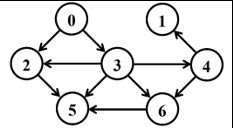
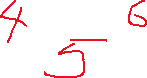
***Задание 1.*** Ориентированный граф **G** взять в соответствии с вариантом. Представить его в отчете в виде матрицы смежности, матрицы инцидентности, списка смежных вершин.





Матрица инцидентности:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 |
| 2 | -1 | 0 | -1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | -1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | 1 |
| 5 | 0 | 0 | 0 | -1 | -1 | -1 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1 | -1 | -1 | 0 | 0 |

Строки = Кол-во вершин

Столбцы = Кол-во рёбер

0 – вершина не инцидентна ребру

1 – дуга выходит из вершины

-1 – дуга входит в вершину

Матрица смежности:

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

Список смежных вершин:

0 → {2, 3}

1 → {}

2 → {5}

3 → {2, 4, 5, 6}

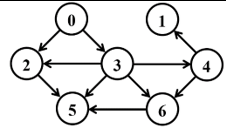
4 → {1,6}

5 → {}

6 → {5}

***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**



Шаг 1

Посещенные вершины: {}

Очередь: {2, 3}

Текущая вершина: 0

Шаг 2

Посещенные вершины: {0}

Очередь: {3, 5}

Текущая вершина: 2

Шаг 3

Посещенные вершины: {0, 2}

Очередь: {5,4,6}

Текущая вершина: 3

Шаг 4

Посещенные вершины: {0, 2, 3}

Очередь: {4,6}

Текущая вершина: 5

Шаг 5

Посещенные вершины: {0, 2, 3, 5}

Очередь: {6, 1}

Текущая вершина: 4

Шаг 6

Посещенные вершины: {0, 2, 3, 5, 4}

Очередь: {1}

Текущая вершина: 6

Шаг 7

Посещенные вершины: {0, 2, 3, 5, 4, 6}

Очередь: {}

Текущая вершина: 1

Шаг 8

Посещенные вершины: {0, 2, 3, 5, 4, 6, 1}

Очередь: {}

Текущая вершина: -

**Поиск в глубину**

Шаг 1

Посещенные вершины: {}

Стек: {2,3}

Текущая вершина: 0

Шаг 2

Посещенные вершины: {0}

Стек: {3, 5}

Текущая вершина: 2

Шаг 3

Посещенные вершины: {0,2}

Стек: {3}

Текущая вершина: 5

Шаг 4

Посещенные вершины: {0,2,5}

Стек: {4,6}

Текущая вершина: 3

Шаг 5

Посещенные вершины: {0,2,5,3}

Стек: {6, 1}

Текущая вершина: 4

Шаг 5

Посещенные вершины: {0,2,5,3,4}

Стек: {6}

Текущая вершина: 1

Шаг 6

Посещенные вершины: {0,2,5,3,4,ё}

Стек: {}

Текущая вершина: 6

Шаг 7

Посещенные вершины: {0,2,5,3,4,1,6}

Стек: {}

Текущая вершина: -

**Топологическая сортировка**

1. Находим какую-либо вершину, в которую не входят дуги, нумеруем ее.
2. Помечаем дуги, выходящие из помеченной вершины, как «не существующие».
3. Повторяем шаги (1) и (2), пока не будут занумерованы все вершины.

Шаг 1



Шаг 2

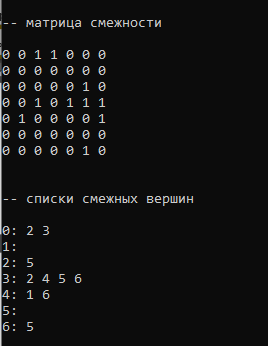


Шаг 3-7



Результирующая последовательность: 0 3 2 4 1 6 5

***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.



Изображение выглядит как текст, Шрифт, снимок экрана, типография

Контент, сгенерированный ИИ, может содержать ошибки.

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

Изображение выглядит как текст, Шрифт, снимок экрана, типография

Контент, сгенерированный ИИ, может содержать ошибки.

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

Изображение выглядит как текст, Шрифт, снимок экрана, типография

Контент, сгенерированный ИИ, может содержать ошибки.

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

Веса ребер принять:

W:

W(e0,1)=8; W(e1,0)=5;

W(e0,2)=1; W(e2,0)=3;

W(e0,3)=2; W(e3,0)=8;

W(e1,3)=11; W(e3,1)=4;

W(e1,4)=5; W(e4,1)=3;

W(e2,3)=7; W(e3,2)=9;

W(e2,5)=11; W(e5,2)=10;

W(e4,3)=4; W(e3,4)=1;

W(e4,6)=10; W(e6,4)=2;

W(e5,6)=2; W(e6,5)=6;

W(e5,3)=3; W(e3,5)=6;

W(e6,3)=7; W(e3,6)=9;

Алгоритм Прима

Шаг 1



Находим минимальный элемент (неориентированный граф)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 0 | 0 | 0 | 1 | 2 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 11 | 0 |
| 3 | 0 | 0 | 9 | 0 | 1 | 6 | 9 |
| 4 | 0 | 3 | 0 | 0 | 0 | 0 | 10 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 6 | 0 |

Шаг 1. вычёркиваем **строку 0**, выделяем **столбец 0**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 0 |  |  | 1 | 2 |  |  |  |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 |  |  |  |  |  |  |  |
| 3 | 0 | 0 | 9 | 0 | 1 | 6 | 9 |
| 4 | 0 | 3 | 0 | 0 | 0 | 0 | 10 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 6 | 0 |

**Шаг 2. Находим минимальный элемент среди оставшихся, связанных с 0 или 2**

вычёркиваем строку 3, выделяем столбец 3

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 0 |  |  | 1 | 2 |  |  |  |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 |  |  |  |  |  |  |  |
| 3 |  |  |  |  | 1 | 6 |  |
| 4 | 0 | 3 | 0 | 0 | 0 | 0 | 10 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 6 | 0 |

Шаг 3. Доступные рёбра из 3:

Добавляем **3–4 = 1**, вычёркиваем строку 4, выделяем столбец 4

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

Шаг 4. Доступные рёбра из 4:

Добавляем **4–1 = 3**, вычёркиваем строку 1, выделяем столбец 1

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

Шаг 5. Доступные рёбра:

Добавляем **3–5 = 6**, вычёркиваем строку 5, выделяем столбец 5

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

**Шаг 6. Осталась вершина 6**

Добавляем **6–5 = 6**, вычёркиваем строку 6, выделяем столбец 6

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

Минимальное остовное дерево. Вес: 19

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

**Алгоритм Краскала**

Шаг 1

Шаг 2

Шаг 3

Шаг 4

Шаг 5

Шаг 6

