最短路径算法

在网图中，点A到点B的所有路径中，边的权值之和最短的那一条路径，这条路径就是两点之间的最短路径，并称路径上的第一个顶点为源点，最后一个顶点为终点。在非网图中，最短路径是指两点之间经历的变数最少的路径。

Dijkstra算法：初始化；将v0加入s集合；循环n-1次：①找最短距离（v-s）,找到顶点下标v ②顶点v加入s集合 ③再调整当前所找到的从始点v到每个终点vi的最短路径的长度以及两个顶点是否连通

最小支撑树

无向连通树的生成树不是唯一的，连通图的一次遍历所经过的边的集合及图中所有顶点的集合就构成了该图的一棵生成树，对连通图的不同遍历，就可能得到不同的生成树。

Prime算法

数据结构：邻接矩阵

先初始化数据，将v0加入U集合，循环n-1次，记录最短边集以及最短边下标，再调整修改从v0到vi的路径权值。

Kruskal算法

数据结构：

#define MAXMINE <图中的最大边数>

typedef struct{

elemtype v1;

elemtype v2;

int cost;

}EdgeType;

EdgeType edges[MAXMINE];

算法：

事先把数组edges中的各元素按照其cost域值由小到大的顺序排列。在对连通连通分量合并时，采用集合的合并方法。对于有n个顶点的网，设置一个数组father[n]，其初值为father[i]=-1（i=0，1，...，n-1），表示各个点在不同的连通分量上，然后，依次取出edges数组中的每条边的两个顶点，查找它们所属的连通分量，假设f1和f2为两顶点所在的树的根结点在father数组中的序号，若不相等，表明这两条边的两个顶点不属于同一分量，则将这条边作为最小生成树的边输出，并合并它们所属的两个连通分量。