TreeMap分析

- 时间复杂度（平均）
  - 添加、删除、搜索：O(logn)

- 特点
  - Key 必须具备可比较性
  - 元素的分布是有顺序的

- 在实际应用中，很多时候的需求
  - Map 中存储的元素不需要讲究顺序
  - Map 中的 Key 不需要具备可比较性

- 不考虑顺序、不考虑 Key 的可比较性，Map 有更好的实现方案，平均时间复杂度可以达到 O(1)
  - 那就是采取哈希表来实现 Map

需求

- 设计一个写字楼通讯录，存放所有公司的通讯信息
  - 座机号码作为 key（假设座机号码最长是 8 位），公司详情（名称、地址等）作为 value
  - 添加、删除、搜索的时间复杂度要求是 O(1)

```
private Company[] companies = new Company[100000000];
public void add(int phone, Company company) {
    companies[phone] = company;
}
public void remove(int phone) {
    companies[phone] = null;
}
public Company get(int phone) {
    return companies[phone];
}
```

索引	数据
0	
1	
...	
40089008	小码哥
...	
68485438	大码哥
...	
99999999	

- 存在什么问题？
  - 空间复杂度非常大
  - 空间使用率极其低，非常浪费内存空间
  - 其实数组 `companies` 就是一个哈希表，典型的【空间换时间】

哈希表（Hash Table）

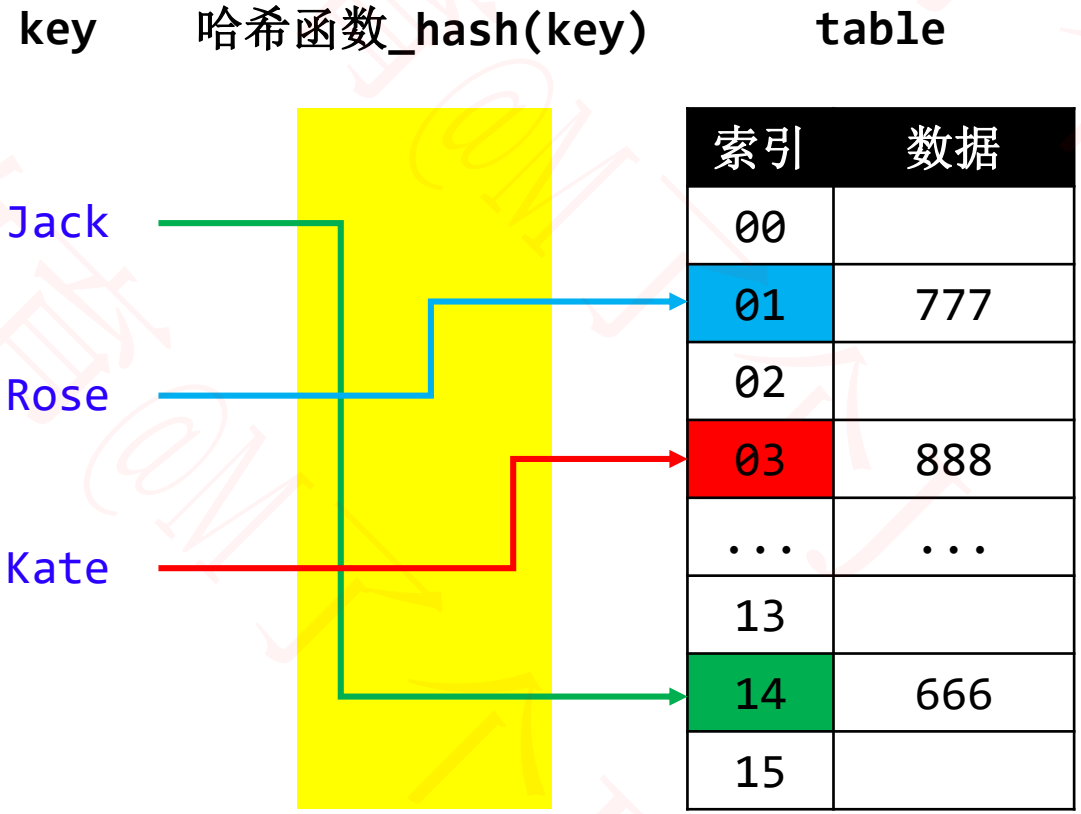
- 哈希表也叫做散列表（hash 有“剁碎”的意思）

- 它是如何实现高效处理数据的？

```
put("Jack", 666);
put("Rose", 777);
put("Kate", 888);
```

- 添加、搜索、删除的流程都是类似的

1. 利用哈希函数生成 key 对应的 index【O(1)】
2. 根据 index 操作定位数组元素【O(1)】



- 哈希表是【空间换时间】的典型应用
- 哈希函数，也叫做散列函数
- 哈希表内部的数组元素，很多地方也叫 Bucket（桶），整个数组叫 Buckets 或者 Bucket Array

小码哥教育 SEEMYGO

TreeMap vs HashMap

- 何时选择TreeMap?
  - 元素具备可比较性且要求升序遍历（按照元素从小到大）

- 何时选择HashMap?
  - 无序遍历