数据结构和算法

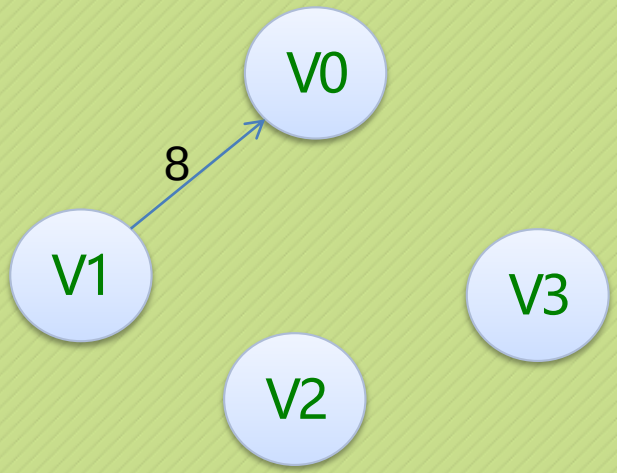
作者：小甲鱼

让编程改变世界

Change the world by program

邻接表（无向图）

- 邻接矩阵看上去是个不错的选择，首先是容易理解，第二是索引和编排都很舒服~
- 但是我们也发现，对于边数相对顶点较少的图，这种结构无疑是存在对存储空间的极大浪费。



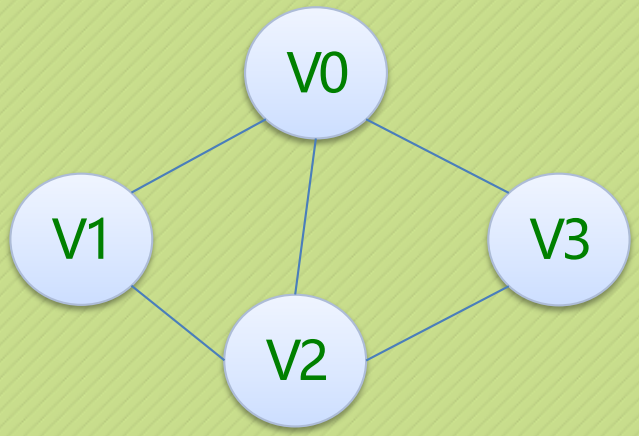
顶点数组：	V0	V1	V2	V3

	V0	V1	V2	V3
V0	0	∞	∞	∞
V1	8	0	∞	∞
V2	∞	∞	0	∞
V3	∞	∞	∞	0

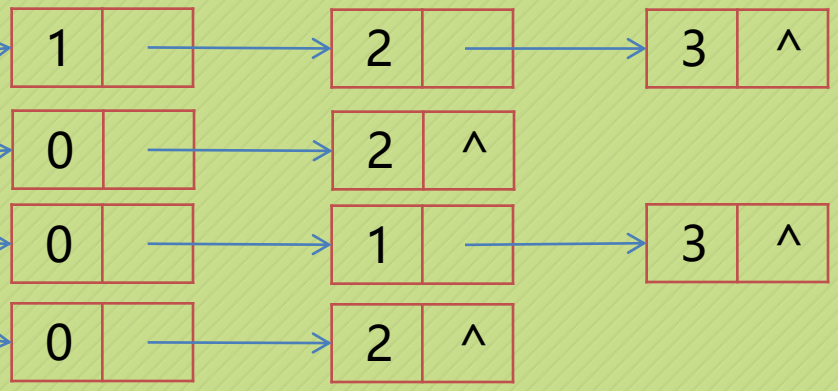
邻接表（无向图）

- 因此我们可以考虑另外一种存储结构方式，例如把数组与链表结合一起来存储，这种方式在图结构也适用，我们称为邻接表(AdjacencyList)。
- 邻接表的处理方法是这样：
  - 图中顶点用一个一维数组存储，当然，顶点也可以用单链表来存储，不过数组可以较容易地读取顶点信息，更加方便。
  - 图中每个顶点Vi的所有邻接点构成一个线性表，由于邻接点的个数不确定，所以我们选择用单链表来存储。

邻接表（无向图）

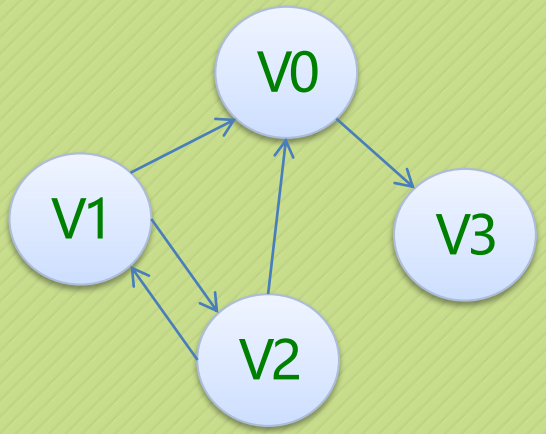


下标	data	first
0	V0	
1	V1	
2	V2	
3	V3	

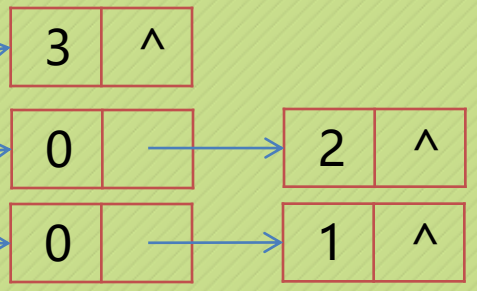


邻接表（有向图）

- 若是有向图，邻接表结构也是类似的，我们先来看下把顶点当弧尾建立的邻接表，这样很容易就可以得到每个顶点的出度：

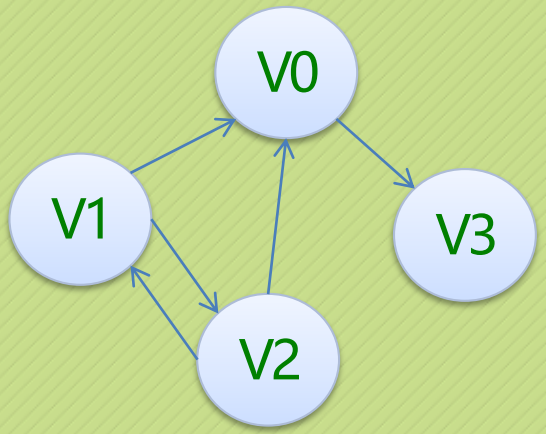


下标	data	first
0	V0	
1	V1	
2	V2	
3	V3	^

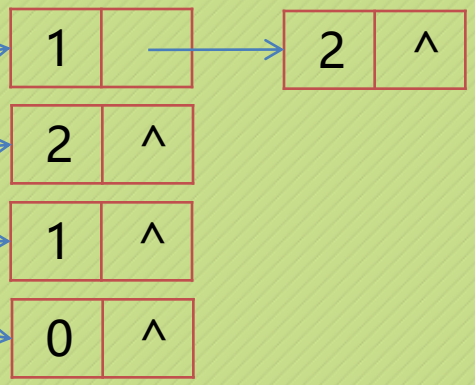


邻接表（有向图）

- 但也有时为了便于确定顶点的入度或以顶点为弧头的弧，我们可以建立一个有向图的逆邻接表：



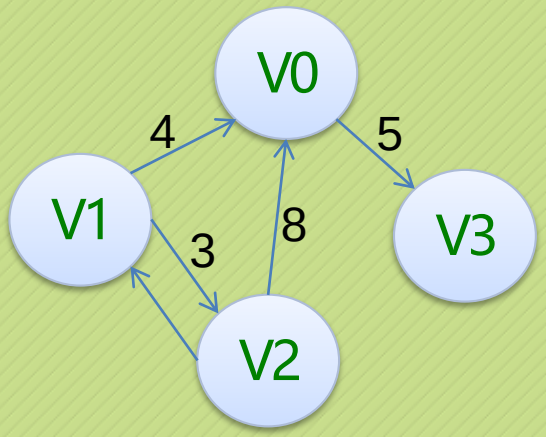
下标	data	first
0	V0	
1	V1	
2	V2	
3	V3	



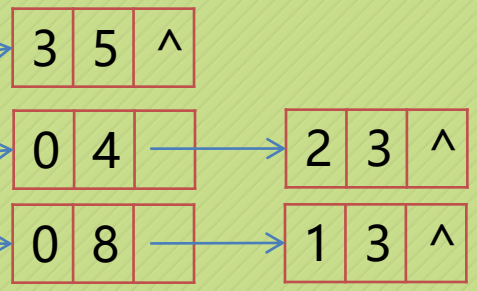
- 此时我们很容易就可以算出某个顶点的入度或出度是多少，判断两顶点是否存在弧也很容易实现。

邻接表（网）

- 对于带权值的网图，可以在边表结点定义中再增加一个数据域来存储权值即可：



下标	data	first
0	V0	
1	V1	
2	V2	
3	V3	^



代码实现

- 作为一个课后作业给大家自己锻炼下，小甲鱼提供的参考答案仅供参考借鉴！