哈希表

Ben

Outline

- 数据结构——平衡的艺术
- 哈希表
- 布隆过滤器

哈希表：定义

- 存放数据的集合
- 操作：根据（Key, Value）进行

  插入，查找，删除（可以没有）
- 空间复杂度：O(m)
- 单次操作时间复杂度：O(1)
- 本质：Key的索引

哈希表：例题

- 给出n个[0, m)范围内的整数，去重
- 快速排序
  - 期望时间复杂度　O(nlogn)
  - 附加空间复杂度　O(1)
- 计数（基数）排序
  - 时间复杂度　　O(n + m)　超越比较排序下限
  - 附加空间复杂度　O(m)

哈希表：思考

- 若n << m，计数排序的大量空间被浪费
- 只需判断是否出现过，优化？
- 将Key区间[0, m) 映射到 [0, p)
- H(key) = key mod p
- 若m > p, 多对一的映射方式

哈希表：实现

- 处理冲突（Key, Value）
  - 开放地址法（数组）
  - 拉链法　　（数组+链表）
- 负载率 = 已有元素大小 / 存储散列大小
- 最坏情况？
- 哈希函数设计

哈希表应用：字符串匹配

- 设字符串A= ‘12314123’
- 求‘123’在A中出现的次数
- 不会写KMP又想要O(n)肿么办？
- Key(‘123’) = ‘1’ * 10^2 + ‘2’ * 10 + ‘3’ * 1

  = 123
- A’ = [123, 231, 314, …, 123]

哈希表应用：字符串匹配

- Key相等时Value有可能不同
- 每次比较Value也是不小的开销，特别是Value可能很大
- 不考虑Value将产生错误率（错误率换时间）
- 多重哈希（降低错误率）

哈希表：麻烦的删除

- 能否直接删除哈希表中的元素？
- 考虑两种不同的实现方式
- 硬删除 vs 软删除

哈希表：思考题

- 设计一个动态平衡的哈希表
- 动态平衡
  - 负载率高 → 增大哈希表空间
  - 负载率低 → 减小哈希表空间
- 正确性测试+压力测试

哈希表：作业题

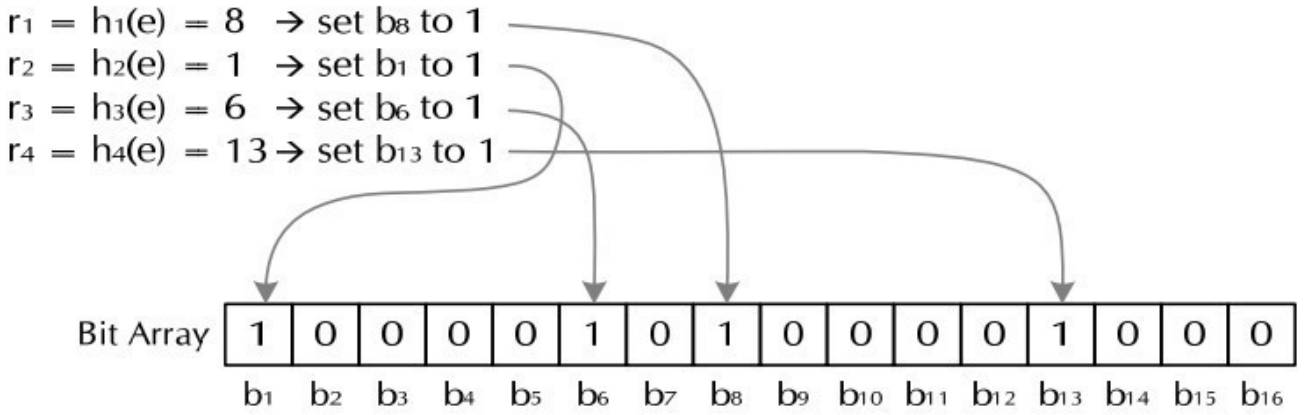
- Leetcode 128. Longest Consecutive Sequence

布隆过滤器：定义

- 判断一个字符串是否出现过的数据结构
- 假设有1亿个字符串，需要多少空间建立Hash索引？
- 1亿 * 8 / 负载率 =
- 哈希表 → 空间换时间
- 布隆过滤器 → 错误率换空间

布隆过滤器：实现

- 由01的数字序列构成
- 插入：多个不同hash函数计算Key，置1
- 查找：有一个为0不可能存在，全为1可能存在
- 空间？



布隆过滤器：优缺点

- 优点
  - 时间和空间
  - 多个hash函数可并行
  - 交差并（位运算）
- 缺点
  - 错误率随着负载率上升而上升
  - 无法删除

布隆过滤器：思考题

- 错误率推导
- 如何进一步优化空间？（位操作）