

向量

无序向量：查找

02-02

他便站将起来，背着手踱来踱去，侧眼把那些人逐个个觑将去，内中一个果然衣领上挂着一寸来长短彩线头。

邓俊辉

deng@tsinghua.edu.cn

无序向量：判等器

```
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

    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

    /* ... */

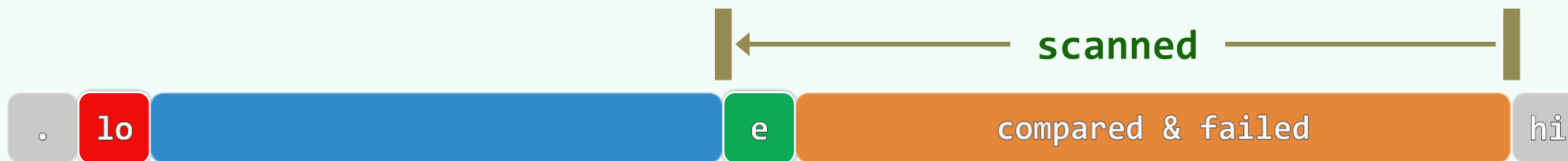
};
```

有序向量：比较器

```
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  
    bool operator== ( Entry<K, V> const& e ) { return key == e.key; } //等于  
    bool operator!= ( Entry<K, V> const& e ) { return key != e.key; } //不等于  
    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  
}; //得益于比较器和判等器，从此往后，不必严格区分词条及其对应的关键码
```

顺序查找

```
template <typename T> Rank Vector<T>:: //  $O(hi - lo) = O(n)$   
find( T const & e, Rank lo, Rank hi ) const { //  $0 \leq lo < hi \leq \_size$   
    while ( (lo < hi--) && (e != _elem[hi]) ); // 逆向查找  
    return hi; // 返回值小于lo即意味着失败；否则即命中者的秩（有多个时，返回最大者）  
}
```



❖ 输入敏感 (input-sensitive) : 最好 $O(1)$, 最差 $O(n)$