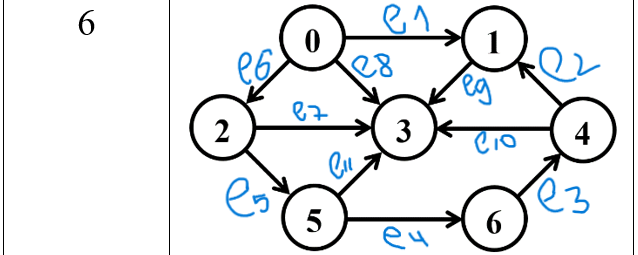
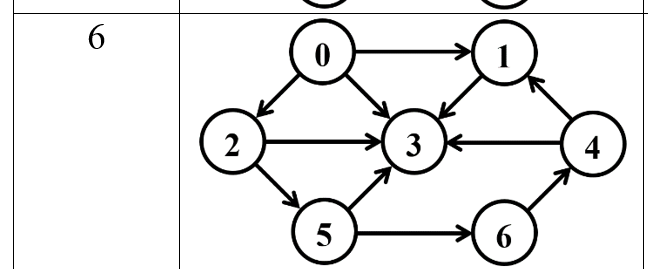
**Лабораторная работа 6. АЛГОРИТМЫ НА ГРАФАХ**

**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



**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 1 0 0 0

2 0 0 0 1 0 0 0

3 0 0 0 0 0 0 0

4 0 1 0 1 0 0 0

5 0 0 0 1 0 0 1

6 0 0 0 0 1 0 0

Матрица инцидентности:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Вершина | |  | | --- | | **e1**  **(0→1)** |  |  | | --- | |  | | |  | | --- | | **e2**  **(4→1)** |  |  | | --- | |  | | |  | | --- | | **e3 (6→4)** |  |  | | --- | |  | | |  | | --- | | **e4 (5→6)** | |  |  |  | | --- | |  | | |  | | --- | | **e5 (2→5)** |  |  | | --- | |  | | |  | | --- | | **e6 (0→2)** |  |  | | --- | |  | | |  | | --- | | **e7 (2→3)** |  |  | | --- | |  | | | **e8 (0→3)** | | --- |  |  | | --- | |  | | **e9 (1→3)** | **e10 (4→3)** | **e11**  **(5→3)** |
| 0 | 1 | -1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 1 | -1 | 1 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | -1 | -1 | -1 | -1 |
| 4 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 5 | 0 | 0 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 6 | 0 | 0 | 1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Cписки смежных вершин:

0: 1 2 3

1: 3

2: 3,5

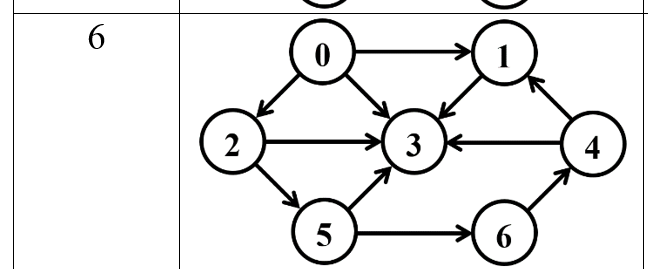
3:

4: 1,3

5: 3,6

6:4

***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**

1.Выбираем стартовую вершину 0. Добавляем ее в очередь.

**Очередь: [0]**

**Посещенная очередь: []**

2.Посещаем вершину 0 и записываем в очередь новые вершины:1,2,3.

**Очередь: [1,2,3]**

**Посещенная очередь: [0]**

3.Посещаем вершину 1.

**Очередь: [2,3]**

**Посещенная очередь: [0,1]**

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

**Очередь: [3,5]**

**Посещенная очередь: [0,1,2]**

5.Посещаем вершину 3.

**Очередь: [5]**

**Посещенная очередь: [0,1,2,3]**

6.Посещаем вершину 5 и записываем в очередь новые вершины: 6.

**Очередь: [6]**

**Посещенная очередь: [0,1,2,3,5]**

7.Посещаем вершину 6 и записываем в очередь новые вершины: 4.

**Очередь: [4]**

**Посещенная очередь: [0,1,2,3,5,6]**

8.Посещаем вершину 4.

**Очередь: []**

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

**Поиск в глубину**

1.Выбираем стартовую вершину 0

**Стек: [0]**

**Посещенные вершины: []**

2.Посещаем 0. Добавляем в стек:1,2,3

**Стек: [1,2,3]**

**Посещенные вершины: [0]**

3.Посещаем 1.

**Стек: [2,3]**

**Посещенные вершины: [0,1]**

4.Посещаем 2. Добавляем в стек: 5.

**Стек: [5,3]**

**Посещенные вершины: [0,1,2]**

5.Посещаем 5. Добавляем в стек: 6.

**Стек: [6,3]**

**Посещенные вершины: [0,1,2,5]**

6.Посещаем 6. Добавляем в стек: 4.

**Стек: [4,3]**

**Посещенные вершины: [0,1,2,5,6]**

7.Посещаем 4.

**Стек: [3]**

**Посещенные вершины: [0,1,2,5,6,4]**

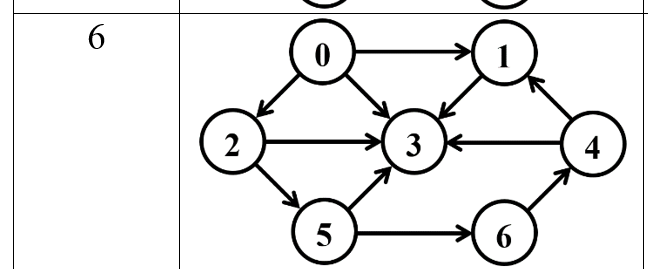
7.Посещаем 3.

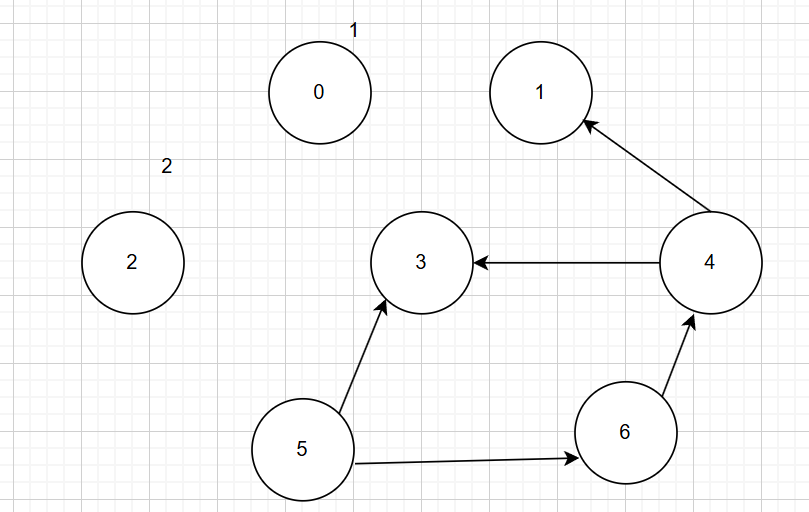
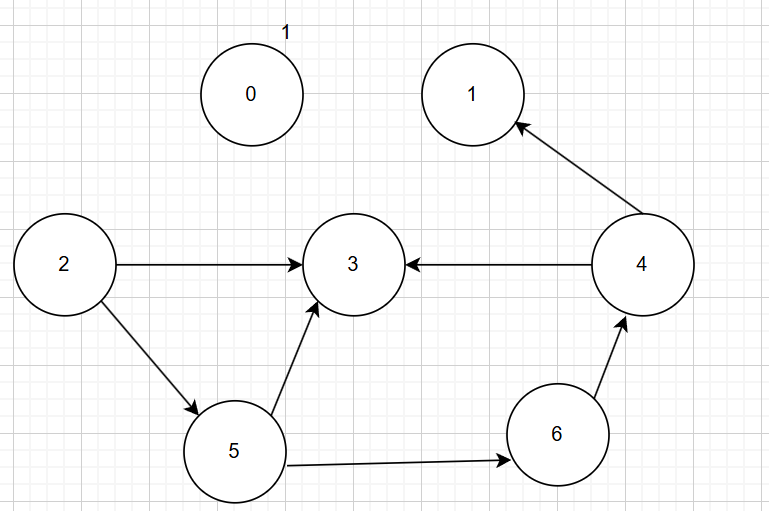
**Стек: []**

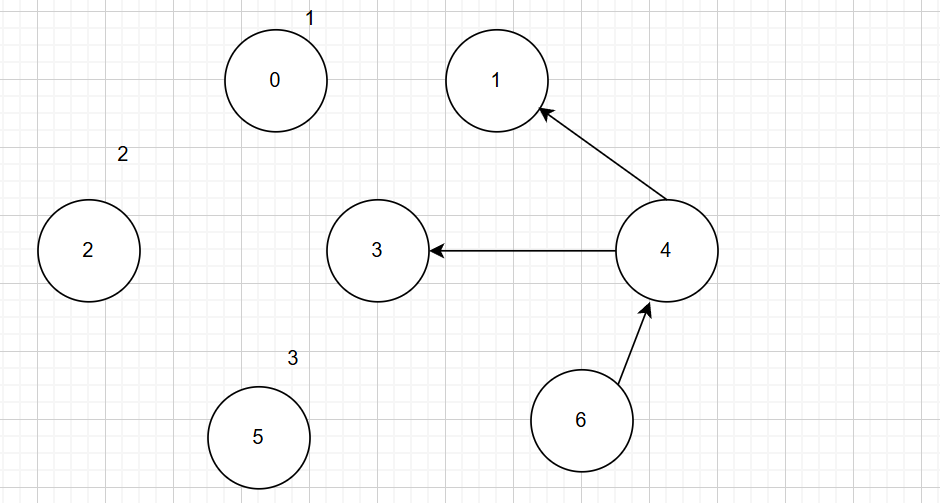
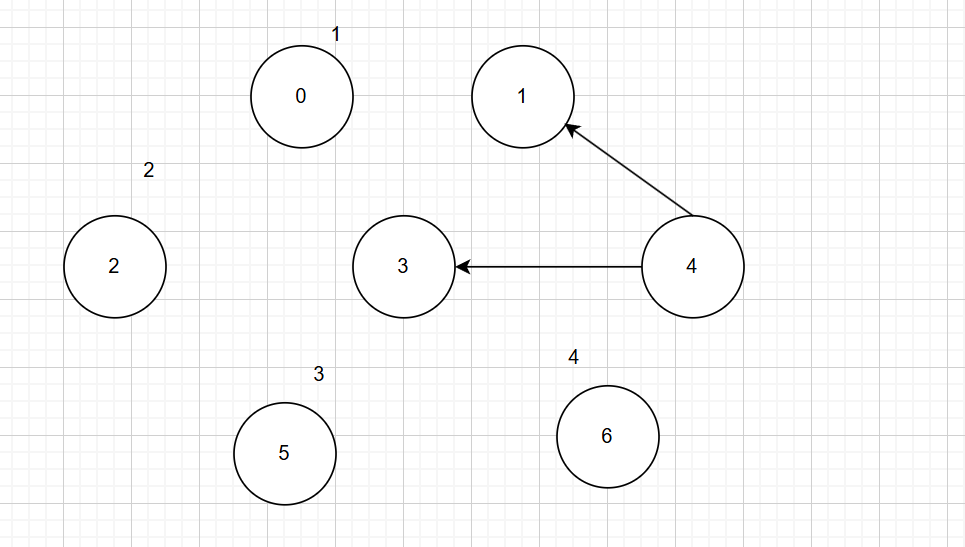
**Посещенные вершины: [0,1,2,5,6,4,3]**

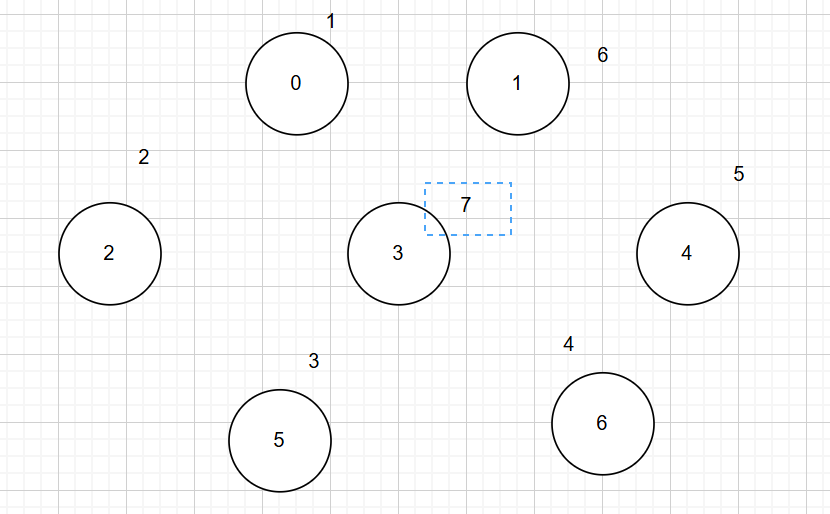
**Алгоритм топологической сортировки.**

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.



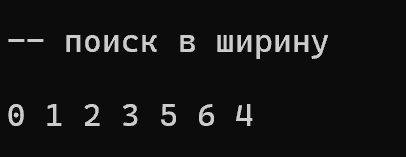
****

** **

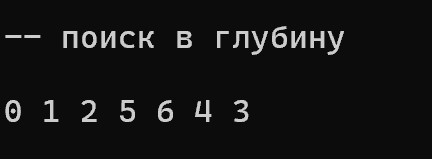
****

**Результат: 0,2,5,6,4,1,3**

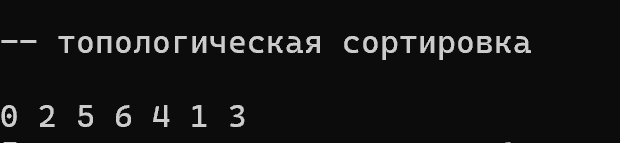
***Задание 3.*** Осуществить программную реализацию алгоритмов на C++. Разработать структуры **AMatrix** и **АList**  для представления ориентированного графа матричным и списковым способом. Разработать функции преобразования из одного способа представления в другой. Разработать функцию **BFS** обхода 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.

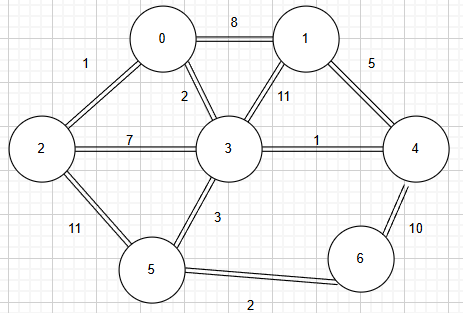


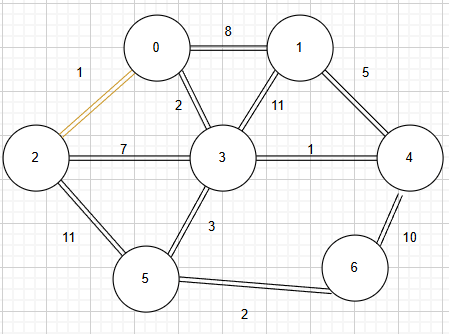
***Задание 6.*** По графу, соответствующему варианту составить минимальное остовное дерево по алгоритму Прима. Шаги построения отразить в отчете.

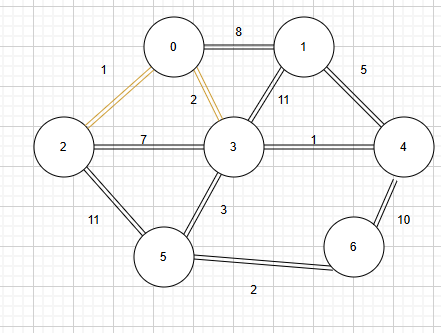
**Остовное связное дерево** – подграф, включающий вершины исходного графа G, не содержащего циклы, каждая вершина которого достижима из любой другой.

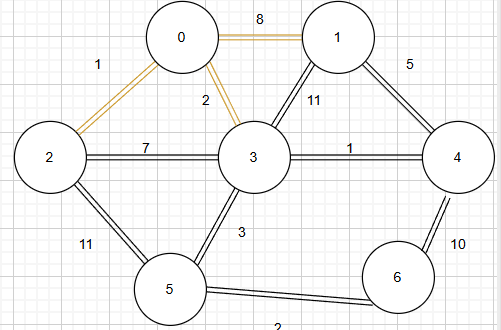
* **Начало алгоритма: с произвольной вершины**
* **К текущему дереву присоединяется смежная вершина с кратчайшим ребром.**
* **Окончание алгоритма: либо все вершины подключены, либо невозможно подключить ни одно ребро.**

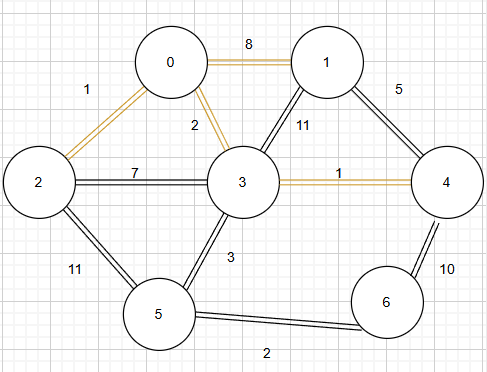
Наинаем с 0

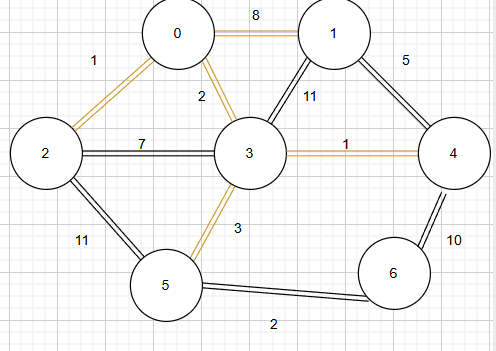


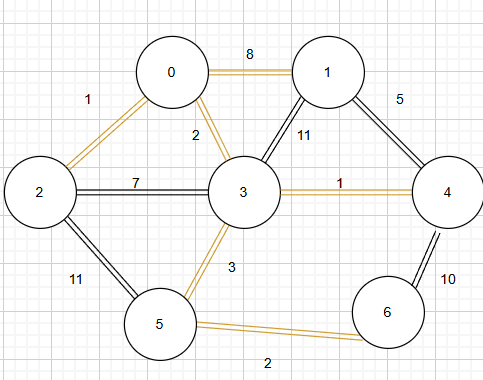












Вес будет равен 27

***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:

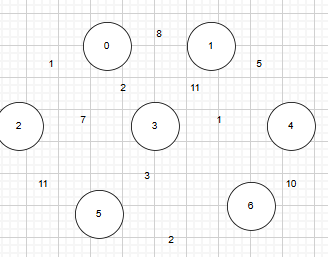
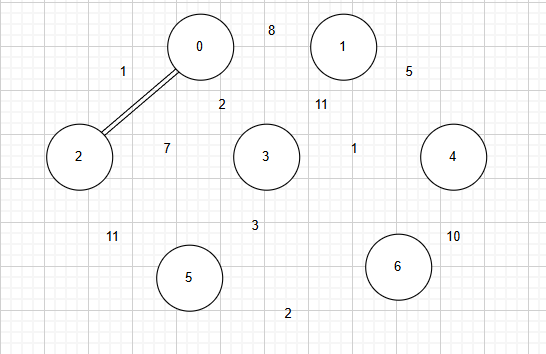
а) обе вершины включаемого ребра принадлежат одноэлементным подмножествам, тогда они объединяются в новое, связное подмножество;

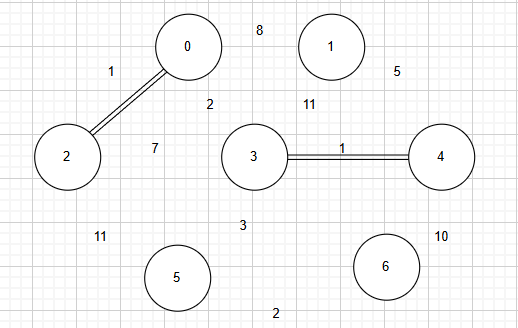
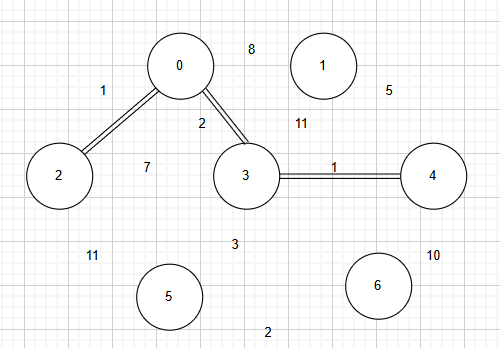
б) одна из вершин принадлежит связному под-множеству, другая нет, тогда включаем вторую в подмножество, которому принадлежит первая;

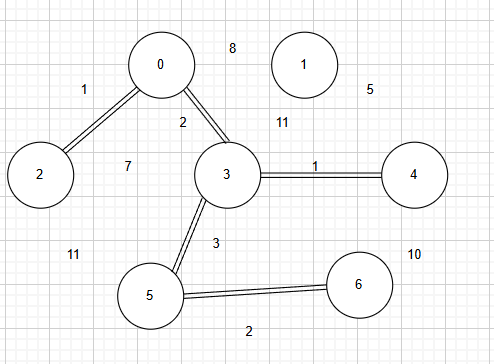
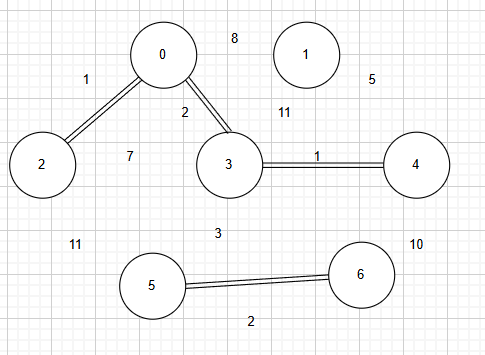
в) обе вершины принадлежат разным связным подмножествам, тогда объединяем подмножества;

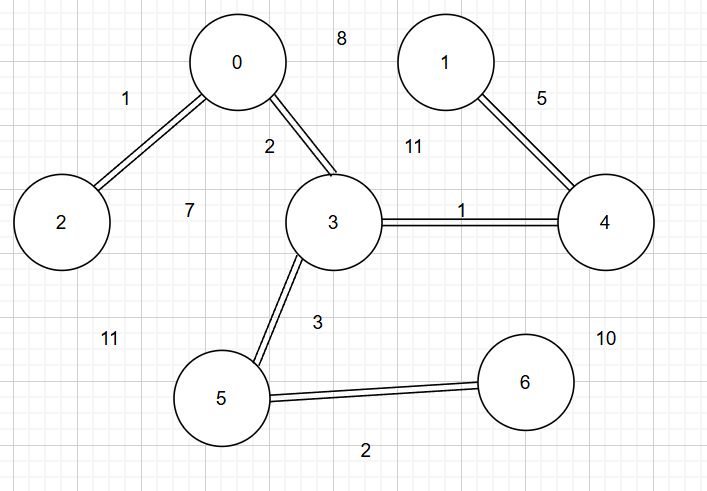
г) обе вершины принадлежат одному связному подмножеству, тогда исключаем данное ребро.

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





Вес будет равен 14