

11-C7

词典

排解冲突：重散列

“天地會壞否？”

“不會壞。只是相將人無道極了，便一齊打合，混沌一番，人物都盡，又重新起。”

邓俊辉

deng@tsinghua.edu.cn

Rehashing

❖ `template <typename K, typename V> //随着装填因子增大，冲突概率、排解难度都将激增`

`void Hashtable<K, V>::rehash() { //此时，不如“集体搬迁”至一个更大的散列表`

`int oldM = M; Entry<K, V>** oldHt = ht;`

`ht = new Entry<K, V>*[M = primeNLT(4 * N)]; N = 0; //新表“扩”容`

`memset(ht, 0, sizeof(Entry<K, V>*) * M); //初始化各桶`

`release(removed); removed = new Bitmap(M); L = 0; //懒惰删除标记`

`for (int i = 0; i < oldM; i++) //扫描原表`

`if (oldHt[i]) //将每个非空桶中的词条`

`put(oldHt[i]->key, oldHt[i]->value); //转入新表`

`release(oldHt); //释放——因所有词条均已转移，故只需释放桶数组本身`

`}`

插入

❖ template <typename K, typename V>

```
bool Hashtable<K, V>::put( K k, V v ) { //散列表词条插入
```

```
    if ( ht[ probe4Hit( k ) ] ) return false; //雷同元素不必重复插入
```

```
    int r = probe4Free( k ); //为新词条找个空桶 ( 只要装填因子控制得当, 必然成功 )
```

```
    ht[ r ] = new Entry<K, V>( k, v ); ++N; //插入
```

```
    if ( removed->test( r ) ) { removed->clear( r ); --L; } //懒惰删除标记
```

```
    if ( (N + L)*2 > M ) rehash(); //若装填因子高于50%, 重散列
```

```
    return true;
```

```
}
```

删除

❖ template <typename K, typename V>

```
bool Hashtable<K, V>::remove( K k ) { //散列表词条删除算法
```

```
    int r = probe4Hit( k ); if ( !ht[r] ) return false; //确认目标词条确实存在
```

```
    release( ht[r] ); ht[r] = NULL; //清除目标词条
```

```
    removed->set(r); --N; ++L; //更新标记、计数器
```

```
    if ( 3*N < L ) rehash(); //若懒惰删除标记过多，重散列
```

```
    return true;
```

```
}
```