最小生成树(MST)问题的扩展

北京大学　　郭炜

本文大量内容引至郑聃崴同名讲义

用prioirty_queue实现 Prim + 堆

```
#define INFINITE 900000000

struct XEdge
{
        int v;
        int w;
        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(可以不要)
        vector<int> vUsed(n);
        int nDoneNum = 0;
        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) {
                    if( vDist[k] > G[xDist.v][i].w ) {
                        vDist[k] = G[xDist.v][i].w;
                        pq.push(XEdge(k,G[xDist.v][i].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山会超时。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;
}
```

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，通讯距离不超过d。

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。

定理

- 如果在最小生成树中去掉所有长度大于d的边后，该最小生成树被分成k个连通支，那么图也被分成k个连通支。

答案

- 只需先求最小生成树，d 的最小值即为第 k 长边！

例题2：POJ 1639 Picnic Planning

矮人虽小却喜欢乘坐巨大的轿车，车大到可以装下无论多少矮人。某天，N(N≤20)个矮人打算到野外聚餐。为了集中到聚餐地点，矮人A 要么开车到矮人B 家中，留下自己的轿车在矮人B 家，然后乘坐B 的轿车同行；要么直接开车到聚餐地点，并将车停放在聚餐地。虽然矮人的家很大，可以停放无数量轿车，但是聚餐地点却最多只能停放K 辆轿车。给你一张加权无向图，描述了N 个矮人的家和聚餐地点，求出所有矮人开车最短总路程。

例题2：POJ 1639 Picnic Planning

这是在一个特殊点V0所连边数有限制(不大于K)条件下求最小生成树的问题。

- 最土解法: 枚举去掉V0的一些边，使得V0的度变为K的方案数，对每种方案求一个最小生成树，然后在所有的最小生成树里再取最小的。
- 复杂度极高，但是本题数据弱，可以过。

- 还是应当掌握度限制生成树的正规算法

最小度限制生成树的定义

- 设G=(V,E,ω)是连通的无向图，v0 ∈V是特别指定的一个顶点，k为给定的一个正整数。V0在T中的度为$D_T(v0)$。

- 如果T是G的一个生成树且$D_T(v0)=k$，则称T为G的k度限制生成树。G中权值和最小的k度限制生成树称为G的最小k度限制生成树。

概念

- T为图G的一个生成树，a,b是G的边，T+a-b记作(+a,-b)，如果T+a-b仍然是一个生成树，则称(+a,-b）是T的一个可行交换。

- T为图G的一个生成树，由T进行一次可行交换得到的新的生成树所组成的集合，称为T的邻集,记为N(T)。

定理

* 定理：设T是图G的最小k度限制树，E0是G中与v0有关联的边的集合，

  E1=E0\E(T)，

  E1即是和V0相连，但是不在T中的边的集合

  E2=E(T)\E0，

  E2即是T中不和V0相连的边的集合

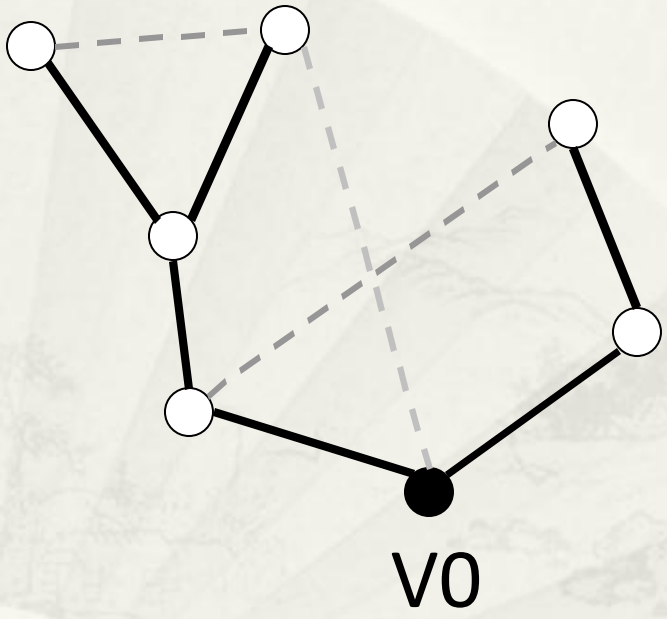
  A={(+a,-b)| a∈E1,b∈E2}

  设ω(a')－ω(b')=min{ω(a)－ω(b)| (+a,-b)∈A}，则T + a'-b'是G的一个最小k+1度限制生成树。

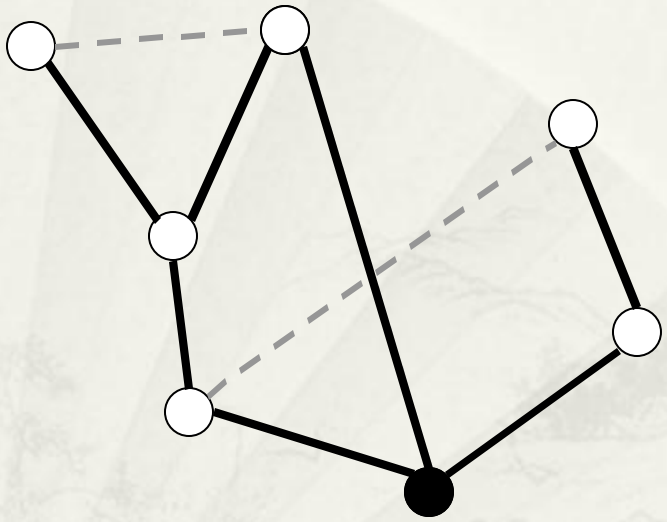
* 即最小k+1度限制生成树属于最小k度限制生成树的邻集。

- 假设我们已经得到了最小p度限制生成树，如何通过它来求最小p+1度限制生成树呢？

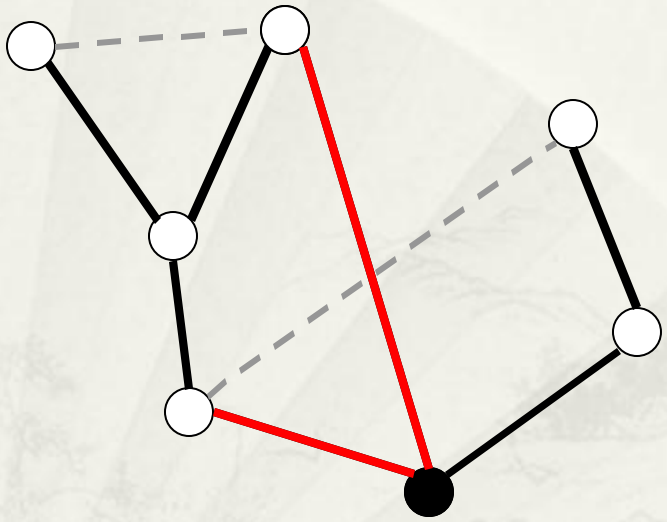
如图，假设我们已经得到了v0度为2时的最小生成树，现在要求v0度为3时的最小生成树。



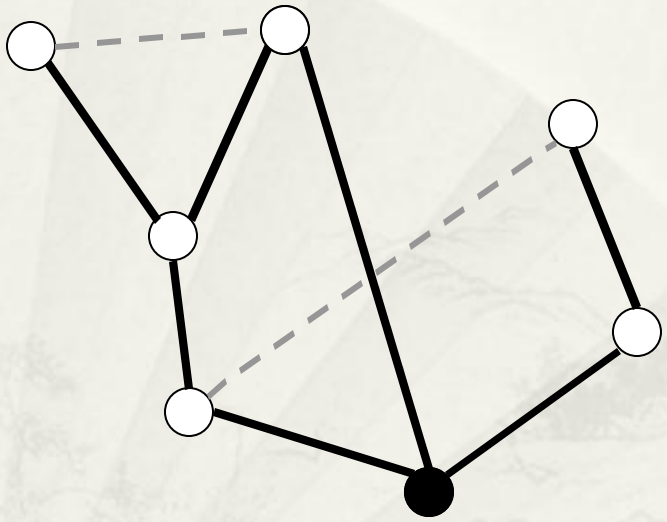
枚举与V0关联且不在树上的边，添加到树上。



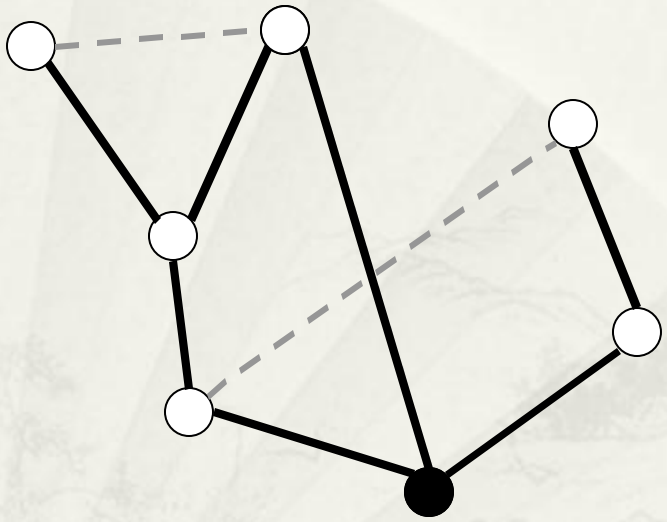
为了使V0的度增加，下面两条边红色的边是不能删除的。



删去边的权值越大，所得到的生成树的权值和就越小，因此，需要找到环上可删除的权值最大的边并将其删除。



简单的枚举，时间复杂度非常高。



应该使用动态规划！

动态规划

- 设最小p度限制生成树为T，T是无根树，为了简便，我们把v0作为该树的根。
- 定义Father(v)为T中v的父结点，Father(v0)无意义。
- 设Best(v)为路径v0->v上与v0无关联且权值最大的边。

动态规划

* Best(v)的状态转移方程为
* Best(v)=max(Best(Father(v)),ω(Father(v),v))

* 边界条件为
* Best[v0]=-∞，{Best[v’]=-∞|(v0,v’)∈E(T)}。

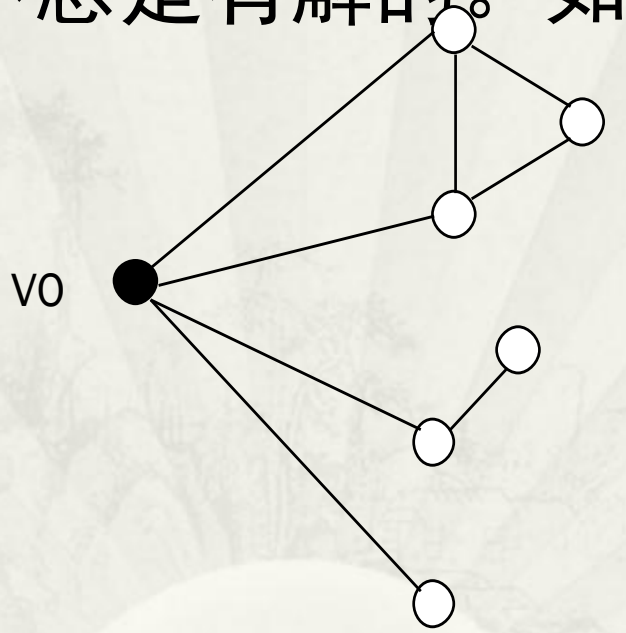
动态规划

- 状态总共|V|个，而状态转移的时间复杂度为O(1),因而总的时间复杂度是O(V),即通过最小p度限制生成树求最小p+1度限制生成树的时间复杂度是O(V)。

- 具体实现：从V0开始做一遍广搜即可。
- 问题：最先求几度的最小度限制生成树呢？即p从多少开始求？

答案

- 因为求最小k度限制生成树，当 $k < D_G(v0)$ 时，问题并不总是有解的。如图。



- 所以如果将v0去掉，图会被分成m个连通支，那么就先求最小m度限制生成树。

如何求最小m度限制生成树

* 1）我们可以先删去v0，对各个连通分量求最小生成树。

  具体实现：枚举每个顶点作为起点执行Prim算法，执行的过程中对同一连通支的顶点用相同数字标记。

* 2）再在每个连通分量中找最小边和v0相连。

* 3）得到最小m度限制生成树。

时间复杂度分析

* 求最小m度生成树：O(VlogV+E);
* 最多k次从p度到p+1度的递推：k*O(V);
* 总复杂度：O(VlogV+E+kV);

例题3：还是通讯网络

- 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。

例题3：还是通讯网络

- 问：怎样合理的分配卫星和铺设线路，使得在保证每两座村庄之间都可以直接或间接地通讯的前提下，铺设线路的总长度最短。？
- 数据规模：0 <= k <= n <= 500

分析

- 我们增加一个节点：卫星。卫星到所有节点的权值都是 0 。
- 那么，我们要求的仍是一个最小生成树。只不过卫星最多只能与 k 个节点相连。
- 也就是说，节点：卫星的度限制为 k 。
- 这是一个最小度限制生成树的问题！

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

- 定理：次小生成树是最小生成树的邻集。

证明

- 证明：
- 可以证明下面一个强一些的结论：
- T是某一棵最小生成树，T0是任一棵异于T的树，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’的权比uv小 (否则,uv就应该在T中)。把u‘v’替换uv即得树T(i+1)。

- 特别地：取T0为任一棵次小生成树，则T(n-1)满足定义且跟T差一条边。上述定理得证。

算法

- 利用该定理，可以得到O(V^2)的算法，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)


谢谢