

Table of Contents

1	HashSet
2	Usage
2.1	Create
2.2	Insert & Contain
2.3	Remove
2.4	Size & Capacity
2.5	Clear
2.6	Iteration
2.7	Set Operations

HashSet

A mutable hash set based on a Robin Hood hash table.

Usage

Create

You can create an empty set using `new()` or construct it using `from_array()`.

```
1
2  test {
3      let _set1 = @hashset.of([1, 2, 3, 4, 5])
4      let _set2 : @hashset.HashSet[String] = @hashset.new()
5
6  }
```

Insert & Contain

You can use `insert()` to add a key to the set, and `contains()` to check whether a key exists.

```
1
2  test {
3      let set : @hashset.HashSet[String] = @hashset.new()
4      set.add("a")
5      assert_eq(set.contains("a"), true)
6  }
```

Remove

You can use `remove()` to remove a key.

```
1
2  test {
3      let set = @hashset.of(["a", "b", "c"])
4      set.remove("a")
5      assert_eq(set.contains("a"), false)
6  }
```

Size & Capacity

You can use `size()` to get the number of keys in the set, or `capacity()` to get the current capacity.

```

1
2  test {
3    let set = @hashset.of(["a", "b", "c"])
4    assert_eq(set.size(), 3)
5    assert_eq(set.capacity(), 8)
6  }

```

Similarly, you can use `is_empty()` to check whether the set is empty.

```

1
2  test {
3    let set : @hashset.HashSet[Int] = @hashset.new()
4    assert_eq(set.is_empty(), true)
5  }

```

Clear

You can use `clear` to remove all keys from the set, but the allocated memory will not change.

```

1
2  test {
3    let set = @hashset.of(["a", "b", "c"])
4    set.clear()
5    assert_eq(set.is_empty(), true)
6  }

```

Iteration

You can use `each()` or `eachi()` to iterate through all keys.

```

1
2  test {
3    let set = @hashset.of(["a", "b", "c"])
4    let arr = []
5    set.each(k => arr.push(k))
6    let arr2 = []
7    set.eachi((i, k) => arr2.push((i, k)))
8  }

```

Set Operations

You can use `union()`, `intersection()`, `difference()` and `symmetric_difference()` to perform set operations.

```
1
2 test {
3   let m1 = @hashset.of(["a", "b", "c"])
4   let m2 = @hashset.of(["b", "c", "d"])
5   fn to_sorted_array(set : @hashset.HashSet[String]) {
6     let arr = set.to_array()
7     arr.sort()
8     arr
9   }
10
11   assert_eq(m1.union(m2) |> to_sorted_array, ["a", "b", "c", "d"])
12   assert_eq(m1.intersection(m2) |> to_sorted_array, ["b", "c"])
13   assert_eq(m1.difference(m2) |> to_sorted_array, ["a"])
14   assert_eq(m1.symmetric_difference(m2) |> to_sorted_array, ["a", "d"])
15 }
```