词典

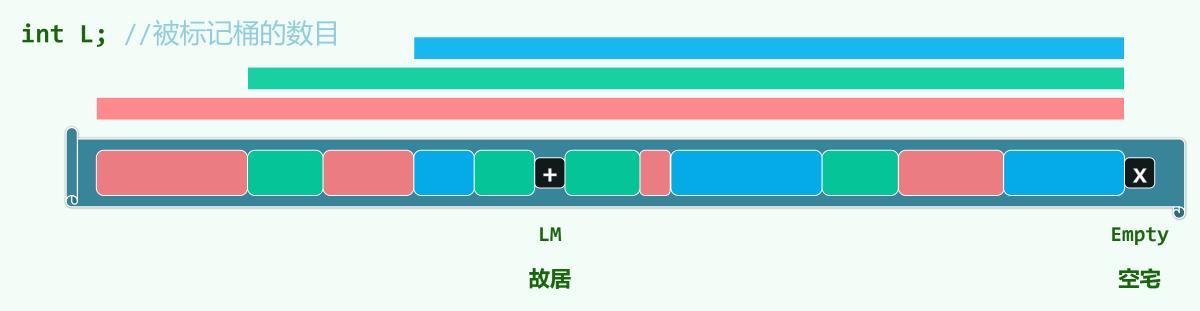
排解冲突: 懒惰删除

邓俊辉 deng@tsinghua.edu.cn

江山故宅空文藻,云雨荒台岂梦思 最是楚宫俱泯灭,舟人指点到今疑

Lazy Removal: 故居 ~ 空宅

❖ Bitmap* removed; //用Bitmap懒惰地标记被删除的桶



- ❖ 仅做标记,不对试探链做更多调整——此后,带标记的桶,角色因具体的操作而异
 - 查找词条时,被视作"必不匹配的非空桶",试探链在此得以延续
 - 插入词条时,被视作"必然匹配的空闲桶",可以用来存放新词条

两种试探算法

```
template <typename K, typename V> int Hashtable<K, V>::probe4Hit(const K& k) {
  int r = hashCode(k) % M; //按除余法确定试探链起点
  while ( ( ht[r] && (k != ht[r]->key) ) | removed->test(r) )
     r = (r + 1) % M; //线性试探 (跳过带懒惰删除标记的桶)
  return r; //调用者根据ht[r]是否为空及其内容,即可判断查找是否成功
template <typename K, typename V> int Hashtable<K, V>::probe4Free(const K& k) {
  int r = hashCode(k) % M; //按除余法确定试探链起点
  while ( ht[r] ) r = (r + 1) % M; //线性试探, 直到空桶 (无论是否带有懒惰删除标记)
  return r; //只要有空桶, 线性试探迟早能找到
```