算法导论23-1 23-3 23-4三题详解

把牛妹带回家

23-1

设 $G=(V, E)$ 是一个无向连通图，在其上定义了权值函数 $w: E\to\mathbf{R}$，并假设 $|E|\geqslant|V|$，且所有边的权值都是不同的。

所谓次最优的最小生成树是这样定义的：设 $\mathcal{T}$ 为 G 的所有生成树的集合，并设 T' 为 G 的一棵最小生成树。那么，次最优的最小生成树就是这样的一棵最小生成树 T，它满足 $w(T)=\min\limits_{T''\in\mathcal{T}-\{T'\}}\{w(T'')\}$。

a）证明最小生成树是唯一的，但次最优最小生成树未必一定是唯一的。

b）设 T 是 G 的一棵最小生成树，证明存在边 $(u, v)\in T$ 和 $(x, y)\notin T$，使得 $T-\{(u, v)\}\cup\{(x, y)\}$ 是 G 的一棵次最优最小生成树。

c）设 T 是 G 的一棵生成树，且对任意两个顶点 u、$v\in V$，设 $\max[u, v]$ 是 T 中 u 和 v 之间唯一通路上的具有最大权值的边。请给出一个运行时间为 $O(V^2)$ 的算法，在给定 T 和所有顶点 u、$v\in V$ 以后，它可以计算出 $\max[u, v]$。

d）写出一个有效的算法来计算 G 的次最优最小生成树。

把以上翻译成人话就是

边的权值不相同，证明最小生成树（MST）唯一，次优次小生成树（SST不唯一）

把牛妹带回家

算法计算SST

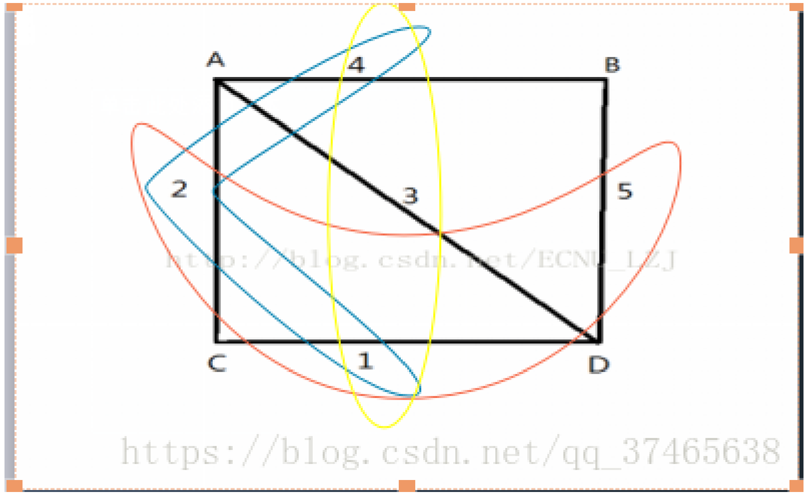
第一问

边的权值不相同，证明最小生成树（MST）唯一，次优次小生成树（SST不唯一）

假设T,T'都为MST，并且{e1,e2……en}∈T，{e'1,e'2……e'n}∈T'，因为权值都不相同，假设ek是所有边中最小的，连接的点时(u,v),而在T'中连接(u,v)的是e'k,因为w(ek)<w(e'k)这意味着如果T'用ek作为连接(u,v)的边，他可以更小，与T'为最小生成树矛盾，所有最小生成树唯一

SST不为一

如图，1,2,4为MST，但是可以看到SST有两条



第二问

把牛妹带回家

如果最小生成树T和次小生成树有两条边不同，即T' = T - {(u1, v1)} + {(x1, y1)} - {(u2, v2)} + {(x2, y2)}，则可以构造出一棵和最小生成树只有一条边不同的生成树T'' = T - {(u1, v1)} + {(x1, y1)}，使得w(T) < w(T'') < w(T')。这和T'是SST生成树矛盾，所以SST和MST只有一条边不同。

第三问

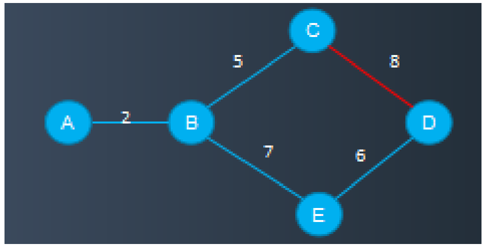
令max（u，v）是两点间最大权值的边，用O（V^2）算法实现

列出所有边的权重，由于权重都是随机排布的，先排序然后找到中值，然后不断朝右边做搜寻，最终找到权值最大的边，这样的算法是典型的找最值的办法，网上的方法很多，就不赘述了

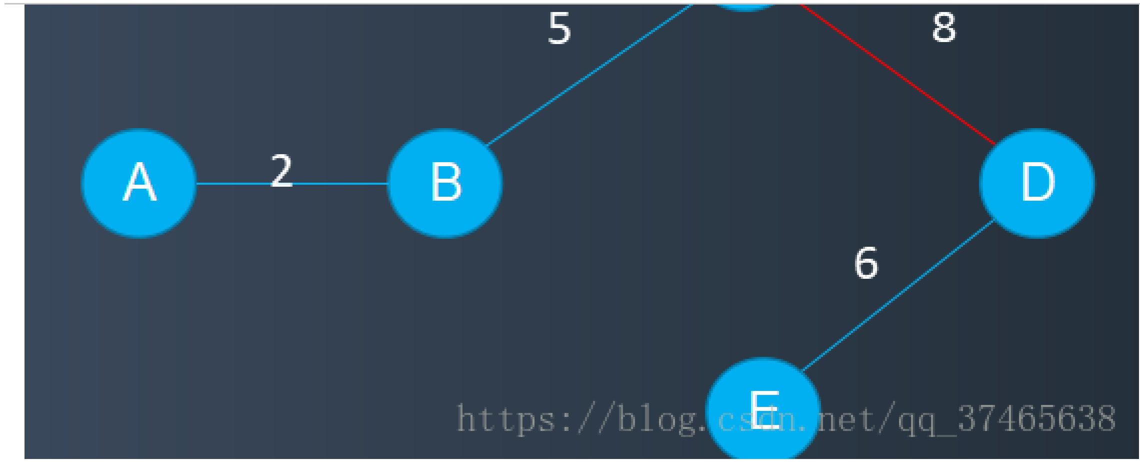
第四问

算法计算SST(代码我目前的水平还写不了，本来想直接复制的，感觉不太负责任，想看的话网上可以搜)

先计算MST，然后再纳入一条不属于MST的（u,v），此时会形成环，所以去掉一条max(u,v),必须是(u,v)路线中属于MST的最大权，否则就得不到SST



从图中可以看出，蓝色边为MST，



此时连接CD，形成环，在环中找到除了新加入的边之外的最大权值边，就是BE然后删掉

就得到了SST

23-3

23-3 （瓶颈生成树） 无向图 G 的**瓶颈生成树** T 是 G 的一棵生成树，其最大边的权重是 G 的所有生成树中最小的。我们称瓶颈生成树 T 的值是 T 中最大权重边的权重。

a. 证明：最小生成树是瓶颈生成树。

本题的(a)部分显示，找出一棵瓶颈生成树并不比找出一棵最小生成树更难。在本题余下的部分，我们就来演示如何在线性时间内找到一棵瓶颈生成树。

b. 请给出一个线性时间的算法，在给定图 G 和整数 b 的情况下，能够判断瓶颈生成树的值是否最大不超过 b。

c. 使用本题(b)部分的算法，设计一个瓶颈生成树问题的线性时间算法，该算法将以(b)部分的算法作为子程序。（提示：考虑使用一个子程序来对边的集合进行收缩，就如思考题 23-2 中所描述的 MST-REDUCE 算法一样。）

把牛妹带回家

假设e为MST的最大权值边，连接着(u,v)两个点，那么存在BST，其连接(u,v)两点的边为e',,可知e'<e,也就是说MST中e换成e'后可以更小，与原假设矛盾，所有MST就是BST

b要让值最大不超过b,方法是去除掉图中所有值超过b的边，然后检验一下剩下的边是否可以形成一棵BST，可以使用广度优先遍历(BFS）或者深度优先遍历(DFS)，使用邻接表可以保证是线性时间算法

c这个理解起来有点复杂需要二分法+MSTREDUCE收缩

首先我们要理解一个概念:MSTREDUCE收缩

简单讲就是有两个点u,v，找到点间的最小权重的边w(u,v)那么就把两点合成一个点，以后只要有其他点连上其中一个，就顺带连上了另一个，照样可以生成最小生成树，同时也简化了图

现在我们用b中的算法进行检查，将所有权值从小到大排序，取中值分两半，在较小一半进行BFS,

如果能找到一棵树，那么意味着这棵树的最大权重边最多是整个图权重边的中位数，然后继续二分循环，直到找不到树为止，那么上一棵树就是BST

如果一开始就没有树，那么就收缩一次，然后按上面的方法计算

运行k次MST-REDUCE时间为O(kE)，故总时间为

O(kE’) + O(E+V’lgV‘)= O(kE+V‘lgV’)，由于每运行一次MST-REDUCE，顶点数目至少减半，故k次后,V‘ <= ((1/2)^k)V，最好情况k = lglgV时，时间为O(Elg(lgV))。

23-4

23-4 （第三种最小生成树算法） 在本题中，我们给出三种不同算法的伪代码。每种算法的输入都是一个连通图和一个权重函数，返回值都是一个边的集合 T。对于每种算法，要么证明 T 是一棵最小生成树，要么证明 T 不是一棵最小生成树。同时给出每种算法的最有效的实现（不管该算法是否能够计算出最小生成树）。

a. MAYBE-MST-A(G, w)

```
sort the edges into nonincreasing order of edge weights w
T = E
for each edge e, taken in nonincreasing order by weight
    if T - {e} is a connected graph
        T = T - {e}
return T
```


这道题看着挺长，其实很简单

a让证明将边按非递增排列，然后逐一减去最大的边，剩下的就是MST

正确，因为每次减掉的都是权值最大的边，所以作为MST组成部分的轻量边会一直保存，直到图中出现一棵树，那就是最小生成树

也可以这样理解，生成树的过程就是不断去掉环的过程，逐一减去最大的边意味着每次都把一个环中的权值最大的边去掉，最后必定生成最小生成树

```
for each edge e, taken in arbitrary order
    if T∪{e} has no cycles
        T=T∪{e}
return T
```

c. MAYBE-MST-C(G, w)

```
T=∅
for each edge e, taken in arbitrary order
    T=T∪{e}
    if T has a cycle c
        let e′ be a maximum-weight edge on c
        T=T−{e′}
return T
```

b错误，因为去除的边是随机的，所以如果去掉了某个环中的轻量边，保留了权值较大的边，就不是MST

c正确，理由类似于a中的第二种方法