Table of Contents

Sorted Set

A mutable set backed by a red-black tree.

Usage

Create

You can create an empty SortedSet or a SortedSet from other containers.

```

test {
  let _set1 : @sorted_set.SortedSet[Int] = @sorted_set.new()
  let _set2 = @sorted_set.singleton(1)
  let _set3 = @sorted_set.from_array([1])

}
```

Container Operations

Add an element to the SortedSet in place.

```

test {
  let set4 = @sorted_set.from_array([1, 2, 3, 4])
  set4.add(5)
  let set6 = @sorted_set.from_array([1, 2, 3, 4, 5])
  assert_eq(set6.to_array(), [1, 2, 3, 4, 5])
}
```

Remove an element from the SortedSet in place.

```

test {
  let set = @sorted_set.from_array([3, 8, 1])
  set.remove(8)
  let set7 = @sorted_set.from_array([1, 3])
  assert_eq(set7.to_array(), [1, 3])
}
```

Whether an element is in the set.

```

test {
  let set = @sorted_set.from_array([1, 2, 3, 4])
  assert_eq(set.contains(1), true)
  assert_eq(set.contains(5), false)
}
```

Iterates over the elements in the set.

```

test {
  let arr = []
  @sorted_set.from_array([1, 2, 3, 4]).each(v => arr.push(v))
  assert_eq(arr, [1, 2, 3, 4])
}
```

Get the size of the set.

```

test {
  let set = @sorted_set.from_array([1, 2, 3, 4])
  assert_eq(set.size(), 4)
}
```

Whether the set is empty.

```

test {
  let set : @sorted_set.SortedSet[Int] = @sorted_set.new()
  assert_eq(set.is_empty(), true)
}
```

Set Operations

Union, intersection and difference of two sets. They return a new set that does not overlap with the original sets in memory.

```

test {
  let set1 = @sorted_set.from_array([3, 4, 5])
  let set2 = @sorted_set.from_array([4, 5, 6])
  let set3 = set1.union(set2)
  assert_eq(set3.to_array(), [3, 4, 5, 6])
  let set4 = set1.intersection(set2)
  assert_eq(set4.to_array(), [4, 5])
  let set5 = set1.difference(set2)
  assert_eq(set5.to_array(), [3])
}
```

Determine the inclusion and separation relationship between two sets.

```

test {
  let set1 = @sorted_set.from_array([1, 2, 3])
  let set2 = @sorted_set.from_array([7, 2, 9, 4, 5, 6, 3, 8, 1])
  assert_eq(set1.subset(set2), true)
  let set3 = @sorted_set.from_array([4, 5, 6])
  assert_eq(set1.disjoint(set3), true)
}
```

Stringify

SortedSet implements to_string (i.e. Show trait), which allows you to directly output it.

```

test {
  let set = @sorted_set.from_array([1, 2, 3])
  assert_eq(set.to_string(), "@sorted_set.from_array([1, 2, 3])")
}
```