1 G = (V, E) 代表一个图,有限的顶点集合,边集合(顶点的偶对)
2 稀疏图,密集图,完全图(边数<(n-1)n/2),无向图,有向图,标号图(顶点带有标号的图),带权图
3 (V1,V2)无向边 <V1,V2>有向边
4 不考虑顶点到自身的边,不允许同一条边重复出现
5 邻接点:一条边连接的顶点,
6 顶点的度:无向图:顶点的相关联边的个数, 有向图:入度/出度, 终端节点
7 子图:(V', E'), 且 E'中边相关联的顶点都在 V'中,才成为子图
8 路径,简单路径(经过的顶点互不相同),路径长度(边的条数),回路,简单回路,无环图,有向无环图(DAG)
9 有根图:(有向图中)存在一个顶点,有路径到达其他的所有顶点,根
10 连通图:(无向图)有一条路径, 任意两点连通->无向图连通
11 连通分量/连通分支:无向图的最大连通子图
12 强连通性:(有向图)任意两个顶点有一条有向路径
13 强连通分量:(有向图)强连通的最大子图
14 网络:带权的连通图
15 自由树:不带简单回路的无向图,连通,只有|V|-1 条边
16 相邻矩阵,空间代价 $O(n^2)$与边数无关
17 邻接表: 顶点表:包括顶点数据和指向边表的指针 边链表:顶点序号和指向边表下一条目的指针
18 无向图需要|V|+2|E|个存储单元,边链表中表目顺序往往按照顶点编号排列
19 有向图保存出边表和入边表之一即可,|V|+|E|个单元
20 十字链表:顶点表:data/firstinarc 第一条指向该顶点的边/firstoutarc 第一条出边
21 边链表:fromvex,tovex,info,fromnextarc(下一个以 fromvex 起点的边),tonextarc
22 周游:系统访问所有的顶点一次 考虑:非连通,存在回路->设置标志位
23 DFS,深度优先搜索树 时间复杂度:邻接表:有向图 $\theta(V+E)$,无向图 $\theta(V+2E)$,相邻矩阵: $\theta(V^2)$
24 BFS,广度优先搜索树 实质上相同,时间复杂度相同
25 拓扑排序:以某种线性顺序组织任务,满足先决条件 有向无环图在不违反先决关系条件下排成线性序列,拓扑序列,不唯一
26 选择一入度为 0 的顶点输出,删掉此顶点和所有的出边,循环 环存在<==>排序结束仍有顶点没有输出
27 基于邻接矩阵的实现(太慢 $\theta(V^3)$)
28 基于邻接表的实现:BFS/DFS $\theta(2V+E)$
29 BFS:入度为 0 的顶点入列,每出列一个顶点,它的出边到达的顶点度数-1,为 0 入列,循环,可以判断环的存在
30 DFS:标记访问过的顶点,递归访问过所有儿子后,自己入队列;最后得到某个 top 排序的逆序,不能判断环的存在(可以加一个祖先的栈,每次询问是否会走到祖先处,有则有环)
31 带权图的最短路径问题 单源最短路径 Dijkstra 每对顶点的最短路径 Floyd
32 Dijkstra(边权非负),生成由近及远的 V0 为根的有向树
33 时间复杂度:(不采用最小堆)每次寻找权值最小结点:V 次扫描,每次 V 个顶点,更新 D 值处 E 次,总时间代价 $\theta(V^2)$ (采用最小堆)不删除原有值,只插入新元素,总时间复杂度 $\theta((V+E)\log E)$
34 Floyd 算法(插点法) 一类动态规划的算法, n 次迭代,更新路径长度,pre 值,允许边权值为负数 时间复杂度 $O(n^3)$ 适合稠密图
35 最小生成树(支撑树) 带权的连通无向图的子图,满足连通性,边权值总和最小,不唯一,但是最小的权值确定

36 Prim 找两个集合的割边中权值最小的(维护最小堆),将边的另一个顶点加入第一个集合,循环直到所有的顶点都加入,时间复杂度同 Dijkstra 算法

37 Kruskal 分成 V 个等价类,按权的大小处理(最小堆),合并等价类(路径压缩),直到只剩一个等价类,时间代价 O(ElogE)