红黑树与AVL树

红黑树不追求"完全平衡"，即不像AVL那样要求节点的 |balFact| <= 1，它只要求部分达到平衡，但是提出了为节点增加颜色，红黑是用非严格的平衡来换取增删节点时候旋转次数的降低，任何不平衡都会在三次旋转之内解决，而AVL是严格平衡树，因此在增加或者删除节点的时候，根据不同情况，旋转的次数比红黑树要多。

插入:插入节点导致树失衡的情况，AVL和RB-Tree都是最多两次树旋转来实现复衡rebalance，旋转的量级是O(1)  
删除:删除节点导致失衡，AVL需要维护从被删除节点到根节点root这条路径上所有节点的平衡，旋转的量级为O(logN)，而RB-Tree最多只需要旋转3次实现复衡，只需O(1)，所以说RB-Tree删除节点的rebalance的效率更高，开销更小！

针对插入和删除节点导致失衡后的rebalance操作，红黑树能够提供一个比较"便宜"的解决方案，降低开销，是对search，insert ，以及delete效率的折衷，总体来说，RB-Tree的统计性能高于AVL.

故引入RB-Tree是功能、性能、空间开销的折中结果。  
5.1 AVL更平衡，结构上更加直观，时间效能针对读取而言更高；维护稍慢，空间开销较大。  
5.2 红黑树，读取略逊于AVL，维护强于AVL，空间开销与AVL类似，内容极多时略优于AVL，维护优于AVL。  
基本上主要的几种平衡树看来，红黑树有着良好的稳定性和完整的功能，性能表现也很不错，综合实力强，在诸如STL的场景中需要稳定表现。

红黑树的查询性能略微逊色于AVL树，因为其比AVL树会稍微不平衡最多一层，也就是说红黑树的查询性能只比相同内容的AVL树最多多一次比较，但是，红黑树在插入和删除上优于AVL树，AVL树每次插入删除会进行大量的平衡度计算，而红黑树为了维持红黑性质所做的红黑变换和旋转的开销，相较于AVL树为了维持平衡的开销要小得多

总结：实际应用中，若搜索的次数远远大于插入和删除，那么选择AVL，如果搜索，插入删除次数几乎差不多，应该选择RB。