Учреждение образования

«Белорусский государственный технологический университет»

**Кафедра информационных систем и технологий**

**Лабораторная работа №5**

**Тема «**АЛГОРИТМЫ НА ГРАФАХ

(алгоритмы поиска в ширину и глубину, топологическая сортировка)**»**

Вариант 8

Выполнила:

Студентка 2 курса 7 группы ФИТ

Курносенко Софья Андреевна

Проверил:

Барковский Евгений Валерьевич

**ЦЕЛЬ РАБОТЫ:** Освоить сущность и программную реализацию: а) способов представления графов; б) алгоритмов поиска в ширину и глубину; в) алгоритма топологической сортировки графов.

**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.

|  |  |
| --- | --- |
| Вариант | Граф G |
| 8 |  |

**Поиск в ширину**

Алгоритм подразумевает, что задана исходная (*стартовая*) 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**

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.

**‣ C – массив окраски вершин**

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.

‣ D – массив расстояний

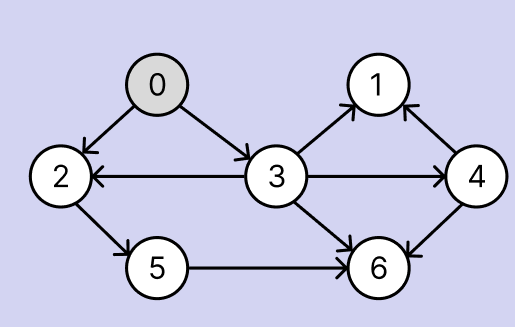
В массиве **D** для каждой вершины хранятся расстояния от стартовой вершины. На первом шаге для стартовой вершины в массиве **D** устанавливается значение 0, а для остальных вершин – значение «бесконечность» (∞). На последнем шаге алгоритма для всех доступных вершин будут заполнены значения, равные их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

‣ P – массив предшествующих вершин

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.

Шаг 1

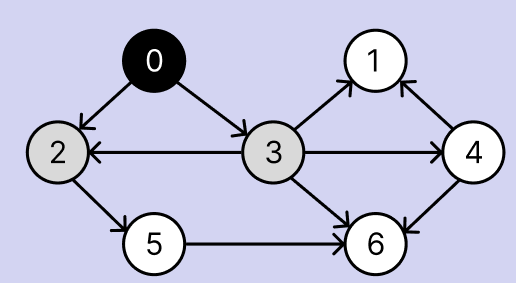
Выбираем вершину «0» стартовой.



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

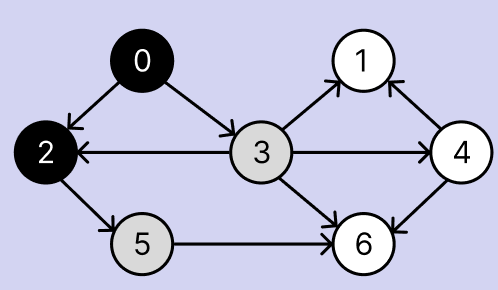
Шаг 2

Фиксируем посещение вершины 0. Идем в смежные ей вершины.



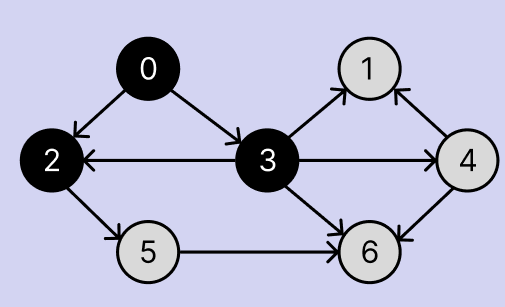
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Q | 2 | 3 |  |  |  |  |  |
| C | B | W | G | G | W | W | W |
| D | 0 | ∞ | 1 | 1 | ∞ | ∞ | ∞ |
| P | N | N | 0 | 0 | N | N | N |

Шаг 3



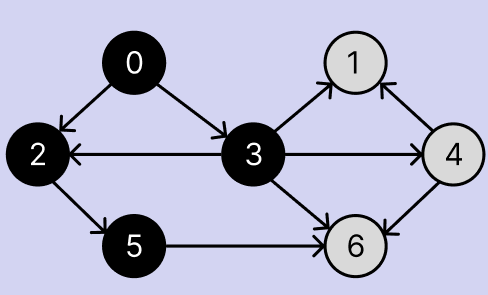
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Q | 3 | 5 |  |  |  |  |  |
| C | B | W | B | G | W | G | W |
| D | 0 | ∞ | 1 | 1 | ∞ | 2 | ∞ |
| P | N | N | 0 | 0 | N | 2 | N |

Шаг 4



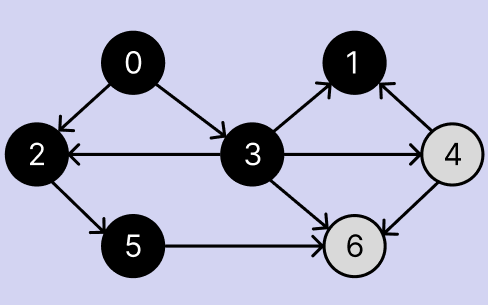
|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Q | 1 | 4 | 5 | 6 |  |  |  |
| C | B | G | B | B | G | G | G |
| D | 0 | 2 | 1 | 1 | 2 | 2 | 2 |
| P | N | 3 | 0 | 0 | 3 | 2 | 3 |

Шаг 5



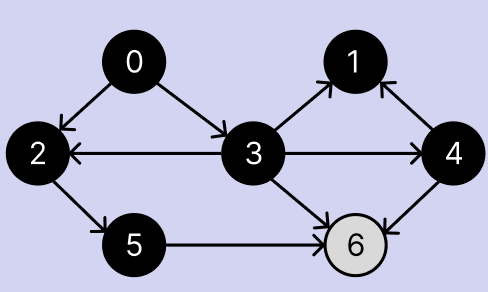
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Q | 1 | 4 | 6 |  |  |  |
| C | B | G | B | B | G | B | G |
| D | 0 | 2 | 1 | 1 | 2 | 2 | 2 |
| P | N | 3 | 0 | 0 | 3 | 2 | 3 |

Шаг 6



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

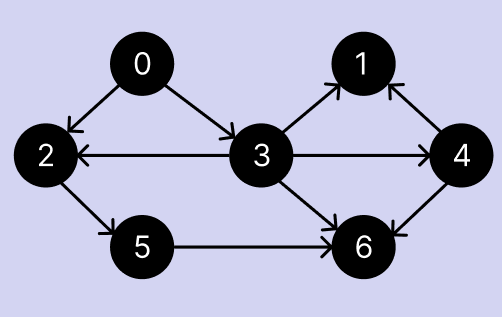
Шаг 7



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Q | 6 |  |  |  |
| C | B | B | B | B | B | B | G |
| D | 0 | 2 | 1 | 1 | 2 | 2 | 2 |
| P | N | 3 | 0 | 0 | 3 | 2 | 3 |

Шаг 8

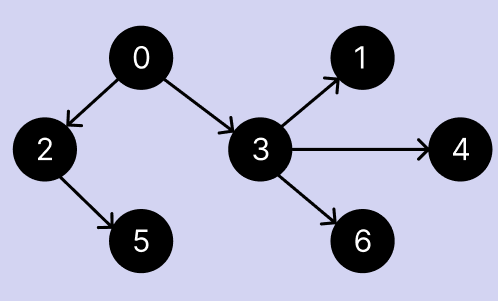
Все вершины чёрного цвета, соответственно, алгоритм закончил свою работу.



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

Ответ: В результате получили вот такое BFS-дерево:

Удаляем ни разу не посещенные вершины:



**Поиск в глубину**

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.

Текущее состояние алгоритма хранится в следующих структурах памяти:

‣ C – массив окраски вершин

‣ D – время окраски вершин в серый цвет

В массиве **D** для каждой вершины записывается время обнаружения (шаг окраски в серый цвет).

‣ P – массив предшествующих вершин

‣ F – время окраски в чёрный цвет

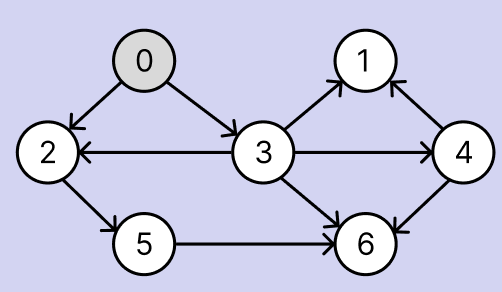
Массив **F** предназначен для хранения времени фиксации (шага окраски в черный цвет) вершины.

‣ t – номер шага алгоритма

Кроме того, используется переменная **t**, текущее значение которой – номер шага алгоритма.

Шаг 1

Стартовой выбираем вершину 0 и окрашиваем её в серый.

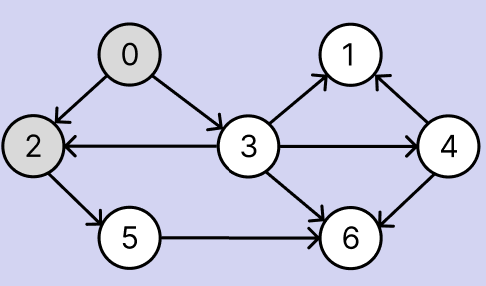


|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| C | G | W | W | W | W | W | W |
| D | 1 | - | - | - | - | - | - |
| P | N | N | N | N | N | N | N |
| F | - | - | - | - | - | - | - |

t = 1 – стартовый шаг

Шаг 2

Смежными к вершине 0 являются вершины 2 и 3. Выбираем одну из них, например 2, и посещаем её.

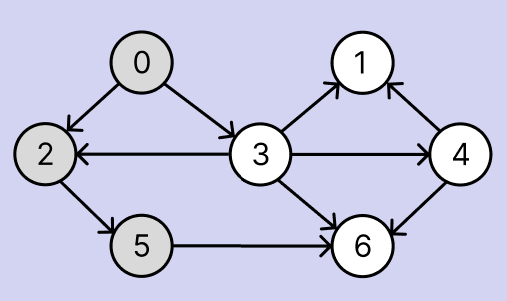


|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| C | G | W | G | W | W | W | W |
| D | 1 | - | 2 | - | - | - | - |
| P | N | N | 0 | N | N | N | N |
| F | - | - | - | - | - | - | - |

t = 2

Шаг 3

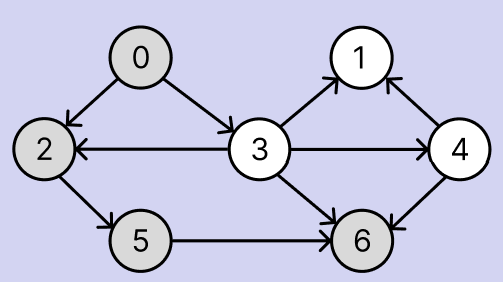
Продолжаем опускаться в глубину. Выбираем вершину 5, как не посещенную смежную вершину к той, на которой остановились на прошлом шаге.



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

t = 3

Шаг 4

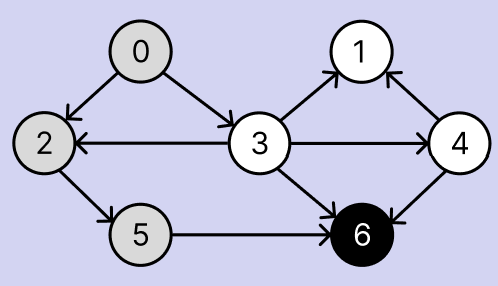


|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| C | G | W | G | W | W | G | G |
| D | 1 | - | 2 | - | - | 3 | 4 |
| P | N | N | 0 | N | N | 2 | 5 |
| F | - | - | - | - | - | - | - |

t = 4

Шаг 5

У вершины 6 нет смежных не посещенных вершин – значит мы дошли до самой последней вершины выбранной ветки и можем зафиксировать этот шаг.

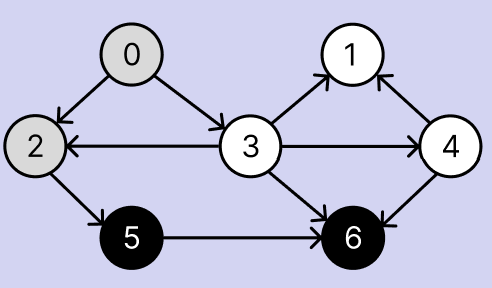


|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| C | G | W | G | W | W | G | B |
| D | 1 | - | 2 | - | - | 3 | 4 |
| P | N | N | 0 | N | N | 2 | 5 |
| F | - | - | - | - | - | - | 5 |

t = 5

Шаг 6

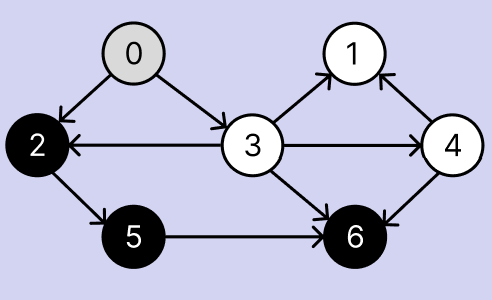
Возвращаемся к предыдущей вершине 5, у неё нет смежных белых вершин – значит, окрашиваем её в чёрный.



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

t = 6

Шаг 7

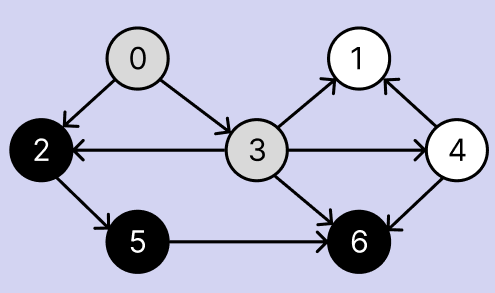


|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| C | G | W | B | W | W | B | B |
| D | 1 | - | 2 | - | - | 3 | 4 |
| P | N | N | 0 | N | N | 2 | 5 |
| F | - | - | 7 | - | - | 6 | 5 |

t = 7

Шаг 8

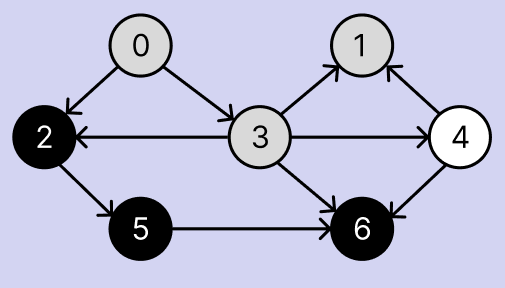
Возвращаемся в вершину 0, находим, что у неё есть смежная не посещенная (белая) вершина 3. Посещаем её.



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

t = 8

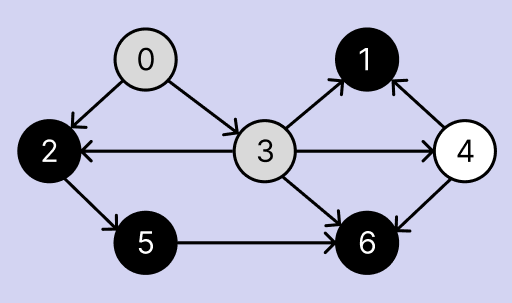
Шаг 9



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

t = 9

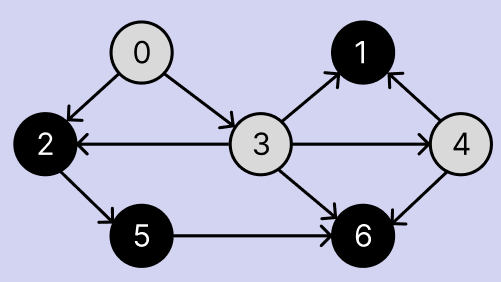
Шаг 10



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

t = 10

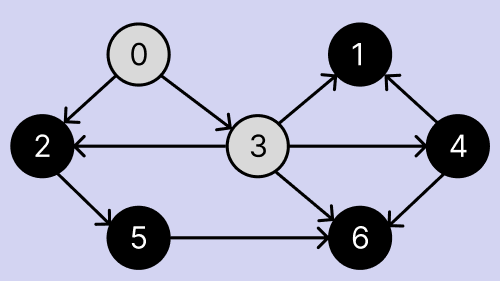
Шаг 11



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| C | G | B | B | G | G | B | B |
| D | 1 | 9 | 2 | 8 | 11 | 3 | 4 |
| P | N | 3 | 0 | 0 | 3 | 2 | 5 |
| F | - | 10 | 7 | - | - | 6 | 5 |

t = 11

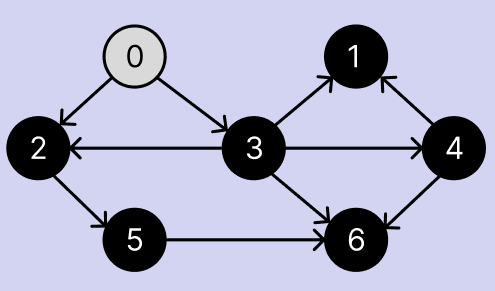
Шаг 12



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| C | G | B | B | G | G | B | B |
| D | 1 | 9 | 2 | 8 | 11 | 3 | 4 |
| P | N | 3 | 0 | 0 | 3 | 2 | 5 |
| F | - | 10 | 7 | - | 12 | 6 | 5 |

t = 12

Шаг 13

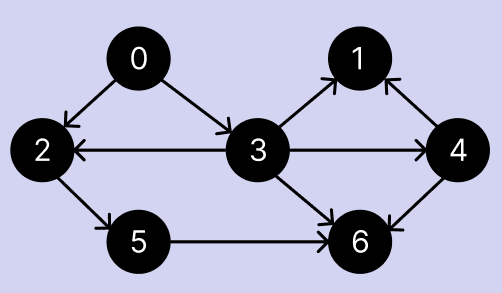


|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| C | G | B | B | G | G | B | B |
| D | 1 | 9 | 2 | 8 | 11 | 3 | 4 |
| P | N | 3 | 0 | 0 | 3 | 2 | 5 |
| F | - | 10 | 7 | 13 | 12 | 6 | 5 |

t = 13

Шаг 14

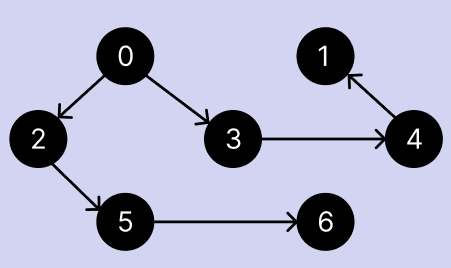
У вершины 0 нет смежных вершин, поэтому окрашиваем её в черный и на этом завершаем алгоритм.



|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| C | G | B | B | G | G | B | B |
| D | 1 | 9 | 2 | 8 | 11 | 3 | 4 |
| P | N | 3 | 0 | 0 | 3 | 2 | 5 |
| F | 14 | 10 | 7 | 13 | 12 | 6 | 5 |

t = 14

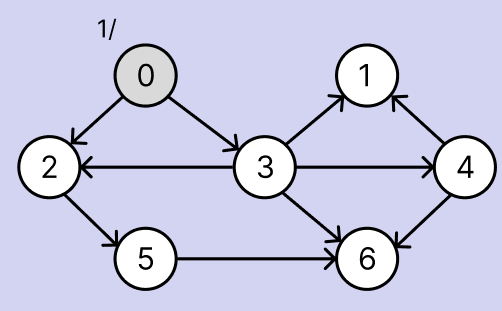
Ответ: DFS-дерево имеет вид:



**Топологическая сортировка**

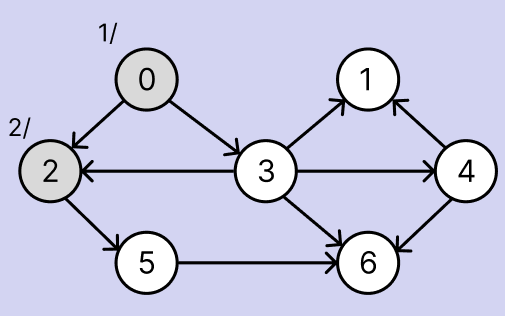
**Топологическая сортировка** − это процедура упорядочивания вершин ориентированного графа, не имеющего циклов (ациклического графа). В результате топологической сортировки для вершин графа определяется такой порядок, что если их расположить на рисунке в соответствии с этим порядком сверху вниз, то дуги 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.

Шаг 1



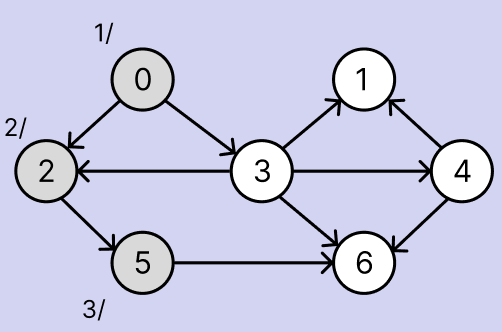
|  |
| --- |
|  |

Шаг 2



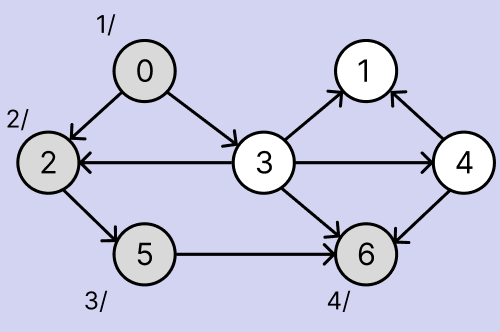
|  |
| --- |
|  |

Шаг 3



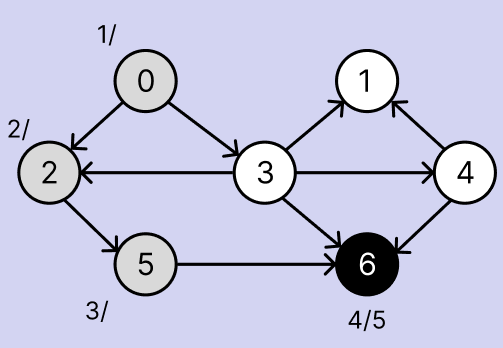
|  |
| --- |
|  |

Шаг 4



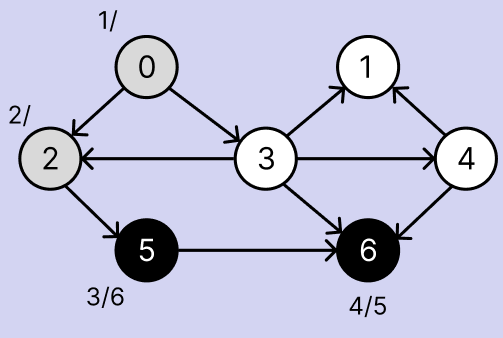
|  |
| --- |
|  |

Шаг 5



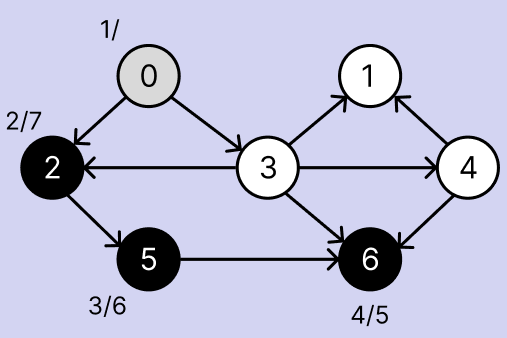
|  |
| --- |
| 6 |

Шаг 6



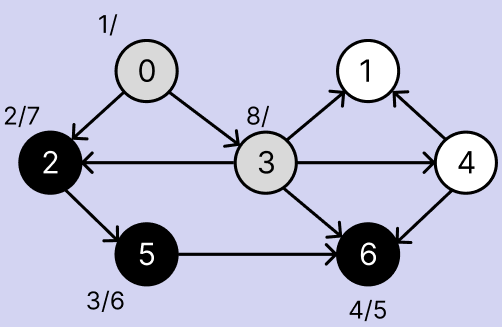
|  |
| --- |
| 6, 5 |

Шаг 7



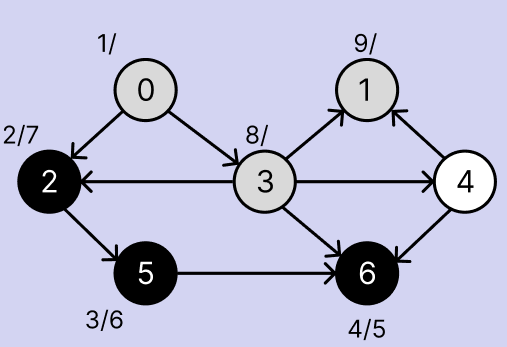
|  |
| --- |
| 6, 5, 2 |

Шаг 8



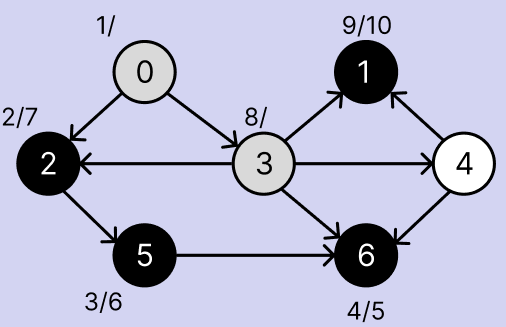
|  |
| --- |
| 6, 5, 2 |

Шаг 9



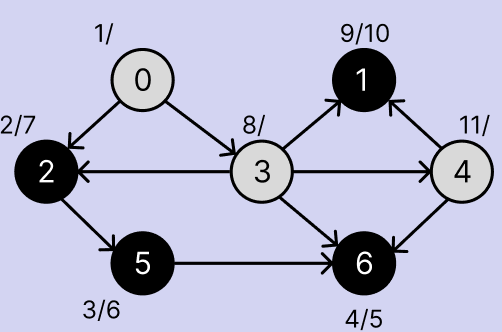
|  |
| --- |
| 6, 5, 2 |

Шаг 10



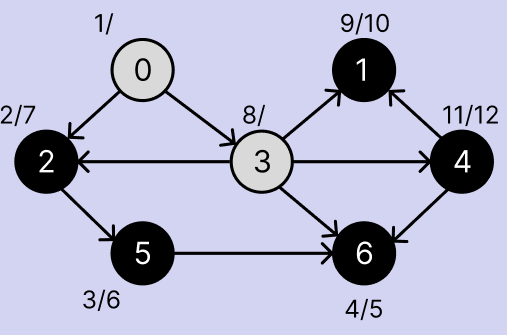
|  |
| --- |
| 6, 5, 2, 1 |

Шаг 11



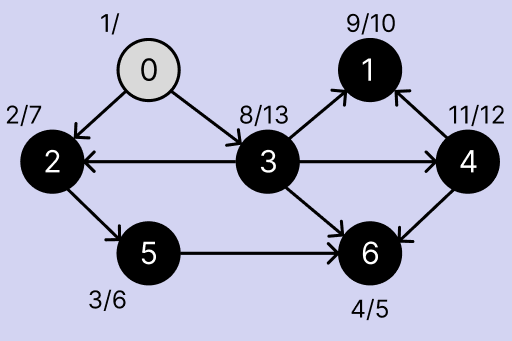
|  |
| --- |
| 6, 5, 2, 1 |

Шаг 12



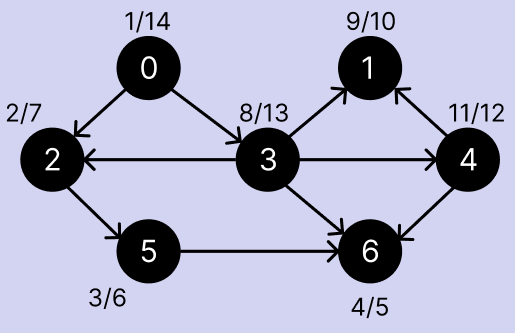
|  |
| --- |
| 6, 5, 2, 1, 4 |

Шаг 13



|  |
| --- |
| 6, 5, 2, 1, 4, 3 |

Шаг 14

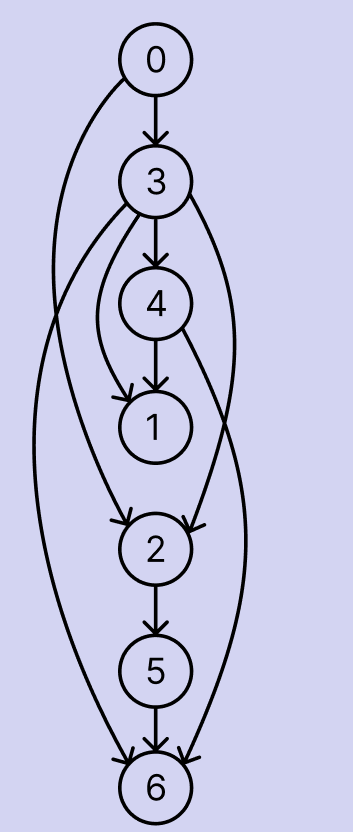


|  |
| --- |
| 6, 5, 2, 1, 4, 3, 0 |

Полученный список нужно развернуть в противоположном направлении:

|  |
| --- |
| 0, 3, 4, 1, 2, 5, 6 |

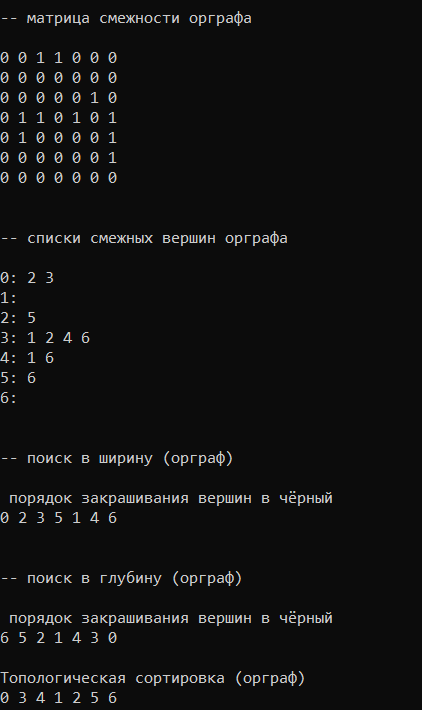
Теперь берем наш граф и выстраиваем его вершины в порядке обратном закрашиванию в черный, добавляем дуги:



**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.



**Вывод:** Я освоила сущность и программную реализацию способов представления графов; алгоритмов поиска в ширину и глубину; алгоритма топологической сортировки графов.