最小生成树(MST)问题及其扩展

本文大量内容引自北京大学信息学院程序设计实习（实验班）郑聃崴、陈国鹏等同学的讲义，在此致谢

图的生成树

- 在一个连通图G中，如果取它的全部顶点和一部分边构成一个子图G’，即：

  V(G’)=V(G);E(G’) ∈E(G)

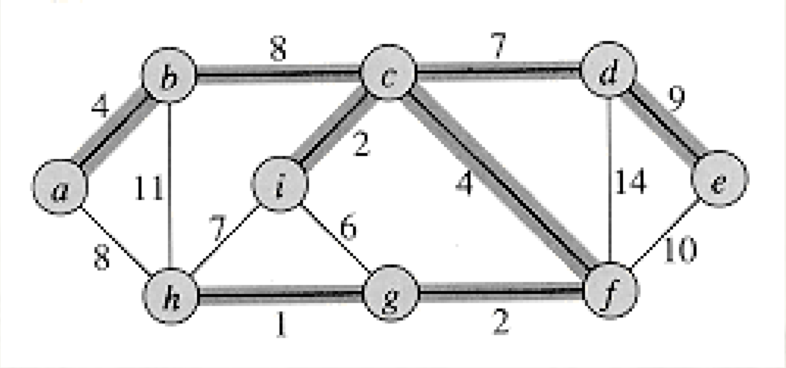
  若边集E(G’)中的边既将图中的所有顶点连通又不形成回路，则称子图G’是原图G的一棵生成树。
- 一棵含有n个点的生成树，必含有n-1条边。

最小生成树

- 对于一个连通网（连通带权图，假定每条边上的权均为大于零的实数）来说，每棵树的权（即树中所有边的权值总和）也可能不同
- 具有权最小的生成树称为最小生成树。

最小生成树

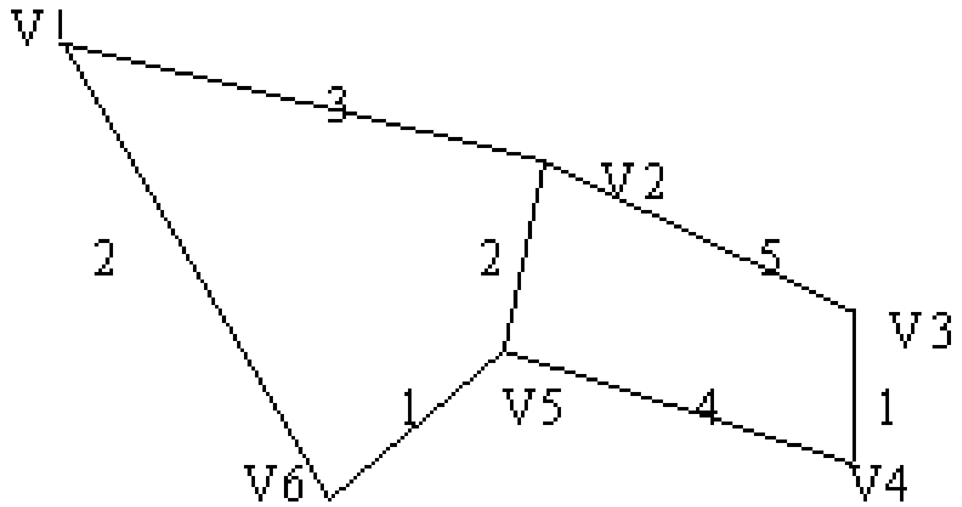
- 生成树
  - 无向连通图的边的集合
  - 无回路
  - 连接所有的点
- 最小
  - 所有边的权值之和最小

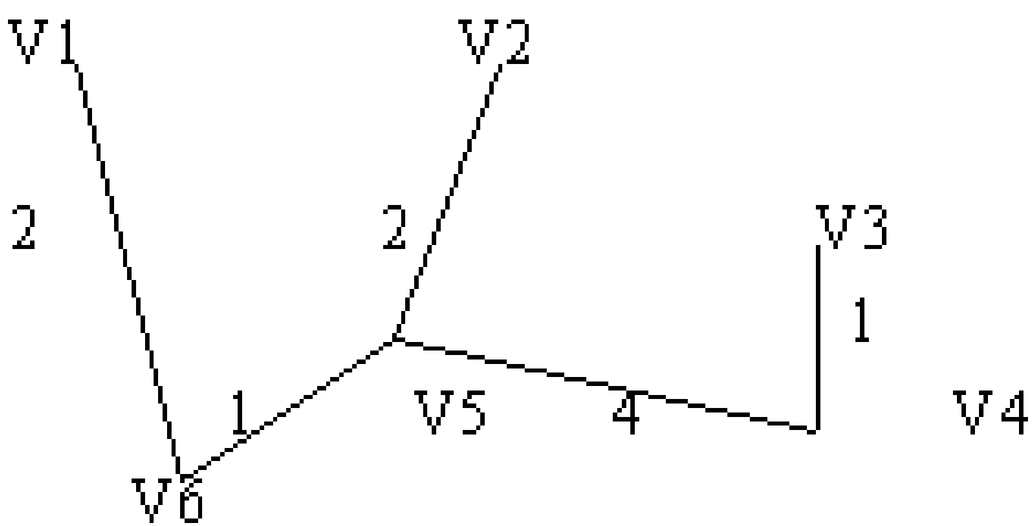


Prim算法

- 假设G=(V,E)是一个具有n个顶点的连通网，T=(U,TE)是G的最小生成树，U,TE初值均为空集。

- 首先从V中任取一个顶点（假定取v1），将它并入U中，此时U={v1}，然后只要U是V的真子集(U∈V)，就从那些一个端点已在T中，另一个端点仍在T外的所有边中，找一条最短边，设为(v_i, v_j)，其中$v_i \in U, v_j \in V-U$，并把该边(v_i, v_j)和顶点v_j分别并入T的边集TE和顶点集U，如此进行下去，每次往生成树里并入一个顶点和一条边，直到n-1次后得到最小生成树。


V1
3
V2
2
2
5
V3
V5
1
4
1
V6
V4


V1
V2
2
2
V3
1
1
V5
4
V4
V6

关键问题

- **每次如何从连接T中和T外顶点的所有边中，找到一条最短的**
- 1） 如果用邻接矩阵存放图，而且选取最短边的时候遍历所有点进行选取，则总时间复杂度为O(V^2)， V 为顶点个数
- 2)用邻接表存放图,并使用堆来选取最短边，则总时间复杂度为O(ElogV)
- 不加堆的Prim 算法适用于密集图，加堆的适用于稀疏图

用prioirty_queue实现 Prim + 堆

```
#define INFINITE 900000000

struct XEdge
{
        int v; //边端点
        int w; //边权值
        XEdge(int v_ = 0, int w_ = INFINITE):v(v_),w(w_) { }
};

vector<vector<XEdge> > G(30); //图的邻接表
bool operator <(const XEdge & e1, const XEdge & e2)
{
        return e1.w > e2.w;
}
```

```
int HeapPrim(const vector<vector<XEdge> > & G,  int n)
//G是邻接表,n是顶点数目，返回值是最小生成树权值和
{
        int i,j,k;
        XEdge xDist(0,0);
        priority_queue<XEdge> pq;
        vector<int> vDist(n); //各顶点到已经建好的那部分树的距离
        vector<int> vUsed(n);//标记顶点是否已经被加入最小生成树
        int nDoneNum = 0; //已经被加入最小生成树的顶点数目
        for( i = 0;i < n;i ++ ) {
                vUsed[i] = 0;
                vDist[i] = INFINITE;
        }
        nDoneNum = 0;
        int nTotalW = 0;
        pq.push(XEdge(0,0));
```

```
    while( nDoneNum < n && !pq.empty() ) {
        do {
            xDist = pq.top();    pq.pop();
        } while( vUsed[xDist.v] == 1 && ! pq.empty());
        if( vUsed[xDist.v] == 0 ) {
            nTotalW += xDist.w;    vUsed[xDist.v] = 1;
nDoneNum ++;
            for( i = 0;i < G[xDist.v].size();i ++ ) {
                int k = G[xDist.v][i].v;
                if( vUsed[k] == 0) {
                    int w = G[xDist.v][i].w ;
                    if( vDist[k] > w ) {
                        vDist[k] = w;
                        pq.push(XEdge(k,w));
                    }
                }
            }
        }
    }
    if( nDoneNum < n )
        return -1; //图不连通
    return nTotalW;
}
```

用prioirty_queue实现 dijkstra + 堆的 POJ 3159 Candies (30000点， 150000 边求最短路)

```
#include <stdio.h>
#include <iostream>
#include <vector>
#include <queue>
using namespace std;
struct CNode
{
        int k;
        int w;
};
bool operator < ( const CNode & d1, const CNode & d2 ) {
        return d1.w > d2.w;   //priority_queue总是将最大的元素出列
}
int aDist[30010];
priority_queue<CNode> pq;
bool bUsed[30010]={0};
//vector<CNode> v[30010];  error,如果用这个，则在poj上会超时。说明vector对象的初始化，也是需要可观时间的
vector<vector<CNode> > v;
const unsigned int INFINITE = 100000000;
```

```
int main()
{
        int N,M,a,b,c;
        int i,j,k;

        CNode p, q;

        scanf("%d%d", & N, & M );

        v.clear();
        v.resize(N+1);

        memset( bUsed,0,sizeof(bUsed));
        for( i = 1;i <= M; i ++ ) {

                scanf("%d%d%d", & a, & b, & c);
                p.k = b;
                p.w = c;
                v[a].push_back( p);
        }
        p.k = 1;
        p.w = 0;
        pq.push ( p);
```

```
	while( !pq.empty ()) {
		p = pq.top ();
		pq.pop();
		if( bUsed[p.k])
			continue;
		bUsed[p.k] = true;
		if( p.k == N )
			break;

		for( i = 0, j = v[p.k].size(); i < j; i ++ ) {
			q.k = v[p.k][i].k;
			if( bUsed[q.k] )
				continue;
			q.w = p.w + v[p.k][i].w ;
			pq.push ( q);
		}
	}
	printf("%d", p.w ) ;
	return 0;
}
```

Kruskal算法

- 假设G=(V,E)是一个具有n个顶点的连通网，T=(U,TE)是G的最小生成树，U=V,TE初值为空。
- 将图G中的边按权值从小到大依次选取，若选取的边使生成树不形成回路，则把它并入TE中，若形成回路则将其舍弃，直到TE中包含N-1条边为止，此时T为最小生成树。

关键问题

- **如何判断欲加入的一条边是否与生成树中边构成回路。**
- 将各顶点划分为所属集合的方法来解决，每个集合的表示一个无回路的子集。开始时边集为空，N个顶点分属N个集合，每个集合只有一个顶点，表示顶点之间互不连通。
- 当从边集中按顺序选取一条边时，若它的两个端点分属于不同的集合，则表明此边连通了两个不同的部分，因每个部分连通无回路，故连通后仍不会产生回路，此边保留，同时把相应两个集合合并

算法：Kruskal 和 Prim

- Kruskal:将所有边从小到大加入，在此过程中判断是否构成回路
  - 使用数据结构：并查集
  - 时间复杂度：$O(E\log E)$
  - 适用于稀疏图
- Prim:从任一节点出发，不断扩展
  - 使用数据结构：堆
  - 时间复杂度：$O(E\log V)$ 或 $O(V\log V+E)$(斐波那契堆）
  - 适用于密集图
  - 若不用堆则时间复杂度为$O(V^2)$

例题1： POJ 2349 Arctic Network

- 某地区共有n座村庄，每座村庄的坐标用一对整数(x, y)表示，现在要在村庄之间建立通讯网络。
- 通讯工具有两种，分别是需要铺设的普通线路和无线通讯的卫星设备。
- 只能给k个村庄配备卫星设备，拥有卫星设备的村庄互相间直接通讯。
- 铺设了线路的村庄之间也可以通讯。但是由于技术原因，两个村庄之间线路长度最多不能超过 d，否则就会由于信号衰减导致通讯不可靠。要想增大 d 值，则会导致要投入更多的设备（成本）

例题1： POJ 2349 Arctic Network

* 已知所有村庄的坐标（ x ， y ），卫星设备的数量 k 。
* 问：如何分配卫星设备，才能使各个村庄之间能直接或间接的通讯，并且 d 的值最小？求出 d 的最小值。
* 数据规模：0 <= k <= n<= 500

(From Waterloo University 2002)

思路

* 假设 d 已知，把所有铺设线路的村庄连接起来，构成一个图。需要卫星设备的台数就是图的连通支的个数。
* d越小，连通支就可能越多。
* 那么，只需找到一个最小的d，使得连通支的个数小于等于卫星设备的数目。

答案

把整个问题看做一个完全图，村庄就是点，图上两点之间的边的权值，就是两个村庄的直线距离。

只需在该图上求最小生成树，d 的最小值即为第 K 长边！

因为：最小生成树中的最长k-1条长边都去掉后，正好将原树分成了k 个连通分支，在每个连通分支上摆一个卫星设备即可

为什么d不可能比第k长边更小？

假设最小生成树T上，第k长边连接的点是a,b，那么将边<a,b>去掉后，树就分成了两个部分T1 和T2

要使T1和T2能够通讯，必须在T1中找一点p和T2中的点q相连，若边<p,q>的长度小于<a,b>，则在T 上用<p,q>替换<a,b>就能得到更小的生成树，矛盾。因此找不到长度小于<a,b>的<p,q>。

对任何第i长边e(i<k)，同理也不可能找到替代e的边。

因此 d不可能更小了

最小生成树可能不止一棵，为什么第k长边长度一定相同？因为有以下结论：

* 一个图的两棵最小生成树，边的权值序列排序后结果相同

证明：假设某个最小生成树T1的边权从小到大排序后的序列为:
a1, a2 …. an
某个最小生成树T2的边权从小到大排序后的序列为:
b1,b2…bn
两者若不同，则必然存在一个最小的i,使得 ai > bi

假设T2中有m条边的权为bi，那么，T1中最多只有m-1条边的权和bi相同。

但是对于T2中任何一条不在T1中的权为bi的边，如果将其从T2去掉，则T2被分成A,B两个部分。那么在T1中连接A,B这两个部分的边，必然权值是等于bi的，否则经过替换，要么T1的权值可以变得更小，要么T2的权值可以变得更小，这和T1,T2是最小生成树矛盾。对T2中每个权值为bi的边，都可以在T1中找到一个权值相同且不在T2的边与其对应，而这些边由于是连接不同部分的，所以不可能相同，因此，在T1中也应该有m条权值为bi的边，这和T1中最多m-1条权值为bi的边矛盾。因此，不存在i,使得的ai>bi，即两个边权序列应该相同。

次小生成树

例题4：POJ1679 The Unique MST

题目：要求判断给定图的最小生成树是否唯一

解：显然，这是一个求次小生成树的问题，如果求出来的次小生成树权值和与最小生成树相同，则不唯一

次小生成树的定义

* 设G=(V，E，ω)是连通的无向图，T是图G的一个最小生成树。如果有另一棵树T1，满足不存在树T’，T’≠T，ω（T’）<ω（T1），则称T1是图G的次小生成树。

* 次小生成树有可能也是最小生成树

定理 1

* 最小生成树的邻集里包含次小生成树。即次小生成树可以通过由最小生成树换一条边来得到。

定理1证明

- T是某一棵最小生成树，T0是任一棵异于T的树，则定可通过变换 T0 --> T1 --> T2 --> ... --> Tn (T) 变成最小生成树。
- 所谓的变换是，每次把Ti中的某条边换成T中的一条边，而且树T(i+1)的权小于等于Ti的权。

具体操作

1. 在Ti中任取一条不在T中的边uv。
2. 把边uv去掉，就剩下两个连通分量A和B，在T中，必有唯一的边u‘v’ 连结A和B。
3. 显然u‘v’的权不比uv大（否则,uv就应该在T中）。把u‘v’替换uv即得树T(i+1)。
4. 特别地：取T0为任一棵次小生成树，T(n-1)也会是次小生成树且跟T差一条边. 定理1得证(每次换边都不会使得树的权值变小，除了从T(n-1)变到T以外.

算法

- 利用该定理，可以得到O(V^2)的算法求次小生成树，具体如下：

Step 1.

- 先用prim求出最小生成树T。在prim的同时，用一个矩阵max_val[u][v] 记录在T中连结任意两点u,v的唯一的路中权值最大的那条边的权值。

- 这是很容易做到的，因为prim是每次增加一个结点s，而设已经标号了的结点集合为W，则易求W中所有点到s的路中的最大边权值

- 设 u 属于W,且 s是被连接到W中的v点的，

- 则

- Max_val[v][s] = 边(v,s)的权
- Max_val[u][s] = Max(Max_val[v][s], Max_val[u][v])

- 用时O(V^2)。

算法

Step　2.

- 枚举所有不在T中的边uv，加入边uv则必然替换权为max_val[u][v]的边，枚举一次就得到一棵新的生成树，如果在这些生成树中有权值和与原最小生成树相等的，则最小生成树不唯一
- 用时O(E)。
- 总复杂度：O(V^2)

2011 ACM/ICPC亚洲区预选赛北京赛站

Problem A. Qin Shi Huang’s National Road System

题目大意：一个无向完全图，边有正权值，点也有正权值。可以选择一条边，将其边权值变为0。要求选定这条边(假定为e0)并将其权值变为0后，满足以下条件：A/B最大。其中A是e0连接的两个点的点权值和，B是修改后的图的最小生成树的边权值和。

解题思路：先求一棵最小生成树，求的过程中，每加入一个点，就记录已经在树上的所有点到该点的路径（树上的路径）上的最长边的权值。然后枚举权值要变成0的边uv，如果uv不是树边，则用它替换uv路径上的最大权值边，o(1)时间即得新最小生成树的边权值和；uv是树边，新最小生成树的边权值和即为原最小生成树的边权值和减去边uv的权值。

最优比例生成树

* Desert King（POJ2728）
* 题意:

　　一个图，每条边有花费C和长度l 两个非负参数求一个生成树，使得花费之和与长度之和的比最小

分析

- 本题的目标就是求

$$r = (C_1+C_2+\cdots+C_{n-1})/(l_1+l_2+\cdots+l_{n-1})$$

最小。

- 看上去有点像最小生成树，但又有本质区别：

1.按长度最小贪心？

2.按花销最小贪心？

分析

- 我们先假设这个比例为 r，则对于任意 r 应该有：

  $r(min) <= r <= r(max)$

- 生成树有n-1条边，则整理得：

$r(max) * l_1 - c_1 + \cdots + r(max) * l_{n-1} - c_{n-1} >= 0$

$r(min) * l_1 - c_1 + \ldots + r(min) * l_{n-1} - c_{n-1} <= 0$

因为我们这里要求r(min)所以现在重点看第二个式子。

$r = (C_1 + C_2 + \ldots + C_{n-1}) / (l_1 + l_2 + \ldots + l_{n-1})$

分析

- $r(min) <= (C_1+C_2+\cdots+C_{n-1})/(l_1+l_2+\cdots+l_{n-1})$
- $r(min) * l_1 - C_1+\cdots+r(min) * l_{n-1} - C_{n-1} <= 0$

这个式子的意思是只要存在一个建生成树的方案，那么r(min)一定满足这个条件。反过来，也就是r(min)对所有的建生成树的方案都满足上述式子。

分析

考虑不等式：

$r * l_1 - C_1 + \cdots + r * l_{n-1} - C_{n-1} <= 0$

左边的值随着r增加而增加，随着r减少而减少。要使上式对任何一种生成树建法都成立，r可以无限小，但我们要求的答案 r(min)是能使得上式在任何建法下都成立的最大的r的取值。

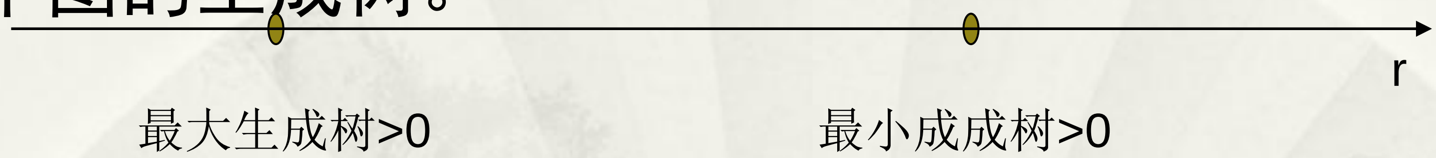
（若 x > r(min)，则r(min)对应的取法，就能使r=x时不等式不成立）

分析

对于每一个假定的r，我们能很快判断上式是否对任何生成树取法都成立，等式左边的值随着r增加而增加，随着r减少而减少，　所以可以用二分查找。

分析

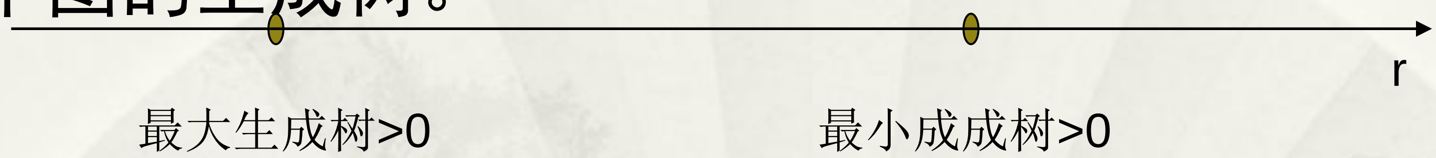
* 以$r * l_i - c_i$为每条边的权值，重新构图，考虑这
个图的生成树。



* 二分r，每次求最大生成树， 新图上最大生成树的取法，就对应于老图上一种生成树的取法。

分析

- 以$r * l_i - c_i$ 为每条边的权值，重新构图，考虑这
个图的生成树。



- 如上图所示，随着r的增加，首先其最大生成树总权值由小于0变成超过0，在这个时候之前的所有r对

$$r * l_1 - c_1 + \cdots + r * l_{n-1} - c_{n-1} <= 0$$

都是成立的，所以这个临界点的r值就是所求的r(min)

分析

- 具体实现的时候就一个二分加最大生成树就好。
- 最大生成树的写法完全类似最小生成树。