# 图

概念：图由两个集合V和E组成，记为G=(V,E)

V(G):顶点的有限集合

E(G)：连接V中两个不同顶点的边的有效集合

(V,E)代表无向图

<V,E>代表有向图

度,出度，入度

在无向图中，每个顶点的度为端点的边数

在有向图中，顶点的度=入度+出度

入度：以该顶点为终点的入边数目

在一个无向图中，若存在一条边(i,j)，则称顶点i、j为该边的两个端点，并称它们互为邻接点。

在一个有向图中，若存在一条边<i,j>，则称此边是顶点i的一条出边，同时也是顶点j的一条入边，称顶点i和j分别为此边的起始端点(起点)和终止端点(终点)。

1、在无向图中，每个顶点的度为端点的边数；

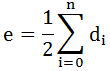
2、在有向图中，顶点v的度=入度+出度

(1)入度：以该顶点为终点的入边数目；

(2)出度：以该顶点为起点的出边数目；

边数、顶点数之间的关系

图(有向图或无向图)中有n个顶点、e条边，每个顶点的度为di(0≤i≤n-1)：



结论：一个图中所有顶点的度之和等于边数的两倍。

(图中每条边被两个邻接点的度重复计算)

子图

设有两个图G=(V,E)和G‘=(V’,E‘)，若V’是V的子集，即V‘ 属于V，且E'是E的子集，即E‘ 属于E，则称G'是G的子图。

完全无向图、完全有向图

1、完全无向图：具有n(n-1)/2条边的无向图；

2、完全有向图：具有n(n-1)条边的有向图。

简单路径：路径中顶点序列中的顶点不重复出现

回路(环)：路径中的开始点和结束点为同一点

简单回路(简单环)：

回路中除开始点、结束点相同外，其余顶点不同

连通、连通图、连通分量：

1、无向图中的两顶点有路径，称两顶点是连通的；

2、连通图：图中的任意两顶点都是连通的；

3、非连通图：图中至少有一对顶点不是连通的；

4、连通分量：无向图中的极大连通子图。

强连通图、强连通分量：

1、强连通图：有向图中的任意两顶点都是连通的；

3、强连通分量：有向图中的极大连通子图。

权、网：

1、边的权：边上标注的具有某种含义的数值；

3、网(带权图)：边上带权的图。

约定：边的权为非负数。

生成树、生成森林

假设一个连通图有 n 个顶点和 e 条边，其中 n-1 条边和 n 个顶点构成一个极小连通子图，称该极小连通子图为此连通图的生成树。

对非连通图，则称由各个连通分量的生成树构成的集合为此非连通图的生成森林。

图的基本运算

① 建立图CreateGraph(G)：建立图G的某种存储结构。

② 销毁图DestroyGraph(G)：释放图G占用的内存空间。

③ 输出图DispGraph(G)：显示图G的结构。

④ 添加顶点AddaVex(G)：在图G中添加顶点，编号自动累加求得。

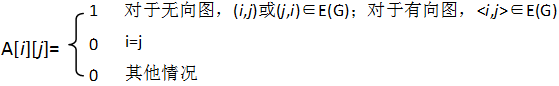
⑤ 插入边InsertEdge(G,u,v,w)：在图G中插入边<u,v>，其权值为w。

⑥ 求顶点的度Degree(G,v)：求图G中顶点v的度。

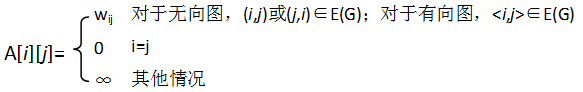
邻接矩阵：表示顶点间相邻关系的矩阵

1、已知图G=(V,E) ，顶点编号依次为0、1、…、n-1，

则G的邻接矩阵是n阶方阵A：



1. 已知图G=(V,E) 是带权图或网，其中wij为边(i,j)或<i,j>的权，则G的邻接矩阵A



邻接表：图的链式存储结构

1、每个顶点建立一个带头结点的单链表，用来串接该顶点的所有邻接点；

2、所有的头结点构成一个数组，称为头结点数组(adjlist[])；

3、第i个单链表adjlist[i]中的结点表示依附于顶点i的边；

4、单链表中的结点由3个域组成：

(1)顶点域adjvex：指示该邻接点在头结点数组中的下标；

(2)权值域weight：存放对应边的权值；

(3)指针域nextarc：指向依附于顶点i的下一条边对应结点。

约定：(1)不带权图的weight域为1，

(2)带权图的weight为边的权值。

图的遍历

从图G=(V,E)中的任意顶点v出发，访问图G中的所有顶点，每个顶点仅被访问一次。

说明：图的遍历要比树的遍历复杂。

建议：为避免同一顶点被多次重复访问，设立辅助数组visited[] ，初值为0，表示顶点没被访问过，一旦顶点i被访问，将visited[i]设置为1。

深度优先遍历(Depth First Search：DFS)

1、任选一个顶点v，并访问v;

2、从顶点v的邻接点中选择一个没有被访问过的邻接点w，从顶点w出发深度优先遍历图；

3、重复2，直到图中所有与v有路径相通的顶点都被访问。

说明：深度优先遍历类似树的先根遍历

DFS思路：

一步一步向前走，当没有可走的相邻顶点时便回退。



广度优先遍历(Breath First Search：BFS)

1、首先访问初始点vi，并将其访问标记设置为1；

2、接着访问顶点vi所有未被访问过的邻接点 vi1, vi2, …,vit ，并将访问标记均设置为1；

3、再按照 vi1, vi2, …, vit 的次序访问每一个顶点所有未被访问过的邻接点，并将访问标记均设置为1；

4、直到图中所有与初始点vi有路径相通的顶点都被访问。

说明：广度优先遍历类似树的层次遍历

顺序一致：用队列实现

无向图的生成树

无向连通图G的全部顶点和部分边构成的子图G’满足：

(1)子图G’中的所有顶点连通；

(2)子图G’中没有回路；

称子图G’是原图G的一棵生成树。

无向图的生成树：由遍历得到

(1) 连通图

遍历一次连通图所经过的边集、所有顶点的集合就构成了该图的一棵生成树(不唯一)；

(2)非连通图(由多个连通分量构成)

从每个连通分量的任一顶点出发进行遍历，可得该连通分量的顶点集，每个连通分量产生的生成树的集合构成整个非连通图的生成树。

说明：

1. 深度优先生成树：深度优先遍历
2. 广度优先生成树：广度优先遍历产生的生成树。

最小生成树：各边的权之和最小的生成树

是一棵树：

(1)无回路(2) V个顶点有V-1条边。

是生成树：

(1)包含全部顶点(2) V-1条边都在图里。

最小生成树的构造算法

(1) 普里姆算法：构造性算法

(2) 克鲁斯卡尔(Kruskal)算法：

按权值递增次序选择边来构造最小生成树。

普里姆(Prim)算法：构造法

1、假设：

①G=(V,E)是具有n个顶点的带权、无向、连通图；

② T=(U,TE)是G的最小生成树，U是T的顶点集，TE是T的边集。

2、从起始顶点 v构造最小生成树T的步骤：

(1)初始化U={v}。以v到其他顶点的所有边为候选边；

(2)重复以下步骤n-1次，使得其他n-1个顶点并入U中。

普里姆算法关键：在顶点集U、U-V之间选择最小边

1、辅助数组closest[n]

已知j∈V-U，closest[j]为该边在U中的顶点编号；

2、辅助数组lowcost[n]

已知j∈V-U，lowcost[j]存储该边的权值。

① 从候选边中挑选权值最小的边加入TE，设该边在V-U中的顶点是k，将k加入U中；

② 考察当前V-U中的所有顶点j，修改j的候选边：

如果(k,j) < j的候选边，用(k,j)取代j的候选边。