1. **问题背景**

图和树是两种重要的数据结构，图的遍历是所有关于图的问题的基础，我们已知有两种：深度优先遍历与广度优先遍历。可以试想这样一个问题：假设目前有N个顶点，每个顶点连接的路径不一样，能否设计一个算法，快速查找出能覆盖所有顶点的路径。本问题的核心并不是求解亮点之间的最短路径，而是需要设计一个路线，能够覆盖所有的点。生活中比较常见的旅行路线设计问题就是基于这个模型。

1. **问题引入及描述**

如下图所示，在这样一个连通图中，我们需要找到权重最小并且能够覆盖所有点的方案。

**图片包含 小, 绿色, 电脑, 建筑

描述已自动生成**

能够覆盖所有路径的方案可以有如下几种，我们选取两种作为比较说明

权重总和为：11+26+20+22+18+21+24+19 = 161



权重总和为：8+12+10+11+17+19+16+7 = 100

通过这样的比较，我们不难发现，第二种方案的权重值更小，我们可以带入一个问题场景，假设每一个点都是一个城市，权重值代表了两个城市之间所需要的花费，对于旅行者或者旅游公司来讲，如何设计一个路线使得每个城市都玩一次的成本最低，这是很经典的一个问题。

1. **问题定义**

## 3.1 定义：在一个无向图且给出权重的连通图中，将给出的所有点连接起来（即从一个点可到任意一个点），且连接路径之和最小的图叫最小生成树。

## 3.2 数据结构：树形结构，因为当n个点相连，且路径和最短，那么将它们相连的路一定是n-1条。

## 3.3 实现思路：每次取出树中点的连接的最小路径，且该路径连接的点不在树中，然后将该路径连接的点加入树中，重复并进行路径更新，当取出边达到n-1条时，树已建立成功。

## 算法分析：

在最小生成树问题中，主要会采用贪心算法。贪心算法在求解某个问题时，总是做出眼前的最大利益，也就是说只顾眼前不顾大局，所以它是局部最优解。贪心算法不是对所有问题都能得到整体最好的解决办法，关键是贪心策略的选择，选择的贪心策略必须具备无后效性，即某个状态以前的状态不会影响以后的状态，只与当前状态有关。

贪心算法有两个特点：

1. 贪心选择：一个问题的整体最优解可通过一系列局部的最优解的选择达到，并且每次的选择可以依赖以前作出的选择，但不依赖于后面要作出的选择。这就是贪心选择性质。
2. 最优子结构性质：通过问题的局部最优解能够得出全局最优解。

对于最小生成树问题，结合贪心算法的性质，可以有两种贪心策略，最近顶点策略和最短边策略。

**4.1 最近顶点策略（Prim Algorithm）**

本策略的核心思想是：任选一个顶点，并以此建立起生成树，每一步的贪心选择是简单地把不在生成树中的最近顶点添加到生成树中。这就是著名的Prim算法。它使生成树以一种自然的方式生长，即从任意顶点开始，每一步为这个顶点加一个分枝，直到生成树中包含全部顶点。

Prim算法在找最小生成树时，将顶点分为两类，一类是在查找的过程中已经包含在树中的（假设为 A 类），剩下的是另一类（假设为 B 类）。对于给定的连通网，起始状态全部顶点都归为 B 类。在找最小生成树时，选定任意一个顶点作为起始点，并将之从 B 类移至 A 类；然后找出 B 类中到 A 类中的顶点之间权值最小的顶点，将之从 B 类移至 A 类，如此重复，直到B类中没有顶点为止。所走过的顶点和边就是该连通图的最小生成树。我们通过一个例子来说明：

图片包含 游戏机, 文字, 地图

描述已自动生成

假如从顶点A出发，顶点B、C、D 到顶点A的权值分别为2、4、2，所以，对于顶点A来说，顶点B和顶点D到A的权值最小，假设先找到的顶点B。

图片包含 游戏机, 地图

描述已自动生成

继续分析顶点C和D，顶点C到B的权值为 3，到A的权值为4；顶点D到A的权值为2，到B不连通（如果之间没有直接通路，设定权值为无穷大）。所以顶点D到A的权值最小。

图片包含 游戏机, 钟表

描述已自动生成

最后，只剩下顶点C，到A的权值为4，到B的权值和到D的权值一样大，为3。所以该连通图有两个最小生成树。

算法效率分析：对于Prim算法，设连通网中有 n 个顶点，需要两个循环，第一个进行初始化的循环语句需要执行 n-1 次，第二个循环共执行 n-1 次，内嵌两个循环，其一是在长度为 n 的数组中求最小值，需要执行 n-1 次，其二是调整辅助数组，需要执行 n-1 次，所以，Prim 算法的时间复杂度为O()。

**4.2 最短边策略**

Kruskal算法就应用了这个贪心策略，它使生成树以一种随意的方式生长，先让森林中的树木随意生长，没生长一次就将两棵树合并，到最后合并成一棵树。

先构造一个只含n个顶点、而边集为空的子图，把子图中各个顶点看成各棵树上的根结点，之后，从图的边集E中选取一条权值最小的边，若该条边的两个顶点分属不同的树，则将其加入子图，即把两棵树合成一棵树，反之，若该条边的两个顶点已落在同一棵树上，则不可取，而应该取下一条权值最小的边再试之。依次类推，直到森林中只有一棵树，也即子图中含有n-1条边为止。

相比于Prim算法，通俗来讲就是以边为主导地位，每次选择权值最小的边，判断该边连接的两点是否连通，若不连通，则合并两点（合并操作以并查集实现）。记录合并的次数，当次数等于n-1时结束。我们通过一个下图来说明Kruskal算法的工作过程：

地图上有字

描述已自动生成

首先，在初始状态下，对各顶点赋予不同的标记，如下图所示：

有钟表的地图

描述已自动生成

对所有边按照权值的大小进行排序，按照从小到大的顺序进行判断，首先是（1，3），由于顶点1和顶点3标记不同，所以可以构成生成树的一部分，遍历所有顶点，将与顶点3标记相同的全部更改为顶点1的标记，如下图所示：

有钟表的地图

描述已自动生成

其次是（4，6）边，两顶点标记不同，所以可以构成生成树的一部分，更新所有顶点的标记为：

图片包含 地图

描述已自动生成

其次是（2，5）边，两顶点标记不同，可以构成生成树的一部分，更新所有顶点的标记为：

有钟表的地图

描述已自动生成

然后最小的是（3，6）边，两者标记不同，可以连接，遍历所有顶点，将与顶点 6 标记相同的所有顶点的标记更改为顶点 1 的标记：

图片包含 游戏机, 人们

描述已自动生成

当选取的边的数量相比与顶点的数量小 1 时，说明最小生成树已经生成。所以最终采用克鲁斯卡尔算法得到的最小生成树为上图所示。

算法效率分析：Kruskal算法为了提高每次贪心选择时查找最短边的效率，可以先将图G中的边按代价从小到达排序，则这个操作的时间复杂度为O()，其中e为无向连通网中边的个数。

Kruskal算法只与网中边的条数有关，而与网中顶点的个数无关，当使用时间复杂度为O（elog2e）的排序方法时，克鲁斯卡尔算法的时间复杂度即为O（log2e），因此当网的顶点个数较多、而边的条数较少时，使用Kruskal算法构造最小生成树效果较好。

1. **问题应用**

在通信行业中，如果要在n个城市之间铺设光缆，主要目标是要使这 n 个城市的任意两个之间都可以通信，但铺设光缆的费用很高，且各个城市之间铺设光缆的费用不同，因此另一个目标是要使铺设光缆的总费用最低。这就需要找到带权的最小生成树。

同理，旅行商问题也是利用最小生成树的经典问题之一。