Table of Contents

Priority Queue

A priority queue is a data structure capable of maintaining maximum/minimum valu
es at front of the queue, which may have other names in other programming langua
ges (C++ std::priority_queue / Rust BinaryHeap). The priority queue here is i
mplemented as a pairing heap and has excellent performance.

Usage

Create

You can use new() or of() to create a priority queue.

```
1
2   test {
3     let queue1 : @priority_queue.T[Int] = @priority_queue.new()
4     let queue2 = @priority_queue.of([1, 2, 3])
5     @json.inspect(queue1, content=[])
6     @json.inspect(queue2, content=[3, 2, 1])
7   }
```

Note, however, that the default priority queue created is greater-first; if you
need to create a less-first queue, you can write a struct belongs to Compare tra
it to implement it.

Length

You can use length() to get the number of elements in the current priority que
ue.

```
1
2   test {
3     let pq = @priority_queue.of([1, 2, 3, 4, 5])
4     assert_eq(pq.length(), 5)
5   }
```

Similarly, you can use the is_empty to determine whether the priority queue is e
mpty.

```
1
2   test {
3     let pq : @priority_queue.T[Int] = @priority_queue.new()
4     assert_eq(pq.is_empty(), true)
5   }
```

Peek

You can use peek() to look at the head element of a queue, which must be eithe
r the maximum or minimum value of an element in the queue, depending on the natu
re of the specification. The return value of peek() is an Option, which means
that the result will be None when the queue is empty.

```
1
2   test {
3     let pq = @priority_queue.of([1, 2, 3, 4, 5])
4     assert_eq(pq.peek(), Some(5))
5   }
```

Push

You can use push() to add elements to the priority queue.

```
1
2   test {
3     let pq : @priority_queue.T[Int] = @priority_queue.new()
4     pq.push(1)
5     pq.push(2)
6     assert_eq(pq.peek(), Some(2))
7   }
```

Pop

You can use pop() to pop the element at the front of the priority queue, respe
ctively, and like Peek, its return values are Option, loaded with the value of t
he element being popped.

```
1
2   test {
3     let pq = @priority_queue.of([5, 4, 3, 2, 1])
4     assert_eq(pq.pop(), Some(5))
5   }
```

```
1
2   test {
3     let pq = @priority_queue.of([5, 4, 3, 2, 1])
4     assert_eq(pq.length(), 5)
5   }
```

Clear

You can use clear to clear a priority queue.

```
1
2   test {
3     let pq = @priority_queue.of([1, 2, 3, 4, 5])
4     pq.clear()
5     assert_eq(pq.is_empty(), true)
6   }
```

Copy and Transfer

You can copy a priority queue using the copy method.

```
1
2   test {
3     let pq = @priority_queue.of([1, 2, 3])
4     let _pq2 = pq.copy()
5
6   }
```