3.1

因为y是x的父节点，且该二叉树的每个节点都是不同的。

所以我们分两种情况讨论

- x是y的左子节点，证明y是T树中比x的更大的节点中的最小值

因为x是y的左叶子节点，所以$x < y$，反证法，如果y不是T树中比x的更大的节点中的最小值，m是的。那么有$x < m < y$，所以m肯定在节点y的左子树中，但是y的左子树只有一个x叶子节点。所以假设不成立。及证y是T树中比x的更大的节点中的最小值

- x是y的右子节点，证明y是T树中比x的更小的节点中的最大值

因为x是y的右叶子节点，所以$x > y$，反证法，如果y不是T树中比x的更小的节点中的最大值，n是的。那么有$x > n > y$，所以n肯定在节点y的右子树中，但是y的右子树只有一个x叶子节点。所以假设不成立。及证y是T树中比x的更小的节点中的最大值

3.2

最坏情况的时间复杂度为$O(n^2)$

当插入的数据是已经排好序的数据时，比如1,2,3....n，由于二叉树的特性。这样的数据序列会产生一棵高度为n的倾斜二叉树，所以插入的总时间为$O(0+1+2+...n-1)=O(n^2)$，排序好之后的中序遍历时间复杂度为$O(n)$，因此总的时间复杂度还是$O(n^2)$

最好的情况时间复杂度为$O(nlogn)$

当插入的数据刚好形成一颗满二叉平衡树的时候，数据的每次插入的时间为$O(logn)$，所以总的插入时间复杂度为$O(nlogn)$，排序好之后的中序遍历时间复杂度为$O(n)$，因此总的时间复杂度还是$O(nlogn)$

3.3

一共有n个元素，m个桶，任意两个元素在一起就会产生一次冲突，所以n个元素在一个桶内总共可产生$C_n^2=\frac{n(n-1)}{2}$次冲突

因为有m个桶，所以任意两个元素在某个桶内发生冲突的概率为

$$p=\frac{1}{m}\cdot\frac{1}{m}\cdot m=\frac{1}{m}$$

所以期望数为

$$C_n^2\cdot p=\frac{n(n-1)}{2m}$$

3.4

(a)

最坏情况的时间复杂度为$\Omega(n)$,因为采取的是一致性哈希。所以每个元素落在某个桶内的概率都是相同的。因此根据鸽巢定理，因为$M > n^2$，所以至少有一个桶里的元素至少有n个(在插入完所有的M个元素之后)。所以当search的元素哈希到含有n个元素的桶里的时候，无论链表是如何设计的，最坏的情况下会搜索完所有n个元素，所以最坏的时间复杂度肯定为$\Omega(n)$

(b)

我们将每个桶内的链表换成红黑树。h(x)存储的是指向红黑树的root结点的指针

我们首先关注期望的时间复杂度是多少，虽然我们将链表的结构转换为了红黑树的结构，但是对于增删查操作期望时间复杂度依然是O(1)，因为对所有的u_i，满足$h(u_i) = h(u_j)$的u_j个数依然是$O(1)$，这也意味着$A[h(u_i)]$中红黑树的期望大小也为$O(1)$

然后是最坏的情况下，因为在最坏情况下每个桶内的元素个数为n，及红黑树结点个数为n，由于红黑树是一颗自平衡的二叉树，所以它的高度不会超过$2log(n)$，所以最坏情况下的增删查的时间复杂度为，哈希上的复杂度乘以红黑树上的复杂度:$O(1) * O(logn) = O(logn)$

3.5

(a)

每个网站需要$log(M)$bits来存储，该黑名单中有w个恶意网站，所以一共需要$b = wlog(M)$bits

(b)

证明:

- (Small space)

首先我们需要用n = 100w位比特来存储哈希表，并且还需要d位比特来存储哈希函数，所以总共需要b = 100w+ d位比特

- (No false negatives)

当我们对一个恶意网站x进行哈希的时候,$T[h(x)] = 1$,因为在我们初始化哈希表的时候我们已经将黑名单中的网站哈希之后将T置为1了。

所以如果某个恶意网站m经过$isMalicious(m)$，返回值一定为true，所以任何一个恶意网站都不可能被误分类。

- (Unlikely false positives)

根据universal hash family的特征，对于任意一个网站y，所有其他任意网站x与它冲突的概率最多为$1/n$，$(P(h(y) = h(x)) <= \frac{1}{n})$，因此所有网站与黑名单中的网站发生冲突的概率为$\frac{w}{n} = \frac{w}{100w} = \frac{1}{100} = 0.01$

所以如果一个网站x不在黑名单中，那它只有0.01的概率会被认为$T[h(x)] = 1$，至少有0.99的概率不会被认为是恶意网站。及$T[h(x)] = 0$，返回false

(c)

证明:

- (Small space)

每个哈希表的bucket里需要1个比特，所以一个哈希表就需要$n = 2w$个比特，10个哈希表就需要$10n = 20w$个比特。因为h为universal hash family里的随机哈希函数，所以每个哈希函数需要$d = O(logM)$比特的空间，10个哈希函数一共就需要10d个比特的空间。所以一共为$b = 10d + 20w$比特的空间

- (No false negatives)

因为我们在初始化的时候，将黑名单中的每一个网站w都哈希进了10个哈希表对应的桶中，及

$$T_1(h_1(w)) = 1, T_2(h_2(w)) = 1 ... T_1 0(h_1 0(w)) = 1$$

所以如果某个恶意网站m经过$isMalicious(m)$，返回值一定为true，所以任何一个恶意网站都不可能被误分类。

- (Unlikely false positives)

根据universal hash family的特征，对于任意一个网站y，所有其他任意网站x与它冲突的概率最多为$1/n$，$(P(h(y) = h(x)) <= \frac{1}{n})$，因此所有网站与黑名单中的网站发生冲突的概率为$\frac{w}{n} = \frac{w}{2w} = \frac{1}{2} = 0.5$，但是这仅是一个哈希表的桶里发生了冲突，而导致$isMalicious(m)$判断为true则至少需要在10个桶内都发生冲突才行，概率也就是

$$(\frac{w}{n})^{10} = (\frac{w}{2w})^{10} = (\frac{1}{2})^{10} = \frac{1}{1024} < 0.001$$

所以任何一个非恶意的网站被$isMalicious()$判断为安全网站的概率大于$1 - 0.001 = 0.999$

因此，这种方法**在准确率方面**比第二题(b)中的0.99还要高

在存储空间上

$$\begin{gathered} 10d + 20w < d + 100w \\ 80w > 9d \\ w > \frac{9}{80}d \end{gathered}$$

当黑名单的数量大于$\frac{9}{80}d$时，第三题中的存储会比第二题中的存储还要小。