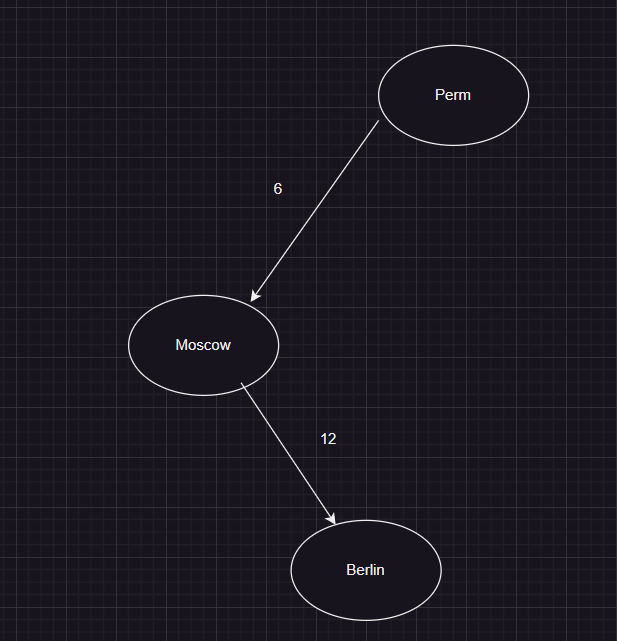
**ЛЕКЦИЯ 11**

**ТЕОРИЯ ГРАФОВ**

1. Граф – множество узлов и связей между ними;
2. Каждый узел называется вершиной, а каждая связь ребром;
3. Каждое ребро соединяет только две вершины.



Круги – вершины (узлы)

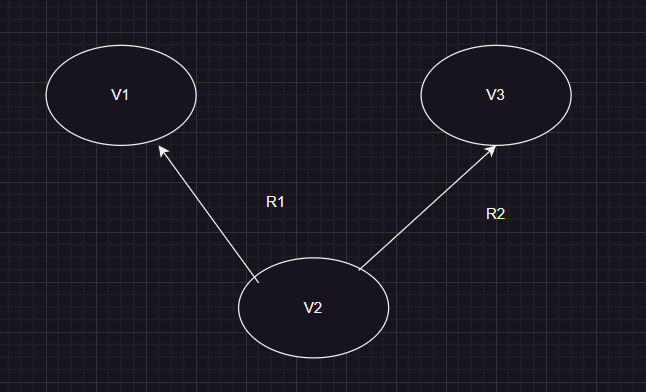
Стрелки – ребра (связи). Числа – веса ребр

**ВИДЫ ГРАФОВ**

1. Ориентированный граф – у ребер есть направление. Движение возможно только по направлению;
2. Неориентированный – возможно двигаться по ребрам в любом направлении;
3. Смешанный - некоторый ребра имеют направления, некоторые нет;
4. Взвешенный граф – каждому ребру поставлено некоторое числовое значение – вес;
5. Если числового значения нет, то граф называется невзвешенным.
6. В название графа указывается его ориентация (ориентированный или нет) и взвешенность (взвешенный или нет);
7. Связный граф – между любой парой вершиной существует как минимум один путь;
8. Несвязный – хотя бы одна вершина не связанна с какой - либо другой;
9. Зачастую связность упускается в названии графа;
10. Плотный – число ребер близко к максимальному;
11. Разреженный граф – имеет малое число ребер;
12. Любой связный граф является плотным, но не каждый плотный является связным.
13. Петля – ребро, которое начинается и заканчивается на одной вершине;

**ПОНЯТИЯ**

1. Инцидентность – используется только в отношении ребра и вершины. Если вершина соединяется с чем - то через ребро, то эта вершина инцидентна этому ребру;
2. Смежность – используется в отношении двух ребер, которые инцидентны одной вершине. Две вершины инцидентны одному ребру;

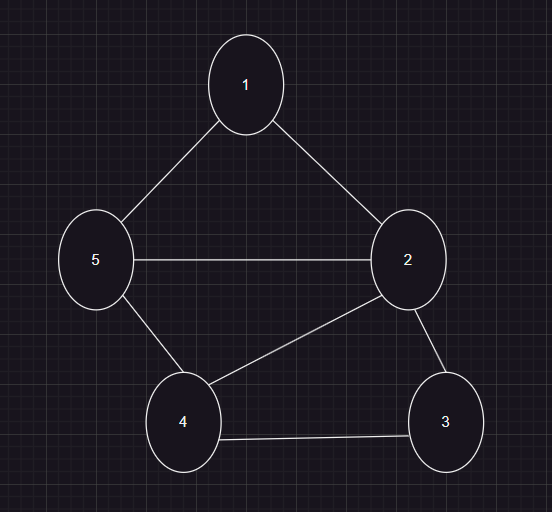
****

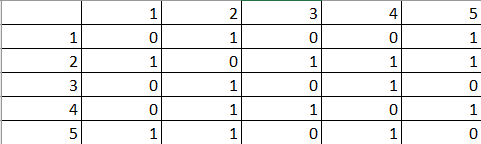
R1 и R2 смежны так как инциденты одному ребру V2.

1. Маршрут – проход по графу через заданную последовательность вершин. Если начало и конец маршрута совпадают, то он замкнут, иначе он называется открытым;
2. Цепь – маршрут, все ребра которого различны. Если все вершины также различны, то цепь называется простой;
3. Цикл – цепь, в которой начало и конец совпадают;

**МАТРИЦА СМЕЖНОСТИ ГРАФА**

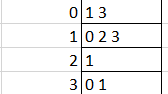
1. Квадратная матрица, содержащая 0 и 1 для невзвешенного графа. Единица – ребро существует между вершинами, 0 – нет ребра (или нет возможности по нему пройти). Столбцы и строки – номера вершин. Если граф взвешенный, то вместо 0 и 1 будут веса ребер между элементами;

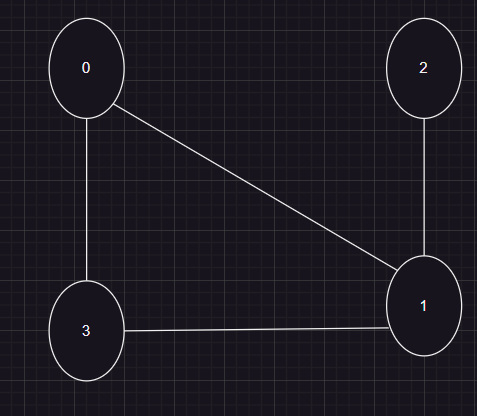




**ВЕКТОР СМЕЖНОСТИ**

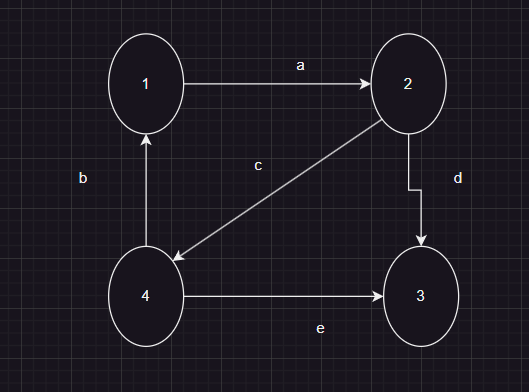
1. Новая форма представления графа. Для каждой вершины хранит номера соседей;
2. Каждой вершине присваивается индекс от 0 до кол-ва вершин – 1;

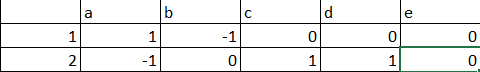




**МАТРИЦА ИНЦИНДЕНТНОСТИ**

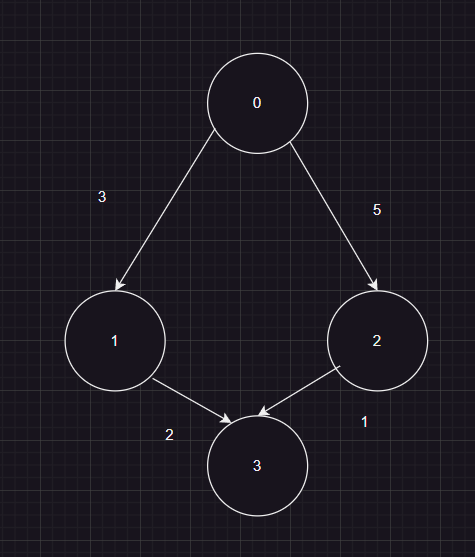
1. Если граф не ориентирован и рассматриваемая вершина не инцидента ребру, то в матрицу записывается 0, иначе если инцидента, то в матрицу записывается единица;
2. Если ориентирован, то если вершина инцидента ребру и является его началом, то в матрицу ставится 1, 0 если не инцидента и -1 если является концом;

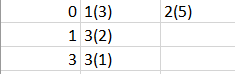




**СПИСОК СМЕЖНОСТИ**

1. Каждой вершины графа соответствует список из соседей, если не взвешенный. Если взвешенный, то рядом с соседней вершиной ставит вес ребра;





**СОЗДАНИЕ ГРАФА НА C++**

1. Создается шаблонный класс граф, где узел содержит шаблонный тип;
2. В классе граф создаем вектор вершин, где индекс вершины – индекс в матрице смежности;
3. В матрице для вершин I и j проверяется значение, стоящие на пересечении I и j;
4. В приватном поле графа хранится вектор вершин и матрица смежности.

**МЕТОДЫ ОБХОДЫ ГРАФОВ**

1. Обход в глубину. Начало – одна из вершин (фиксируем информация, что она посещена). Далее идем по ребру к соседним вершинам, отмечая посещение соседей. Если вершина попалась несколько раз, то обрабатывается она единожды. Пример: правило левой руки;
2. Обход в ширину. Поиск начинается с начальной вершины, которая обрабатывается, маркируется и помещается в очередь. Обработанную вершину удалили и в очередь записали все соседние вершины обработанной.

**АЛГОРИТМЫ ДЕЙСКСТРЫ**

1. Алгоритм поиск минимального пути от заданной вершины до всех остальных;
2. Каждой вершине присваиваем некоторое очень большое число, кроме начальной, е1 ставим 0;
3. Вычисляем новое число для каждой вершины как сумму числа вершины и веса ребра между ними. Если оно меньше, чем число, которое закреплено за соседней, то к соседней присваиваем найденную сумму;
4. Переходим к ближайшему соседу. Повторяем, пока не обработаем все вершины.

**ЛАБОРАТОРНАЯ РАБОТА**

1. Необходимо сделать вертикальную печать графа;
2. Реализовать все обходы и алгоритм Д. Граф указан в варианте.

**ЗАДАЧА КОММИВОЯЖЕРА**

1. Условие: человек двигается от пункта А с возвратом в тот же пункт. Каждый город на графе должен быть посещен хотя бы один раз. Необходимо найти минимальный путь по времени либо расстоянию;

**МЕТОД ВЕТВЕЙ И ГРАНИЦ**

1. Общий принцип – нахождение наиболее эффективным маршрутов из существующих;
2. Шаги метода:

- Нумеруем города;

- Заносим расстояния между городами;

- Находим минимум в каждой строке матрицы;

- Уменьшаем каждое значение элемента строки на ранее найденный минимум;

- Аналогичное действие для столбцов после предыдущей редукции;

- Делаем редукция столбцов;

- Оценкой для каждой нулевой клетки – сумма минимальных элемента по столбцу и строке, где найден 0 (сам 0 не учитывается);

- Выбираем нулевой элемент с наибольшей оценкой. Его заносим в итоговый путь;

- Все остальные возможные пути закрываем;

- Если все отрезки пути еще не найдены, то продолжаем делать редукции;

- Таким образом мы получим то из какого города мы двигаемся в какой;

- Редуцированные M считаются 0.