09-C4

词典

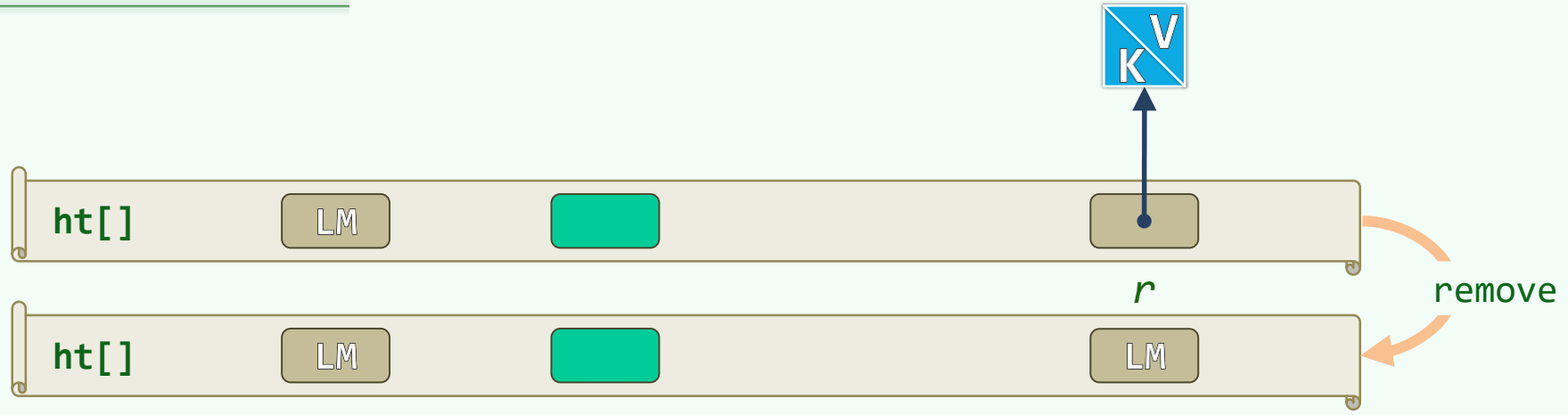
排解冲突：重散列

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

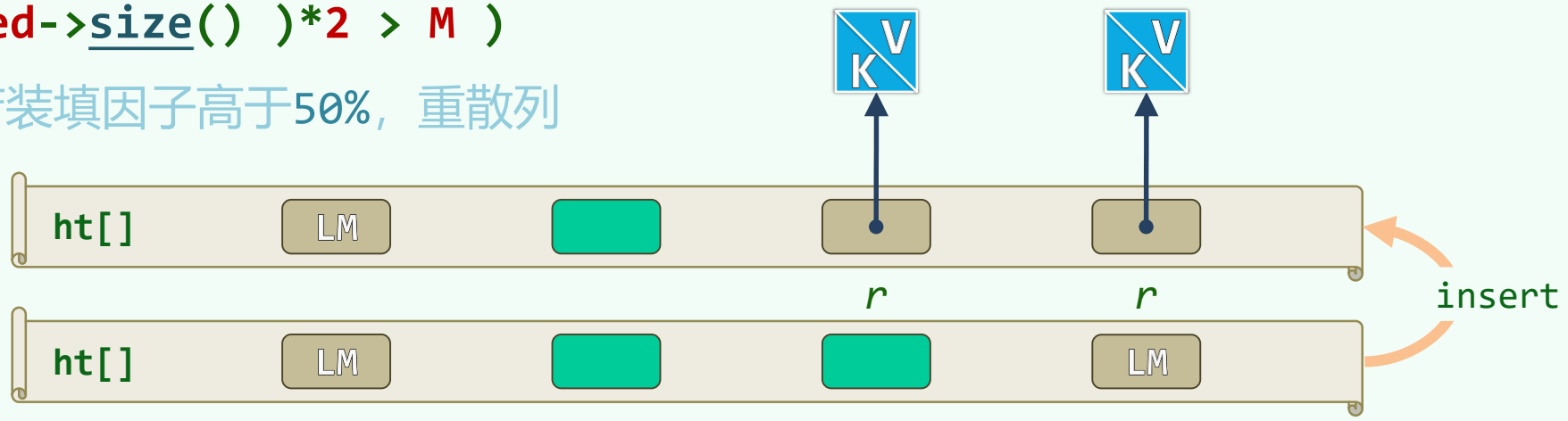
删除

```
template <typename K, typename V> bool Hashtable<K, V>::remove( K k ) {
   Rank r = probe4Hit( k ); if ( !ht[r] ) return false; //确认目标词条确实存在
   ---------------------------------------------------------------------
   delete ht[r]; ht[r] = NULL; --N; //清除词条
   removed->set(r); //设置LM标记
   ---------------------------------------------------------------------
   return true;
}
```



插入

```
template <typename K, typename V> bool Hashtable<K, V>::put( K k, V v ) {
   if ( ht[ probe4Hit( k ) ] ) return false; //忽略相等元素
   Rank r = probe4Free( k ); //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; //插入新词条
   removed->clear( r ); //清除LM标记
   if ( ( N + removed->size() )*2 > M )
      rehash(); //若装填因子高于50%，重散列
   return true;
}
```



Rehashing：随着装填因子攀升，冲突激增；超过阈值后，便需要扩容

```
template <typename K, typename V> void Hashtable<K, V>::rehash() {
   Rank 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
   delete removed; removed = new Bitmap(M); //懒惰删除标记
   for ( Rank i = 0; i < oldM; i++ )
      if ( oldHt[i] ) //原表中的每个词条
         put( oldHt[i]->key, oldHt[i]->value ); //逐个转入新表
   delete[] oldHt; //释放——因所有词条均已转移，故只需释放桶数组本身
}
```