**ЗАДАНИЕ ДЛЯ ВЫПОЛНЕНИЯ:**

**Задание 1.**  Ориентированный граф **G** взять в соответствии с вариантом. Осуществить алгоритмы поиска в ширину и глубину, а также алгоритма топологической сортировки аналогично примерам, рассмотренным на лекциях. Оформить отчет, включив в него **каждый** шаг выполнения алгоритмов.

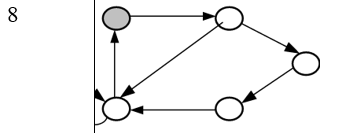
**Задание 2.** Осуществить программную реализацию алгоритмов на C++. Разработать структуры **AMatrix** и **АList** для представления ориентированного графа матричным и списковым способом. Разработать функции преобразования из одного способа представления в другой. Разработать функцию **BFS** обхода вершин графа, используя метод поиска в ширину. Продемонстрировать работу функции. Копии экрана вставить в отчет.

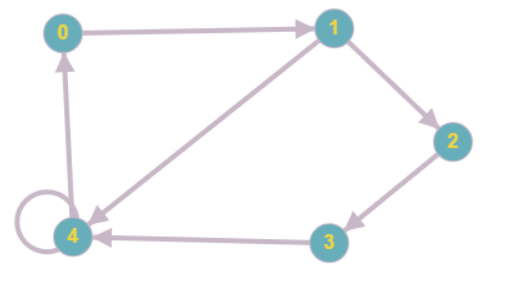
**Задание 3.**  Разработать функцию **DFS** обхода вершин графа, используя метод поиска глубину. Продемонстрировать работу функции. Копии экрана вставить в отчет.

**Задание 4.** Доработайте функцию **DFS**,для выполнения топологической сортировки графа. Продемонстрировать работу функции. Копии экрана вставить в отчет.

**Алгоритм поиска в ширину(BFS)**

Исходный граф:





Алгоритм подразумевает, что задана исходная (***стартовой***) вершина, и основывается на простом правиле: при выборе очередной вершины предпочтение отдается ближайшей.

При этом считается, что все дуги графа имеют единичную длину.

Сначала посещается стартовая вершина, затем все вершины, смежные ей (т. е. находящиеся на расстоянии 1), после чего вершины, находящиеся на расстоянии 2 от стартовой и т.д.

Текущее состояние алгоритма хранится в следующих структурах памяти: **Q** – очередь вершин, **С** – массив окраски вершин, **D** – массив расстояний и **P** – массив предшествующих вершин.

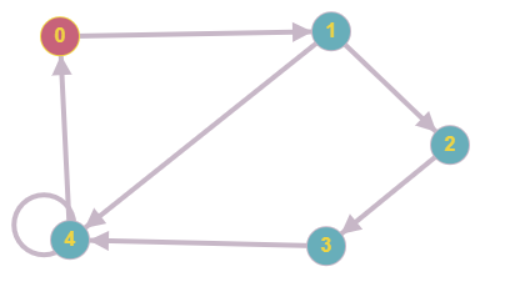
Очередь **Q** (структура памяти, реализующая алгоритм «первый вошел − первый вышел»), используется для промежуточного хранения номеров вершин. На каждом шаге алгоритма, в очередь помещаются номера вершин в порядке их обнаружения. На каждом шаге, кроме первого, из очереди извлекается очередной номер вершины, подлежащей отметке о посещении. На первом шаге алгоритма в очередь помещается номер стартовой вершины. На последнем шаге очередь пуста.

Массив **C** используется для хранения состояния вершин. С каждым из трех возможных состояний обычно связывают цвет: белый (**W**) – вершина не посещалась, серый (**G**) – вершина посещалась, черный (**B**) – фиксирован факт посещения вершины. На первом шаге алгоритма стартовая вершина окрашивается в серый цвет, а остальные – в белый. На последнем шаге все вершины становятся черными.

В массиве **D** для каждой вершины хранятся расстояния от стартовой вершины. На первом шаге для стартовой вершины в массиве **D** устанавливается значение 0, а для остальных вершин – значение «бесконечность» (**I**). На последнем шаге алгоритма для всех доступных вершин будут заполнены значения, равные их расстоянию от стартовой вершины.

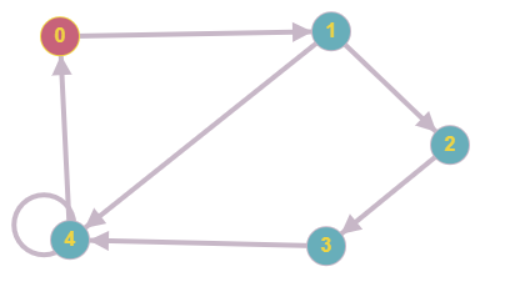
Массив **P** позволяет восстановить порядок обхода вершин и хранит для каждой вершины, кроме стартовой, предшествующую в обходе вершину. На первом шаге алгоритма всем элементам массива присваивается значение «пустота» (**N**). На последнем шаге алгоритма для всех доступных вершин будут заполнены значения, равные номеру предшествующей вершины в порядке обхода.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 0 |  |  |  |  |
| C | G | W | W | W | W |
| D | 0 | I | I | I | I |
| P | N | N | N | N | N |



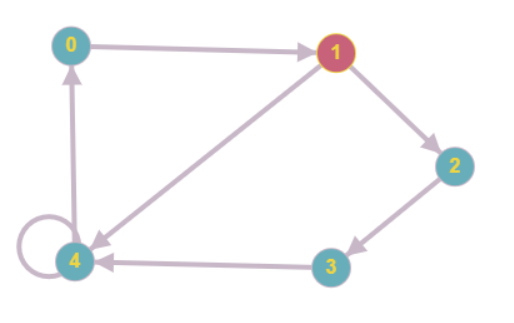
Шаг 1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 1 |  |  |  |  |
| C | B | G | W | W | W |
| D | 0 | 1 | I | I | I |
| P | N | 0 | N | N | N |



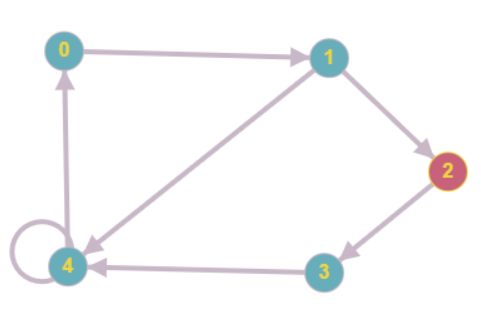
Шаг 2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Q | 2 | 4 |  |  |  |  |
| C | B | | B | G | W | G |
| D | 0 | | 1 | 2 | I | 2 |
| P | N | | 0 | 1 | N | 1 |



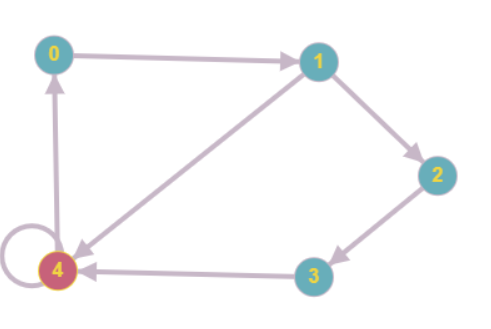
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 4 | 3 |  |  |  |
| C | B | | B | B | G | G |
| D | 0 | | 1 | 2 | 3 | 2 |
| P | N | | 0 | 1 | 2 | 1 |

Шаг 3



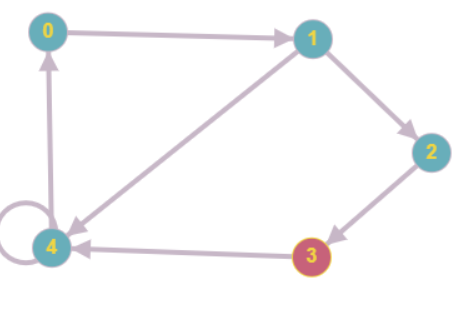
Шаг 4

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q | 3 |  |  |  |  |
| C | B | B | B | G | B |
| D | 0 | 1 | 2 | 3 | 2 |
| P | N | 0 | 1 | 2 | 1 |



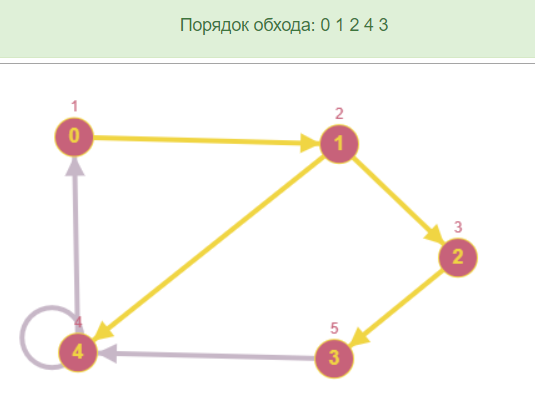
Шаг 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q | 3 |  |  |  |
| C | B | B | B | B | B |
| D | 0 | 1 | 2 | 3 | 2 |
| P | N | 0 | 1 | 2 | 1 |



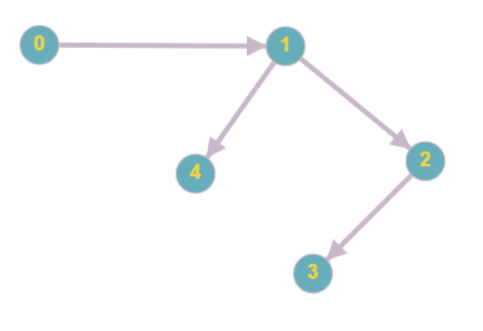
Шаг 6 Все вершины чёрного цвета, соответственно, алгоритм закончил свою работу.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Q |  |  |  |  |  |
| C | B | B | B | B | B |
| D | 0 | 1 | 2 | 4 | 3 |
| P | N | 0 | 1 | 2 | 1 |



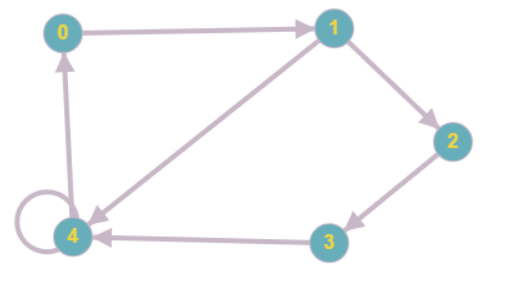
***BFS-дерево*** – это дерево, множество вершин которого является подмножеством вершин исходного графа, связанных дугами в порядке их посещения (в соответствии с массивом **P**), а корнем – стартовая вершина.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P | N | 0 | 1 | 2 | 1 |



**Алгоритм поиска в глубину (DFS)**

Исходный граф:

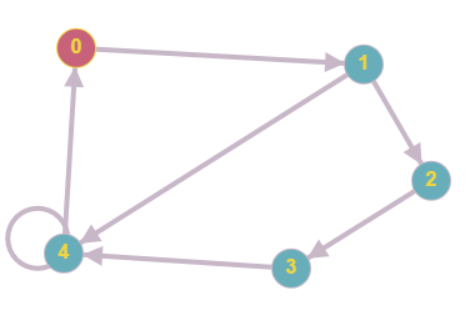


Как и для поиска в ширину, задается стартовая вершина. Алгоритм описывается следующим образом: для каждой не пройденной вершины, начиная со стартовой, необходимо найти все смежные вершины и повторить поиск для каждой.

Назначение и размерность массивов **С** (массив окраски вершин) и **P** (массив предшествующих вершин) такие же, как и в алгоритме BFS. В массиве **D** для каждой вершины записывается время обнаружения (шаг окраски в серый цвет). Массив **F** предназначен для хранения времени фиксации (шага окраски в черный цвет) вершины. Кроме того, используется переменная **t**, текущее значение которой – номер шага алгоритма.

Шаг 1

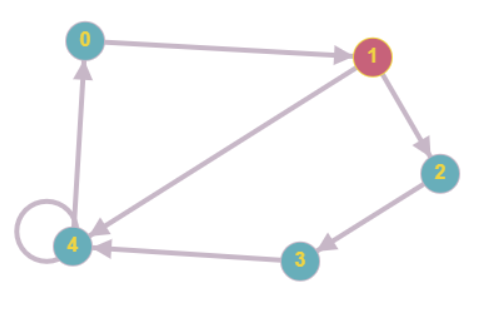
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | W | W | W | W |
| D | 1 | I | I | I | I |
| P | N | N | N | N | N |
| F | 0 | 0 | 0 | 0 | 0 |



t = 1 – стартовый шаг

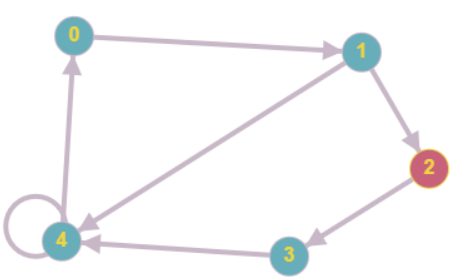
Шаг 2

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | W | W | W |
| D | 1 | 2 | I | I | I |
| P | N | 0 | N | N | N |
| F | 0 | 0 | 0 | 0 | 0 |

  
t = 2

Шаг 3

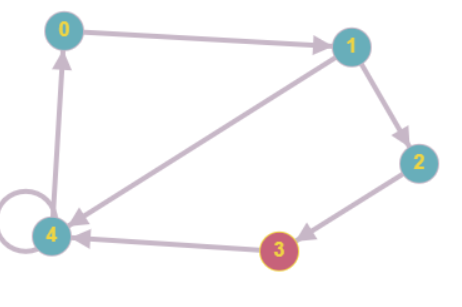
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | G | W | W |
| D | 1 | 2 | 3 | I | I |
| P | N | 0 | 1 | N | N |
| F | 0 | 0 | 0 | 0 | 0 |



t = 3

Шаг 4

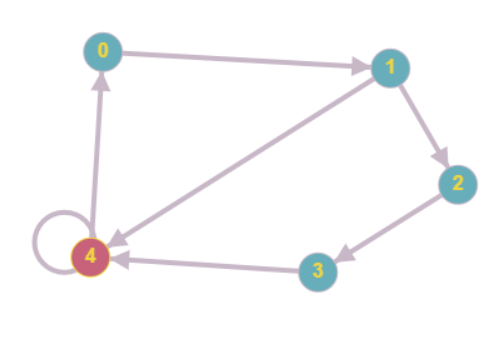
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | G | G | W |
| D | 1 | 2 | 3 | 4 | I |
| P | N | 0 | 1 | 2 | N |
| F | 0 | 0 | 0 | 0 | 0 |



t = 4

Шаг 5

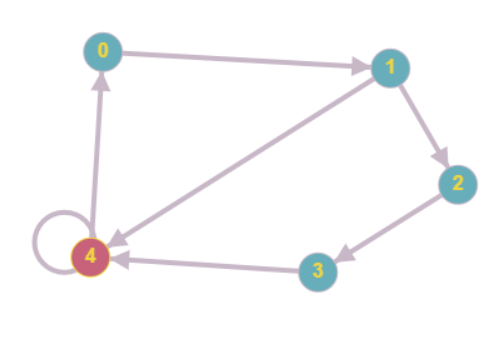
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | G | G | G |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 0 | 0 | 0 | 0 |



t = 5

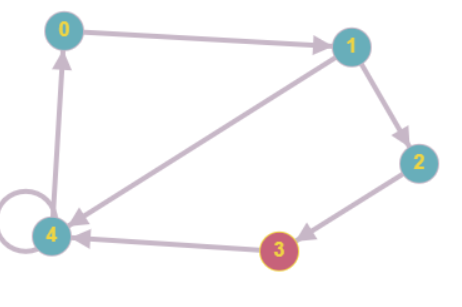
Шаг 6 Белых вершин, смежный с 4, нет, следовательно, мы погрузились в глубину. Закрашиваем 4 вершину в чёрный цвет.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | G | G | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 0 | 0 | 0 | 6 |

t = 6

Шаг 7. На предыдущем и последующих шагах массивы D и P – не изменяются;

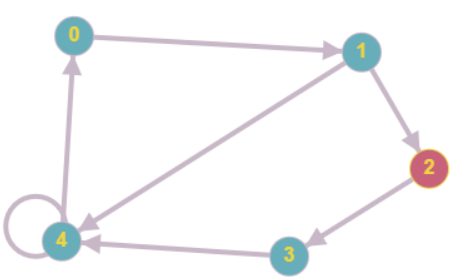
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | G | B | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 0 | 0 | 7 | 6 |



t = 7

Шаг 8

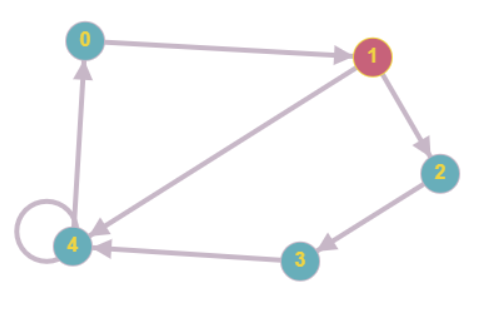
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | G | B | B | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 0 | 8 | 7 | 6 |



t = 8

Шаг 9

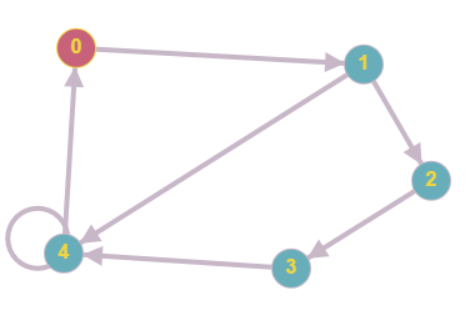
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | G | B | B | B | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 0 | 9 | 8 | 7 | 6 |



t = 9

Шаг 10 Последний шаг: окрашиваем 0 вершину в чёрный цвет, проверяем нет ли смежный ей вершин серого цвета. Таких нет. Значит алгоритм закончил свою работу.

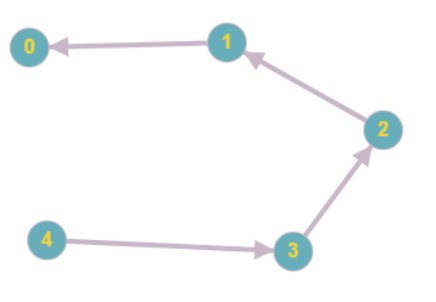
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| C | B | B | B | B | B |
| D | 1 | 2 | 3 | 4 | 5 |
| P | N | 0 | 1 | 2 | 3 |
| F | 10 | 9 | 8 | 7 | 6 |



t = 10

Ответ: DFS-дерево имеет вид:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P | N | 0 | 1 | 2 | 3 |



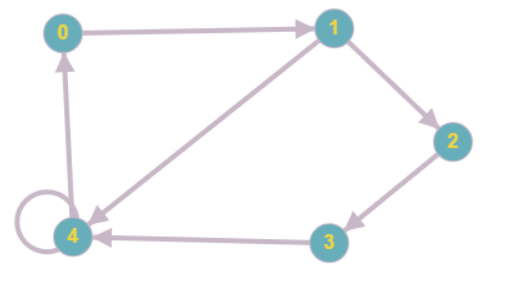
***Топологическая сортировка вершин графа***

***Топологическая сортировка*** − это процедура упорядочивания вершин ориентированного графа, не имеющего циклов (ациклического графа). В результате топологической сортировки для вершин графа определяется такой порядок, что если их расположить на рисунке в соответствии с этим порядком сверху вниз, то дуги будут направлены только от верхних вершин к нижним. Обычно после выполнения топологической сортировки вершины переименовываются (перенумеровываются) в соответствии с полученным порядком. После такого переименования граф обладает свойством: начальная вершина каждой дуги имеет номер (имя) меньший, чем номер конечной вершины этой дуги.

Наиболее известны два способа топологической сортировки графа: алгоритмы Демукрона и алгоритм, применяющий поиск в глубину.

Шаг 1

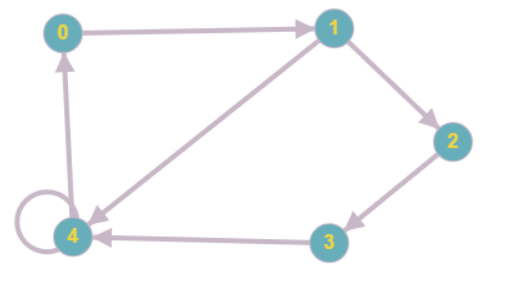
1/



Шаг 2

2/

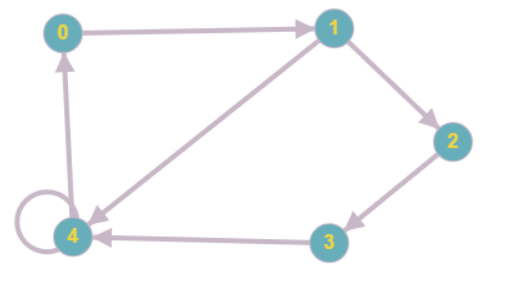
1/



Шаг 3

2/

1/



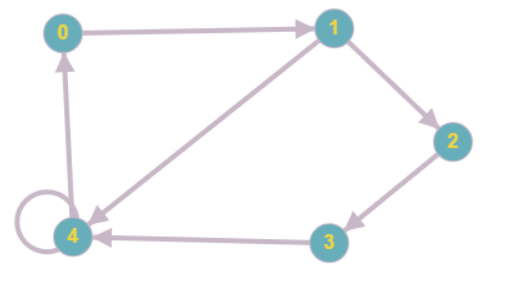
3/

0

Шаг 4

1/

2/



4/

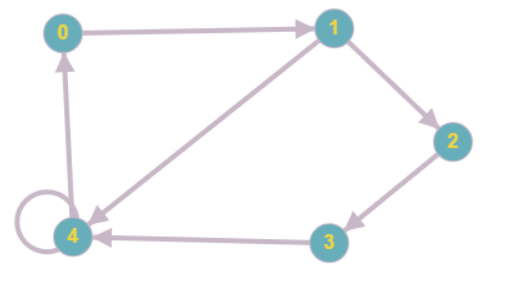
3/

Шаг 5

0

1/

2/



4/

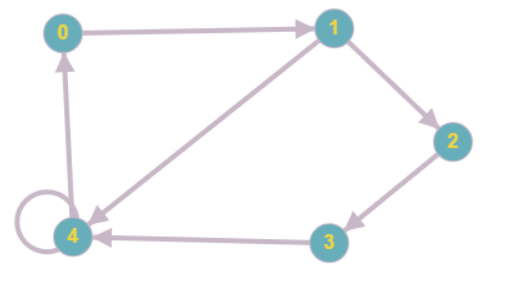
3/

5/

Шаг 6

1/

2/



4/

3/

5/6

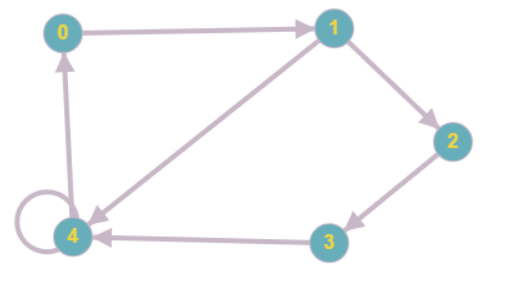
4

Шаг 7

2/

1/

|  |  |
| --- | --- |
| 4 | 3 |



4/7

3/

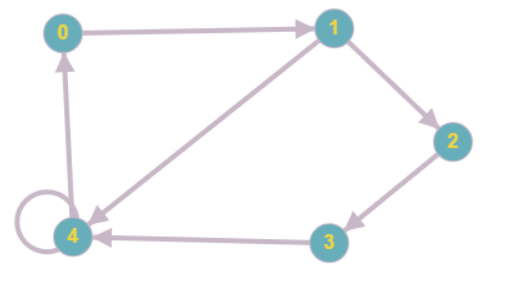
5/6

Шаг 8

1/

2/

|  |  |  |
| --- | --- | --- |
| 4 | 3 | 2 |



4/7

3/8

5/6

4/7

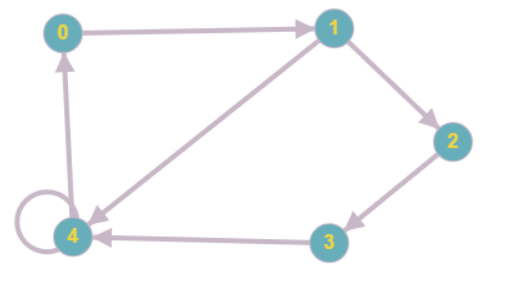
0

Шаг 9

1/

2/9

|  |  |  |  |
| --- | --- | --- | --- |
| 4 | 3 | 2 | 1 |



4/7

3/8

5/6

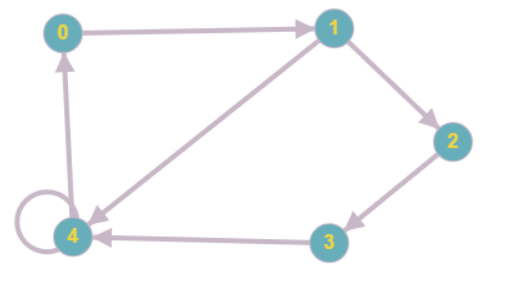
Шаг 10

0

1/10

2/9

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 4 | 3 | 2 | 1 | 0 |



4/7

3/8

5/6

Формируем очередь по мере окрашивания вершин в чёрный цвет: 4 3 2 1 0

