研究生算法课课堂笔记

上课日期： 12 月 19 日 第(2)节课

组长学号及姓名： 1601214515 马银萍

组员学号及姓名： 1601214443 向耀程 1601214527 张芳芳

本节课要点：贪心法及最小生成树算法课后习题讲解

1. **题目描述：**

   假定海岸线是无限长的直线，陆地在海岸线的一边，大海在另一边，每个小岛都视为坐落在海面上的一个点。每个在海岸线上架设的雷达监测站，都只能覆盖以 R 为半径的一个圆，所以海面上的小岛能被覆盖的条件是它距离海岸线上的雷达监测站的距离小于或者等于 R。给定 n 个小岛以及这些小岛的位置，并输入雷达的辐射面积 R，问最少需要多少个雷达站才能覆盖所有小岛。

   问题分析：

   与上一课所讲的进程监督问题类似，可以将问题转化为求区间选择问题的最大不冲突区间数。以每个小岛为圆心，R 为半径画圆，圆在 x 轴下方的部分即为可以覆盖该小岛的雷达架设区域，易知其他条件不变时雷达架在海岸线比内陆能覆盖更广的区域，因此只需要考虑海岸线上的点。如图 1 所示，以每个小岛为圆心的圆都与海岸线的相交所得到的弦即为我们可以架设雷达的区间，设小岛 T 的坐标为（x,y），如果 y>R,那么这个问题无解，无论在哪里建站都覆盖不到这个小岛。如果 y<=R,那么区间的起始时间为：x-$\sqrt{R^2-y^2}$ ，结束时间为：x+$\sqrt{R^2-y^2}$,我们按照结束时间对区间进行排序，在考察每个区间的时候，需要判断该区间的起始时间是否在上一区间的结束时间之前，本问题已完全转化为区间选择问题。下图中$T_i(i=1,2,\ldots,n)$表示海面上的小岛。

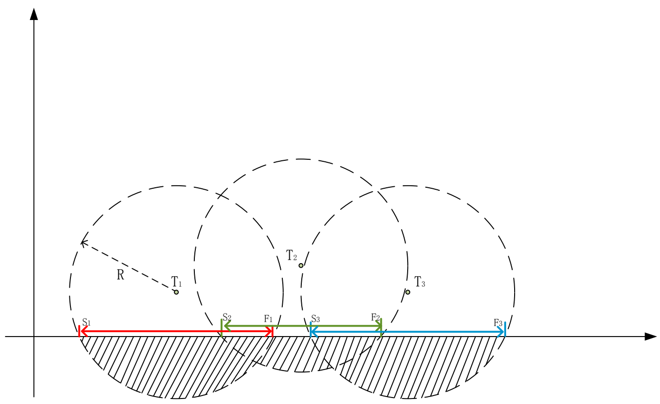

图 1：雷达问题示意图

伪代码如下：

Interval-selection(T,R)

```
For Each  t ∈ T , (x, y) ← (t.x, t.y)
        If  y > R  then
            RETURN FALSE
        Else
                t.start = x − √(R² − y²)  t.end = x + √(R² − y²)
          End
End
SORT(T) // sort by the end time
```

```
Cur = 0 Count = 0
For  i  in  range(T.size())
        If T[i].start >= T[cur].end  Continue
        Else
            Count++
            Cur = i
        End
End
Return Count
```

---

2. 你认为下面的论述中是真还是假，做出你的假定，如果是真，给出一个简短的证明，如果是假，给出一个反例。

设G是任意一个无向连通图，每条边e具有不同的权重c(e)。假设 e*是G中权重最小的边；即对每条边$e \neq e^*$,$c(e^*) < c(e)$.那么 G 中存在一棵包含边 e*的最小生成树T。

这个论述是真的。

证明：假设所有的最小生成树中都不包含 e*，任取某个最小生成树MST_i，将 e*加入该树中，则必然会形成一个环路，而 e*一定是该环路上权重最小的边，删掉该环路上除 e*以外的任何一条边都可以得到一个比MST_i更小的生成树，所以矛盾。

思考：

（1）如果图中可能有相等的边，但是最小边是唯一的，那么同样的，最小的边也一定在任何一棵最小生成树中。

（2）假设存在一个连通图G，权重最大的边如果唯一，则最大边必不在最小生成树里？

解答：这个观点是错误的，如果权重最大的边 e 在 G 中连接的是两个连通分量，并且去除边 e 之后 G 不再连通，这种情况下 e 一定在最小生成树中。所以在这里应该加一个限制条件，该权重最大的边如果出现在一个环中，并且唯一，那么它将不会出现在任何一棵最小生成树中。同时，如果环中权重最大的边不唯一，那么有可能权重最大的边会出现在某个最小生成树中（极端情况：图中所有的边的权重都相同，而且存在环，这时，其中某些边也必须出现在最小生成树中）

3. （a）假设给出图G上的最小生成树问题的一个实例，边的费用都是正的且不相等，设T是关于这个实例的一棵最小生成树，现在我们把每条边的权重c_e用它的平方c_e^2替换，因此产生这个问题的一个新的实例，它有同样的拓扑结构但是具有不同的权重。T 对于这个新的实例仍旧是一棵最小生成树吗？

（b）假设给定有向图G上的最短$s-t$路径问题的一个实例，我们假设所有的边的权重都是正的且不相等。设 P 是对于这个实例的一个最小权重的$s-t$路径，现在我们把每条边的权重c_t用它的平方c_t^2来替换，因此产生这个问题的一个新的实例，它有同样的拓扑结构但具有不同的权重。

P 对于这个新的实例一定仍旧是一条最小权重的$s-t$路径吗？

证明：

（a）首先回顾一下 Kruskal 算法的执行过程：

1). 记 Graph 中有 v 个顶点，e 个边；

2). 新建图 Graphnew，Graphnew 中拥有原图中相同的 e 个顶点，但没有边；

3). 将原图 Graph 中所有 e 个边按权值从小到大排序；

4). 循环：从权值最小的边开始遍历每条边，如果这条边连接的两个节点在图 Graphnew 中的不同连通分量中，添加这条边到图 Graphnew 中，直至图 Graph 中所有的节点都在同一个连通分量中。

对所有c_e进行升序排序得到序列 A，对所有c_e^2按照升序排序得到序列 B，由于边的费用是正的，那么根据 $y = x^2$ 在 $(0, +\infty)$ 单调递增的性质可知c_e^2在 A 中的序号应与c_e在 B 中的序号是一样的。那么根据 Kruskal 算法进行最小生成树的构建时，两个图构建时边权重顺序相同，并且由于图的结构没有发生改变，因此在判断一条边连接的两个点是否在同一个连通分量的规则是相同的，因此构造出来的图也是相同的。

（b）这个结论是错误的，可以举出以下反例。

图 3 中，s 为源点，那么求得的 t 到源点 s 最短距离应为 2，最短路径为$s \to t$。

图 4 是将图 3 中边权重平方之后得到的新图，在这个图中 t 到 s 的最短距离应该是 3，最短路径为$s \to a \to b \to t$。显然$s \to t$的最短路径发生了改变。

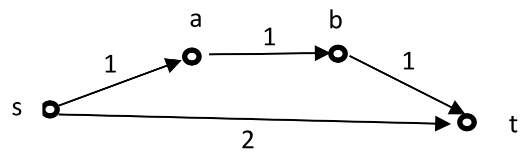

图 3：以 s 为源点的单源最短路径图

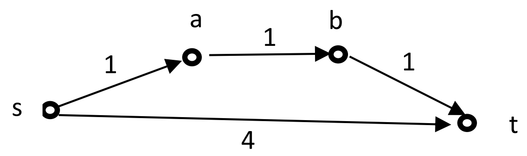

图 4：图 3 中边权重平方得到的图

4. 假设给你一个连通图G，它的边的费用都是不相同的。证明G有唯一的一棵最小生成树。

   证明：假设 G 的最小生成树不唯一，设T_1和T_2分别是图 G 的两棵不同的最小生成树，那么必然存在一条边$e \in T_1$ 但 $e \notin T_2$，否则T_1包含于T_2，而T_2是最小生成树，那么T_1和T_2是相同的。那么将 e 添加到T_2中，则会形成一个环 C，并且 e 不会是这个环上权重最大的一条边，否则由 red rule 知生成T_1时 e 必然不会在最小生成树T_1中，那么将环 C 中权重最大的一条边去掉，将 e 加入到T_2中，会形成一棵新的生成树，这棵生成树的权重会比T_2小，这与T_2是最小生成树是矛盾的。由反证法可知 G 中最小生成树是唯一的。

5. 最小瓶颈树：设$G = < V, E >$是一个具有n个顶点m条边的连通图，假设边的权重都是不同的。设$T = < V, E' >$是G的一棵生成树；我们定义T的瓶颈边是 T 中具有最大权重的边。那么G的一棵生成树T是一棵最小瓶颈生成树。

（a）G 的每棵最小瓶颈生成树都是G的一棵最小生成树吗？

（b） G 的每棵最小生成树都是G的一棵最小瓶颈生成树吗？

证明：

（a）最小瓶颈生成树不一定是最小生成树，举反例证明如下：

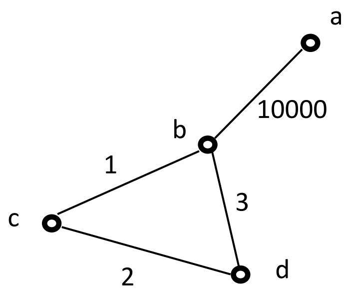

图 5：5（a）反例

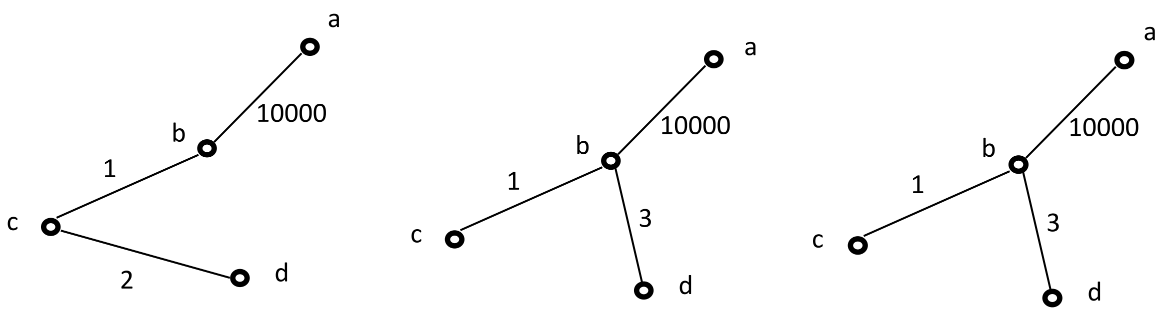

图 6 ：图 5 的瓶颈生成树

图 5 的最小瓶颈生成树有 3 棵，但是只有图 6 中最左边一棵是最小生成树。

在这个例子中，a—> b是瓶颈边，在最小瓶颈生成树中，瓶颈边之外的其它比它小的边怎么选是没有影响的，但是却会影响最小生成树。也就是说，最小瓶颈生成树只是保证了瓶颈边最小，其他部分还有好多自由度。

（b）假设最小生成树 T 不是一棵最小瓶颈生成树，那么最小瓶颈生成树中权重最大的边一定小于最小生成树中权重最大的边。设 e 为最小生成树中权重最大的边，它的两个端点是 u 和 v，那么 e 肯定不在最小瓶颈生成树中，由于在最小瓶颈生成树中 u 和 v 是连通的，将 e 加入到最小瓶颈生成树中，必定会形成一个环，并且 e 是环上权重最大的一条边，由 red rule 知环上权重最大的边一定不在最小生成树中，这与 e 在最小生成树 T 中矛盾，因此证明最小生成树是一棵最小瓶颈生成树。

思考：Dijkstra 算法中边不能为负，但 Prim 算法可以，为什么？

DIJKSTRA (V, E, s)

Create an empty priority queue.
FOR EACH $v \neq s$: $d(v) \leftarrow \infty$; $d(s) \leftarrow 0$.
FOR EACH $v \in V$: *insert* v with key $d(v)$ into priority queue.
WHILE (the priority queue *is not empty*)
  $u \leftarrow$ *delete-min* from priority queue.
  FOR EACH edge $(u, v) \in E$ leaving u:
    IF $d(v) > d(u) + \ell(u, v)$
      decrease-key of v to $d(u) + \ell(u, v)$ in priority queue.
      $d(v) \leftarrow d(u) + \ell(u, v)$.

PRIM (V, E, c)

Create an empty priority queue.
$s \leftarrow$ any node in V.
FOR EACH $v \neq s$: $d(v) \leftarrow \infty$; $d(s) \leftarrow 0$.
FOR EACH v : *insert* v with key $d(v)$ into priority queue.
WHILE (the priority queue *is not empty*)
  $u \leftarrow$ *delete-min* from priority queue.
  FOR EACH edge $(u, v) \in E$ incident to u:
    IF $d(v) > c(u, v)$
      decrease-key of v to $c(u, v)$ in priority queue.
      $d(v) \leftarrow c(u, v)$.

图 7：Dijkstra 算法和 Prim 算法伪代码

解答：Dijkstra 算法和 Prim 算法不同之处在于 decrease-key 这一步，Dijkstra 算法每次从优先队列中取出点 u 加入到S集合之后，从源点 s 到该点 u 的最短路径就已经确定下来了，不会再变化，这时如果有其他到达 u 的路径的权重为负数，使得从源点到点 u 距离更短，但是 Dijkstra 算法并不会更新 u 点到源点的距离值，因此会产生矛盾。比如图 8 中，s 为源点，根据 Dijkstra 算法首先会将 a 点加入到集合 S 中，并将源点到 a 点的值更新为 1，然后将 b 点加入到集合 s 中，这时尽管有 b 到 a 的权重为-3 的边，Dijkstra 也不会将源点到 a 点的距离更新为-1。

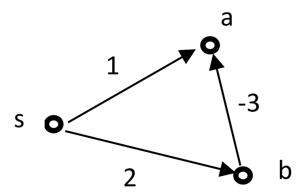

图 8：带负权重边的图

而Prim算法中的 cost 为 attachment cost，每次更新时，都是将当前横跨两个 partition 的边中权重最小的边加入最小生成树中，所以边为负数只会使得总权重变小，不会有其他变化，因此Prim算法适用于存在负权重边的情况。