图的多种存储方法

刚好最近课程进行到了图论，而图论最首要的就是存图。所以总结一下我所知道的一些图的存储方法。

邻接矩阵

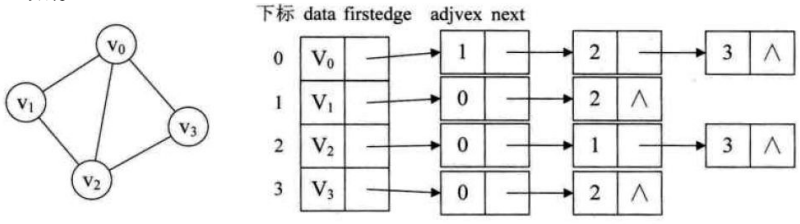
相当基础的存储方法。
图的构成无非点与边，而边则是由两个点连成，因此我们可以用n*n□n为点数）的方阵来存储图。
在n*n的方阵中□F(i,j)记录i结点和j结点之间的联通关系。
对于无权图，邻接矩阵表示如下：

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

其中0表示两点不连通，1表示两点连通。
不难发现对于无向图，邻接矩阵是沿对角线对称的。而有向图不一定。
对于有权图而言，可以用F(i,j)记录权值。
邻接矩阵的优点是对于特定的两点，只要O(1)的时间就可以判断连通关系。缺点是对于结点较多的稀疏空间浪费太大，且需要O(n)时间来遍历一个点连通结点。

邻接表

邻接表在邻接矩阵的基础上进行了优化（某些方面）。
我们发现邻接矩阵的空间浪费主要来源于大量的不连通点。既然如此，我们可以只记录与一个点相连的那些点。



如上图，采用链表的方式来存储边，对于有权图可以多开一个权值变量。
优点是存储空间占用较小，遍历起来方便。缺点是对于给定两点的连通关系无法直接判断。

边集数组

以上两种都是以点为核心的图存储方式，在大多数情况能发挥很大用处。不过当边数很多，且题目以边为核心时，我们有另一种存储方法。
事实上这个存储方法非常接近图的输入方式，用两个一维数组存储，一个记录点（如果按1,2,...n顺序的话其实也不用记），另一个记录边，这个边数组的每个元素由三部分构成：起点、终点、权值。
发现其实按着输入输完就建完了。

前向星

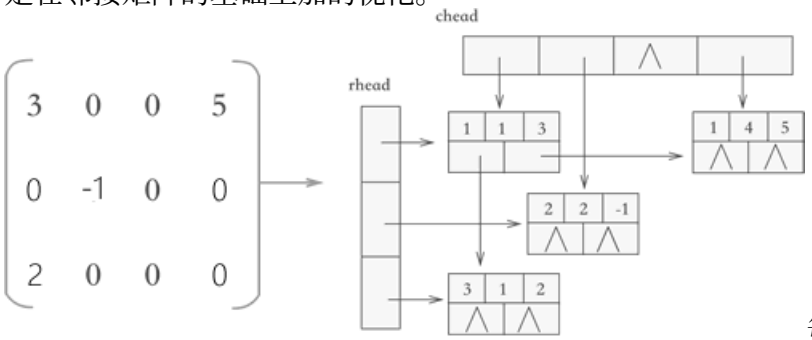
有的文章会把前向星归类为邻接表的优化，不过我个人觉得它在形式上跟接近边集数组。
和边集数组的区别就是在边集数组的基础上按起点进行了排序，像是一个“ 竖着排 ”的邻接表。
然后需要一个一维数组来记录某个点为起点的所有边在数组中的起始位置。
用起来和邻接表差不多。但因为需要做一遍排序而且插入删除边很麻烦，因此只能进行一些离线的操作。

链式前向星

针对前向星的种种缺点进行了优化。
形象点的理解就是“ 乱糟糟 ”的邻接表。
想要在边集数组的基础上遍历同一个起点又不想排序怎么办？只要加一个next模拟指针让一条边指向与它同起点的下一条边就行了。
也就是在无序的边集数组里进行一些连接操作。
具体操作就是构建一个head数组，这个数组记录以i为起点的最后输入的那条边在边集数组中的位置。
我们可以看出在读入过程中这个head是会不断更新的。正好，读到一条以i为起点的边时，让它指向当前的head[i][](next[当前的边]=head[i])
嗯，其实这个连接顺序是与读入顺序相反的。
链式前向星的空间占用小，而且删除操作也比较简单（只要更改next就行了）

十字链表

是在邻接矩阵的基础上加的优化。



每个元素包含五个部分:起点、终点、权值、右指针、下指针。