六、图（下）

6.1.最短路径问题Shortest Path：边的权值和最小的路径

起点：源点Source

终点：Destination

6.2.问题分类：

a.单源最短路径问题：从固定源点出发到所有其他定点的最短路径

无权图：路径按递增（非递减）顺序生成，BFS（Queue实现），链表实现O(V+E)

有权图：

负值圈:negative-cost cycle会使算法都挂掉

Dijkstra算法：路径按递增（非递减）顺序生成，不能解决有负边的情况

令s={源点s+已经确定了最短路径的顶点vi}

对任意未收录的顶点v，求dist时，仅经过s中已收录的定点

每次从未收录的定点中选一个dist最小的收录（**贪心算法**）

（真正的最短路径必须只经过s中的顶点）

增加一个v进入s，可能影响另一个w的dist值！！

那么v一定在s到w的路径上且v与w邻接

dist[w]=dist[v]+<v,w>的权重 或不变

时间复杂度：

a.直接扫描所有未收录顶点O(V)：O(V^2 +E)，对稠密图效果好

b.将dist存在最小堆中O(logV):O(VlogV +ElogV)=O(ElogV)，假设V<E

稀疏图效果好

b.多源最短路径问题：求任意两点间的最短路径

方法一：直接将单源最短路径算法调用V遍：O(V^3+E*V)，稀疏图效果好

方法二：Floyd算法：O(V^3)，稠密图效果好

1.初始化D^(-1)[i][j]为邻接矩阵，两点有边值为1，**无边值为Max**

2.D^0到D^(V-1)慢慢给出i到j的真正最短距离

6.3.最小生成树问题：Miminum Spanning Tree：让所有节点都连通

6.3.1.概念：

树：无回路；V个顶点一定有V-1条边

是生成树：包含全部顶点；V-1条边都在图里

生成树中任加一条边都一定构成回路

边的权重和最小

最小生成树存在=图连通

6.3.1.贪心算法：每一步都要最好（权重最小的边）的

只能用图里有的边、**只能正好用掉V-1**条边，不能有回路

a.Prim算法（收集的是点）：每次加一个到小树路径最小的结点：注意不能有回路！！

与已收录结点不连通的初始dist要设定为正无穷，收录进后dist变为0

初始parent设为-1：parent用来回溯路径

O(V^2)，适合稠密图

b.Kruskal算法（收集的是边）：森林合并成树

依次把权重最小的边连上

造成回路的不连

取权重最小边：把边放入最小堆中，取的时候为O（logN）

并查集（两点是否在一个集合（一个树）中）：判断E(v,w)是否构成回路

如果MST中不到V-1条边，则生成树不存在（不连通）

T=O（ElogE）

6.3.拓扑排序：

AOV网络：Activity On Vertex e.g.先修课程

拓扑序：满足（v到w有有向路径，v一定排在w前面）的顶点序列

拓扑排序：获得一个拓扑序的过程

AOV如果有合理的拓扑序，则必定是有向无环图**DAG**：Directed Acyclic Graph

拓扑序算法：

每次输出没有入度的结点V

V结点输出后，V的每个临接点W的入度-1（减完后入度为0则入容器）

若图中有未输出结点，但每个都有入度，则代表图中有回路，不是合理AOV（DAG）

随时将入度变为0的顶点放到一个Queue容器里（减少时间复杂度）

AOE网络：Activity On Edge

活动写在边上，数据包括：持续时间C<i,j>,机动时间D<i,j>

顶点表示活动到此结束

顶点数据包括：顶点编号+最早完成时间+最晚完成时间

关键路径问题：

最早完成时间：初始结点为0，从前往后，前一个最早+C，每个选最大

最晚完成时间：末结点最早=最晚，从后往前倒推，后一个最晚-C，选最小

机动时间：后一个最晚-前一个最早-C

关键路径：绝对不允许延误的活动路径（没有机动时间）