邻接矩阵与邻接表实现

邻接矩阵

邻接矩阵基本介绍

图的邻接矩阵（Adjacency Matrix）存储方式是用两个数组来表示图。一个一维的数组存储图中顶点信息，一个二维数组（称为邻接矩阵）存储图中的边或弧的信息

邻接矩阵是一种常用地用来描述图的方法

邻接矩阵实现方案

根据上述邻接矩阵的定义，可以给出邻接矩阵的实现思路

- 先使用一维数组储存所有的顶点信息
  - 描述顶点信息可以使用一个结构体一维数组实现
  - 在结构体数组中加入相关数据域完成对节点信息的描述
- 再使用二维数组来描述所有顶点之间的关系
  - 二维数组相当于一个矩阵，矩阵中的值为 0 或 1
  - 为了保证二维数组的可扩展性，我们可以采用 STL 中的 vector 数据结构来实现

> 我们还可以考虑新的实现方式：直接在描述顶点信息的结构体中嵌套和其它顶点关系的`bool`数组，事实上，本作业代码即采取这种方式

邻接表

邻接表基本介绍

邻接表数据结构的出现事实上是因为邻接矩阵对于存储空间的浪费而发明的

事实上邻接表可以理解为一种二维数组，第一维代表顶点的个数，而第二位的长度实际上取决于每个顶点的“邻居”的个数

邻接表实现方案

根据上面的定义，我们很容易给出邻接表的实现方案

- 先使用一维数组储存所有的顶点信息
  - 描述顶点信息可以使用一个结构体一维数组实现
  - 在结构体数组中加入相关数据域完成对节点信息的描述
- 再使用链表来描述所有顶点之间的关系
  - 链表可以调用 STL 库中的 list 数据结构来实现
  - 链表可以直接内嵌在各节点的结构体中，这种写法会使数据结构的使用更加抽象，但是会更加符合逻辑。本次作业即采用这种方法

两种数据结构的选择策略

邻接矩阵的优点是可以快速判断两个顶点之间是否存在边，可以快速添加边或者删除边。而其缺点是如果顶点之间的边比较少，会比较浪费空间。

而邻接表的优点是节省空间，只存储实际存在的边。其缺点是关注顶点的度时，就可能需要遍历一个链表。另一个缺点是，对于无向图，如果需要删除一条边，就需要在两个链表上查找并删除。

因此，我们对邻接表和邻接矩阵的选择策略是：

- 当为稀疏图时，选择邻接表，减少空间浪费
- 当为稠密图时，选择邻接矩阵，选择邻接矩阵，加快相关操作的速度

程序清单

本次作业根据上述方案实现了比较简易的邻接表和邻接矩阵类，支持增加节点和删除节点的操作，可以呈现大体的数据结构设计思路

文件路径：

- 邻接矩阵：`/coding/code5/AdjacencyMatrix.cpp`
- 邻接表：`/coding/code5/AdjacencyTable.cpp`