Динамические структуры данных

Прикладное программирование

Понятие графа

Граф – это совокупность двух конечных множеств: множества точек и множества линий, попарно соединяющих некоторые из этих точек. Множество точек называется вершинами (узлами) графа. Множество линий, соединяющих вершины графа, называются ребрами (дугами) графа.

Основные понятия теории графов

- **Ориентированный граф (орграф) – граф, у которого все ребра ориентированы, т.е. ребрам которого присвоено направление.**
- **Неориентированный граф (неорграф) – граф, у которого все ребра неориентированы, т.е. ребрам которого не задано направление.**
- **Смешанный граф – граф, содержащий как ориентированные, так и неориентированные ребра.**

Основные понятия теории графов

- Петлей называется ребро, соединяющее вершину саму с собой. Две вершины называются смежными, если существует соединяющее их ребро. Ребра, соединяющие одну и ту же пару вершин, называются кратными.
- Простой граф – это граф, в котором нет ни петель, ни кратных ребер.
- Мультиграф – это граф, у которого любые две вершины соединены более чем одним ребром.

Основные понятия теории графов

- Маршрутом в графе называется конечная чередующаяся последовательность смежных вершин и ребер, соединяющих эти вершины.
- Маршрут называется открытым, если его начальная и конечная вершины различны, в противном случае он называется замкнутым.
- Маршрут называется цепью, если все его ребра различны. Открытая цепь называется путем, если все ее вершины различны.

Основные понятия теории графов

- Замкнутая цепь называется циклом, если различны все ее вершины, за исключением концевых.
- Граф называется связным, если для любой пары вершин существует соединяющий их путь.

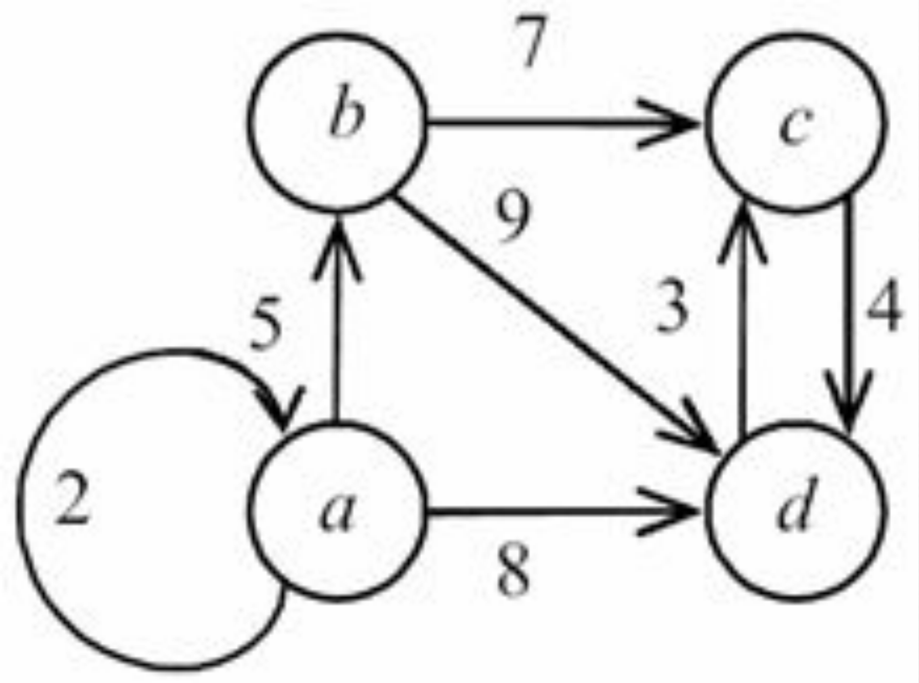
Основные понятия теории графов

- Вес вершины – число (действительное, целое или рациональное), поставленное в соответствие данной вершине (интерпретируется как стоимость, пропускная способность и т. д.). Вес (длина) ребра – число или несколько чисел, которые интерпретируются по отношению к ребру как длина, пропускная способность и т. д.
- Взвешенный граф – граф, каждому ребру которого поставлено в соответствие некое значение (вес ребра).

Представление графа в памяти компьютера

- Выбор структуры данных для хранения графа в памяти компьютера имеет принципиальное значение при разработке эффективных алгоритмов. Рассмотрим несколько способов представления графа.
- Пусть задан граф, у которого количество вершин равно n, а количество ребер – m. Каждое ребро и каждая вершина имеют вес – целое положительное число. Если граф не является помеченным, то считается, что вес равен единице.

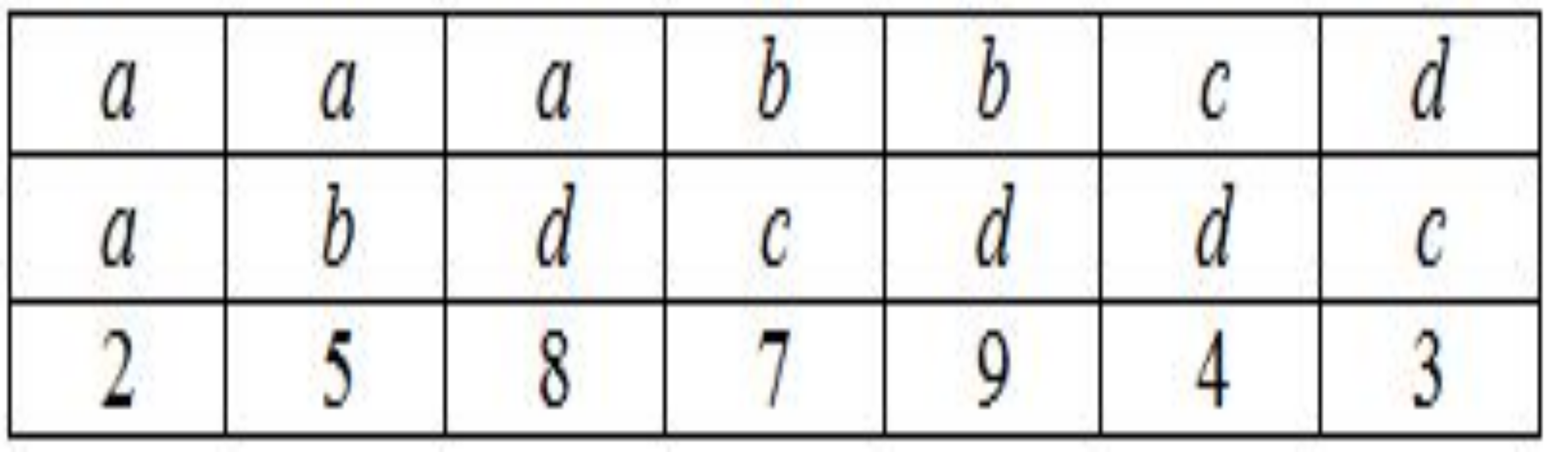
Представление графа в памяти компьютера



Представление графа в памяти компьютера

Список ребер – это множество, образованное парами смежных вершин. Для его хранения обычно используют одномерный массив размером m, содержащий список пар вершин, смежных с одним ребром графа. Список ребер более удобен для реализации различных алгоритмов на графах по сравнению с другими способами.

Представление графа в памяти компьютера

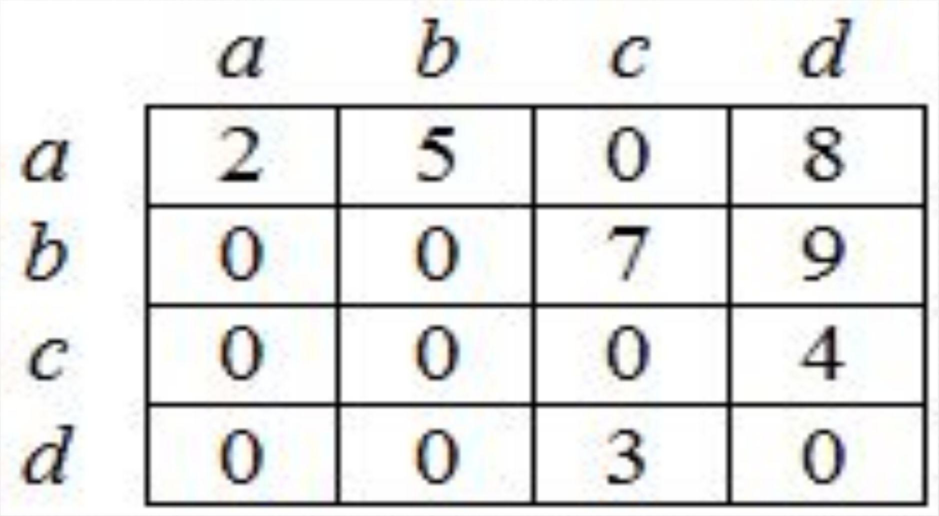


a	*a*	*a*	*b*	*b*	*c*	*d*
a	*b*	*d*	*c*	*d*	*d*	*c*
2	5	8	7	9	4	3

Представление графа в памяти компьютера

- Матрица смежности – это двумерный массив размерности n x n, значения элементов которого характеризуются смежностью вершин графа. При этом значению элемента матрицы присваивается количество ребер, которые соединяют соответствующие вершины. Данный способ действенен, когда надо проверять смежность или находить вес ребра по двум заданным вершинам.

Представление графа в памяти компьютера

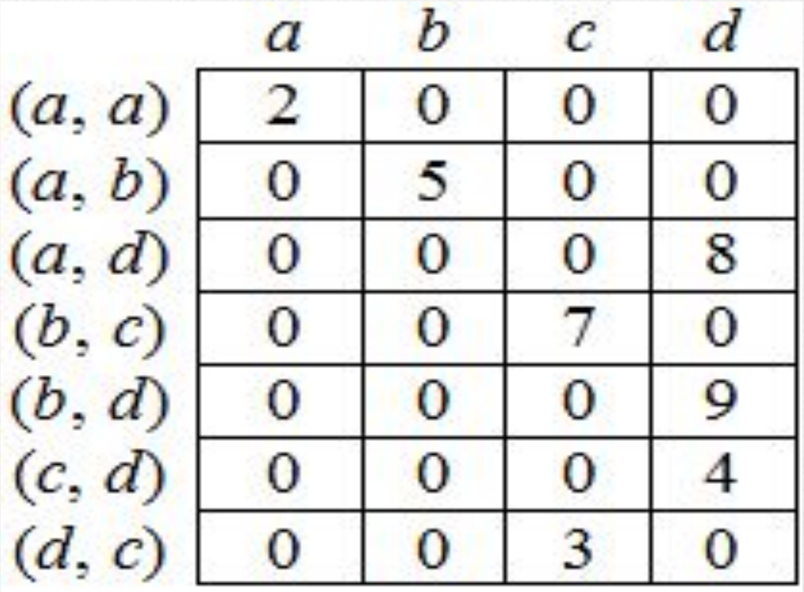


	a	*b*	*c*	*d*
a	2	5	0	8
b	0	0	7	9
c	0	0	0	4
d	0	0	3	0

Представление графа в памяти компьютера

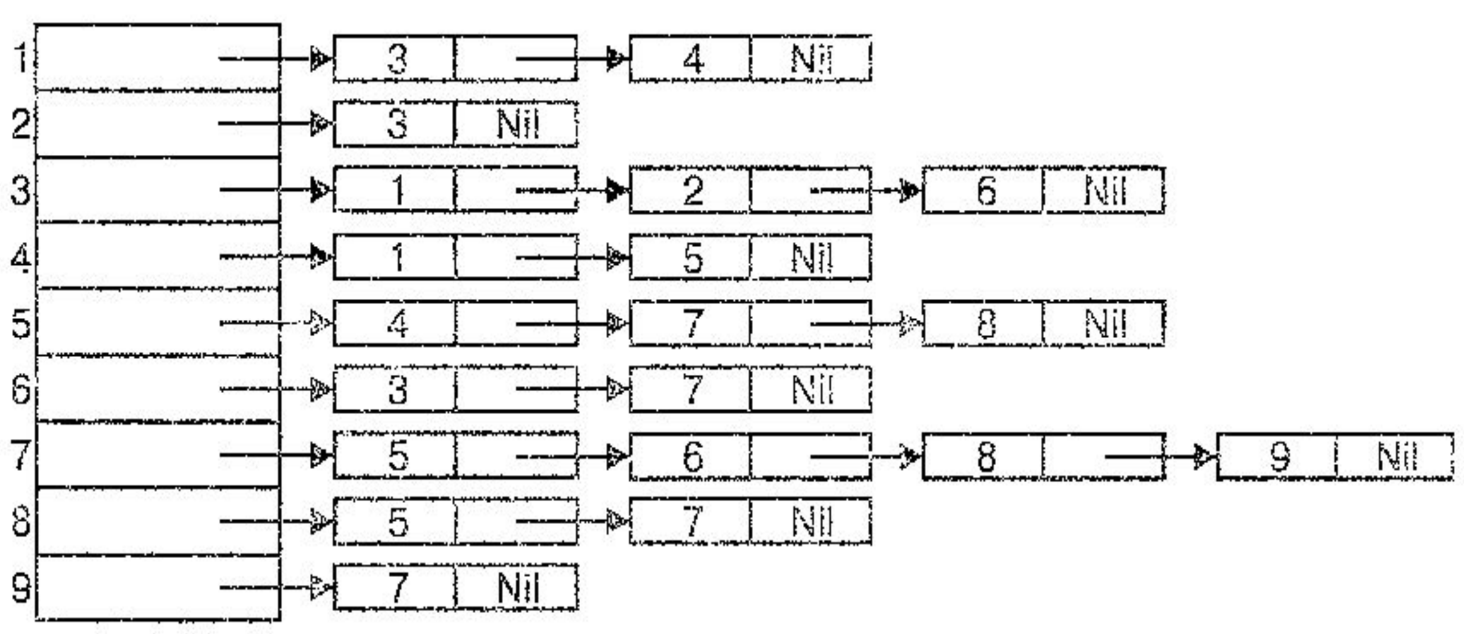
- Матрица инцидентности – это двумерный массив размерности n x m, в котором указываются связи между инцидентными элементами графа (ребро и вершина). Столбцы матрицы соответствуют ребрам, строки – вершинам. Ненулевое значение в ячейке матрицы указывает связь между вершиной и ребром. Данный способ является самым емким для хранения, но облегчает нахождение циклов в графе.

Представление графа в памяти компьютера



	a	b	c	d
(a, a)	2	0	0	0
(a, b)	0	5	0	0
(a, d)	0	0	0	8
(b, c)	0	0	7	0
(b, d)	0	0	0	9
(c, d)	0	0	0	4
(d, c)	0	0	3	0

Представление графа в памяти компьютера



Поиск в графе

Существует много алгоритмов на графах, в основе которых лежит систематический перебор вершин графа, такой что каждая вершина просматривается (посещается) в точности один раз. Поэтому важной задачей является нахождение хороших методов поиска в графе.

Поиск в графе

- Под обходом графов (поиском на графах) понимается процесс систематического просмотра всех ребер или вершин графа с целью отыскания ребер или вершин, удовлетворяющих некоторому условию.

Поиск в графе

- При решении многих задач, использующих графы, необходимы эффективные методы регулярного обхода вершин и ребер графов. К стандартным и наиболее распространенным методам относятся:
- поиск в глубину (Depth First Search, DFS);
- поиск в ширину (Breadth First Search, BFS).

Поиск в графе

- **При поиске в глубину посещается первая вершина, затем необходимо идти вдоль ребер графа, до попадания в тупик. Вершина графа является тупиком, если все смежные с ней вершины уже посещены. После попадания в тупик нужно возвращаться назад вдоль пройденного пути, пока не будет обнаружена вершина, у которой есть еще не посещенная вершина, а затем необходимо двигаться в этом новом направлении.**

Поиск в графе

- Процесс оказывается завершенным при возвращении в начальную вершину, причем все смежные с ней вершины уже должны быть посещены.
- Таким образом, основная идея поиска в глубину – когда возможные пути по ребрам, выходящим из вершин, разветвляются, нужно сначала полностью исследовать одну ветку и только потом переходить к другим веткам (если они останутся нерассмотренными).

Поиск в графе

Алгоритм поиска в глубину

Шаг 1. Всем вершинам графа присваивается значение не посещенная. Выбирается первая вершина и помечается как посещенная.

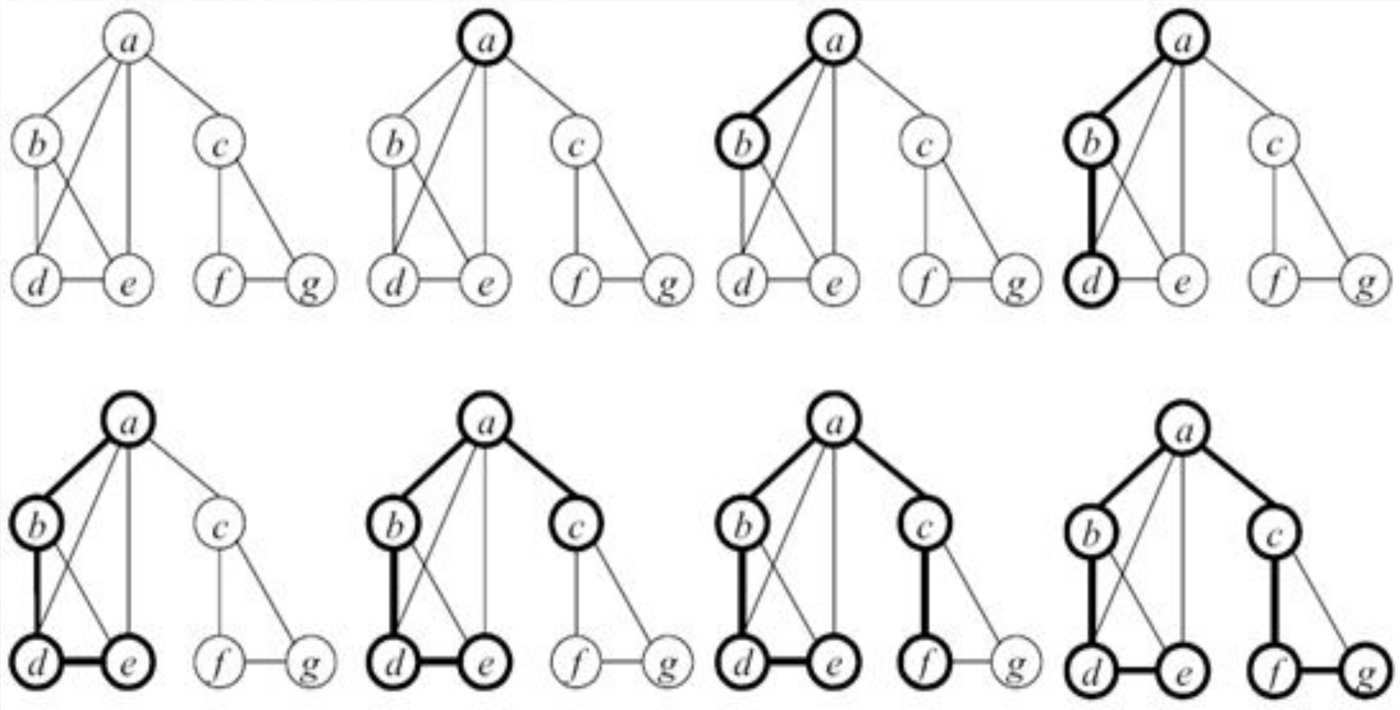
Шаг 2. Для последней помеченной как посещенная вершины выбирается смежная вершина, являющаяся первой помеченной как не посещенная, и ей присваивается значение посещенная. Если таких вершин нет, то берется предыдущая помеченная вершина.

Поиск в графе

Шаг 3. Повторить шаг 2 до тех пор, пока все вершины не будут помечены как посещенные.

Также часто используется нерекурсивный алгоритм поиска в глубину. В этом случае рекурсия заменяется на стек. Как только вершина просмотрена, она помещается в стек, а использованной она становится, когда больше нет новых вершин, смежных с ней.

Поиск в графе



Поиск в графе

- //Описание функции алгоритма поиска в глубину
- void Depth_First_Search(int n, int **Graph, bool *Visited,
- int Node){
- Visited[Node] = true;
- cout << Node + 1 << endl;

Поиск в графе

- for (int i = 0 ; i < n ; i++)
- if (Graph[Node][i] && !Visited[i])
- Depth_First_Search(n,Graph,Visited,i);
- }

Поиск в графе

Поиск в ширину

При поиске в ширину, после посещения первой вершины, посещаются все соседние с ней вершины. Потом посещаются все вершины, находящиеся на расстоянии двух ребер от начальной. При каждом новом шаге посещаются вершины, расстояние от которых до начальной на единицу больше предыдущего.

Поиск в графе

- Чтобы предотвратить повторное посещение вершин, необходимо вести список посещенных вершин. Для хранения временных данных, необходимых для работы алгоритма, используется очередь – упорядоченная последовательность элементов, в которой новые элементы добавляются в конец, а старые удаляются из начала.

Поиск в графе

- Таким образом, основная идея поиска в ширину заключается в том, что сначала исследуются все вершины, смежные с начальной вершиной (вершина с которой начинается обход). Эти вершины находятся на расстоянии 1 от начальной. Затем исследуются все вершины на расстоянии 2 от начальной, затем все на расстоянии 3 и т.д. Обратим внимание, что при этом для каждой вершины сразу находятся длина кратчайшего маршрута от начальной вершины.

Поиск в графе

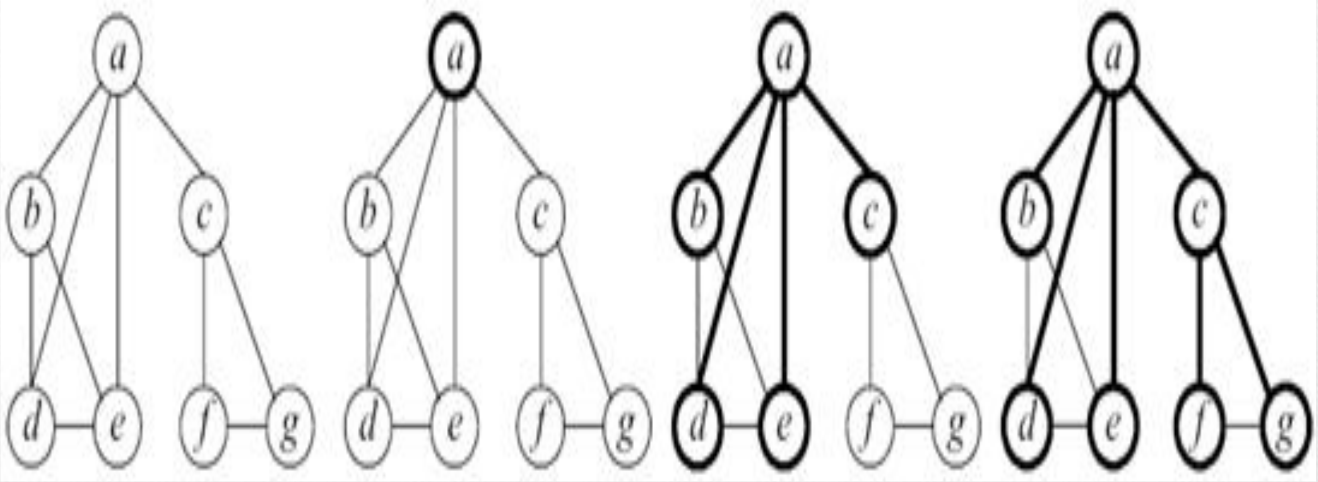
Алгоритм поиска в ширину

Шаг 1. Всем вершинам графа присваивается значение не посещенная. Выбирается первая вершина и помечается как посещенная (и заносится в очередь).

Шаг 2. Посещается первая вершина из очереди (если она не помечена как посещенная). Все ее соседние вершины заносятся в очередь. После этого она удаляется из очереди.

Шаг 3. Повторяется шаг 2 до тех пор, пока очередь не пуста

Поиск в графе



Поиск в графе

```
//Описание функции алгоритма поиска в ширину
void Breadth_First_Search(int n, int **Graph,
                          bool *Visited, int Node){
 int *List = new int[n]; //очередь
 int Count, Head;       // указатели очереди
 int i;
```

Поиск в графе

```
// начальная инициализация
for (i = 0; i < n ; i++)
  List[i] = 0;
Count = Head = 0;
```

Поиск в графе

- // помещение в очередь вершины Node
- List[Count++] = Node;
- Visited[Node] = true;
- while (Head < Count) {
- //взятие вершины из очереди
- Node = List[Head++];
- cout << Node + 1 << endl;

Поиск в графе

- // просмотр всех вершин, связанных с вершиной Node
- for (i = 0 ; i < n ; i++)
- // если вершина ранее не просмотрена
- if (Graph[Node][i] && !Visited[i]){
- // заносим ее в очередь
- List[Count++] = i;
- Visited[i] = true;
- }
- }
- }