**Лабораторная работа 6. АЛГОРИТМЫ НА ГРАФАХ**

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

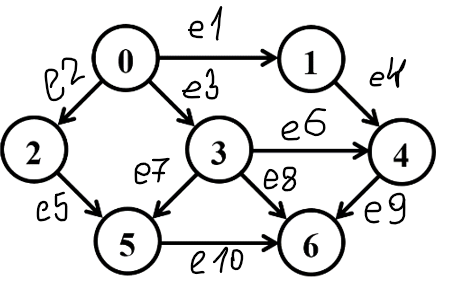
|  |  |
| --- | --- |
| 11 |  |

**ЗАДАНИЕ ДЛЯ ВЫПОЛНЕНИЯ:**

***Задание 1.*** Ориентированный граф **G** взять в соответствии с вариантом. Представить его в отчете в виде матрицы смежности, матрицы инцидентности, списка смежных вершин.

0 1 2 3 4 5 6 – матрица смежности

0 0 1 1 1 0 0 0

1 0 0 0 0 1 0 0

2 0 0 0 1 0 0 0

3 0 0 0 0 1 1 1

4 0 0 0 0 0 0 0

5 0 0 0 0 0 0 0

6 0 0 0 0 0 0 0

Матрица инцидентности:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Вершина | |  | | --- | | **e1**  **(0→1)** |  |  | | --- | |  | | |  | | --- | | **e2**  **(0→2)** |  |  | | --- | |  | | |  | | --- | | **e3 (0→3)** |  |  | | --- | |  | | |  | | --- | | **e4 (1→4)** |  |  | | --- | |  | | |  | | --- | | **e5 (2→5)** |  |  | | --- | |  | | |  | | --- | | **e6 (3→4)** |  |  | | --- | |  | | |  | | --- | | **e7 (3→5)** |  |  | | --- | |  | | | **e8 (3→6)** | | --- |  |  | | --- | |  | | **e9 (4→6)** | **e10 (5→6)** |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | -1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | -1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | -1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 4 | 0 | 0 | 0 | -1 | 0 | -1 | 0 | 0 | 1 | 0 |
| 5 | 0 | 0 | 0 | 0 | -1 | 0 | -1 | 0 | 0 | 1 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | -1 | -1 |

Cписки смежных вершин:

0: 1 2 3

1: 4

2: 5

3: 4 5 6

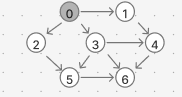
4: 6

5: 6

6:

***Задание 2.*** 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.

**Поиск в ширину**

Выбираем стартовую вершину 0. Добавляем ее в очередь.

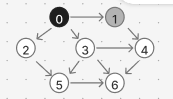
**Очередь: [0]**

**Посещенная очередь: []**

Посещаем вершину 0 и записываем в стек новые вершины:1,2.

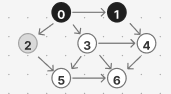
**Очередь: [1,2,3]**

**Посещенная очередь: [0]**

Посещаем вершину 1 и записываем в стек новые вершины:4.

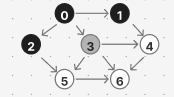
**Очередь: [2,3,4]**

**Посещенная очередь: [0,1]**

****Посещаем вершину 2 и записываем в стек новые вершины:5.

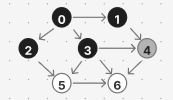
**Очередь: [3,4,5]**

**Посещенная очередь: [0,1,2]**

Посещаем вершину 3 и записываем в стек новые вершины:6.

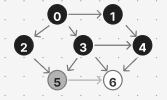
**Очередь: [4,5,6]**

**Посещенная очередь: [0,1,2,3]**

Посещаем вершину 4 и записываем в стек новые вершины:ничего.

**Очередь: [5,6]**

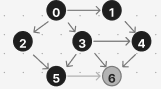
**Посещенная очередь: [0,1,2,3,4]**



Посещаем вершину 5 и записываем в стек новые вершины:ничего.

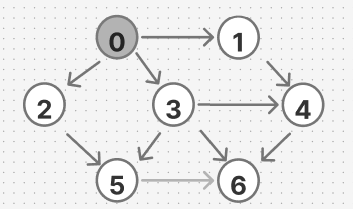
**Очередь: [6]**

**Посещенная очередь: [0,1,2,3,4,5]**

Посещаем вершину 6 и записываем в стек новые вершины:ничего.

**Очередь: []**

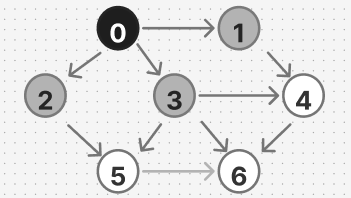
**Посещенная очередь: [0,1,2,3,4,5,6]**

**Поиск в глубину**

Выбираем стартовую вершину 0

**Стек: [0]**

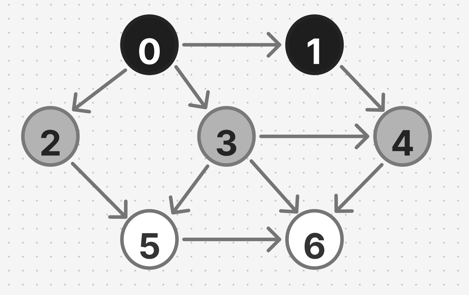
**Посещенные вершины: []**



Посещаем 0. Добавляем в стек:1,2,3

**Стек: [1,2,3]**

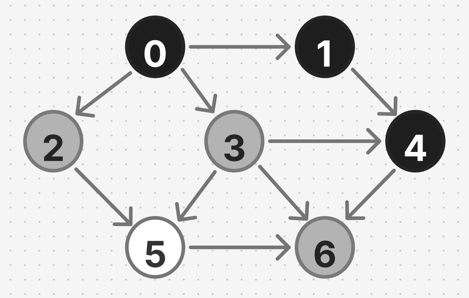
**Посещенные вершины: [0]**



Посещаем 1. Добавляем в стек: 4.

**Стек: [4,2,3]**

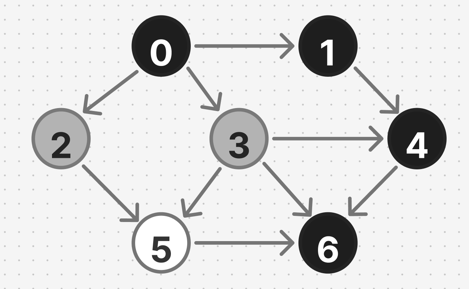
**Посещенные вершины: [0,1]**

****

Посещаем 4. Добавляем в cтек: 6.

**Стек: [6,2,3]**

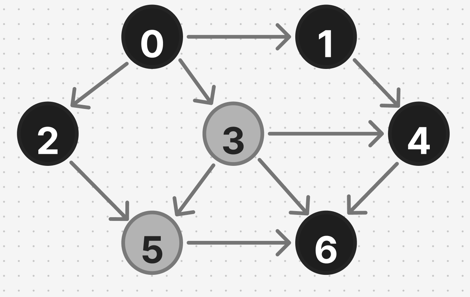
**Посещенные вершины: [0,1,4]**

****

Посещаем 6.

**Стек: [2,3]**

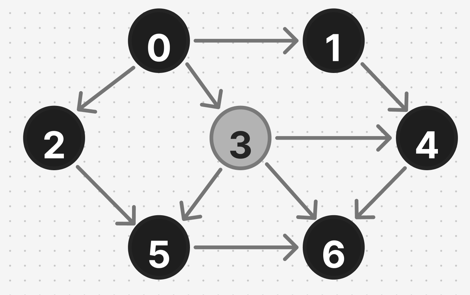
**Посещенные вершины: [0,1,4,6]**

****

Посещаем 2. Добавляем в стек: 5.

**Стек: [5,3]**

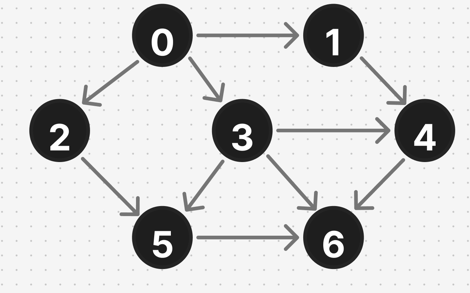
**Посещенные вершины: [0,1,4,6,2]**

****

Посещаем 5.

**Стек: [3]**

**Посещенные вершины: [0,1,4,6,2,5]**

****

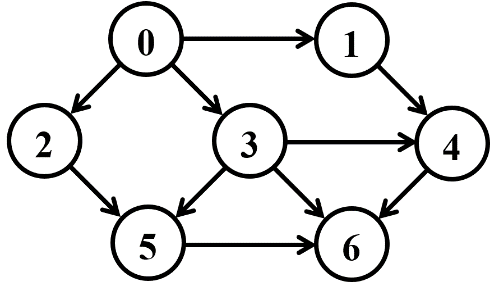
Посещаем 3. Добавляем в стек: ничего.

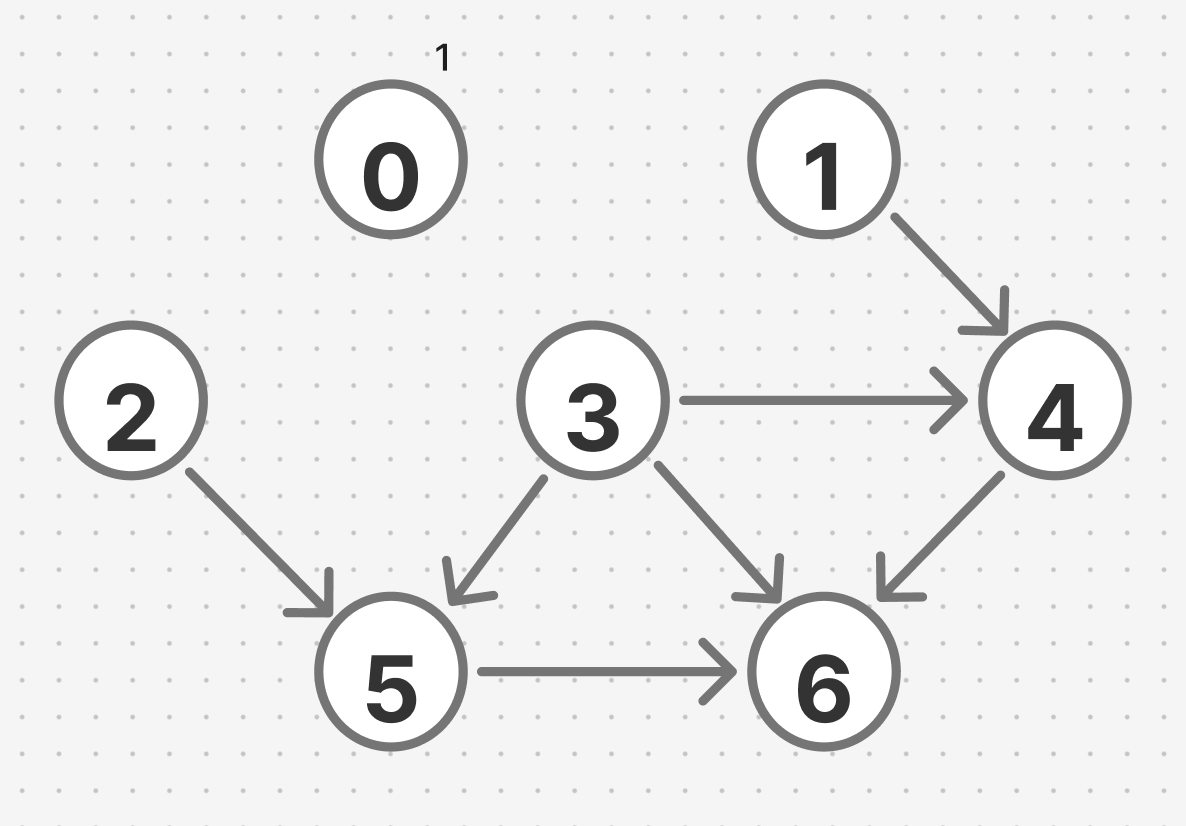
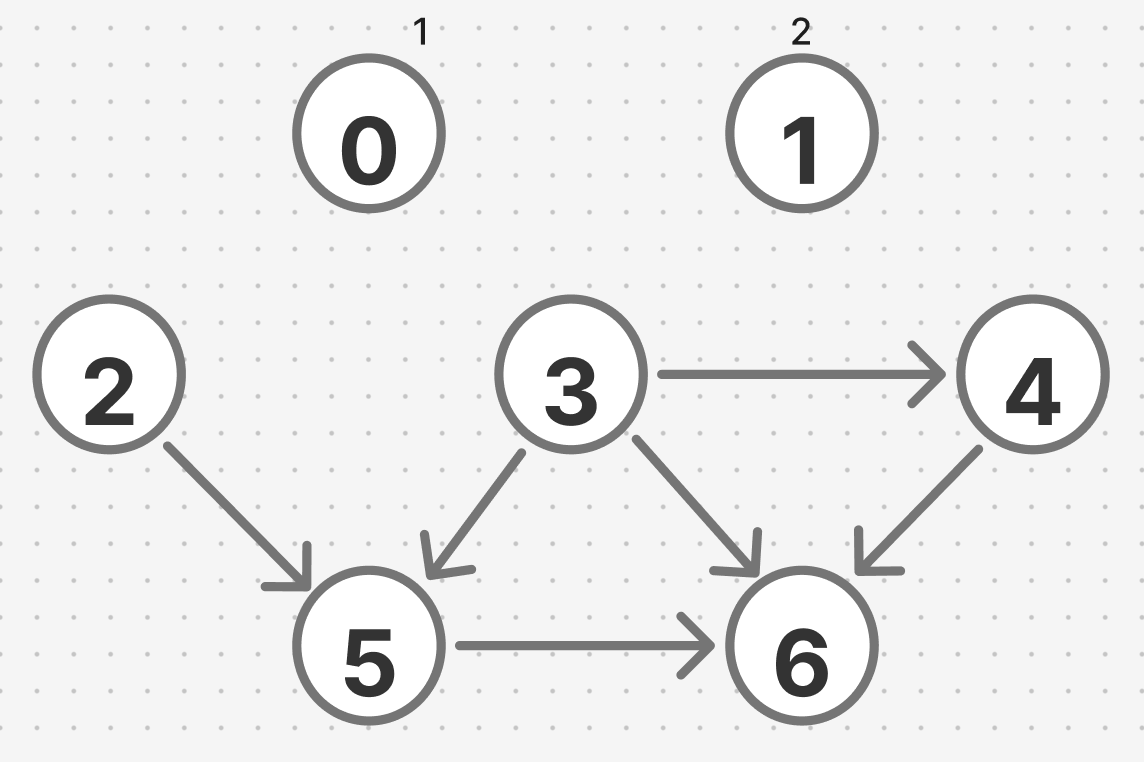
**Стек: []**

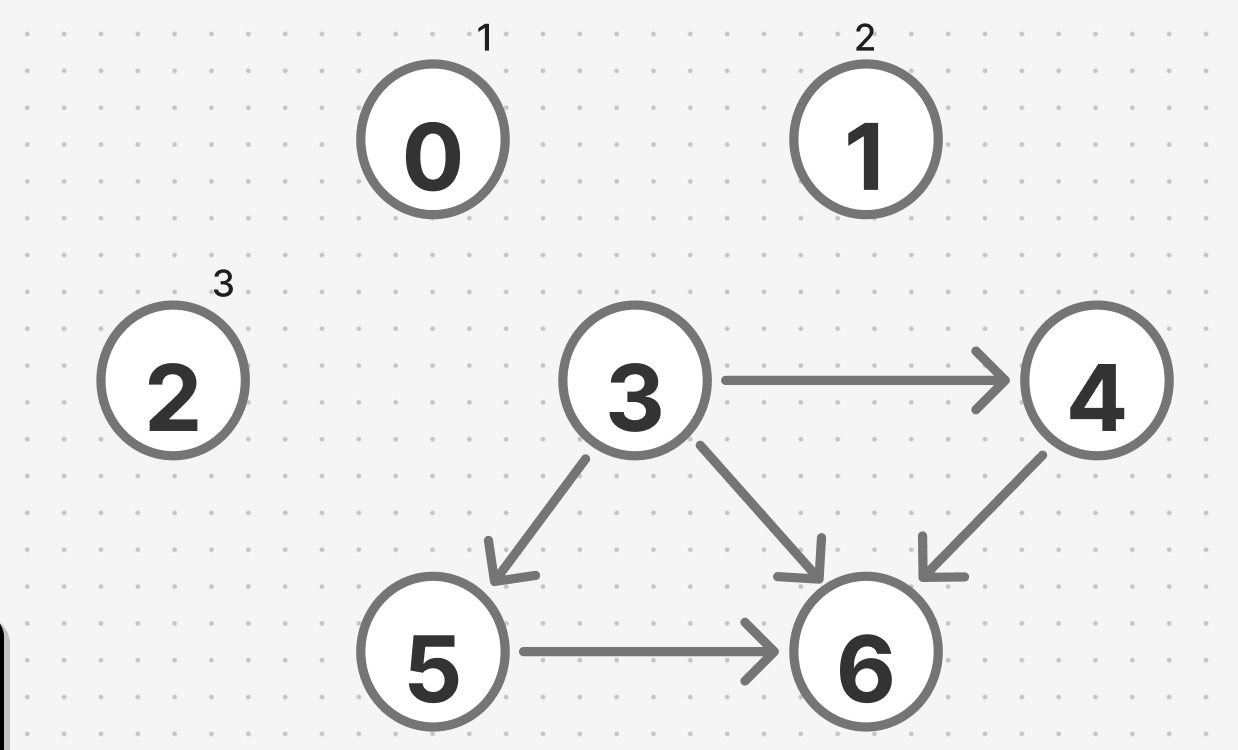
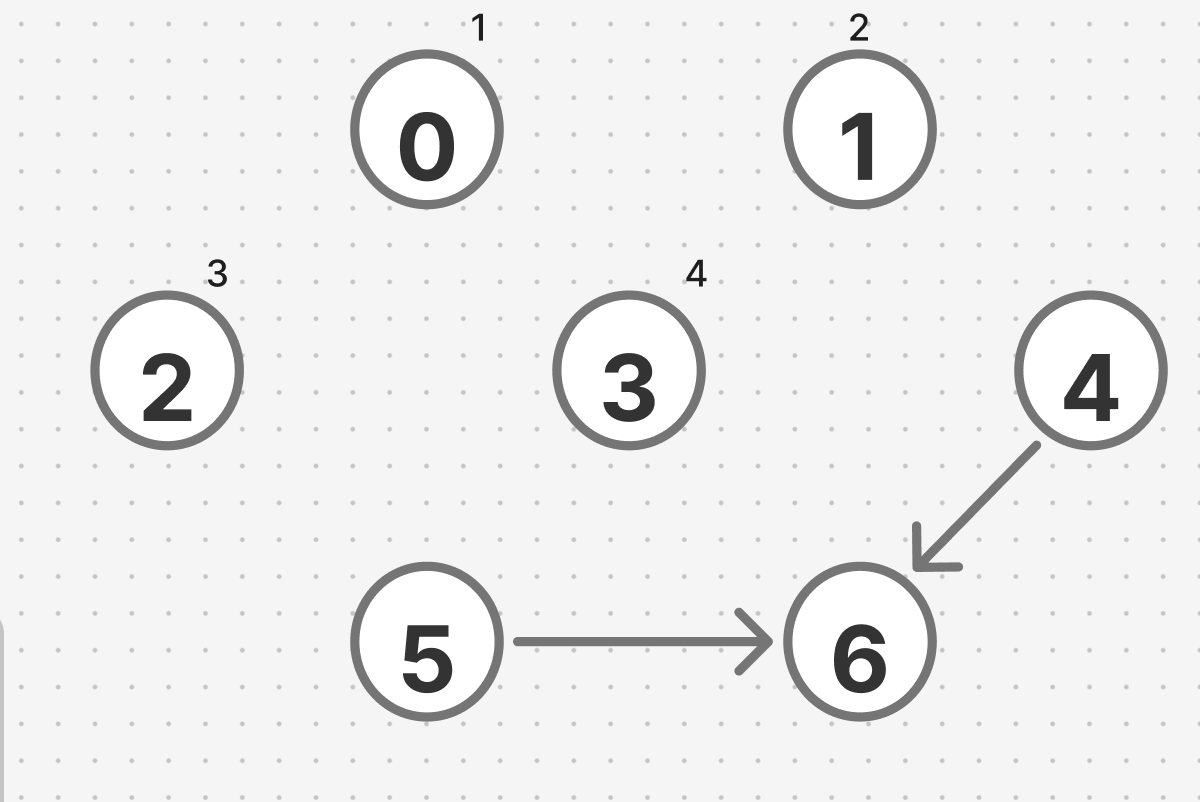
**Посещенные вершины: [0,1,4,6,2,5,3]**

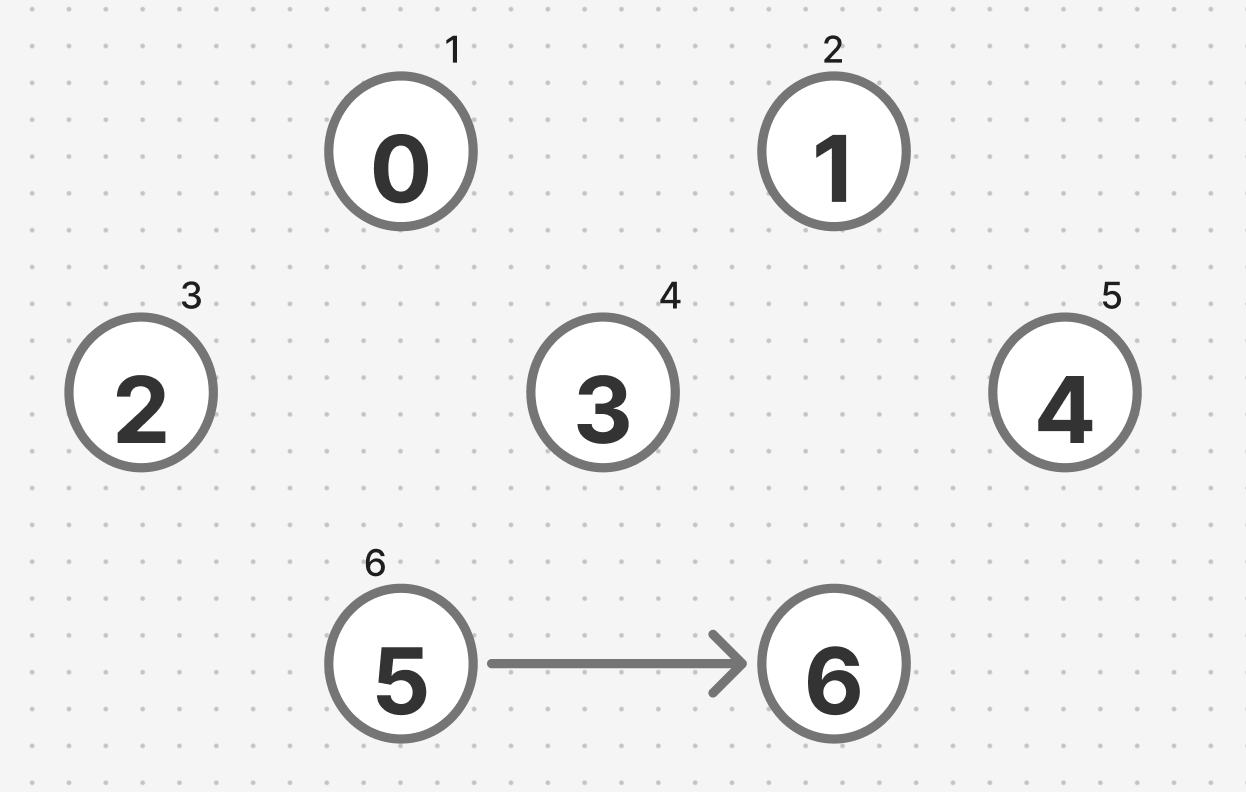
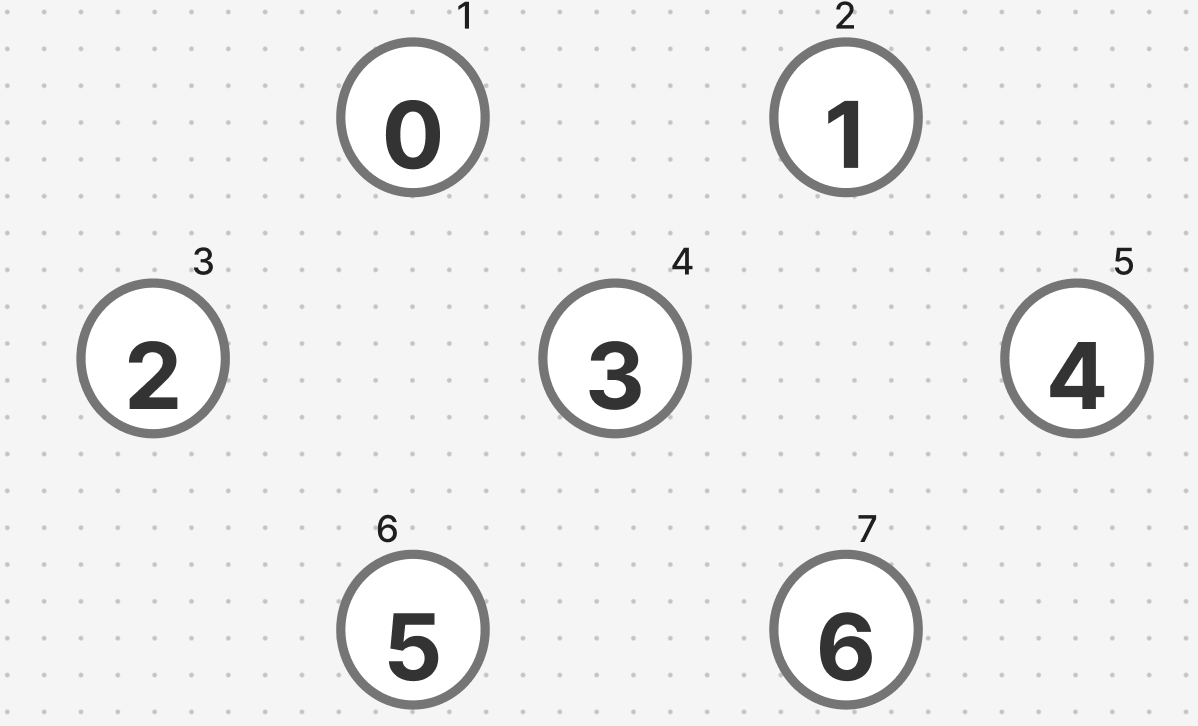
**Алгоритм топологической сортировки.**

1. Ищем вершину, в которую не входят дуги и нумеруем ее.
2. Помечаем дуги, выходящие из этой вершины как несуществующие.
3. Повторяем шаги (1) и (2), пока не будут занумерованы все вершины.



Результат: 0,1,2,3,4,5,6

Или 0,2,1,3,5,4,6

Или 0,3,1,2,4,5,6

Или 0,3,2,5,1,4,6

и тд

***Задание 3.*** 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 4.*** Разработать функцию **DFS**  обхода вершин графа, используя метод поиска глубину. Продемонстрировать работу функции. Копии экрана вставить в отчет.



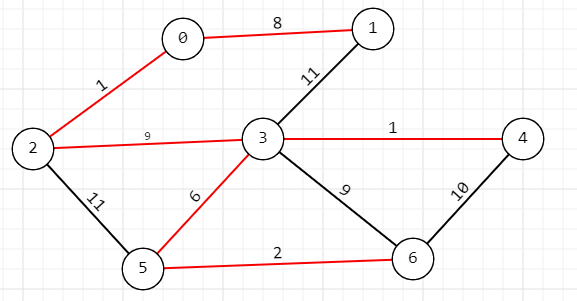
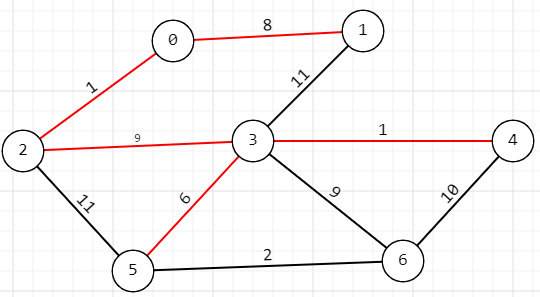
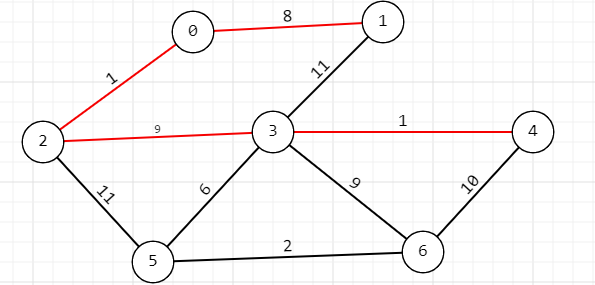
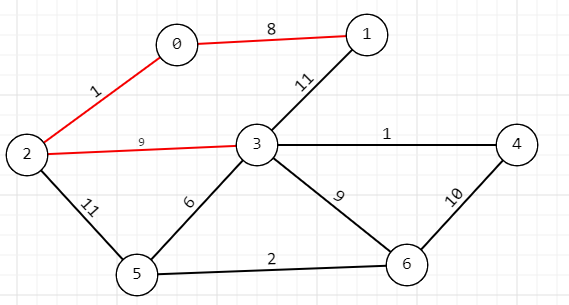
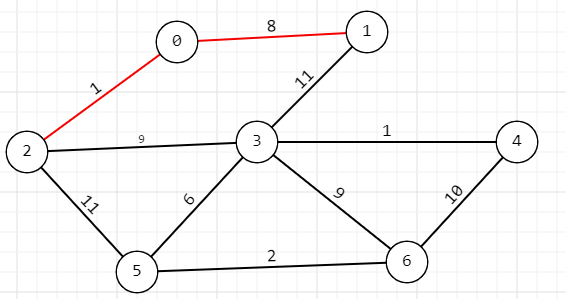
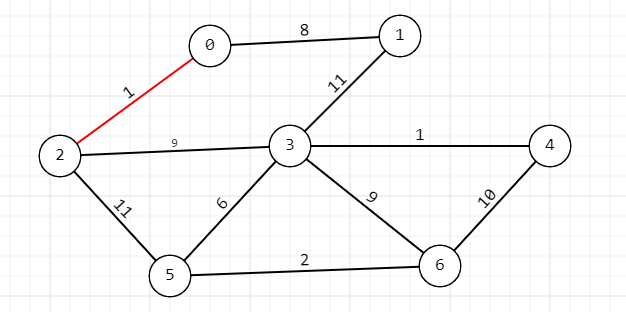
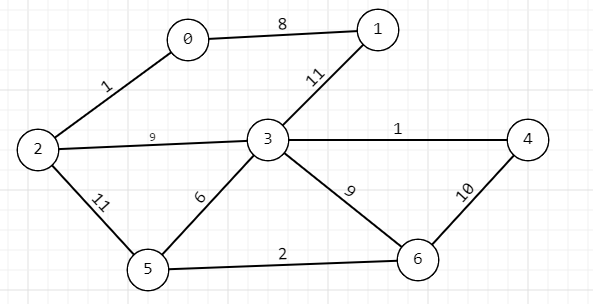
***Задание 5.*** Доработайте функцию **DFS**,для выполнения топологической сортировки графа. Продемонстрировать работу функции. Копии экрана вставить в отчет.



***Задание 6.*** По графу, соответствующему варианту составить минимальное остовное дерево по алгоритму Прима. Шаги построения отразить в отчете.

**Остовное связное дерево** – подграф, включающий вершины исходного графа G, не содержащего циклы, каждая вершина которого достижима из любой другой.

* **Начало алгоритма: с произвольной вершины**
* **К текущему дереву присоединяется смежная вершина с кратчайшим ребром.**
* **Окончание алгоритма: либо все вершины подключены, либо невозможно подключить ни одно ребро.**

Наинаем с 0

Вес будет равен 28

***Задание 7.*** По графу, соответствующему варианту составить минимальное остовное дерево по алгоритму Крускала. Шаги построения отразить в отчете.

1. Удалить все ребра и получить остовной подграф с изолированными вершинами.
2. Отсортировать ребра по возрастанию.
3. Ребра последовательно, по возрастанию их весов, включаются в остовное дерево. Возможны случаи:

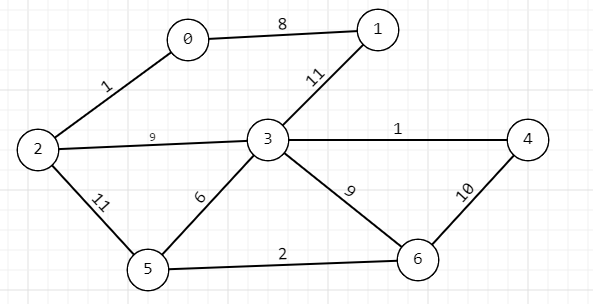
а) обе вершины включаемого ребра принадлежат одноэлементным подмножествам, тогда они объединяются в новое, связное подмножество;

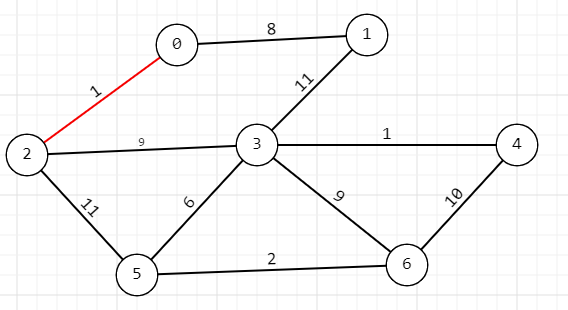
б) одна из вершин принадлежит связному под-множеству, другая нет, тогда включаем вторую в подмножество, которому принадлежит первая;

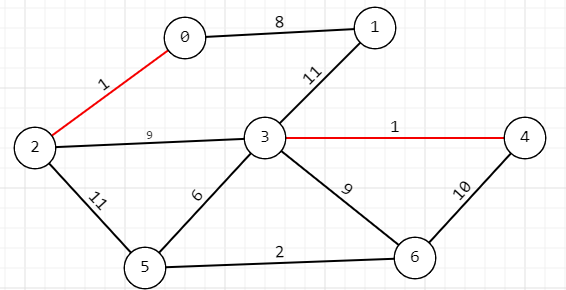
в) обе вершины принадлежат разным связным подмножествам, тогда объединяем подмножества;

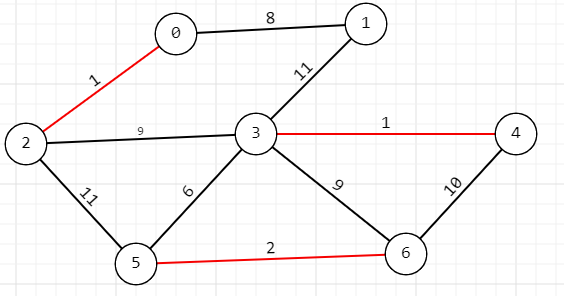
г) обе вершины принадлежат одному связному подмножеству, тогда исключаем данное ребро.

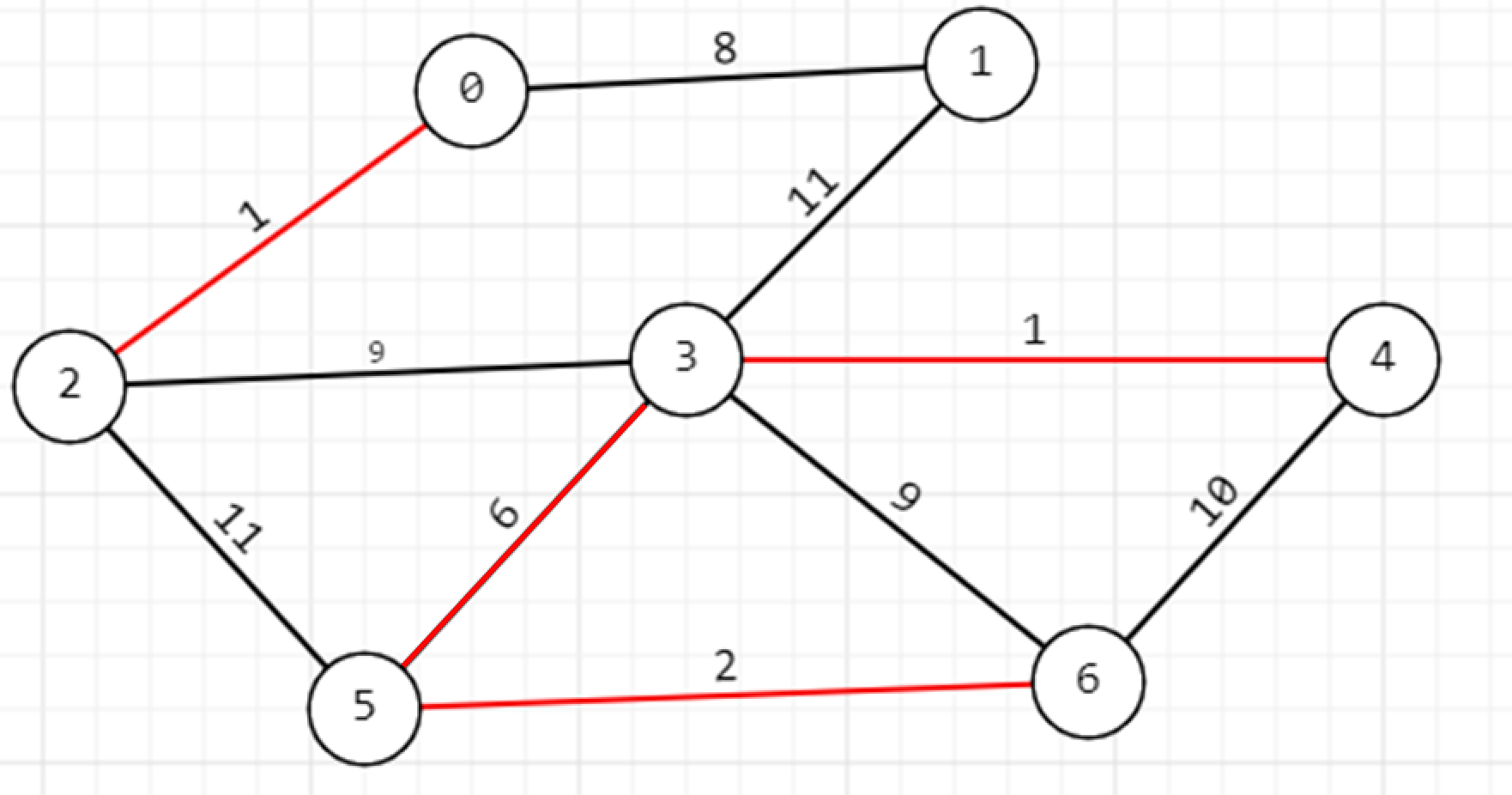
1. Алгоритм завершается, когда все вершины будут объединены в одно множество.

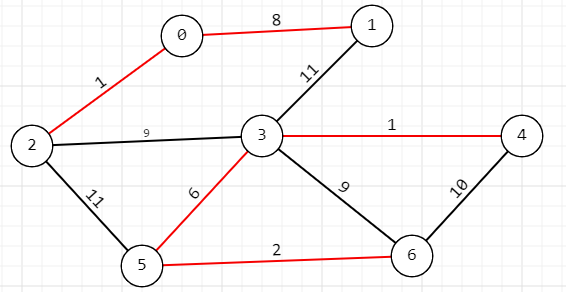


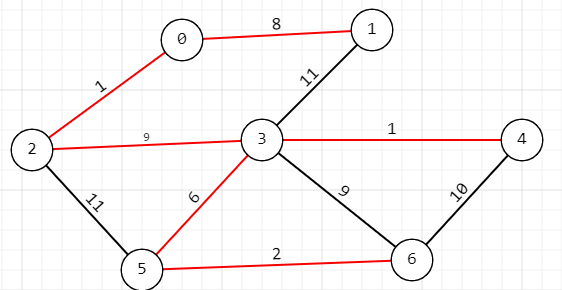












Вес будет равен 28

**Ответы на вопросы**

**1. Какие представления графов Вы знаете?**

Матрица смежности, список смежности и список ребер.

**2. В чем заключается поиск в ширину? Где рационально его использовать?**

Поиск в ширину — алгоритм обхода графа, начинающийся с заданной вершины и постепенно обходящий все ее соседей, затем все соседей ее соседей и т.д. Рационально использовать для поиска кратчайшего пути или нахождения компонент связности графа.

**3. В чем заключается поиск в глубину? В каких ситуациях рационально его использовать?**

Поиск в глубину — алгоритм обхода графа, начинающийся с заданной вершины и идущий вглубь каждой ветви до тех пор, пока не достигнет конца пути, затем возвращается на ближайшую непосещенную вершину и продолжает поиск. Рационально использовать для поиска циклов, топологической сортировки и генерации лабиринтов.

**4. В чем смысл топологической сортировки? Для чего она применяется?**

Топологическая сортировка — упорядочивание вершин графа так, чтобы для каждого ребра (u, v) вершина u предшествовала вершине v. Применяется, например, для определения порядка зависимостей между задачами в проектном управлении или определения порядка компиляции файлов в компьютерных программах.

**5. Что такое минимальное остовное дерево?**

Минимальное остовное дерево — это подмножество ребер графа, которое связывает все вершины и имеет минимальную суммарную длину.

**6. В чем заключается стандартный алгоритм построения минимального остовного дерева?**

Стандартный алгоритм построения минимального остовного дерева — это алгоритм Крускала или алгоритм Прима, которые оба имеют временную сложность O(E log E), где E — количество ребер графа.

**7. К какой категории алгоритмов относятся алгоритмы Прима и Крускала?**

Алгоритмы Прима и Крускала относятся к алгоритмам построения минимального остовного дерева.

**8. Опишите один шаг алгоритма Крускала? Когда алгоритм прекращает свою работу?**

Один шаг алгоритма Крускала заключается в выборе минимального по весу ребра, которое не образует цикл с уже выбранными ребрами. Алгоритм прекращает свою работу, когда все вершины графа становятся связанными или когда все ребра были рассмотрены.