[**面试题——轻松搞定面试中的红黑树问题**](https://www.cnblogs.com/wuchanming/p/4444961.html)

版权所有，转载请注明出处，谢谢！  
<http://blog.csdn.net/silangquan/article/details/18655795>

   连续两次面试都问到了红黑树，关键两次都没有答好，这次就完整地来学习整理一下。

没有学习过红黑树的同学请参考：

<<Introduction to Algorithms>> Chapter 13 Red-Black Trees Chapter 14 Augmenting Data Structures

[教你透彻了解红黑树](http://blog.csdn.net/v_JULY_v/article/details/6105630)

**1.stl中的set底层用的什么数据结构？**

**2.红黑树的数据结构怎么定义的？**

**3.红黑树有哪些性质？**

**4.红黑树的各种操作的时间复杂度是多少？**

**5.红黑树相比于BST和AVL树有什么优点？**

**6.红黑树相对于哈希表，在选择使用的时候有什么依据？**

**7.如何扩展红黑树来获得比某个结点小的元素有多少个？**

**8.扩展数据结构有什么步骤？**

**9 为什么一般hashtable的桶数会取一个素数**

**详细解答**

**1.stl中的set底层用的什么数据结构？**

红黑树

**2.红黑树的数据结构怎么定义？**

**[cpp]** [view plaincopy](http://blog.csdn.net/silangquan/article/details/18655795)

1. **enum Color**
2. {
3. RED = 0,
4. BLACK = 1
5. };
7. **struct RBTreeNode**
8. {
9. **struct RBTreeNode\*left, \*right, \*parent;**
10. int   key;
11. int data;
12. Color color;
13. };

**3.红黑树有哪些性质？**

一般的，红黑树，满足以下性质，即只有满足以下全部性质的树，我们才称之为红黑树：  
1）每个结点要么是红的，要么是黑的。  
2）根结点是黑的。  
3）每个叶结点（叶结点即指树尾端NIL指针或NULL结点）是黑的。  
4）如果一个结点是红的，那么它的俩个儿子都是黑的。  
5）对于任一结点而言，其到叶结点树尾端NIL指针的每一条路径都包含相同数目的黑结点。

**4.红黑树的各种操作的时间复杂度是多少？**

能保证在最坏情况下，基本的动态几何操作的时间均为O（lgn）

**5.红黑树相比于BST和AVL树有什么优点？**

红黑树是牺牲了严格的高度平衡的优越条件为代价，它只要求部分地达到平衡要求，降低了对旋转的要求，从而提高了性能。红黑树能够以O(log2 n)的时间复杂度进行搜索、插入、删除操作。此外，由于它的设计，任何不平衡都会在三次旋转之内解决。当然，还有一些更好的，但实现起来更复杂的数据结构能够做到一步旋转之内达到平衡，但红黑树能够给我们一个比较“便宜”的解决方案。

相比于BST，因为红黑树可以能确保树的最长路径不大于两倍的最短路径的长度，所以可以看出它的查找效果是有最低保证的。在最坏的情况下也可以保证O(logN)的，这是要好于二叉查找树的。因为二叉查找树最坏情况可以让查找达到O(N)。

红黑树的算法时间复杂度和AVL相同，但统计性能比AVL树更高，所以在插入和删除中所做的后期维护操作肯定会比红黑树要耗时好多，但是他们的查找效率都是O(logN)，所以红黑树应用还是高于AVL树的. 实际上插入 AVL 树和红黑树的速度取决于你所插入的数据.如果你的数据分布较好,则比较宜于采用 AVL树(例如随机产生系列数),但是如果你想处理比较杂乱的情况,则红黑树是比较快的

**6.红黑树相对于哈希表，在选择使用的时候有什么依据？**

权衡三个因素: 查找速度, 数据量, 内存使用，可扩展性。  
　　总体来说，hash查找速度会比map快，而且查找速度基本和数据量大小无关，属于常数级别;而map的查找速度是log(n)级别。并不一定常数就比log(n) 小，hash还有hash函数的耗时，明白了吧，如果你考虑效率，特别是在元素达到一定数量级时，考虑考虑hash。但若你对内存使用特别严格， 希望程序尽可能少消耗内存，那么一定要小心，hash可能会让你陷入尴尬，特别是当你的hash对象特别多时，你就更无法控制了，而且 hash的构造速度较慢。

红黑树并不适应所有应用树的领域。如果数据基本上是静态的，那么让他们待在他们能够插入，并且不影响平衡的地方会具有更好的性能。如果数据完全是静态的，例如，做一个哈希表，性能可能会更好一些。  
  
在实际的系统中，例如，需要使用动态规则的防火墙系统，使用红黑树而不是散列表被实践证明具有更好的伸缩性。Linux内核在管理vm\_area\_struct时就是采用了红黑树来维护内存块的。

红黑树通过扩展节点域可以在不改变时间复杂度的情况下得到结点的秩。

**7.如何扩展红黑树来获得比某个结点小的元素有多少个？**

这其实就是求节点元素的顺序统计量，当然任意的顺序统计量都可以需要在O(lgn)时间内确定。

在每个节点添加一个size域，表示以结点 x 为根的子树的结点树的大小，则有

size[x] = size[[left[x]] + size [right[x]] + 1;

这时候红黑树就变成了一棵顺序统计树。

利用size域可以做两件事：

1). 找到树中第i小的结点；

**[cpp]** [view plaincopy](http://blog.csdn.net/silangquan/article/details/18655795)

1. OS-SELECT(x;,i)
2. r = size[left[x]] + 1;
3. **if i == r**
4. **return x**
5. elseif i < r
6. **return OS-SELECT(left[x], i)**
7. **else return OS-SELECT(right[x],  i)**

思路：size[left[x]]表示在对x为根的子树进行中序遍历时排在x之前的个数，递归调用的深度不会超过O(lgn);

2).确定某个结点之前有多少个结点，也就是我们要解决的问题；

**[cpp]** [view plaincopy](http://blog.csdn.net/silangquan/article/details/18655795)

1. OS-RANK(T,x)
2. r = x.left.size + 1;
3. y = x;
4. **while y != T.root**
5. **if y == y.p.right**
6. r = r + y.p.left.size +1
7. y = y.p
8. **return r**

思路：x的秩可以视为在对树的中序遍历种，排在x之前的结点个数加上一。最坏情况下，OS-RANK运行时间与树高成正比，所以为O (lgn).

**8.扩展数据结构有什么步骤？**

1).选择基础数据结构；

2).确定要在基础数据结构种添加哪些信息；

3).验证可用基础数据结构上的基本修改操作来维护这些新添加的信息；

4).设计新的操作。

**9 为什么一般hashtable的桶数会取一个素数**

设有一个[哈希函数](http://zhidao.baidu.com/search?word=%E5%93%88%E5%B8%8C%E5%87%BD%E6%95%B0&fr=qb_search_exp)  
H( c ) = c % N;  
当N取一个[合数](http://zhidao.baidu.com/search?word=%E5%90%88%E6%95%B0&fr=qb_search_exp)时，最简单的例子是取2^n，比如说取2^3=8,这时候  
H( 11100(二进制） ) = H( 28 ) = 4  
H( 10100(二进制) ) = H( 20 ）= 4  
  
这时候c的二进制第4位（从右向左数）就”失效”了，也就是说，无论第c的4位取什么值，都会导致H( c )的值一样．这时候c的第四位就根本不参与H( c )的运算，这样H( c )就无法完整地反映c的特性，增大了导致冲突的几率．  
  
取其他[合数](http://zhidao.baidu.com/search?word=%E5%90%88%E6%95%B0&fr=qb_search_exp)时，都会不同程度的导致c的某些位”失效”，从而在一些常见应用中导致冲突．  
但是取[质数](http://zhidao.baidu.com/search?word=%E8%B4%A8%E6%95%B0&fr=qb_search_exp)，基本可以保证c的每一位都参与H( c )的运算，从而在常见应用中减小冲突几率．．  
  
（个人意见：有时候不取[质数](http://zhidao.baidu.com/search?word=%E8%B4%A8%E6%95%B0&fr=qb_search_exp)效率也不会太差..但是无疑取[质数](http://zhidao.baidu.com/search?word=%E8%B4%A8%E6%95%B0&fr=qb_search_exp)之比较保险的..)