MISTAKES

- 重载运算符：友元函数的定义没有搞清楚。申明在一个类内，表示另一个函数（类）为这个类的友元。
- 类模板的实例化、全局类和嵌套类：全局类默认了`Compare=std::less<T>`，类中再调用全局类的指针的时候，`Compare`必须显示地指出从全局类继承而来，不能使用默认参数，否则编译器无法识别为同一个`Compare`类。
- `value_type`的类型为`pair<Key,T>`，无默认构造函数，只有拷贝构造函数，选择指针的方式会更加方便。注意`value_type *`的析构。
- `begin()`、`end()`函数复杂度为`O(1)`。
- `dfs_copy()`函数，新的叶子节点赋`nil`，且注意两棵树不同的`nil`节点是不同的。
- `dfs_destroy()`函数，给最后的根节点赋`nil`。
- `const`型的类的`const`指针，不能赋值给非`const`型的指针，限制修改。
- `const`型的类调用成员函数只可以调用`const`型的成员函数，如果有解引用`*`或引用`&`作为返回值，返回值也需要为`const`类型。
- 求`successor`、`predecessor`的时候，不能从根节点往下二分查找，要从当前节点出发。
- 嵌套类`iterator`、`const_iterator`为全局类`map<>`的友元，反之不成立，所以`map<>`的成员函数可能不能访问`iterator`中的私有成员，这里需要在`iterator`中将`map<>`申明为友元。
- `make_pair`里面的参数为自己制定的类时，好像不太好用，所以尽量用`pair`制定参数后的构造函数，`pair<T1,T2>(val_1,val_2)`。
- `rb_tree`的`erase`操作时，一般是将删除节点转移成其`SUCCESSOR`的问题，因为`SUCCESSOR`至多只有一个儿子。但不能直接交换被删除节点和`SUCCESSOR`的值，这会导致迭代器出问题。应该将两个节点在树中的位置进行交换，这样就保证了迭代器的不变性。注意维护它们和父亲、儿子的关系。