Chapter 8 图

基本概念

图：由顶点集合 V 和顶点间关系集合 E 组成的数据结构，分有向图和无向图

- 不考虑自环
- 不考虑两个顶点间有多条边相联
- 完全图：设图的顶点数为 n，则无向完全图边数为 $\frac{n(n-1)}{2}$，有向完全图边数为 $n(n-1)$
- 权：某些图中与边相关的数值。带权图也被称为网络
- 邻接顶点：有边相联的一对顶点
- 子图，设图$G=(V,E),G'=(V',E')$，若$V'\subseteq V,E'\subseteq E$，则称 G' 为 G 的子图
- 度：与顶点关联的边数，有向图分入度和出度
- 路径：顶点序列
- 路径长度：路径上边的数量
- 简单路径：指路径上顶点不重复
- 连通图与连通分量：无向图中若从顶点 v 到顶点 u 有路径，则称 v 与 u 连通，若图中任意一对顶点是连通的则称此图为连通图，非连通图的极大连通子图叫做连通分量
- 强连通：有向图中任意两顶点到彼此均有路径
- 生成树：极小连通子图

图的存储结构

邻接矩阵

将所有顶点的信息组织为顶点表，然后用一个矩阵表示顶点间的邻接关系（a[i][j] = 1 表示顶点 i 和 j 邻接）

无向图的邻接矩阵是对称的，第 i 行/列的和是顶点 i 的度

有向图第 i 行/列的和是顶点 i 的出度/入度

邻接矩阵在矩阵稀疏时存储利用率很低

邻接表

将邻接矩阵的各行组织为单链表，链表结点存储终顶点（带权图还需额外存储权值）

对于无向图，边链表的结点数就是度

对于有向图，可以额外存储逆邻接表，存储入边

邻接多重表

在处理边为主且每条边处理一次的实际应用中很有用

无向图的邻接多重表中每条边用一个边顶点表示，由五个域组成：

mark	vertex1	vertex2	path1	path2
标记是否已处理	顶点	顶点	依附于顶点1的下一条边	依附于顶点2的下一条边

有向图的类似，只不过按照十字链表组织

图的遍历

需要一个数组标记顶点是否已被访问

深度优先搜索

- 访问当前顶点，并标记为已访问
- 在该顶点的所有邻接顶点中寻找未访问的点
    - 找到未访问的点，将其作为当前顶点，继续深度优先搜索
    - 所有邻接顶点均已访问过，回退，将上一步的顶点作为当前顶点

对于邻接表存储的图，扫描边的时间为$O(e)$，每个顶点访问一次，时间复杂度为$O(v+e)$

对于邻接矩阵存储的图，查找一个顶点所有邻接的边的复杂度为$O(v)$，总的时间复杂度为$O(v^2)$

广度优先搜索

- 访问当前顶点并标记为已访问
- 顺序访问当前顶点的所有邻接顶点
- 顺序访问当前顶点的所有邻接顶点的所有邻接顶点
- ……

使用队列实现，邻接表表示的时间复杂度为$O(v+e)$，邻接矩阵表示的时间复杂度为$O(v^2)$

连通分量

对于非连通的无向图，使用深度优先搜索或广度优先搜索只能访问到出发顶点所在最大连通子图的所有顶点，即该图的一个连通分量。所有连通分量的生成树构成了该图的生成森林

最小生成树

生成树是原图的极大无环子图，即删去一边后不再连通，而加入一边后产生回路

对于带权图，不同的生成树对应的总权值不同，总权值最小的生成树为**最小生成树**

- 只能用网络中的边构造最小生成树
- 恰好有 $n-1$ 条边
- 选用的 $n-1$ 条边不构成回路

同一带权图可能有多个最小生成树，但**各边权值不同的图的最小生成树是唯一的**

Kruskal 算法

任给一个有 n 个顶点的连通网络 $N=\{V,E\}$

- 构造由 n 个顶点组成的不含边的图 $T=\{V,\emptyset\}$，每个顶点自成一个连通分量
- 从 E 中取出权值最小的边，若该边的顶点来自不同的两个连通分量，则将此边加入 T
- 重复直至所有顶点都在一个连通分量上

利用**最小堆**存放边，利用**并查集**查询两个顶点是否在同一个连通分量

对于邻接表，检查邻接表需要 $O(n+e)$ 的时间，构造最小堆需要 $O(elog_2e)$ 的时间，生成最小生成树时需要进行 $O(e)$ 次出堆，$2e$ 次 find 操作以及 $n-1$ 次 union 操作，时间复杂度分别为 $O(elog_2e)$，$O(elog_2n)$，$O(n)$，总的时间复杂度为 $O(n+elog_2e)$

对于邻接矩阵，检查邻接矩阵需要 $O(n^2)$ 的时间，构造最小堆需要 $O(elog_2e)$ 的时间，生成最小生成树时需要进行 $O(e)$ 次出堆，$2e$ 次 find 操作以及 $n-1$ 次 union 操作，时间复杂度分别为 $O(elog_2e)$，$O(elog_2n)$，$O(n)$，总的时间复杂度为 $O(n^2+elog_2e)$

平均时间复杂度为 $O(elog_2n)$

Prim 算法

任给一个有 n 个顶点的连通网络 $N=\{V,E\}$，将顶点集分为不相交的两部分 $V=V_{mst}\cup(V-V_{mst})$，其中 V_{mst} 是最小生成树的点集，

- 选出 $(u,v),u\in V_{mst},v\in V-V_{mst}$ 中权值最小的边，将 v 加入 V_{mst} ，将该边加入 E_{mst}
- 直到 V_{mst} 中有 n 个顶点，E_{mst} 中有 $n-1$ 条边

需要最小堆存储边，每次选出一个顶点在最小生成树，另一个顶点不在最小生成树的权最小的边退出堆，加入生成树，然后将新出现的所有一个顶点在最小生成树，另一个顶点不在最小生成树的边插入堆，重复 $n-1$ 次

算法迭代 $O(n)$ 次，每次平均插入 $\frac{2e}{n}$ 条边，删除 e 条边，故时间复杂度为 $O(elog_2e)$

最短路径

非负权值的单源最短路径

问题：给定一个有向带权图 D 和源点 v ，各边的权值非负，求从 v 到 D 其余各顶点的最短路径

Dijkstra 算法：

集合 S 存放已经求出最短路径的重点，初始时 S 中只有源点 v_0 ，每次求得一条最短路径则将终点加入 S ，直到 S 包含所有顶点。引入辅助数组 dist[]，dist[i]表示当前从 v_0 到 v_i 的最短路径的长度，初始时若 v_0 到 v_i 有边则为边的权值，否则为 ∞ 。引入辅助数组 path[]，path[i]表示最短路径上 v_i 的前一个顶点，初始时若 v_0 和 v_i 间有边则 path[i]=0，否则为 -1

- 求得一条最短路径 $dist[k]=min\{dist[i]\mid v_i\in V-S\}$
- 将其终点 v_k 加入 S ，对所有 $v_i\in V-S$ 修改 dist：
  $dist[i]=min\{dist[i],dist[k]+weight(k,i)\}$
- 重复直至 S 包含所有顶点，此时 dist[i] 即为 v_0 到 v_i 的最短路径

算法的时间复杂度为 $O(n^2)$

所有顶点之间的最短路径

问题：已知一个各边权值均大于0的有向带权图，对每一对顶点 $v_i\neq v_j$ 求最短路径及其长度

解决方法之一是以每个顶点为源点执行 Dijkstra 算法，时间复杂度为 $O(n^3)$

Floyd 算法：

设置一个 $n\times n$ 的方阵 $A^{(k)}$ ，除对角线为0以外，$a^{(k)}[i][j]$ 为顶点 v_i 到 v_j 的有向路径的长度，初始时即为邻接矩阵，$A^{(-1)}=E$

$A^{(k)}[i][j]=min\{A^{(k-1)}[i][j],A^{(k-1)}[i][k]+A^{(k-1)}[k][j]\},k=0,1,\cdots n-1$

Floyd 算法不允许出现负权边组成的回路，时间复杂度为 $O(n^3)$

顶点表示活动的网络（AOV）

AOV 网络：用有向图表示工程，顶点表示活动，有向边 $<V_i,V_j>$ 表示活动 V_i 必须先于活动 V_j 进行

AOV 网络中活动的前驱后继关系具有**传递性**和**反自反性**，因此 AOV 网络中不能出现有向环。可为其构造**拓扑序列**，即将各个顶点排列成一个线性序列，所有的前驱后继关系均能满足。AOV 网络的拓扑序列不唯一

拓扑排序的步骤：

- 输入 AOV 网络，令 n 为顶点个数
- 在 AOV 网络中寻找没有直接前驱的顶点，输出
- 从图中删去该顶点以及从其出发的所有有向边

重复，若全部顶点已经输出，则完成拓扑排序，否则说明图中剩余的顶点均有直接前驱，即图中存在有向环。可以采用一个栈存储所有入度为 0 的顶点，每次从栈中弹出一个顶点，输出，从该顶点出发的边的终点入度-1，若为0则进入栈。进出栈共执行 n 次，入度-1执行 e 次，时间复杂度为 $O(n+e)$

边表示活动的网络（AOE）

AOE 网络：有向带权图表示工程，边表示活动，边的权表示活动的持续时间，顶点表示事件。每一个事件发生表示在它之前的活动已经完成，在它之后的活动可以开始。

称入度为0的点为源点，出度为0的点为汇点

完成工程需要完成所有活动，取决于从源点到汇点的最长路径上所有活动持续时间之和，称为**关键路径**，以下定义几个与计算关键路径有关的量

- 事件最早可能开始时间 $Ve(i)$：从源点 V_0 到事件 V_i 的最长路径
- 事件最迟允许开始时间 $Vl(i)$：保证汇点 V_{n-1} 在 $Ve(n-1)$ 完成的前提下，V_i 最迟允许开始的时间，即 $Ve(n-1)$ 减去 V_i 到 V_{n-1} 的最长路径
- 活动最早可能开始时间 $Ae(k)$：设活动 a_k 在边 $<V_i,V_j>$ 上，则 $Ae(k)=Ve(i)$
- 活动最迟允许开始时间 $Al(k)$：设活动 a_k 在边 $<V_i,V_j>$ 上，则 $Al(k)$ 是保证不会延误的情况下活动最晚开始的时间，$Al(k)=Vl(j)-during(a_k)$
- 用 $Al(k)-Ae(k)$ 表示活动 a_k 的时间余量，时间余量为0的活动为**关键活动**

求 Ve 与 Vl（必须拓扑有序且逆拓扑有序）

- $Ve(0)=0, Ve(i)=max\{Ve(j)+during(<V_j,V_i>)\}, <V_j,V_i>\in S_2, i=1,2,\cdots,n-1$，其中 S_2 为所有指向 V_i 的有向边的集合
- $Vl(n-1)=Ve(n-1), Vl(i)=min\{Vl(j)-during(<V_i,V_j>)\}, <V_i,V_j>\in S_1, i=1,2,\cdots,n-1$，其中 S_1 是所有从 V_i 出发的有向边的集合

然后即可求出 $Ae(k)=Ve(i), Al(k)=Vl(j)-during(<V_i,V_j>)$

$Ae=Al$ 的活动即为关键活动