Hw3
窦嘉伟　518021911160

4-1
True
　假设所有的最小生成树都不包含边 e，即添加 e 边之后会出现环。假设 e 连接 A，B 两点，则切断 B 与其父节点的边 E 后生成一颗新树，由于 e 是最小边，该树比源最小生成树还小。产生矛盾。

4-8
假设存在两条最小生成树，则必然存在不同的边 E，E 存在 T2 而不在 T1 中，在 T1 中，如果添加 E 就会出现一个环，如果 E 是这个环中最大的边，那么 E 就不会属于任何一个最小生成树。矛盾。

4-9
1） False
　假设这样一个 G 有这样的边　（1，2）=3
　（1，3）=1
　（2，3）=2
　（3，4）=8
　（3，4）=10
由 2，3，8 边组成的树也是最小瓶颈树，但不是最小生成树 。

2） True
　假设存在最小生成树 T 不是最小瓶颈树，那么存在最小瓶颈树 S，取 T 中最大边 E 放入 S 中，则有 T 大于 S 中任何一条边，并且生成一个环，E 在该环中最大。所以 E 不是任何最小生成树的边。矛盾

4-29
假设 L={d1，d2，d3.。。。dn} ，在 L 上递归，首先判断有没有 0，如果有，剔除该点。否则，排序，令 d1>d2>d3...>dn。将前 dn 个点入度-1 并去掉点 dn，得到新的 L‘={d1-1,d2-1,d3-1$d_{dn-1}-1, d_{dn}-1.....d(n-1)\}$.
当且仅当存在图满足度为 L’时，存在图满足度为 L。
对 L’不断递归，最后只剩两个度，即可判断是否存在这样的图。
　下面证明上述结论。
左推右：只需添加一个点与前 dn 个点相连即可
右推左：即证有一个点与前 dn 个点相连，删去这个点即可。
　假设不存在这样的点。令删去的点为 Vn。若存在 Vn 与 Vj 相连而不与 Vi 相连（i<=dn<j)，那么由 di>dj 可知存在 V 与 Vj 不相连，与 Vi 相连，那么我们把 Vn-Vj 和 V-Vi 改为 Vn-Vi 和 V-Vj,可保证度数不变。如此下去，可得到这样的 Vn。删去 Vn 即可

5-1
首先将 AB 数据库排序，每个数据库有 n 个元素，A[n/2]为 A 中位数，B 同理。
将比较两者中位数，若 A 中位数> B 中位数，则去除 A 的后半部分和 B 的前半部分，进行递归。
算法如下

```
M(n,a,b) //n 为当前两个数组进行计算的元素个数
{
K=⌈n/2⌉;
If (a[k]>b[k])
Return M(k,a+⌊n/2⌋,b);
Else return M(k,a,b+⌊n/2⌋);
}
```

5-4
利用快速傅立叶变换实现复杂度 O（2logn+n）

5-7
用 B 来代表图 G 的边界，假设
定义性质*：如果 G 包含一个不属于 B 的节点并
且 v 与 B 中的一个节点相邻，并且也满足 v<B 那么 G 就有一个性质（*）。
则在有性质* 的图中，最小不在 B 中（因为 v<B）,因此这个图至少有一个不在边界上的局部最小，把这个局部最小叫做内部局部最小。
因此，假设 G 有性质（*），假设 v 是不在 B 中且比 B 中所有节点小的节点。用 C 来表示 G 中中间的行和列（不算 G 的边界）。用 S 来表示 B 并 C，在 G 中删除 S，得到四个子图。用 T 来表示所有和 S 相连的节点。 用 O(n)的时间，我们找到 S 并 T 中最小的值对应的节点 u，u 不在 B 中，因为图满足性质 *
因此有两种情况：
第一种是 u 在 C 中，这种情况下 u 是内部局部最小，因为 u 所 有的邻居都是内部节点，并且 u 是其中最小的。
第二种情况是 u 在 T 中，用 G1 来表示包含 u 的 G 的子图，同时 G1 也包含在 S 中的边界部分。现在我们说 G1 也满足性质*，因为 u 也与 G1 的边界相连，并且小于边界的值，因此 G1 也有一个内部局部最小，同时这个也是 G 的内部局部最小。继续递归，能够得到最后的内部局部最小。
算法复杂度可以表示为 T（n）=T（n/4）+O（n）　　=O（n）