4. 哈希表（ Hash Table)、映射 (Map)、集合 (Set) 的实现与特性

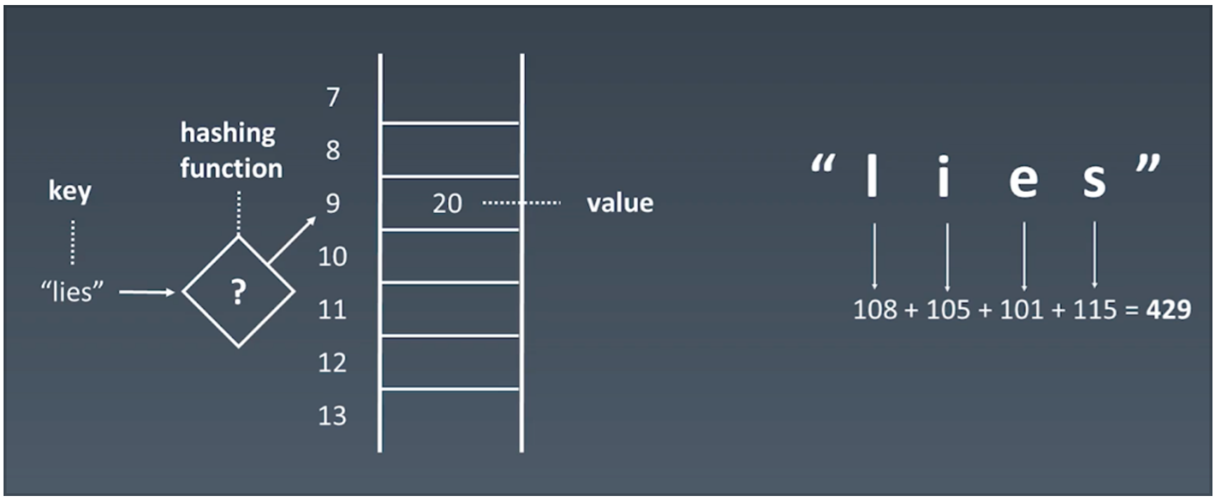
哈希表 Hash Table

Hash Table Java Source Code

> **哈希表（ Hash Table ）**，也叫散列表，是根据 **关键值（ Key value)** 而直接进行访问的数据结构。
> 它通过把关键码值映射到表中一个位置来访问记录，以加快查找的速度。
> 这个映射函数叫做 **散列函数（ Hash Function)**,存放记录的数组叫做哈希表（或散列表）。

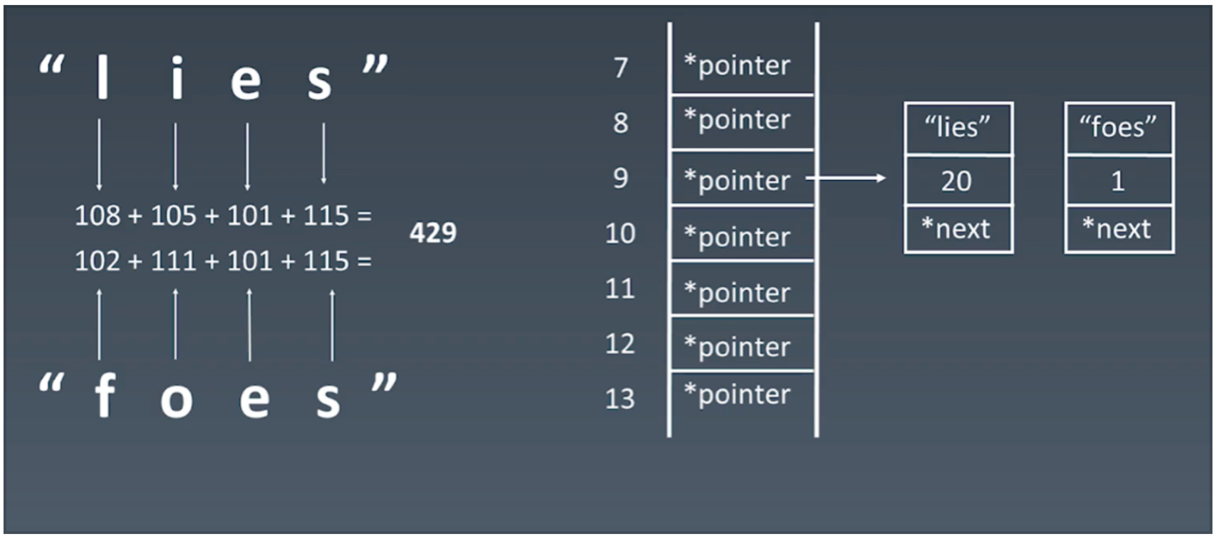
- 工程实践
  1. 电话号码簿；
  2. 用户信息表；
  3. 缓存 （ LRU Cache ）；
  4. 键值队存储 （Redis) .

哈希函数 Hash Function

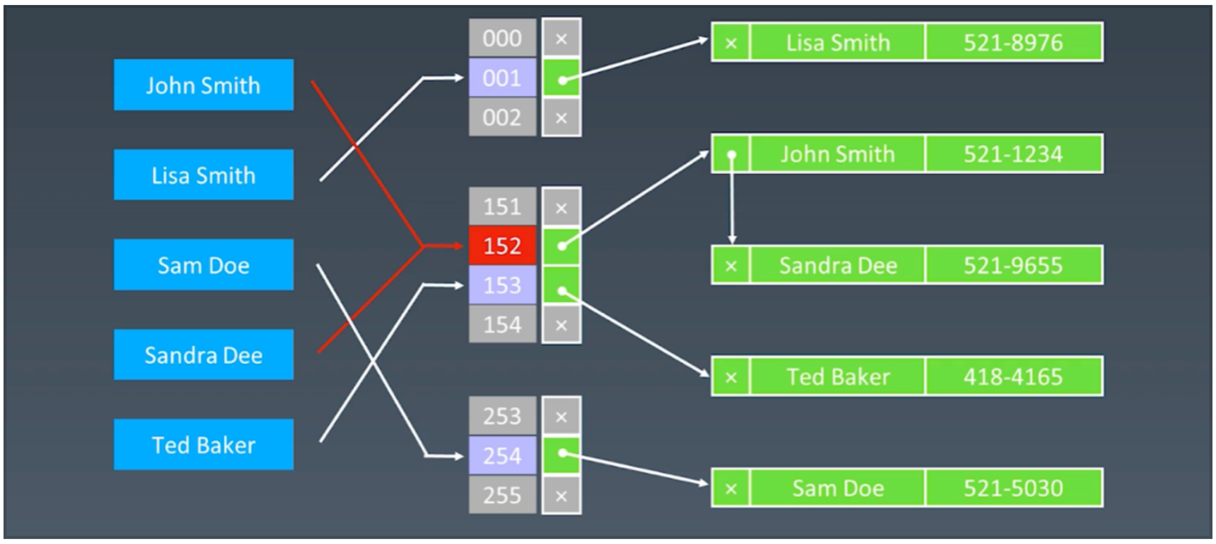


哈希碰撞 Hash Collisions

- 哈希碰撞 Hash Collisions 即 散列冲突，经过 hash function 后得到相同的散列值.
- 解决办法：
  1. 换散列位置，容易造成散列表空间不足；
  2. 增加一个纬度，即改用 linked list 结构，在相同散列值下可以存多个值.



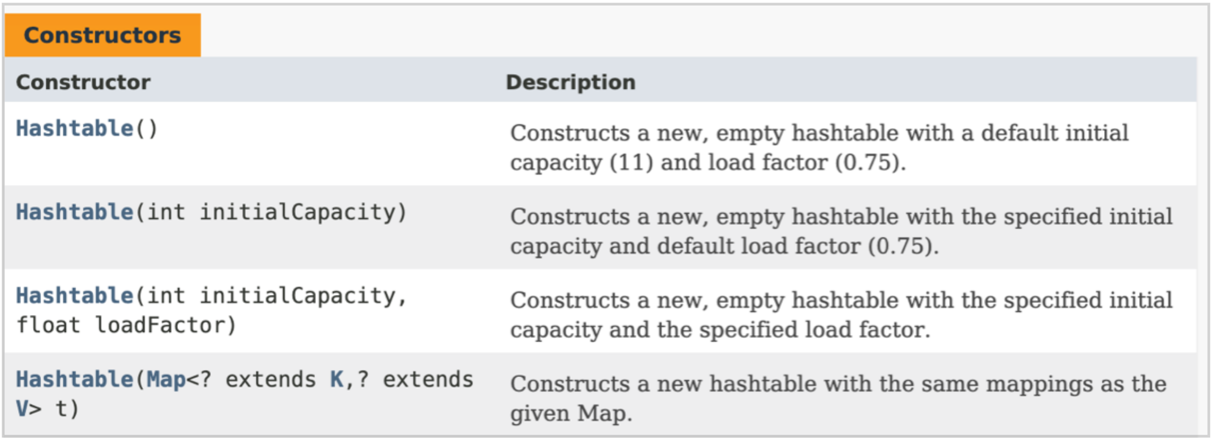
完成结构



Java API

Hashtable Interface

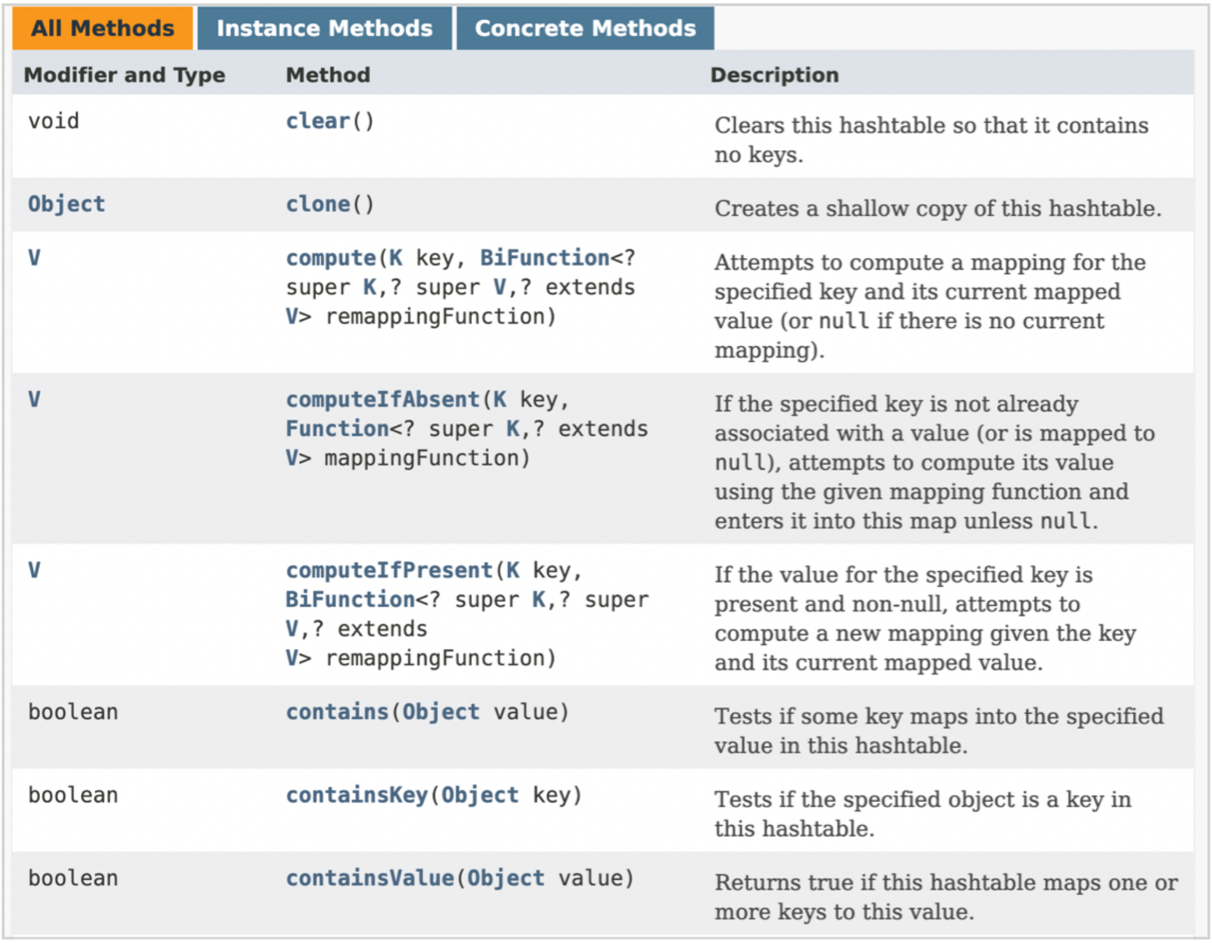
Constructor Summary



Constructors

Constructor	Description
`Hashtable()`	Constructs a new, empty hashtable with a default initial capacity (11) and load factor (0.75).
`Hashtable(int initialCapacity)`	Constructs a new, empty hashtable with the specified initial capacity and default load factor (0.75).
`Hashtable(int initialCapacity, float loadFactor)`	Constructs a new, empty hashtable with the specified initial capacity and the specified load factor.
`Hashtable(Map<? extends K,? extends V> t)`	Constructs a new hashtable with the same mappings as the given Map.

Method Summary

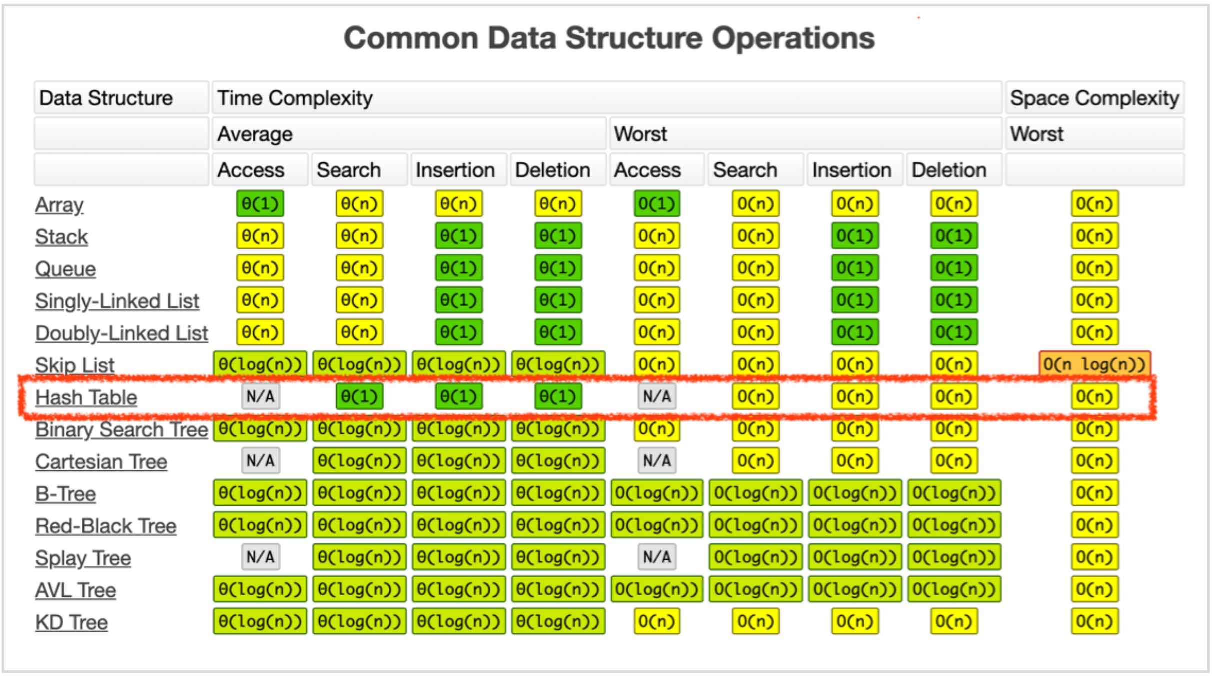


All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`void`	`clear()`	Clears this hashtable so that it contains no keys.
`Object`	`clone()`	Creates a shallow copy of this hashtable.
`V`	`compute(K key, BiFunction<? super K,? super V,? extends V> remappingFunction)`	Attempts to compute a mapping for the specified key and its current mapped value (or `null` if there is no current mapping).
`V`	`computeIfAbsent(K key, Function<? super K,? extends V> mappingFunction)`	If the specified key is not already associated with a value (or is mapped to `null`), attempts to compute its value using the given mapping function and enters it into this map unless `null`.
`V`	`computeIfPresent(K key, BiFunction<? super K,? super V,? extends V> remappingFunction)`	If the value for the specified key is present and non-null, attempts to compute a new mapping given the key and its current mapped value.
`boolean`	`contains(Object value)`	Tests if some key maps into the specified value in this hashtable.
`boolean`	`containsKey(Object key)`	Tests if the specified object is a key in this hashtable.
`boolean`	`containsValue(Object value)`	Returns true if this hashtable maps one or more keys to this value.

Enumeration<V>	elements()	Returns an enumeration of the values in this hashtable.
Set<Map.Entry<K,V>>	entrySet()	Returns a Set view of the mappings contained in this map.
boolean	equals(Object o)	Compares the specified Object with this Map for equality, as per the definition in the Map interface.
V	get(Object key)	Returns the value to which the specified key is mapped, or null if this map contains no mapping for the key.
int	hashCode()	Returns the hash code value for this Map as per the definition in the Map interface.
boolean	isEmpty()	Tests if this hashtable maps no keys to values.
Enumeration<K>	keys()	Returns an enumeration of the keys in this hashtable.
Set<K>	keySet()	Returns a Set view of the keys contained in this map.
V	merge(K key, V value, BiFunction<? super V,? super V,? extends V> remappingFunction)	If the specified key is not already associated with a value or is associated with null, associates it with the given non-null value.
V	put(K key, V value)	Maps the specified key to the specified value in this hashtable.
void	putAll(Map<? extends K,? extends V> t)	Copies all of the mappings from the specified map to this hashtable.
protected void	rehash()	Increases the capacity of and internally reorganizes this hashtable, in order to accommodate and access its entries more efficiently.
protected void	rehash()	Increases the capacity of and internally reorganizes this hashtable, in order to accommodate and access its entries more efficiently.
V	remove(Object key)	Removes the key (and its corresponding value) from this hashtable.
int	size()	Returns the number of keys in this hashtable.
String	toString()	Returns a string representation of this Hashtable object in the form of a set of entries, enclosed in braces and separated by the ASCII characters " , " (comma and space).
Collection<V>	values()	Returns a Collection view of the values contained in this map.

复杂度分析



Common Data Structure Operations

Data Structure	Time Complexity								Space Complexity
	Average				Worst				Worst
	Access	Search	Insertion	Deletion	Access	Search	Insertion	Deletion	
Array	Θ(1)	Θ(n)	Θ(n)	Θ(n)	O(1)	O(n)	O(n)	O(n)	O(n)
Stack	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Queue	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Singly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Doubly-Linked List	Θ(n)	Θ(n)	Θ(1)	Θ(1)	O(n)	O(n)	O(1)	O(1)	O(n)
Skip List	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n log(n))
Hash Table	N/A	Θ(1)	Θ(1)	Θ(1)	N/A	O(n)	O(n)	O(n)	O(n)
Binary Search Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)
Cartesian Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(n)	O(n)	O(n)	O(n)
B-Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Red-Black Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
Splay Tree	N/A	Θ(log(n))	Θ(log(n))	Θ(log(n))	N/A	O(log(n))	O(log(n))	O(log(n))	O(n)
AVL Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(log(n))	O(log(n))	O(log(n))	O(log(n))	O(n)
KD Tree	Θ(log(n))	Θ(log(n))	Θ(log(n))	Θ(log(n))	O(n)	O(n)	O(n)	O(n)	O(n)

映射 Map、集合 Set

- 在 哈希表 Hash Table 的基础上抽象出来的数据结构，经常应用在工程领域.
- Map、Set 的区别：

    Map：可以有重复的 键值对关系，key 是不重复的，但是 value 可以重复；

    Set ：不重复的元素集合，不是一个键值对的关系，就是一个单个元素，且不重复.
- 在 python 中是 映射 Map 是 字典 dict；

经常使用的 API

Java Code

```
// Map
new HashMap() / new TreeMap()
map.set(key, value)
map.get(key)
map.hash(key)
map.size()
map.clear()


// Set
new HashSet() / new TreeSet()
```

```
set.add(value)
set.delete(value)
set.has(value)
```

Python Code

```



list_x = [1,2,3,4]

# Map
map_x = {
  'jack' : 100,
  'Benjamin' : 80,
  'selina' : 90,
  ...
}

# Set
set_x = {'jack','selina','Andy'}
set_y = set(['jack','selina','jack'])
```

Map、Set Interface

> 着重看 Method 如何调用的

- Java Map Interface
- Java Set Interface

作业

- 分析 Hash Table Source Code，着重看 put()、get() Methods

#Algorithm/Part II : Theory/Data Structure#