

二叉搜索树

概述：循关键码访问



08 - A7

There's nothing in your head the sorting hat can't see.  
So try me on and I will tell you where you ought to be.

邓俊辉

deng@tsinghua.edu.cn

# 循关键码访问

- ❖ 很遗憾，向量、列表并不能兼顾静态查找与动态修改
- ❖ 能否综合二者的优点？
- ❖ 各数据项依所持**关键码**而彼此区分：call-by-KEY
- ❖ 当然，关键码之间必须同时支持**比较（大小）**与**比对（相等）**
- ❖ 数据集中的数据项，统一地表示和实现为词条（entry）形式



基本结构	查找	插入/删除
无序向量	$\Theta(n)$	$\Theta(n)$
有序向量	$\Theta(\log n)$	$\Theta(n)$
无序列表	$\Theta(n)$	$\Theta(1)$
有序列表	$\Theta(n)$	$\Theta(n)$

# 词条

```
❖ template <typename K, typename V> struct Entry { //词条模板类  
    K key; V value; //关键码、数值  
  
    Entry( K k = K(), V v = V() ) : key(k), value(v) {}; //默认构造函数  
  
    Entry( Entry<K, V> const & e ) : key(e.key), value(e.value) {}; //克隆  
  
    // 比较器、判等器（从此，不必严格区分词条及其对应的关键码）  
  
    bool operator< ( Entry<K, V> const & e ) { return key < e.key; } //小于  
    bool operator> ( Entry<K, V> const & e ) { return key > e.key; } //大于  
    bool operator==( Entry<K, V> const & e ) { return key == e.key; } //等于  
    bool operator!=( Entry<K, V> const & e ) { return key != e.key; } //不等  
};
```