图(graph)

2019 年 3 月 18 日

1 图（graph）

图是由顶点（vertex）的集合及顶点间的关系（边，edges）的集合组成的一种数据结构。不考虑自环（*self loop*）或多重边（多重图，*multigraph*）。

1.1 术语和概念

有向图(directed graph) 图的边有方向，A到B和B到A不同。边用尖括号即$< A, B >$，$< B, A >$表示

无向图(undirected graph) 图的边没有方向，A到B与B到A是一条边，边用圆括号即（A，B）表示

完全图(complete graph) 任意两个顶点之间都有边的图。无向图有$\frac{n(n-1)}{2}$条边；有向图有$n(n-1)$条边

权(weight) 边上附加的权重系数

邻接顶点(adjacent vertex) 一条边连接的两个顶点互为邻接顶点

顶点的度(degree) 与某顶点相关联的边的个数，称为度

路径(path)和路径长度(path length) 从点A到点B期间经过的点的序列（或边的序列）称为路径。路径长度为期间所有边的长度之和

简单路径 路径上各顶点无重复（无回环，cycle）出现

连通图(connected graph)和连通分量(connected component) 在无向图中，若顶点A到B有路径，则称为连通；若图中任意两点均连通，则称连通图；非连通图的最大连通子图称为连通分量

强连通图(strongly connected digraph)和强连通分量 有向图中，任意两点均存在路径（从A到B和从B到A 两条），则称此图为强连通图；非强连通图的最大强连通子图叫做强连通分量

生成树(spanning tree) 包含图中所有顶点，且有尽可能少的边

2 抽象数据类型接口（ADTI）

图由一系列顶点表示，每条边由一对顶点表示。图除需要构造析构函数外，需要有插入、删除顶点函数，插入、删除边函数，获取邻接顶点函数。

```
explicit Graph();
~Graph();
bool InsertVertex(T vertex);  // 插入顶点
bool RemoveVertex(int index); // 删除顶点

bool InsertEdge(int index1, int index2, E weight);
bool RemoveEdge(int index1, int index2);
bool Display();
```

3 数据类型的实现：邻接矩阵(adjacency matrix)

图常用的存储方法有三种，邻接矩阵（*adjacency matrix*），邻接表（*adjacency list*），邻接多重表（*adjacency multilist*）。

这里只介绍邻接矩阵和邻接表两种方法，分别和其他数据结构的数组实现方法和链表实现方法对应。

邻接矩阵 邻接矩阵是用于表示各个顶点之间关系的矩阵。若两个顶点（a，b）间存在边，则对应的元素（a，b）不为零。

对于有向图，一般使用列坐标（第几行）表示边的起点；行坐标（第几列）表示边的终点。

$$E_{ij} = \begin{cases} 1 & if\ (i,j) \in E\ or < i,j > \in E \\ 0 & else \end{cases}$$

当图是无向图，对应的邻接矩阵为角对称矩阵。同时，某行（列）元素之和，是从改点出（入）射的边的个数，也称之为度。

对于有向图，某行元素之和，是该顶点出射的度；某列元素之和，是该顶点入射的度。

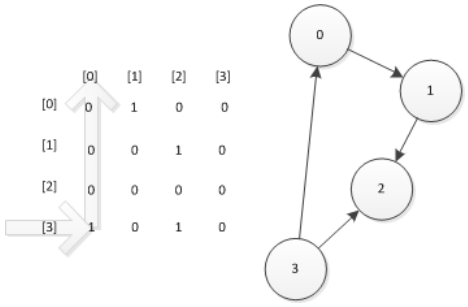

图 1: 有向图的邻接矩阵

因为不考虑自环情况，所以对角线元素均为0。

如果是带权图，不可以继续使用0表示不连接，以防和权值为0混淆。可以用无限∞表示顶点间不连接，连接的话使用具体权重值w。但是因为对角线元素没有意义，所以可以继续使用0。

构造函数 使用邻接矩阵存储图，需要使用一个顺序表（一维数组），用于存放顶点的信息；使用一个二维数组，存放顶点之间的连接关系（边）。在构造图类时，需要根据最大顶点数，分配一个一维数组，和一个二维数组。

其中给定一个指针E**，开辟一个二维数组空间，首先需要开辟一个一维数组，其元素类型为E*；随后对该一维数组的每个元素依次开辟类型为E的一维数组，形成二维数组。

可寻址范围为：

一维数组*[0,vertex_number)*，二维数组*[0,vertex_number)*[0,vertex_number)*

顶点和边的插入 即给数组元素变量赋值

顶点的删除 顶点的插入可以在最后位置插入；删除操作需要将最后一个元素覆盖到待删除元素位置，之后删除最后一个元素。所以删除一个顶点只需要将一维数组和二维数组中最后一个元素，覆盖到待删除结点处，并将记数变量减一，等同于尾删除。

除了删除顶点，还需删除顶点和其他顶点的连接关系。当一顶点被删除后，无论是入射到该点的边，还是从该点出射的边，均要断掉。所以需要清空与该点有关的行、列元素，即先将最后一行和最后一列覆盖到当前顶点所在的行和列，随后将计数变量减一，尾删除。

边的删除 清空边数据，并自减计数变量。

寻找邻接顶点 邻近顶点指该顶点指向的顶点，所以在邻接矩阵中搜索行向量。

邻接矩阵存储图的方式，优点是寻址数据更高效，但当矩阵为稀疏矩阵时，空间利用率低。可使用邻接表的存储方式。

4 数据类型的实现：邻接表(adjacency list)

使用邻接表的存储方式，需要两种结点结构：顶点结点和边结点。

使用领接表表示图，有两种方法：出边表和入边表。“出边”就是当前顶点的指针域中，记录从自己射出到其他顶点的边；“入边”就是在指针域中记录从其他顶点射到当前顶点的边。

其（出边表）基本原理是，顶点结点在一个顺序表中存储。每个顶点结点中的边结点指针域指向“以自己为起点的边的边结点”，而这个边结点的指针域指向“以该结点出射的另一条边的边结点”。即，一个顺序表存储顶点，每个顶点依次链接着以自己为出射结点的边的链表，而边结点中存放着对应的终点结点的id。

所以顶点结点有两个域，分别存放数值和边结点的指针。

边结点中含有三个域，分别是1）自己终点顶点结点的id；2）边的权值；3）指向下一条边结点的指针。

入边表：每个顶点结点链接着“以自己为终点”的边结点。

图的遍历（graph traversal） 图的遍历就是从某个顶点出发，访遍所有顶点且所有顶点只访问一遍。

可分为深度优先搜索（*DFS, depth first search*）和广度优先搜索（*BFS, breadth first search*）。

如果图中存在回路，为了避免重复访问，需要对每个顶点设置一个标志域，用于标志该顶点是否被访问过。

深度优先（DFS, depth first search） 其流程是：1）从当前顶点开始访问（起点），对访问标志域赋值；2）在当前结点所有邻接顶点中，找出未访问的一个顶点，进行访问；3）如果当前结点所有顶点均被访问过，则后退一步；4）重复上述过程，直至起始顶点的所有邻接顶点均被访问过。

广度优先（BFS, breadth first search） 其流程是：1）从当前顶点开始访问，对访问标志域赋值；2）依次访问当前顶点的所有邻接顶点；3）再依次访问邻接顶点的各个未访问的邻接顶点；4）重复上述过程，直至所有顶点均被访问。

连通分量（connected component） 当图为**非连通图**时，从某顶点出发**无法**遍历图的所有顶点，但可以遍历到最大连通子图构成的一个连通分量。

对于非连通图，获取图的连通分量时，可以先从某顶点开始对图进行遍历，至此便可以获知一个连通分量。随后再从另一个之前未被访问的顶点开始对图进行遍历，便可以获知另一连通分量。

在连通分量中，所有顶点的集合和遍历时经过的边可以构成一棵树，称之为生成树。

最短路径（shortest path） 从带权图的某一顶点（源点）出发，寻找一条通往另一顶点（终点）的最短路径。