Белорусский государственный университет
Факультет радиофизики и компьютерных технологий
Кафедра интеллектуальных систем

“АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ”

Лабораторная работа №2
“Базовые операции над графами”

Выберите подходящий тип данных для представления графа.

Напишите и протестируйте процедуры обеспечивающие добавление вершины в граф, вывод элементов графа на экран. Реализуйте процедуры поиска в глубину и поиска в ширину в графе.

Входные параметры, задающие граф, необходимо считывать из файла.

Выполните задание согласно вашему варианту:

1. Найдите все вершины заданного графа, недостижимые от заданной вершины.
2. Определите, выполняется ли условие достижимости между любыми двумя вершинами заданного орграфа (*условие достижимости* - существует путь, связывающий эти вершины).
3. Определите, является ли связным заданный граф.
4. Найдите длину кратчайшего цикла в графе.
5. Для двух выделенных вершин графа постройте соединяющий их простой путь.
6. Найдите самый длинный простой путь в графе.
7. Найдите все вершины графа, к которым существует путь заданной длины от выделенной вершины.
8. Найдите все вершины орграфа, от которых существует путь заданной длины к выделенной вершине.
9. Найдите источник заданного орграфа (*источником* орграфа называется вершина, от которой достижимы все другие вершины).
10. Найдите такую вершину заданного графа, которая принадлежит каждому пути между двумя выделенными (различными) вершинами и отлична от каждой из них.
11. Найдите диаметр графа (*диаметр* графа - максимальное расстояние между всевозможными парами его вершин).
12. Найдите медиану графа (*медиана* графа - это такая его вершина, для которой сумма расстояний от нее до остальных вершин минимальна. Граф неориентированный).
13. Найдите сток заданного орграфа (*стоком* орграфа называют вершину, в которую ведут ребра из всех других вершин графа).

Краткие справочные сведения

Граф — это совокупность непустого множества объектов и множества пар этих объектов. Объекты представляются как вершины, или узлы графа, а связи — как дуги, или рёбра.

Для разных областей применения виды графов могут различаться направленностью, ограничениями на количество связей и дополнительными данными о вершинах или ребрах. Многие структуры, представляющие практический интерес в математике и информатике, могут быть представлены графами. Например, строение учебника можно смоделировать при помощи ориентированного графа (орграф), в котором вершины — это параграфы, а дуги (ориентированные ребра) — гиперссылки.

Граф – G = (V, E) состоит из V — конечного непустого множества элементов (называемых **узлами** или **вершинами**), и E – множества неупорядоченных пар (u, v) вершин из V, называемых **ребрами**.

Ребро s = (u, v) соединяет вершины u, v. Ребро s и вершина u (а также s и v) называются **инцидентными**, а вершины u и v – **смежными**.

Графы бывают: **ориентированные** и **неориентированные**, **связные** и **несвязные**, **полные**, **двудольные**, **мультиграфы** и т.д..

Пустой граф – граф с пустым множеством вершин.

Ориентированный граф, или **орграф**, G = (V, E) отличается от графа тем, что E – множество упорядоченных пар (u, v) вершин u, v ∈ V, называемых **дугами**. Дуга (u, v) ведет от вершины u к вершине v.

Если некоторая пара вершин соединена несколькими ребрами, то граф называется **мультиграфом**.

На практике граф изображают следующим образом (вершины – точками, ребра – линиями (или для ориентированного графа направленными линиями):

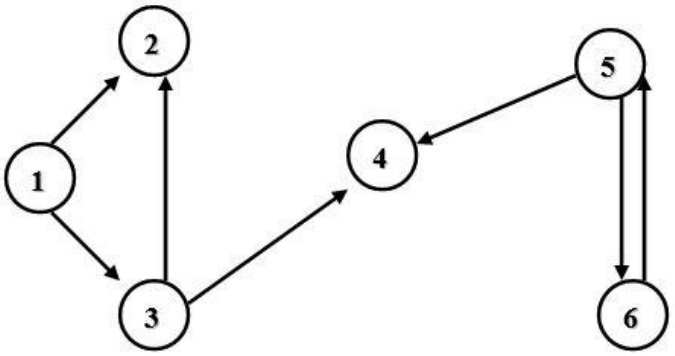


Рисунок 1. Изображение графа

Количество вершин графа называется его **порядком**.

Количество ребер графа называется его **размером**.

Если между вершинами существует ребро, то вершины называются **смежными**.

Число ребер, инцидентных некоторой вершине, называется **степенью** данной **вершины**.

Путь, соединяющий вершины u, v, – это последовательность вершин $v_0, v_1, \ldots, v_n$ такая, что $v_0 = u$, $v_n = v$ и для любого i вершины v_i и v_{i+1} соединены ребром.

Длина пути равна количеству ребер.

Путь **замкнут**, если $v_0 = v_n$.

Путь называется **простым**, если все его вершины различны.

Замкнутый путь, в котором все ребра различны, называется **циклом**.

Расстояние между двумя вершинами – длина кратчайшего пути, соединяющего эти вершины.

Граф называется **связным**, если для любой пары вершин существует соединяющий их путь.

При этом вершину v называют **преемником** вершины u, а u – **предшественником** вершины v.

Деревом называется связный граф без циклов.

Корневое дерево – ориентированный граф, который удовлетворяет следующим условиям:

- имеется точно один узел (корневой), в который не входит ни одна дуга;
- в каждый узел, кроме корня входит ровно одна дуга;
- из корня имеется путь в каждый узел.

Существует множество алгоритмов на графах, в основе которых лежит такой перебор вершин графа, при котором каждая вершина просматривается ровно один раз:

- поиск в глубину;
- поиск в ширину;
- топологическая сортировка.
- нахождение кратчайшего пути;
- максимальный поток;
- минимальное остовное дерево.

Способы представления графа в памяти:

1. С использованием массивов — матрица смежности, матрица инцидентности.
2. С использованием последовательно связанных элементов – список ребер, список смежности.

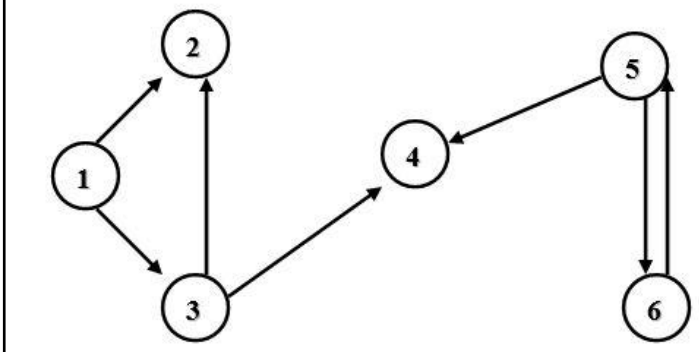
1. Матрица смежности

Классическим способом задания графа является матрица смежности A размера n x n

(n – количество вершин графа), где A[i][j] =1, если существует ребро(дуга), идущее из вершины i в j, и A[i][j] =0 в противном случае.

Матрица смежности неориентированного графа является симметричной матрицей, на главной диагонали которой стоят нули.

	1 2 3 4 5 6
1	0 1 1 0 0 0
2	0 0 0 0 0 0
3	0 1 0 1 0 0
4	0 0 0 0 0 0
5	0 0 0 1 0 1
6	0 0 0 0 1 0



Недостатками такого представления являются:

- объем памяти вне зависимости от количества ребер (дуг) составляет n^2;
- необходимо заранее знать количество вершин графа.

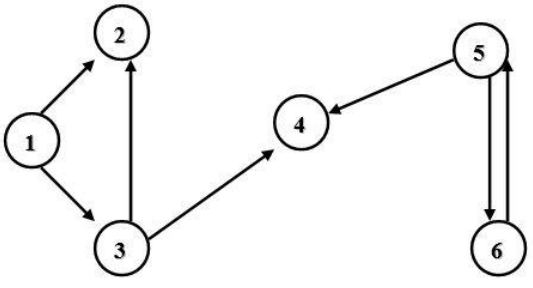
Недостатками такого представления являются:

- объем памяти вне зависимости от количества ребер (дуг) составляет n^2;
- необходимо заранее знать количество вершин графа.

2. Матрица инцидентности

Другим традиционным способом представления графа служит матрица инцидентности. Эта матрица A размера n x m, где n – количество вершин графа, а m – количество ребер графа. Для орграфа столбец, соответствующий дуге (i, j) содержит 1 в строке, соответствующей вершине i, содержит -1 в строке, соответствующей вершине j, и нули во всех остальных строках.

	(1,2)	(1,3)	(3,2)	(3,4)	(5,4)	(5,6)	(6,5)
1	1	1	0	0	0	0	0
2	-1	0	-1	0	0	0	0
3	0	-1	1	1	0	0	0
4	0	0	0	-1	-1	0	0
5	0	0	0	0	1	1	-1
6	0	0	0	0	0	-1	1



Недостатками такого представления являются:

- объем памяти составляет *nm* ячеек, которые в большинстве заполнены нулями;
- чтобы проверить существование дуги между вершинами, необходим перебор столбцов матрицы;
- необходимо заранее знать количество вершин графа.

3. Список пар (список ребер)

Более экономным в отношении памяти, особенно если m << n^2, является метод представления графа с помощью списка пар, соответствующих ребрам: пара i, j соответствует ребру (i, j)

Для орграфа, представленного на рисунке 1:

1 1	3	3	5	5	6
2 3	2	4	4	6	5

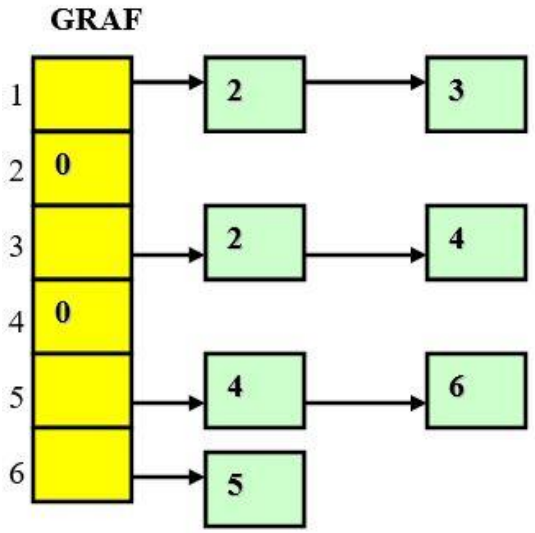
Объем памяти в этом случае составляет O(m), но необходимо большое количество шагов для получения множества вершин, к которым ведут ребра из заданной вершины.

4. Список смежности

Во многих ситуациях наиболее приемлемым способом задания графа является структура данных, которую называют списками смежности.

```
struct  SP
 {
        int   key;              // номер смежной вершины
          SP   *next;           // указатель на следующую смежную вершину
        };
SP  *GRAF[n];
```

В этом случае GRAF – массив, i–й элемент которого является указателем на начало линейного списка вершин, смежных с i–ой вершиной; если из вершины i нет выходящих дуг, то GRAF[i] = NULL. Таким образом, граф представлен в виде n списков смежности, по одному для каждой вершины.



Объем памяти в этом случае O(n+m), но необходимо заранее знать количество вершин графа.

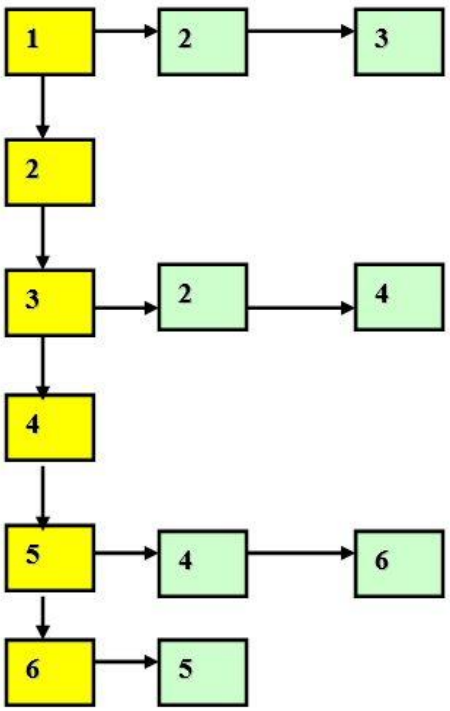
5. Список смежности

Во многих ситуациях вершины в граф добавляются в ходе выполнения программы, т.е. количество вершин графа может быть неизвестно заранее. Тогда наиболее приемлемым способом задания графа является структура данных, которую называют списками смежности, но граф представляется не массивом таких списков, а тоже линейным списком, в котором обязательны поля: номер вершины графа, указатель на список смежных с данной вершин, указатель на следующую вершину графа.

```
struct  SP
 {
        int   key;                  // номер смежной вершины
   SP   *next;              // указатель на следующую смежную вершину
        };

struct GRAF
 {
         int      NV;              //  номер вершины
         SP       *spis_sv;        //указатель на линейный список смежных вершин
    GRAF *next;          //указатель на следующую вершину графа
        };
```

Для орграфа, представленного на рисунке 1:



Поиск в глубину

Поиск в глубину (англ. Depth First Search) является одним из методов систематического исследования всех вершин и ребер графа. Он исходит из процедуры

прохождения графа методом поиска с возвратом и является основой многих алгоритмов на графах, как ориентированных, так и неориентированных.

Общая идея метода состоит в следующем. Поиск начинается с посещения некоторой вершины v0 ∈ V. Пусть v – последняя посещенная вершина. Выбирается произвольное ребро (v, w), инцидентное v. Если вершина w уже пройдена (посещалась ранее), осуществляется возврат в вершину v и выбирается другое ребро.

Если вершина w еще не пройдена, то она посещается и процесс применяется рекурсивно к вершине w. Если все ребра, инцидентные вершине v, уже исследованы, осуществляется возврат назад по ребру (u, v), по которому пришли в вершину v, и продолжается исследование ребер, инцидентных u. Процесс заканчивается, когда делается попытка вернуться назад из вершины v0, с которой начиналось исследование.

Рассмотренная процедура представлена алгоритмом 1.

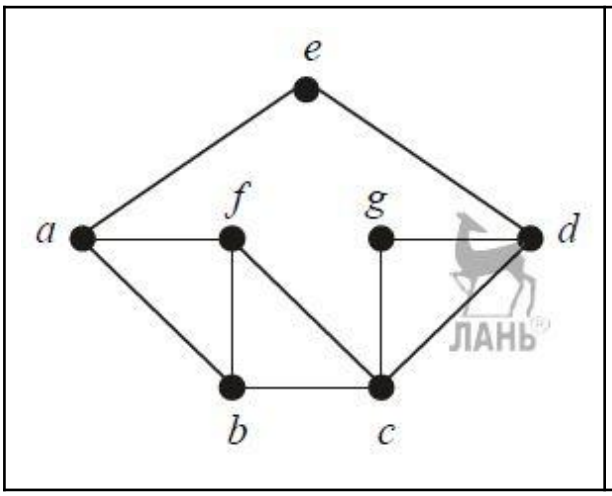
Данный алгоритм осуществляет поиск в глубину в произвольном, необязательно связном графе, заданном структурой смежности. Каждой вершине v ∈ V графа поставлен в соответствие элемент new (v), чтобы отличить уже пройденные вершины графа (new (v) = false) от еще не пройденных (new (v) = true).

```
for x ∈ V do new (x) ← true   // инициализация
for x ∈ V do if new (x) then DFS (x)

procedure DFS (v)
        visit (v)                    // посещение вершины v
        new (v) ← false\
        for w ∈ Adj (v) do if new (w) then DFS (w)
return
```

алгоритм 1. Поиск в глубину (псевдокод)

Для неориентированного графа, представленного ниже, заданного структурой смежности, поиск в глубину дает следующий порядок посещения вершин: a, b, c, d, e, g, f. Очевидно, что порядок прохождения вершин графа не единственен, так как ребра, инцидентные вершине, могут выбираться для рассмотрения в произвольном порядке.



v	Adj(v)
a	b, e, f
b	a, c, f
c	b, d, f, g
d	c, e, g
e	a, d
f	a, b, c
g	c, d

Поскольку для каждой вершины, которая проходится впервые, производится обращение к *DFS* ровно один раз, всего требуется |*V* | обращений к *DFS*. При каждом обращении количество производимых действий пропорционально числу ребер, инцидентных рассматриваемой вершине. Поэтому время поиска в глубину в произвольном графе равно O (|*V* | + | *E* |).

Методика поиска в глубину очевидным образом переносится на ориентированные графы и не требует никаких модификаций. Необходимо только учитывать направленность ребер (дуг).