

	A	B	C	D
A	-	1	0	0
B	1	-	1	0
C	0	1	-	0
D	0	0	0	-

В ячейке AB есть 1, значит есть ребро. AC лежит 0, значит нет ребра

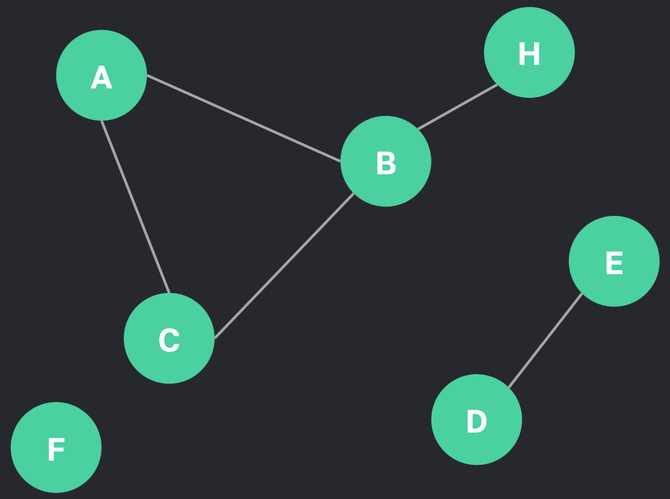
Представление графа

```
Graph {
  vertices: [элементы вершин графа]
  adjacent: [i x j = смежны ли i j]
}
```

Занимаемая память $O(V^2)$

Достижимость и связность

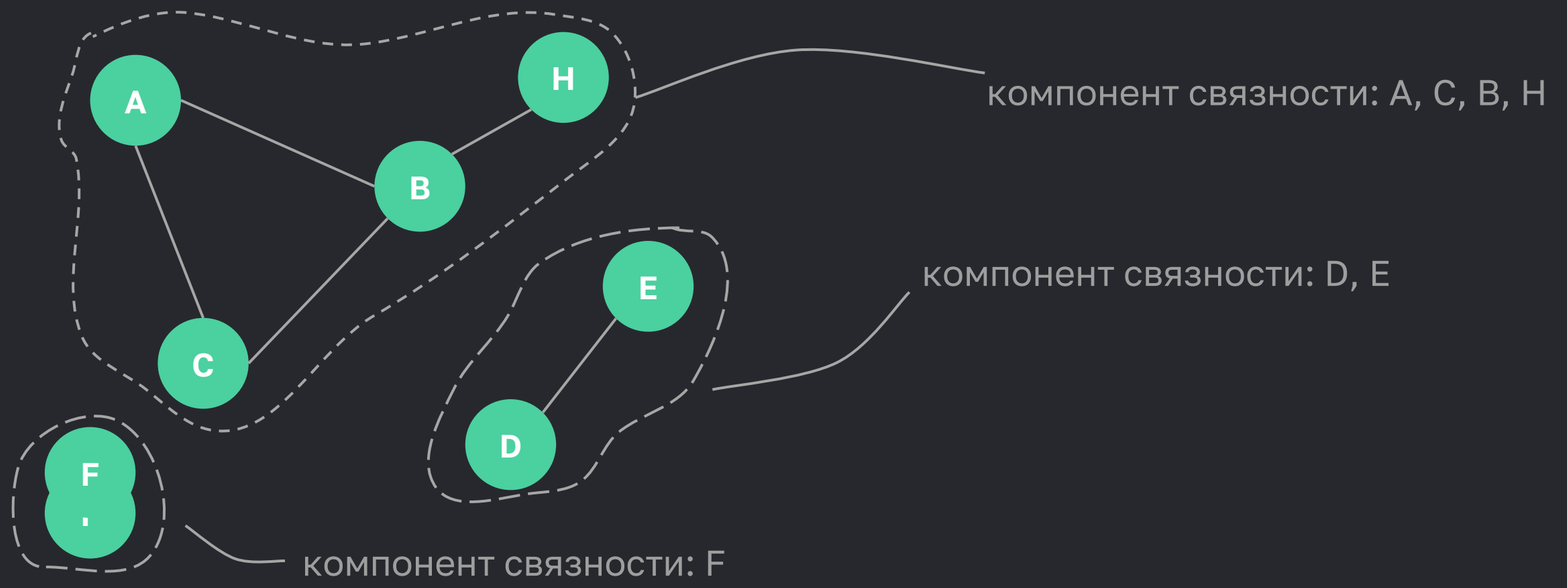
Достижимость. Одна вершина называется достижимой из другой, если до неё можно дойти по рёбрам.



Мы видим, что B достижима из C, но недостижима из E. Заметьте, что если первая вершина достижима из второй, то и вторая достижима из первой

Достижимость и связность

Связность. Граф называется связным, если любая вершина достижима из другой (на примере граф несвязный).



Компонентой связности называется набор из некоторых вершин графа, где любые две вершины достижимы друг из друга и никакую другую вершину нельзя добавить в набор, не нарушив это свойство

Обходы графа

Обход в глубину

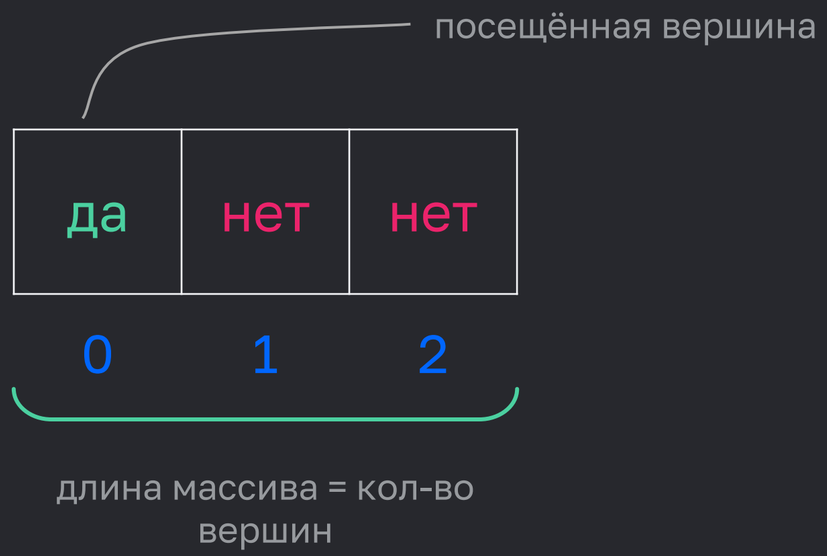
Что такое обход графа? Это алгоритм посещения всех его вершин.

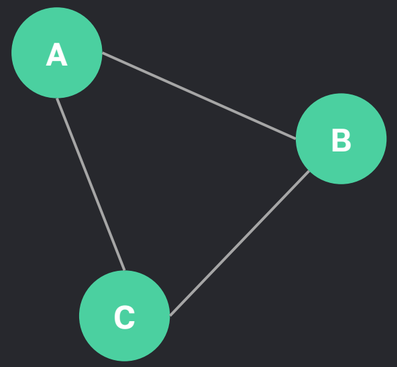
Существует два основных обхода:

- в глубину
- ширину

Обход в глубину

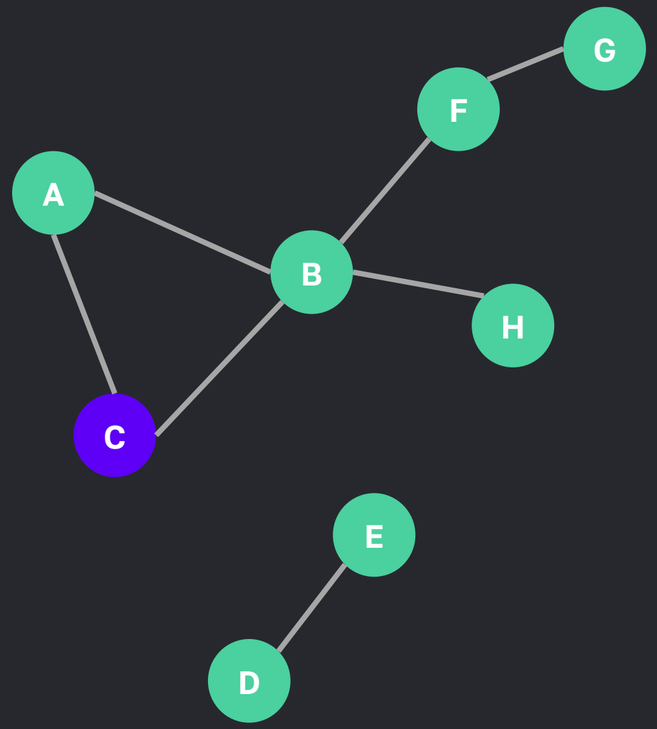
Обход в глубину. Чтобы не ходить по кругу заведем массив,
в котором будем запоминать, какие вершины мы уже посетили





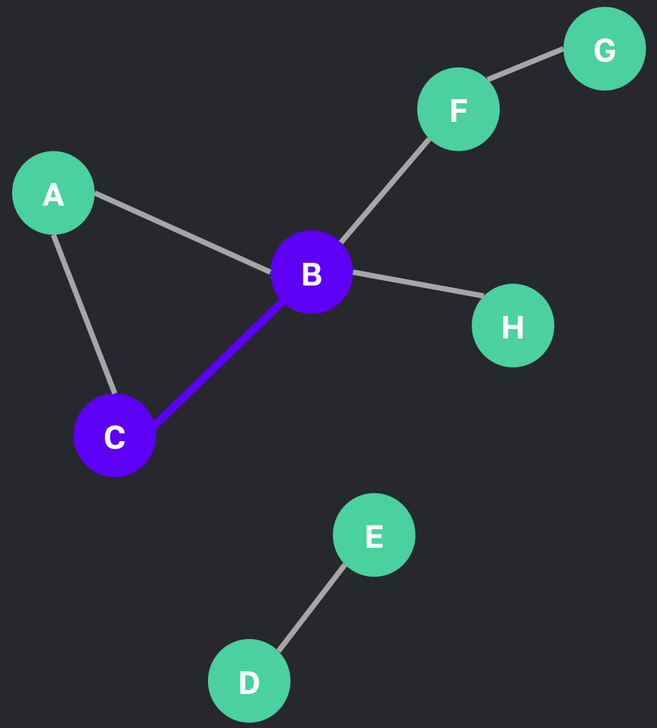
Обход в глубину

Обход в глубину. Пройдёмся циклом по вершинам графа и для каждой непосещённой запустим обход с неё



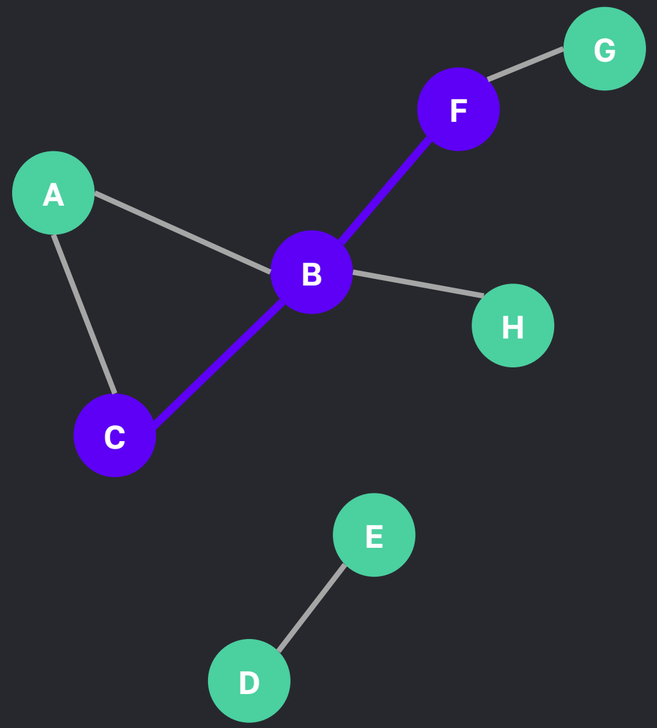
Обход в глубину

Обход в глубину. Пройдёмся циклом по вершинам графа и для каждой непосещённой запустим обход с неё



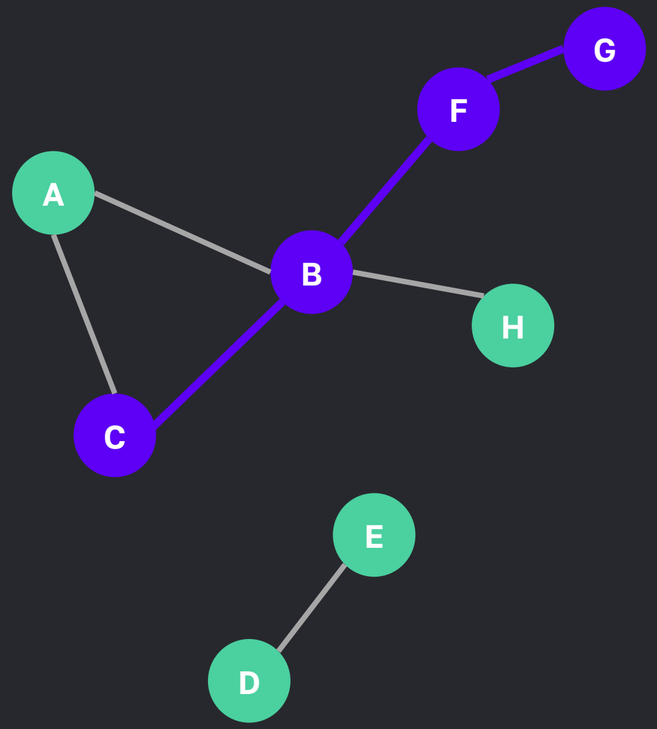
Обход в глубину

Обход в глубину. Пройдёмся циклом по вершинам графа и для каждой непосещённой запустим обход с неё



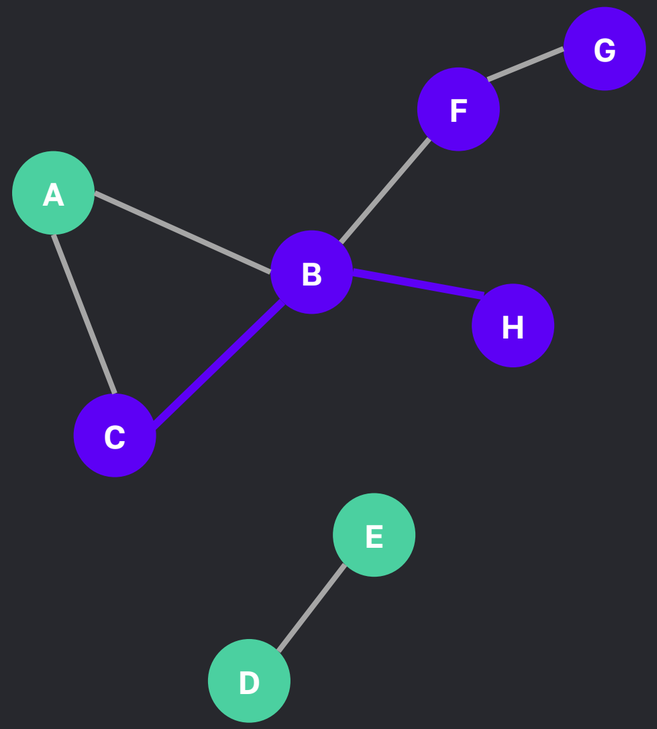
Обход в глубину

Обход в глубину. Пройдёмся циклом по вершинам графа и для каждой непосещённой запустим обход с неё



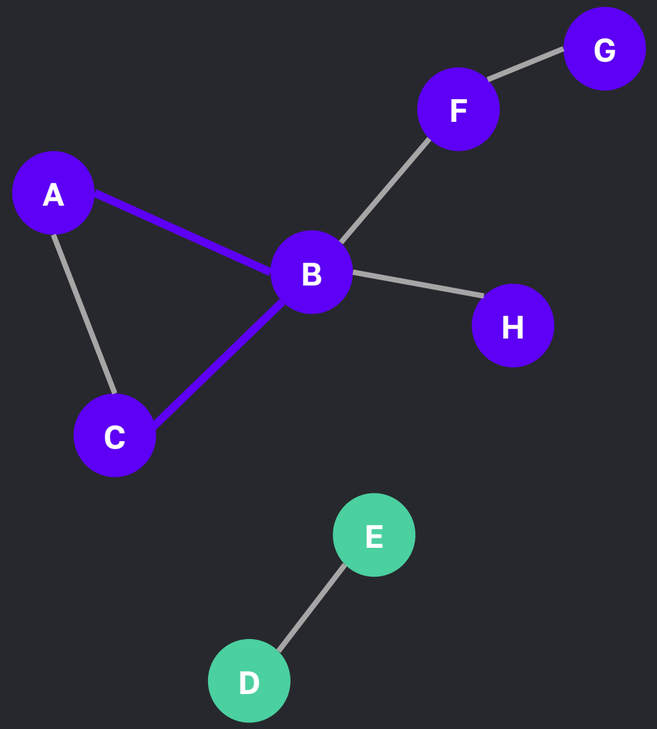
Обход в глубину

Обход в глубину. Пройдёмся циклом по вершинам графа и для каждой непосещённой запустим обход с неё



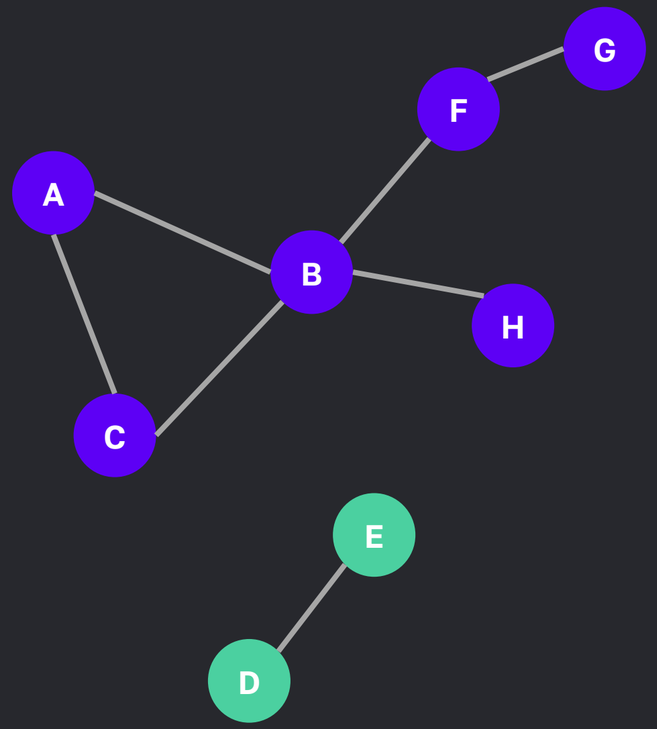
Обход в глубину

Обход в глубину. Пройдёмся циклом по вершинам графа и для каждой непосещённой запустим обход с неё



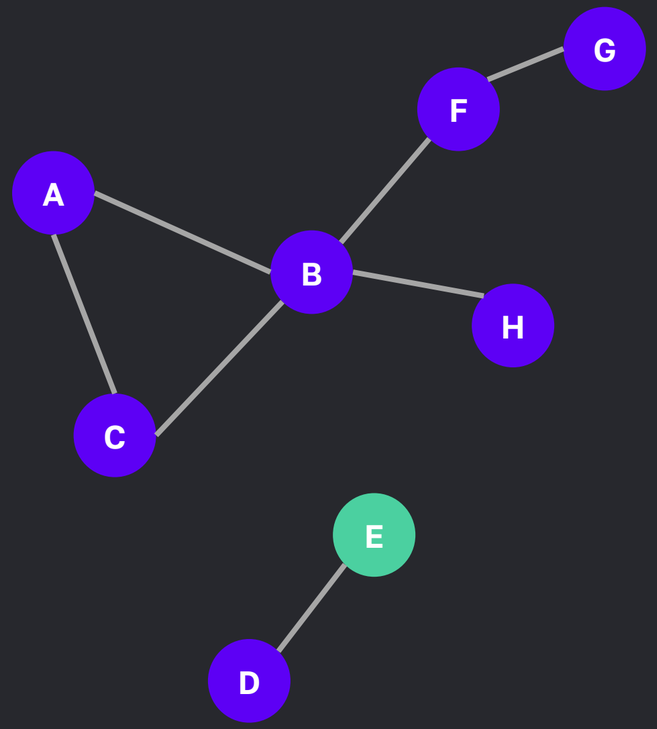
Обход в глубину

Обход в глубину. Пройдёмся циклом по вершинам графа и для каждой непосещённой запустим обход с неё



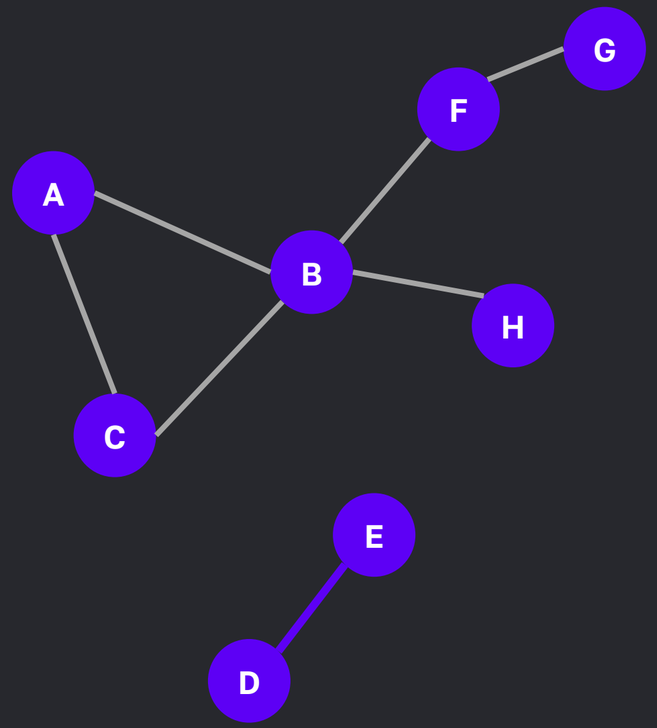
Обход в глубину

Обход в глубину. Пройдёмся циклом по вершинам графа и для каждой непосещённой запустим обход с неё



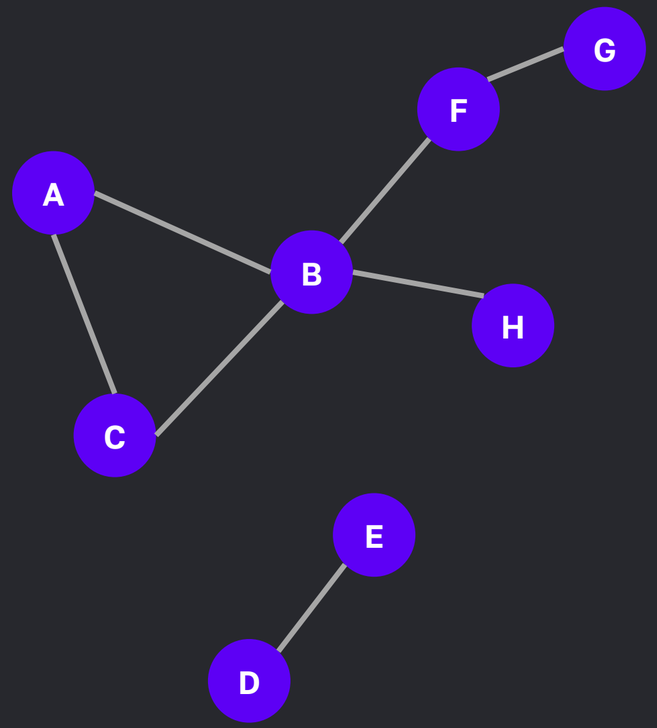
Обход в глубину

Обход в глубину. Пройдёмся циклом по вершинам графа и для каждой непосещённой запустим обход с неё



Обход в глубину

Обход в глубину. Пройдёмся циклом по вершинам графа и для каждой непосещённой запустим обход с неё



Обход в глубину

Псевдокод:

```
dfs(graph, vertex, visited):
  посетили vertex!
  visited[vertex] = да
  for v из смежных(vertex)
    if не visited[v]
      dfs(graph, v, visited)

dfs(graph):
  visited = [V раз нет]
  for v из вершины(graph)
    if не visited[v]
      dfs(graph, v, visited)
```

Комментарий
Запуск обхода в графе с вершины vertex
и со вспомогательным массивом посещённых вершин

Визуализация алгоритма https://visualgo.net/en/dfsbfs

Обход в глубину

Псевдокод:

```
dfs(graph, vertex, visited):
  посетили vertex!
  visited[vertex] = да
  for v из смежных(vertex)
    if не visited[v]
      dfs(graph, v, visited)

dfs(graph):
  visited = [V раз нет]
  for v из вершины(graph)
    if не visited[v]
      dfs(graph, v, visited)
```

Комментарий

Выбираем эту вершину как следующую, посещённую обходом

Обход в глубину

Псевдокод:

```
dfs(graph, vertex, visited):
  посетили vertex!
  visited[vertex] = да
  for v из смежных(vertex)
    if не visited[v]
      dfs(graph, v, visited)

dfs(graph):
  visited = [V раз нет]
  for v из вершины(graph)
    if не visited[v]
      dfs(graph, v, visited)
```

Комментарий
Помечаем, что мы её посетили

Обход в глубину

Псевдокод:

```
dfs(graph, vertex, visited):
  посетили vertex!
  visited[vertex] = да
  for v из смежных(vertex)
    if не visited[v]
      dfs(graph, v, visited)

dfs(graph):
  visited = [V раз нет]
  for v из вершины(graph)
    if не visited[v]
      dfs(graph, v, visited)
```

Комментарий
Перебираем все смежные ей вершины

Обход в глубину

Псевдокод:

```
dfs(graph, vertex, visited):
  посетили vertex!
  visited[vertex] = да
  for v из смежных(vertex)
    if не visited[v]
      dfs(graph, v, visited)

dfs(graph):
  visited = [V раз нет]
  for v из вершины(graph)
    if не visited[v]
      dfs(graph, v, visited)
```

Комментарий

Если в какую-то вершину ещё не заходили обходом, то рекурсивно запустим обход из неё

Обход в глубину

Псевдокод:

```
dfs(graph, vertex, visited):
  посетили vertex!
  visited[vertex] = да
  for v из смежных(vertex)
    if не visited[v]
      dfs(graph, v, visited)

dfs(graph):
  visited = [V раз нет]
  for v из вершины(graph)
    if не visited[v]
      dfs(graph, v, visited)
```

Комментарий
Обход всего графа

Обход в глубину

Псевдокод:

```
dfs(graph, vertex, visited):
  посетили vertex!
  visited[vertex] = да
  for v из смежных(vertex)
    if не visited[v]
      dfs(graph, v, visited)

dfs(graph):
  visited = [V раз нет]
  for v из вершины(graph)
    if не visited[v]
      dfs(graph, v, visited)
```

Комментарий

Создаём вспомогательный массив для отметок посещений вершин

Обход в глубину

Псевдокод:

```
dfs(graph, vertex, visited):
  посетили vertex!
  visited[vertex] = да
  for v из смежных(vertex)
    if не visited[v]
      dfs(graph, v, visited)

dfs(graph):
  visited = [V раз нет]
  for v из вершины(graph)
    if не visited[v]
      dfs(graph, v, visited)
```

Комментарий

Главный цикл: перебирает все вершины графа

Обход в глубину

Псевдокод:

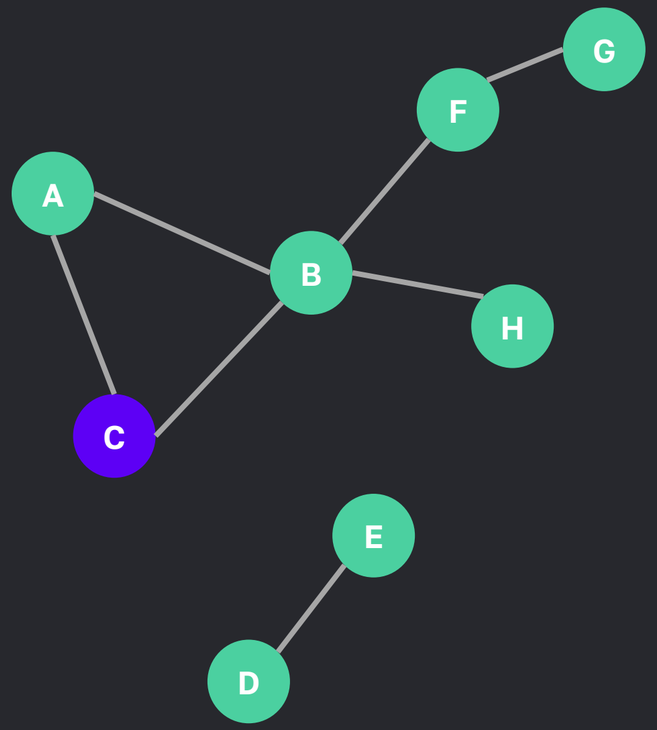
```
dfs(graph, vertex, visited):
  посетили vertex!
  visited[vertex] = да
  for v из смежных(vertex)
    if не visited[v]
      dfs(graph, v, visited)

dfs(graph):
  visited = [V раз нет]
  for v из вершины(graph)
    if не visited[v]
      dfs(graph, v, visited)
```

Комментарий

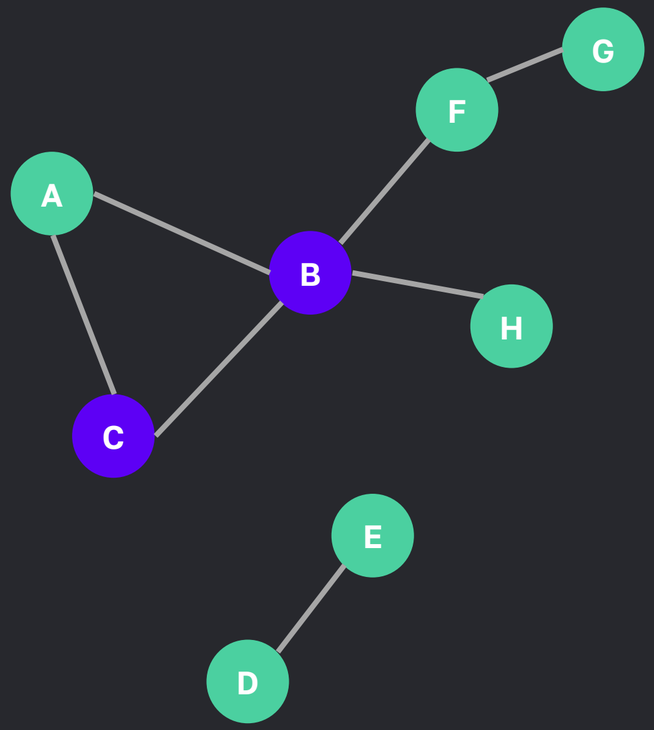
Если вершину ещё не посетили, то запускаем из неё рекурсивный обход через вспомогательный метод, изложенный выше

Обход в ширину

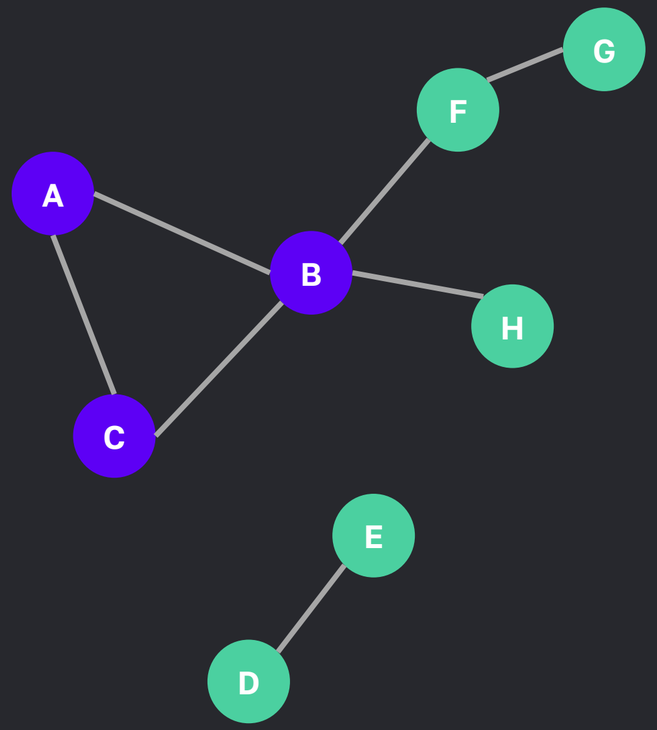
Обход в ширину



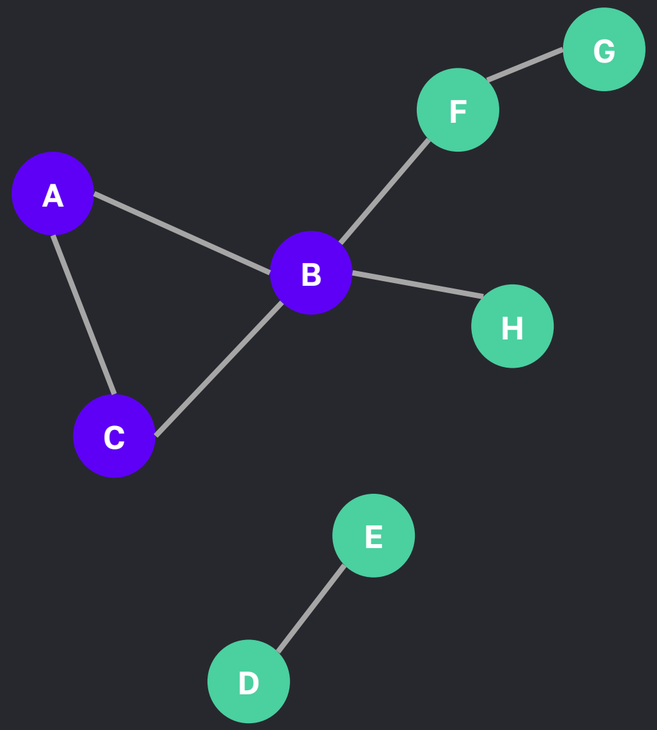
Обход в ширину



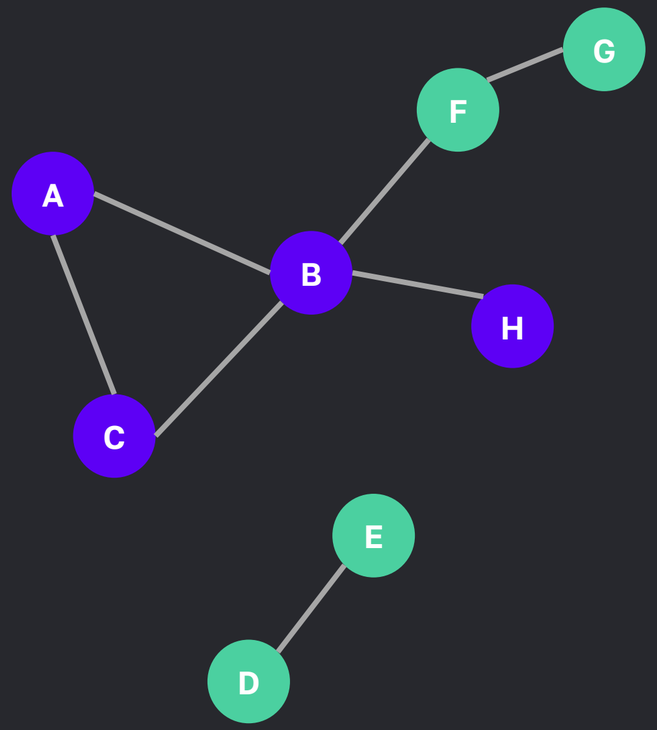
Обход в ширину



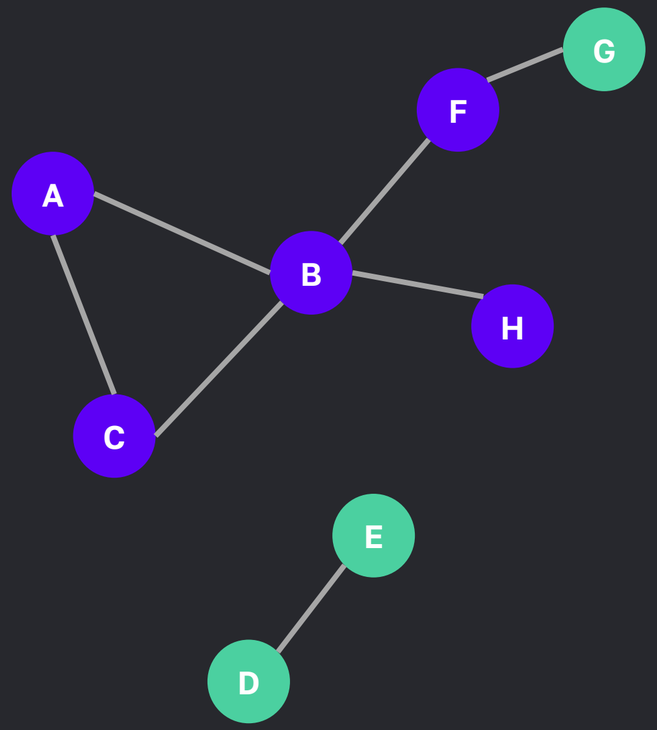
Обход в ширину



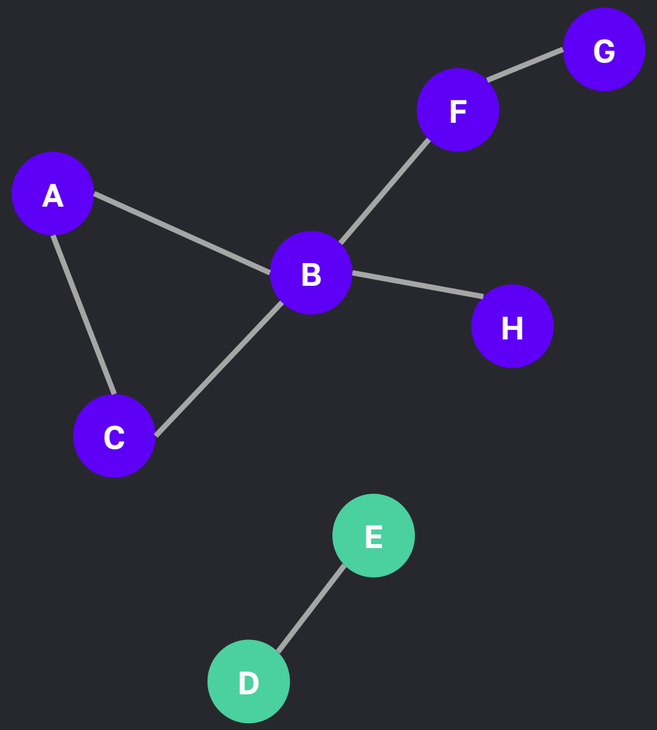
Обход в ширину



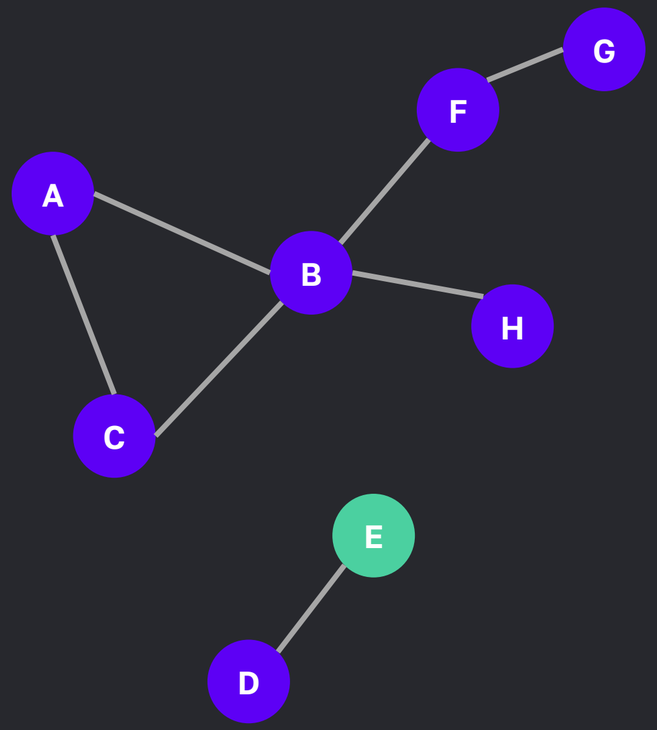
Обход в ширину



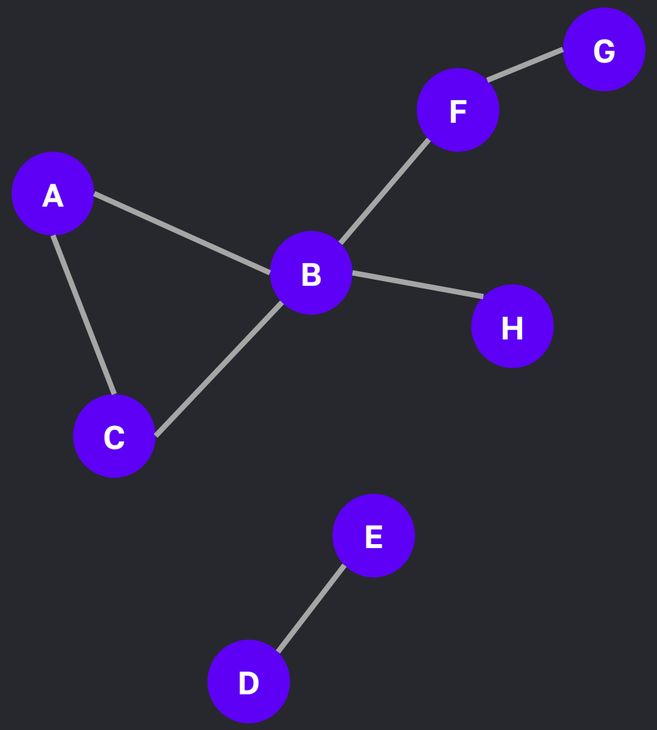
Обход в ширину



Обход в ширину



Обход в ширину



Обход в ширину

Псевдокод:

```
bfs(graph):
  visited = [V раз нет]
  queue = new Queue
  for v из вершины(graph)
    if не visited[v]
      queue.add(v)
      while queue не пуст
        vertex = queue.next()
        посетили vertex!
        visited[vertex] = да
        for v из смежные(vertex)
          if не visited[v]
            queue.add(v)
```

Комментарий
Обход в ширину

Визуализация алгоритма https://visualgo.net/en/dfsbfs

Обход в ширину

Псевдокод:

```
bfs(graph):
  visited = [V раз нет]
  queue = new Queue
  for v из вершины(graph)
    if не visited[v]
      queue.add(v)
      while queue не пуст
        vertex = queue.next()
        посетили vertex!
        visited[vertex] = да
        for v из смежные(vertex)
          if не visited[v]
            queue.add(v)
```

Комментарий
Такой же вспомогательный массив

Обход в ширину

Псевдокод:

```
bfs(graph):
  visited = [V раз нет]
  queue = new Queue
  for v из вершины(graph)
    if не visited[v]
      queue.add(v)
      while queue не пуст
        vertex = queue.next()
        посетили vertex!
        visited[vertex] = да
        for v из смежные(vertex)
          if не visited[v]
            queue.add(v)
```

Комментарий

Заводим очередь для упорядочивания вершин для обхода

Обход в ширину

Псевдокод:

```
bfs(graph):
  visited = [V раз нет]
  queue = new Queue
  for v из вершины(graph)
    if не visited[v]
      queue.add(v)
      while queue не пуст
        vertex = queue.next()
        посетили vertex!
        visited[vertex] = да
        for v из смежные(vertex)
          if не visited[v]
            queue.add(v)
```

Комментарий
Проходимся по всем вершинам графа

Обход в ширину

Псевдокод:

```
bfs(graph):
  visited = [V раз нет]
  queue = new Queue
  for v из вершины(graph)
    if не visited[v]
      queue.add(v)
      while queue не пуст
        vertex = queue.next()
        посетили vertex!
        visited[vertex] = да
        for v из смежные(vertex)
          if не visited[v]
            queue.add(v)
```

Комментарий

Если ещё не посетили вершину

Обход в ширину

Псевдокод:

```
bfs(graph):
  visited = [V раз нет]
  queue = new Queue
  for v из вершины(graph)
    if не visited[v]
      queue.add(v)
      while queue не пуст
        vertex = queue.next()
        посетили vertex!
        visited[vertex] = да
        for v из смежные(vertex)
          if не visited[v]
            queue.add(v)
```

Комментарий

Кладём в очередь вершину, с которой начинаем обход

Обход в ширину

Псевдокод:

```
bfs(graph):
  visited = [V раз нет]
  queue = new Queue
  for v из вершины(graph)
    if не visited[v]
      queue.add(v)
      while queue не пуст
        vertex = queue.next()
        посетили vertex!
        visited[vertex] = да
        for v из смежные(vertex)
          if не visited[v]
            queue.add(v)
```

Комментарий
Пока в очереди что-то есть

Обход в ширину

Псевдокод:

```
bfs(graph):
  visited = [V раз нет]
  queue = new Queue
  for v из вершины(graph)
    if не visited[v]
      queue.add(v)
      while queue не пуст
        vertex = queue.next()
        посетили vertex!
        visited[vertex] = да
        for v из смежные(vertex)
          if не visited[v]
            queue.add(v)
```

Комментарий

Вынимаем из очереди следующую вершину

Обход в ширину

Псевдокод:

```
bfs(graph):
  visited = [V раз нет]
  queue = new Queue
  for v из вершины(graph)
    if не visited[v]
      queue.add(v)
      while queue не пуст
        vertex = queue.next()
        посетили vertex!
        visited[vertex] = да
        for v из смежные(vertex)
          if не visited[v]
            queue.add(v)
```

Комментарий

Выбираем эту вершину как следующую, посещённую обходом

Обход в ширину

Псевдокод:

```
bfs(graph):
  visited = [V раз нет]
  queue = new Queue
  for v из вершины(graph)
    if не visited[v]
      queue.add(v)
      while queue не пуст
        vertex = queue.next()
        посетили vertex!
        visited[vertex] = да
        for v из смежные(vertex)
          if не visited[v]
            queue.add(v)
```

Комментарий
Помечаем вершину как посещённую

Обход в ширину

Псевдокод:

```
bfs(graph):
  visited = [V раз нет]
  queue = new Queue
  for v из вершины(graph)
    if не visited[v]
      queue.add(v)
      while queue не пуст
        vertex = queue.next()
        посетили vertex!
        visited[vertex] = да
        for v из смежные(vertex)
          if не visited[v]
            queue.add(v)
```

Комментарий

Все смежные непосещённые обходом вершины кладём в очередь

Обход в ширину

Существует два основных обхода:

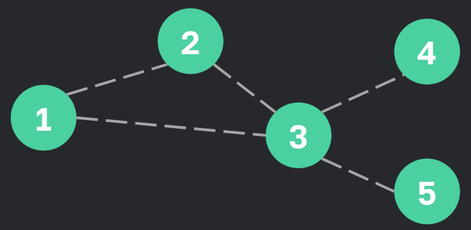
- в глубину
- ширину

Асимптотика. Асимптотика работы обоих алгоритмов зависит от типа реализации графа. Для реализации на ссылках — это $O(V + E)$, для реализации на матрице смежности — это $O(V^2)$

Проверка на наличие цикла

Проверка на наличие цикла

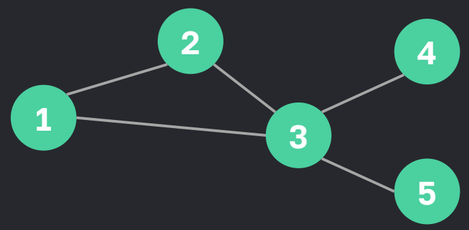
Задача. Дан граф:



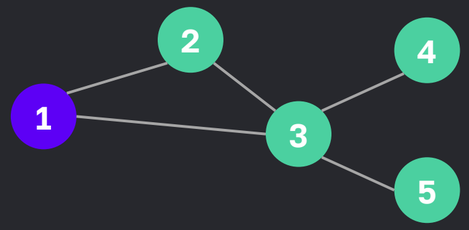
Нужно определить, есть ли в нём циклы

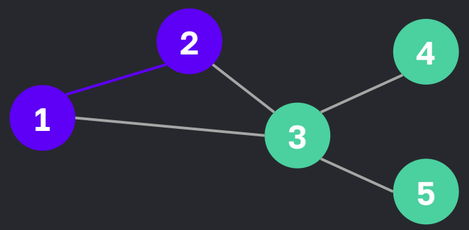
Проверка на наличие цикла

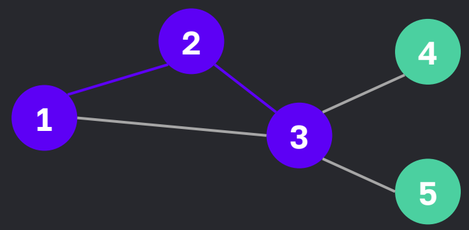
Решение: просто запустим обход в глубину.



Если, перебирая смежные вершины, мы наткнёмся на уже посещённую, то цикл в графе есть

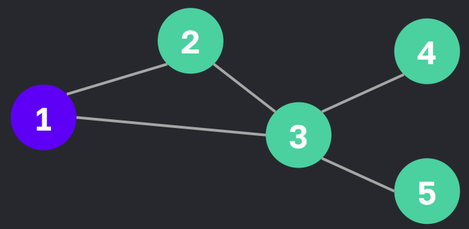


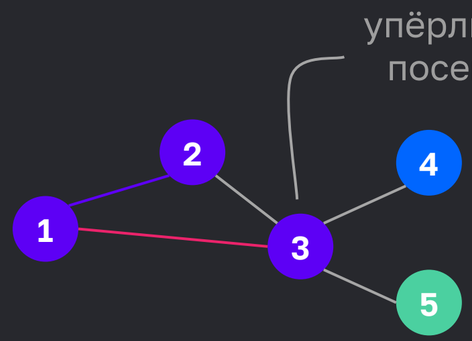


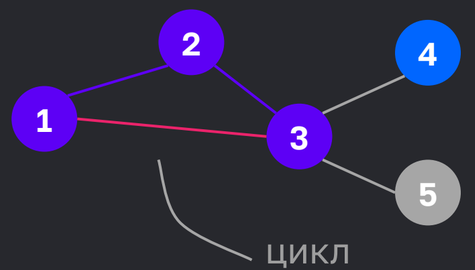


Проверка на наличие цикла

Если, перебирая смежные вершины, мы наткнёмся на уже посещённую, то цикл в графе есть

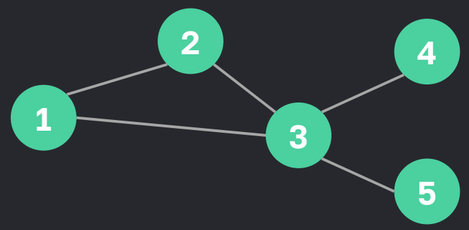


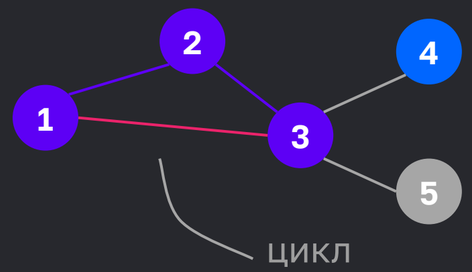




Проверка на наличие цикла

Решение: просто запустим обход в глубину.





Не забываем, что необходимо исключить случай вершины, из которой мы напрямую пришли бы в текущую, иначе такой тривиальный цикл (A B A) существует в любом графе, где есть хотя бы одно ребро.

Асимптотика алгоритма равна асимптотике обхода в глубину

Проверка на наличие цикла

Псевдокод:

```
dfs(graph, vertex, visited, prev):
  visited[vertex] = да
  for v из смежные(vertex)
    if не visited[v]
      dfs(graph, v, visited)
    else if v не prev
      return есть цикл!

is_cyclic(graph):
  visited = [V раз нет]
  for v из вершины(graph)
    if не visited[v]
      if dfs(graph, v, visited, нету)
        есть цикл
  return нет циклов
```

Комментарий

Обход в глубину, который прекращается и возвращает «да», как только мы, перебирая смежные вершины, наткнёмся на уже посещённую (кроме той, из которой пришли в эту вершину)

Проверка на наличие цикла

Псевдокод:

```
dfs(graph, vertex, visited, prev):
  visited[vertex] = да
  for v из смежные(vertex)
    if не visited[v]
      dfs(graph, v, visited)
    else if v не prev
      return есть цикл!

is_cyclic(graph):
  visited = [V раз нет]
  for v из вершины(graph)
    if не visited[v]
      if dfs(graph, v, visited, нету)
        есть цикл
  return нет циклов
```

Комментарий
Отметка, что мы посетили вершину

Проверка на наличие цикла

Псевдокод:

```
dfs(graph, vertex, visited, prev):
  visited[vertex] = да
  for v из смежные(vertex)
    if не visited[v]
      dfs(graph, v, visited)
    else if v не prev
      return есть цикл!

is_cyclic(graph):
  visited = [V раз нет]
  for v из вершины(graph)
    if не visited[v]
      if dfs(graph, v, visited, нету)
        есть цикл
  return нет циклов
```

Комментарий

– Перебираем смежные, рекурсивно запускаем обход от непосещённых вершин

Проверка на наличие цикла

Псевдокод:

```
dfs(graph, vertex, visited, prev):
  visited[vertex] = да
  for v из смежные(vertex)
    if не visited[v]
      dfs(graph, v, visited)
    else if v не prev
      return есть цикл!

is_cyclic(graph):
  visited = [V раз нет]
  for v из вершины(graph)
    if не visited[v]
      if dfs(graph, v, visited, нету)
        есть цикл
  return нет циклов
```

Комментарий

Но если наткнулись на посещённую, и она не та, из которой пришли, то цикл есть

Проверка на наличие цикла

Псевдокод:

```
dfs(graph, vertex, visited, prev):
  visited[vertex] = да
  for v из смежные(vertex)
    if не visited[v]
      dfs(graph, v, visited)
    else if v не prev
      return есть цикл!

is_cyclic(graph):
  visited = [V раз нет]
  for v из вершины(graph)
    if не visited[v]
      if dfs(graph, v, visited, нету)
        есть цикл
  return нет циклов
```

Комментарий

Функция определения наличия циклов в графе

Проверка на наличие цикла

Псевдокод:

```
dfs(graph, vertex, visited, prev):
  visited[vertex] = да
  for v из смежные(vertex)
    if не visited[v]
      dfs(graph, v, visited)
    else if v не prev
      return есть цикл!

is_cyclic(graph):
  visited = [V раз нет]
  for v из вершины(graph)
    if не visited[v]
      if dfs(graph, v, visited, нету)
        есть цикл
  return нет циклов
```

Комментарий

Как в обычном dfs, запускаем везде обход. Если один из вызовов вернул «да» (нашёл цикл), значит цикл есть

Проверка на наличие цикла

Псевдокод:

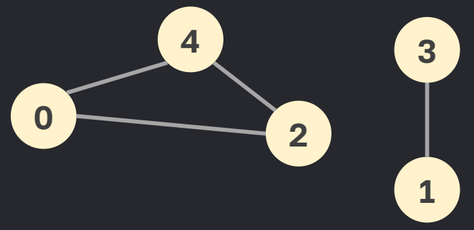
```
dfs(graph, vertex, visited, prev):
  visited[vertex] = да
  for v из смежные(vertex)
    if не visited[v]
      dfs(graph, v, visited)
    else if v не prev
      return есть цикл!

is_cyclic(graph):
  visited = [V раз нет]
  for v из вершины(graph)
    if не visited[v]
      if dfs(graph, v, visited, нету)
        есть цикл
  return нет циклов
```

Комментарий
Возвращаем, что «нет циклов»

Компоненты связности

Компоненты связности

Задача. Дан граф:



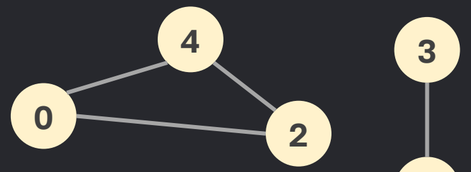
Необходимо найти компоненты связности. Результатом нашего поиска должен быть массив, где для каждой вершины будет указан номер компоненты связности

0	1	0	1	0
0	1	2	3	4

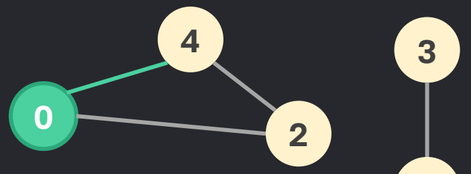
длина массива = кол-во вершин

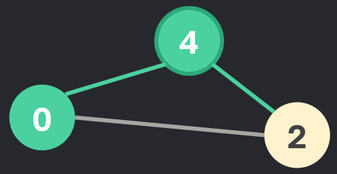
Компоненты связности

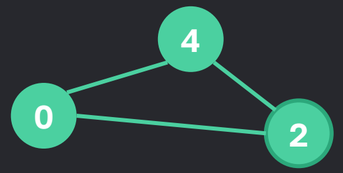
Решение: для этого немного видоизменим алгоритм обхода в глубину.



Теперь вместо хранения «да» для посещённой вершины, мы будем хранить номер компоненты связности



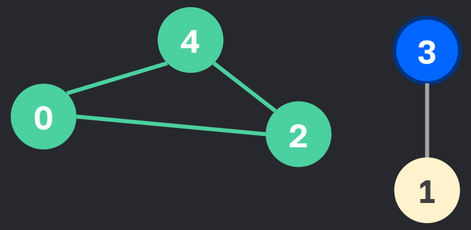


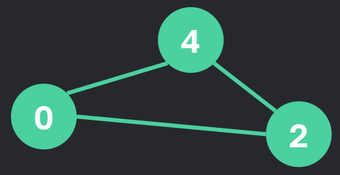


Зёленый помечаем номером 0

Компоненты связности

Каждый новый вызов обхода в глубину из ещё не посещённой вершины в глобальном цикле означает, что мы до неё ещё не дошли ни из одной посещённой вершины, а значит — это новая компонента связности

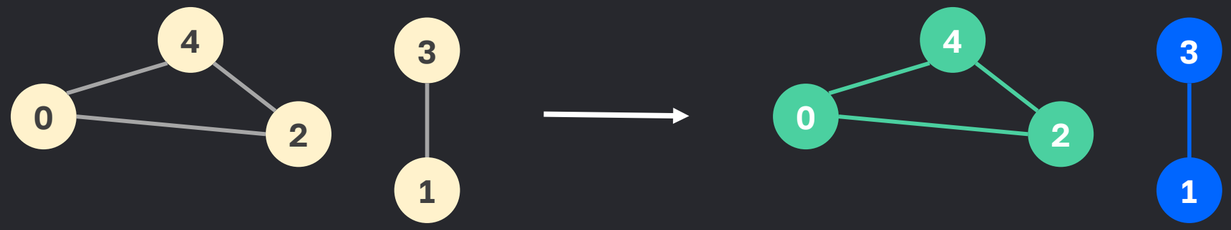




Синий помечаем номером 1

Компоненты связности

Решение: для этого немного видоизменим алгоритм обхода в глубину.



Зёленый помечаем номером 0

Синий помечаем номером 1

Асимптотика аналогична асимптотике обхода в глубину

Компоненты связности

Псевдокод:

```
dfs(graph, vertex, cids, cid):
  cids[vertex] = cid
  for v из смежные(vertex)
    if cids[v] = 0
      dfs(graph, v, visited, cid)

components(graph):
  cids = [V раз 0]
  cid = 1
  for v из вершины(graph)
    if cids[v] = 0
      dfs(graph, v, cids, cid)
      cid += 1
```

Комментарий

– Видоизменённая функция рекурсивного обхода в глубину

Компоненты связности

Псевдокод:

```
dfs(graph, vertex, cids, cid):
  cids[vertex] = cid
  for v из смежные(vertex)
    if cids[v] = 0
      dfs(graph, v, visited, cid)

components(graph):
  cids = [V раз 0]
  cid = 1
  for v из вершины(graph)
    if cids[v] = 0
      dfs(graph, v, cids, cid)
      cid += 1
```

Комментарий

Cids — массив для ответа, cid — номер компоненты связности этого рекурсивного обхода

Выставляем посещаемой вершине этот номер компоненты связности

Компоненты связности

Псевдокод:

```
dfs(graph, vertex, cids, cid):
  cids[vertex] = cid
  for v из смежные(vertex)
    if cids[v] = 0
      dfs(graph, v, visited, cid)

components(graph):
  cids = [V раз 0]
  cid = 1
  for v из вершины(graph)
    if cids[v] = 0
      dfs(graph, v, cids, cid)
      cid += 1
```

Комментарий

Обычный для dfs перебор непосещённых смежных вершин для рекурсивного запуска. Посещённость определяем через непроставленность номера компоненты связности для вершины

Компоненты связности

Псевдокод:

```
dfs(graph, vertex, cids, cid):
  cids[vertex] = cid
  for v из смежные(vertex)
    if cids[v] = 0
      dfs(graph, v, visited, cid)

components(graph):
  cids = [V раз 0]
  cid = 1
  for v из вершины(graph)
    if cids[v] = 0
      dfs(graph, v, cids, cid)
      cid += 1
```

Комментарий

Метод поиска компонента связности

Компоненты связности

Псевдокод:

```
dfs(graph, vertex, cids, cid):
  cids[vertex] = cid
  for v из смежные(vertex)
    if cids[v] = 0
      dfs(graph, v, visited, cid)

components(graph):
  cids = [V раз 0]
  cid = 1
  for v из вершины(graph)
    if cids[v] = 0
      dfs(graph, v, cids, cid)
      cid += 1
```

Комментарий
Массив ответа

Компоненты связности

Псевдокод:

```
dfs(graph, vertex, cids, cid):
  cids[vertex] = cid
  for v из смежные(vertex)
    if cids[v] = 0
      dfs(graph, v, visited, cid)

components(graph):
  cids = [V раз 0]
  cid = 1
  for v из вершины(graph)
    if cids[v] = 0
      dfs(graph, v, cids, cid)
      cid += 1
```

Комментарий
Счётчик компонента связности

Компоненты связности

Псевдокод:

```
dfs(graph, vertex, cids, cid):
  cids[vertex] = cid
  for v из смежные(vertex)
    if cids[v] = 0
      dfs(graph, v, visited, cid)

components(graph):
  cids = [V раз 0]
  cid = 1
  for v из вершины(graph)
    if cids[v] = 0
      dfs(graph, v, cids, cid)
      cid += 1
```

Комментарий
Главный цикл запуска обхода

Компоненты связности

Псевдокод:

```
dfs(graph, vertex, cids, cid):
  cids[vertex] = cid
  for v из смежные(vertex)
    if cids[v] = 0
      dfs(graph, v, visited, cid)

components(graph):
  cids = [V раз 0]
  cid = 1
  for v из вершины(graph)
    if cids[v] = 0
      dfs(graph, v, cids, cid)
      cid += 1
```

Комментарий

Если ещё не посещали вершину, то запускаем из неё рекурсивный обход, после которого увеличиваем счётчик компонента связностей