图（Graph）

回顾逻辑结构——图形结构：

结构中的数据元素之间存在多对多的关系。其特点是每个元素的前驱或后继的个数是任意的。因此结构中可能没有或存在多个开始元素或终端元素。

定义：

图（graph）由两个集合V（vertex）和E（edge）组成，记为G = (V, E)。其中V是顶点的有限集合，记为V（G）；E是**连接**V中顶点对的边的有限集合，记为E（G）。

E（G）可以为空集，此时图G只有顶点

在图G中，如果表示边的序偶是有序的，则称G为有向图（digraph）。有向图的边用 <i , j > 表示，称为有向边；

如果E（G）是对称的，即有<i , j> 必有 <j , i>，则用 (i , j)或(j , i)代替这两个顶点对，称为无向边，称G为无向图（undirgraph）。无向图是有向图的特例

图的基本术语

1. 端点和邻接点

--

在一个无向图中，若存在一条边（i，j），则称顶点i和j为该边的两个端点（endpoint），这两个端点互称邻接点（adjacent），边（i，j）和顶点i、j关联。

在一个有向图中，若存在有向边<i，j>，则此边是i的一条出边、j的一条入边；i为此边的起点、j为终点；j为i 的出边邻接点、i为j 的入边邻接点。

2. 顶点的度、入度和出度

在无向图中，顶点所关联的边的数目称为**顶点的度（degree）**。

在有向图中，顶点的度分为入度和出度，以顶点为终点的边数目称为顶点的**入度**（indegree），以顶点为起点的边数目称为顶点的出度（outdegree），入度与出度的和称为顶点的度。

度数与边数的关系：

一个图中**所有顶点的度数之和是边数的两倍**。因为每条边分别作为邻接点的度各计一次。

3. 完全图

1）无向图中任意两个顶点之间都存在一条边，共n（n-1）/2条边

2）有向图中任意两个顶点之间都存在方向相反的两条边，共n（n-1）条边

4. 稠密图和稀疏图

当一个图接近完全图的时候，称为稠密图（dense degree）。反之，称为稀疏图。

5. 子图

假设有图G=（V，E）中，存在图G’=(V’, E’)满足V’（G’）是V（G）的子集，且E’是E的子集，则称G’是G的子图。注意：子图一定是一个图

6.路径和路径长度

路径（path）是开始点到结束点的一个顶点序列，路径长度（path length）是路径上经过的边数目。与无向图不同，有向图路径上的边用序偶表示。

简单路径（simple path）：除开始点和结束点可以相同以外，其余顶点均不相同。

7. 回路/环

开始点与结束点相同的路径称为回路或环（cycle）。符合本定义的简单路径称为简单回路或简单环（simple cycle）。

8. 连通、连通图和连通分量

在无向图G中，若顶点i和j之间有路径，则称它们是连通的。

若图G中任意两个顶点都是连通的，则称G为连通图（connected graph）。

无向图G中的极大（集合V和E尽可能的大）连通子图称为G的连通分量。

显然，连通图只有一个连通分量，即本身

9. 强连通图和强连通分量

在有向图G中，若从顶点i到j有路径，则称从顶点i到j是连通的。

若图中任意两个顶点都有到彼此之间的路径，则称G为强连通图（strongly connected graph）。注意：强连通图不一定是有向完全图。

有向图G中的极大连通子图称为G的强连通分量（strongly connected component）。

显然，强连通图只有一个强连通分量即本身。

10. 权和网

图中的每一条边都可以附有一个数值，这种与边相关的数值称为**权**，边上带权的图称为带权图（weighted graph）， 也称作网（net）

定理：

n个顶点的强连通图至少有n条边。

属于连通图的有：

无向完全图、无向环；

属于强连通图的有：

有向完全图、有向环。

图的存储结构和基本运算算法

存储结构：

邻接矩阵（adjacency matrix）数组

邻接表（adjacency list）

图的遍历

图的遍历的概念：

从图中的某个顶点出发，沿着图的边访问所有顶点，每个顶点仅访问一次。

如果给定图是连通的或强连通的，则遍历过程**一次**就能完成 ？ 一次是什么意思？

答案：即选定某个起始点就能遍历完整个图，不需要再选另一新起始点重新进行遍历。

辅助存储结构：

访问标记数组

图的遍历方法有两种：

深度优先遍历（Depth First Search，DFS）：访问初始顶点，再依次对没被访问过的邻接点进行深度优先遍历，直到与初始顶点**相邻**的顶点都被访问过。

广度优先遍历（Breadth First Search，BFS）：访问初始顶点v，然后访问每一个未被访问过的邻接点，然后再将刚刚访问过的邻接点作为初始顶点，依次访问它每的一个未被访问过的邻接点，以此类推，直到图中所有和初始点v有路径**相通**的顶点都被访问过为止。

保存临时数据的两种容器：栈、队列