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

a

0 1

d e c b

f

2 3 4

k

g h j

l

5 6

i

Матрица смежности:

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

Матрица инцидентностей:

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | a | b | c | d | e | f | g | h | i | j | k | l |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | -1 | -1 | -1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | -1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 | -1 | -1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 4 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | 1 |
| 5 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | -1 | 1 | 0 | 0 | 0 |
| 6 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | -1 | -1 | 0 | -1 |

Список смежных вершин:

S0 = {1, 2, 3}

S1 = пустое множество

S2 = {3, 5}

S3 = {1, 4, 5, 6}

S4 = {1, 6}

S5 = {6}

S6 = пустое множество

***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.

**Алгоритм поиска в ширину (BFS).**

По условию, граф имеет 7 вершин, пронумерованных начиная с нуля. В качестве стартовой вершины выбрана вершина с номером 0.

Шаг 1.

В качестве стартовой вершины выбираем вершину с номером 0.

0 1

2 3 4

5 6

Шаг 2. У вершины 0 три смежные вершины; для последующего пути выбираю вершину с наименьшим весом из трех – первую, а вторую и третью добавляю в начало очереди. Закрашиваю контур нулевой в красный цвет, как пройденную. А смежные (первая, вторая, третья) – в синий.

0 1

2 3 4

5 6

Шаг 3. У вершины 1 нет смежных вершин; для последующего пути выбираю вершину с наименьшим весом из двух – вторую, а третью добавляю в начало очереди.

0 1

2 3 4

5 6

Шаг 4. У вершины 2 две смежные вершины; вершина 3 уже добавлена в очередь, поэтому помечаю вершину 5 и перехожу к вершине 3.

0 1

2 3 4

5 6

Шаг 5. У вершины 3 три смежные вершины; вершина 5 уже добавлена в очередь, поэтому помечаю вершины 4 и 6 и перехожу к вершине 5.

0 1

2 3 4

5 6

Шаг 6. У вершины 5 только одна смежная вершина ­­– 6, которая уже помечена. Поэтому помечаю 5 вершину как пройденную и перехожу к вершине 4.

0 1

2 3 4

5 6

Шаг 7. У вершины 4 одна смежная вершина — 6 , помечаю ее как пройденную и перехожу к вершине 6.

0 1

2 3 4

5 6

Шаг 8. Вершина 6 является последней, поэтому меняем ее контур на красный и заканчиваем алгоритм.

0 1

2 3 4

5 6

Итоговое BFS-дерево:

0 1

2 3 4

5 6

Порядок обхода: 0, 1, 2, 3, 5, 4, 6

**Алгоритм поиска в глубину (DFS).**

По условию, граф имеет 7 вершин, пронумерованных начиная с нуля. В качестве стартовой вершины выбрана вершина с номером 0.

Шаг 1. В качестве стартовой вершины выбираем вершину с номером 0. Далее будем осуществлять проход по смежным вершинам, пока не сможет достичь того, чтобы не было возможности осуществить дальнейший проход.

0 1

2 3 4

5 6

Шаг 2. Вершина 0 имеет три смежные вершины, переходим в вершину 1, так как она имеет минимальный вес.

0 1

2 3 4

5 6

Шаг 2. Вершина 1 не имеет смежных вершины, поэтому закрашиваем ее контур красным цветом и возвращаемся в вершину 0.

0 1

2 3 4

5 6

Шаг 3. Второй по весу вершиной, смежной вершине 0, является вершина 2. Поэтому переходим к ней.

0 1

2 3 4

5 6

Шаг 4. Находим смежные вершины для второй – это третья и пятая. Переходим в младшую вершину.

0 1

2 3 4

5 6

Шаг 5. Находим смежные вершины для третьей – это четвертая и шестая. Переходим в младшую вершину.

0 1

2 3 4

5 6

Шаг 6. Единственной смежной для четвертой вершины является шестая. Переходим туда.

0 1

2 3 4

5 6

Шаг 7. Вершина шесть не имеет смежных вершин, поэтому возвращаемся к вершине 4.

0 1

2 3 4

5 6

Шаг 8. Вершина четыре не имеет свободных смежных вершин, поэтому возвращаемся к вершине 3.

0 1

2 3 4

5 6

Шаг 9. Вершина три имеет еще одну смежную вершину ­— пятую, переходим к ней.

0 1

2 3 4

5 6

Шаг 10. Вершина пять является глубиной, поэтому закрашиваем ее контур красным цветом и возвращаемся к вершине 3.

0 1

2 3 4

5 6

Шаг 11. Аналогично закрашиваем вершину три и переходим ко второй.

0 1

2 3 4

5 6

Шаг 12. Также закрашиваем вершину два и переходим к нулевой.

0 1

2 3 4

5 6

Шаг 13. Закрашиваем нулевую вершину. Алгоритм окончен.

0 1

2 3 4

5 6

В результате получили DFS-дерево:

0 1

2 3 4

5 6

Порядок обхода: 0, 1, 2, 3, 4, 6, 5.

**Алгоритм топологической сортировки.**

Топологическая сортировка (Topological sort) — один из основных алгоритмов на графах, который применяется для решения множества более сложных задач.

Задача топологической сортировки графа состоит в следующем: указать такой линейный порядок на его вершинах, чтобы любое ребро вело от вершины с меньшим номером к вершине с большим номером. Очевидно, что если в графе есть циклы, то такого порядка не существует.

Топологическая сортировка:

4, 6, 5, 2, 3, 1, 0.

**Задание 3**

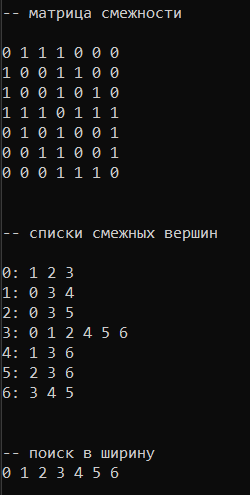


Рисунок 4 – Демонстрация программы с функцией BFS

**Задание 4**

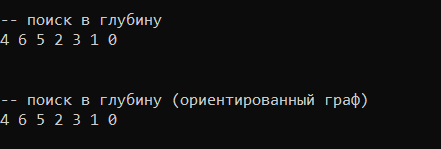


Рисунок 5 – Демонстрация программы с функцией DFS

**Задание 5**



Рисунок 6 – Демонстрация доработанной функции DFS

**Задание 6**

На вход алгоритма подаётся связный неориентированный граф. Для каждого ребра задаётся его стоимость.

Сначала берётся произвольная вершина и находится ребро, инцидентное данной вершине и обладающее наименьшей стоимостью. Найденное ребро и соединяемые им две вершины образуют дерево. Затем, рассматриваются рёбра графа, один конец которых — уже принадлежащая дереву вершина, а другой — нет; из этих рёбер выбирается ребро наименьшей стоимости. Выбираемое на каждом шаге ребро присоединяется к дереву. Рост дерева происходит до тех пор, пока не будут исчерпаны все вершины исходного графа.

Результатом работы алгоритма является остовное дерево минимальной стоимости.

Начальная вершина 0:

1. {0} => ({0,2} = 1), ({0,3} = 2). Идем {0,3}
2. {3} => ({3,4} = 1), ({3,5} = 6). Идем {3,4}
3. {4} => ({4,1} = 3)
4. {5} => ({5,6} = 2)

1+2+1+6+3+2=15

Вес минимального остовного дерева равен 15.

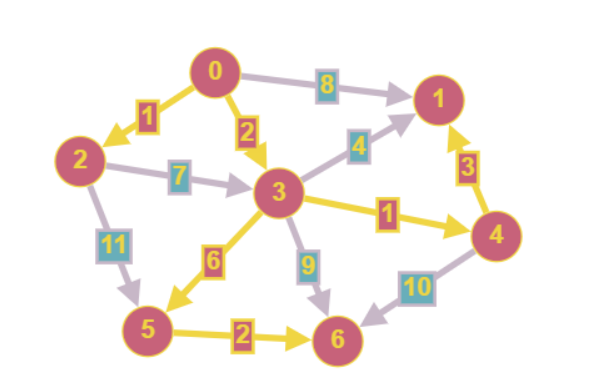


Рисунок 6 – Завершенный алгоритм Прима

**Задание 7**

В начале текущее множество рёбер устанавливается пустым. Затем, пока это возможно, проводится следующая операция: из всех рёбер, добавление которых к уже имеющемуся множеству не вызовет появление в нём цикла, выбирается ребро минимального веса и добавляется к уже имеющемуся множеству. Когда таких рёбер больше нет, алгоритм завершён. Подграф данного графа, содержащий все его вершины и найденное множество рёбер, является его остовным деревом минимального веса.

Начальное ребро {3,4} = 1

1. {3,4} => ({0,2} = 1),
2. {0,2} => ({0,3} = 2),
3. {0,3} => ({5,6} = 2),
4. {5,6} => ({1,4} = 3),
5. {1,4} => ({3,5} = 6)

1+1+2+2+3+6=15

Вес минимального остовного дерева равен 15.

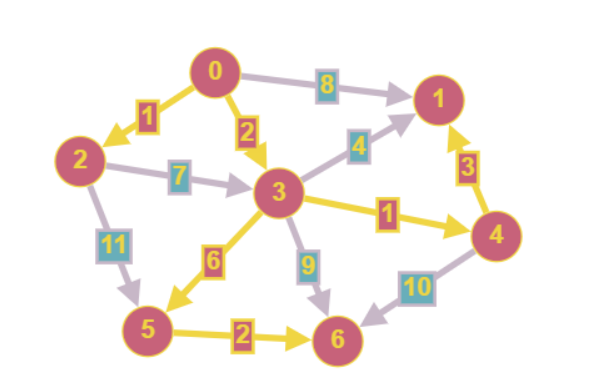


Рисунок 7 – Завершенный алгоритм Краскала