

# HashSet & HashMap

HashMap 哈希表/散列表

HashMap <key, value>

I. 需求 look up → HashMap can solve

key = Integer/long/string  
value = Student class  
key = Integer  
value = String  
list <String/Integer>

a) 查找:

\* key 最好是单值 > 简单 > 简单

① 暴力查找 // O(N) brute-force 暴力法

② Binary-search // O(lgN)

③ HashMap O(1)

(Bucket) []

\* Array = 通过 index 去找对应 element O(1)

⇒ Get  
Value

key → index 希望 O(1) → Bucket 在 Bucket 找 key  
<key, value>

key 的 Type 不定  
could be int, string,  
Myclass

private int hashCode

\* key → index

default behavior:  
定义任何 class 都是 Object 的子类, 会

① class Object { 实体 method class MyClass { 继承 Object

```
public int hashCode() {
    ...
}
```

← customize hashCode

@Override  
public int hashCode()

! 3 → 自定义

★ 生成 index  
// 泛型类

② private int hashCode (K key) { → 作成 hashmap 的 key  
if (key == null) return 0; return Math.abs (key.hashCode % buckets.size); }

\* 为什么一定要大写:

因为要把 primitive 变成 class, 才能点出

e.g int → Integer

来 hashCode

\* 当前方式可能导致 diff key, same index → Conflict

key<sub>1</sub> → index = 1  
 key<sub>2</sub> → index = 1 conflict

⇒ handle conflict in buckets

③ Buckets handle multiple item

List <<key, value>>

key 是 list, 可能是 O(1)

O(1) 正常情况

\* hash Func = hashCode

\* index → Bucket O(1)

查找 hashmap 的 time complexity  
 取决于 bucket 里有多少个 <key, pair>

\* Scan Buckets O(k) k is # bucket

④ <sup>希望</sup> ~~希望~~ hash Map . get () = O(1)

⇒ # (Bucket) size is O(1)

\* 保证 Buckets size 是一个常数 O(1)

↳ # Buckets 足够多 e.g 存 100 元素有 100 个桶且几乎平均分配

↳ # Buckets <sup>多</sup> vs <sup>少</sup> 存的元素

★  $0.75 = \frac{\# \text{key, val}}{\# \text{Buckets}}$

↳ load factor (java 中)

⑤ 理论上 Hashmap.get 不是 O(1) ① Avg O(1) [expected]  
② worst case O(N)  
N is size of buckets

---

⇒ put  
void

put(key, value)  
↑

① key → index

② index → bucket[]

③ Scan bucket[] 找 pair

{ Avg. O(1)  
[worst case O(N)]

④ Insert or update

⇒ Remove remove(key)

return boolean ① key → Index

// True key exist, ② Index → Bucket[]

other False

③ Scan Bucket[] for pair <key, value>

④ delete

⇒ boolean contained(key)

// True key exist, other False

2. 创建和初始化

a) Map <Integer, Integer> map = new HashMap<>();

→ Map's subset

↑  
Interface

..... frequencyMap = new HashMap<>();

3. 散列表

a) data 排步 not 连续, 是分散排 & bucket 可能为 空

## 4. method

Java hashMap 常见方法

- public value **get(Object key)**: Used to get the value of the corresponding key.
- public value **put(K key, V value)**: Inserts the value which is mentioned in the argument for the corresponding key.
- public boolean **containsKey(Object key)**: Decision of whether key is present or not, note that, the return type is Boolean.
- public boolean **containsValue(Object value)**: Decision of whether the value is present or not, note that the return type is Boolean.
- public V **remove(Object key)**: Clears particular key and its value from HashMap as specified in code.
- public void **clear()**: Clears all key and values from the HashMap as mentioned.
- public boolean **isEmpty()**: Verifies whether HashMap is empty or not.
- Object **clone()**: Mappings of a HashMap is returned by this method which we can use for cloning purpose to another HashMap.
- public int **size()**: Returns the size, means how many key-value pair is present in a HashMap.
- public Set<Map.Entry<K, V>> **entrySet()**: The set of mapping in HashMap is returned by this method.
- public Set<K> **keySet()**: The set of key which is present in HashMap is returned by this method.
- public value **getOrDefault(K key, V val)**: get the value of key, if the map don't have the key, then get val.
- Collection **values()**: You can get a collection of all of the values for a HashMap.

→ O(N)

→ O(N)

COPY

→ 帮你去 traverse 遍 hashmap 中所有的 pair base on 从第一个桶到第 N 个桶

## 5. Traverse hashmap

遍历hashMap的几种方法

```
Map<String, Object> map = ...;
// method1: keySet()
for (String key : map.keySet()) {
    System.out.println(map.get(key));
}
// method2: values()
for (Object value : map.values()) { // ... }
// method3: entrySet()
for (Map.Entry<String, Object> entry : map.entrySet()) {
    String key = entry.getKey();
    Object value = entry.getValue();
    // ...
}
```

→ for each key

→ for each value

→ for each entry set

6. 桶 default 开了 16 个 with load factor of 0.75, 希望每个桶是 O(N)

## 7. HashSet

a) only key, no value

b) Hashmap vs hashset

HashSet<String> hset = new HashSet<String>

key only

HashSet vs HashMap

HashMap	HashSet
HashMap实现了Map接口	HashSet实现了Set接口
HashMap存储键值对	HashSet仅仅存储对象
使用put()方法将元素放入map中	使用add()方法将元素放入set中
HashMap中使用键对象来计算hashcode值	HashSet使用成员对象来计算hashcode值, 对于两个对象来说hashcode可能相同, 所以equals()方法用来判断对象的相等性, 如果两个对象不同的话, 那么返回false
HashMap比较快, 因为使用唯一的键来获取对象	HashSet较HashMap来说比较慢

c) e.g give student id is legal or not

e.g 结点是否被处理过

\* array 里一些 String class, 不连续 int 不能变成 index