Package 'collections'

March 7, 2019

Type Package

Title High Performance Container Data Types

Version 0.1.5

Date 2019-3-7

Description Provides high performance container data types such as Queue, Stack, Deque, Dict and OrderedDict. Benchmarks <https://randy3k.github.io/collections/articles/benchmark.html> have shown that these containers are asymptotically more efficient than those offered by other packages.

URL https://randy3k.github.io/collections

License MIT + file LICENSE

Depends R (>= 3.4.0)

NeedsCompilation yes

ByteCompile yes

Imports R6

Suggests testthat

LazyData true

RoxygenNote 6.1.0

Author Randy Lai [aut, cre]

Maintainer Randy Lai <randy.cs.lai@gmail.com>

Repository CRAN

Date/Publication 2019-03-07 08:12:42 UTC

R topics documented:

`collections-package` *collections: High Performance Container Data Types*

Description

Provides high performance container data types such as Queue, Stack, Deque, Dict and OrderedDict. Benchmarks <https://randy3k.github.io/collections/articles/benchmark.html> have shown that these containers are asymptotically more efficient than those offered by other packages.

Author(s)

Maintainer: Randy Lai <randy.cs.lai@gmail.com>

See Also

Useful links:

- https://randy3k.github.io/collections

`Deque` *Double Ended Queue*

Description

The Deque class creates a double ended queue with pairlist backend.

Usage

```
Deque
```

Format

An object of class `R6ClassGenerator` of length 24.

Usage

```
Deque$new()
Deque$push(item)
Deque$pushleft(item)
Deque$pop()
Deque$popleft()
Deque$peek()
Deque$peekleft()
Deque$extend(q)
Deque$extendleft(q)
Deque$remove(item)
Deque$clear()
Deque$size()
Deque$as_list()
```

Arguments

- `item`: any R object
- q: a Deque object

See Also

DequeL

Examples

```
q <- Deque$new()
q$push("foo")
q$push("bar")
q$pushleft("baz")
q$pop()  # bar
q$popleft()  # baz
```

DequeL *Double Ended Queue (list based)*

Description

The DequeL class creates a double ended queue with list backend. Pure R implementation, mainly for benchmark.

Usage

```
DequeL
```

Format

An object of class `R6ClassGenerator` of length 24.

Usage

```
DequeL$new()
DequeL$push(item)
DequeL$pushleft(item)
DequeL$pop()
DequeL$popleft()
DequeL$peek()
DequeL$peekleft()
DequeL$extend(q)
DequeL$extendleft(q)
DequeL$clear()
DequeL$remove(item)
DequeL$size()
DequeL$as_list()
```

Arguments

- `item`: any R object
- q: a DequeL object

See Also

Deque

Examples

```
q <- DequeL$new()
q$push("foo")
q$push("bar")
q$pushleft("baz")
q$pop()  # bar
q$popleft()  # baz
```

Dict *Dictionary*

Description

The `Dict` class creates an ordinary (unordered) dictionary. The key-value pairs are stored in an R environment.

Usage

```
Dict
```

Format

An object of class `R6ClassGenerator` of length 24.

Usage

```
Dict$new()
Dict$set(key, value)
Dict$get(key, default = NULL)
Dict$remove(key)
Dict$pop(key, default = NULL)
Dict$has(key)
Dict$keys()
Dict$values()
Dict$update(d)
Dict$clear()
Dict$size()
Dict$as_list()
```

- `key`: any R object, key of the item
- `value`: any R object, value of the item
- `default`: optional, the default value of an item if the key is not found

See Also

OrderedDict and OrderedDictL

Examples

```
d <- Dict$new()
d$set("apple", 5)
d$set("orange", 10)
d$set("banana", 3)
d$get("apple")
d$as_list()  # unordered
d$pop("orange")
d$as_list()  # "orange" is removed
```

OrderedDict *Ordered Dictionary*

Description

The `OrderedDict` class creates an ordered dictionary. Keys are stored in a double ended queue Deque while items are stored in an R environment.

Usage

```
OrderedDict
```

Format

An object of class `R6ClassGenerator` of length 24.

Usage

```
OrderedDict$new()
OrderedDict$set(key, value)
OrderedDict$get(key, default = NULL)
OrderedDict$remove(key)
OrderedDict$pop(key, default = NULL)
OrderedDict$popitem(last = TRUE)
OrderedDict$has(key)
OrderedDict$keys()
OrderedDict$values()
OrderedDict$update(d)
OrderedDict$clear()
OrderedDict$size()
OrderedDict$as_list()
```

Argument

- `key`: any R object, key of the item
- `value`: any R object, value of the item
- `default`: optinal, the default value of an item if the key is not found
- `d`: an OrderedDict or OrderedDictL

See Also

Dict and OrderedDictL

Examples

```
d <- OrderedDict$new()
d$set("apple", 5)
d$set("orange", 10)
d$set("banana", 3)
d$get("apple")
d$as_list()  # the order the item is preserved
d$pop("orange")
d$as_list()  # "orange" is removed
```

`OrderedDictL` *Ordered Dictionary (list based)*

Description

The `OrderedDictL` class creates an ordered dictionary. The key-value pairs are stored in an R List. Pure R implementation, mainly for benchmark.

Usage

```
OrderedDictL
```

Format

An object of class `R6ClassGenerator` of length 24.

Usage

```
OrderedDictL$new()
OrderedDictL$set(key, value)
OrderedDictL$get(key, default = NULL)
OrderedDictL$remove(key)
OrderedDictL$pop(key, default = NULL)
OrderedDict$popitem(last = TRUE)
OrderedDictL$has(key)
OrderedDictL$keys()
OrderedDictL$values()
OrderedDictL$update(d)
OrderedDictL$clear()
OrderedDictL$size()
OrderedDictL$as_list()
```

Argument

- `key`: any R object, key of the item
- `value`: any R object, value of the item
- `default`: optinal, the default value of an item if the key is not found
- `d`: an OrderedDict or OrderedDictL

See Also

Dict and OrderedDict

Examples

```
d <- OrderedDictL$new()
d$set("apple", 5)
d$set("orange", 10)
d$set("banana", 3)
d$get("apple")
d$as_list()  # the order the item is preserved
d$pop("orange")
d$as_list()  # "orange" is removed
```

PriorityQueue *Priority Queue*

Description

The PriorityQueue class creates a priority queue (a.k.a heap).

Usage

```
PriorityQueue
```

Format

An object of class R6ClassGenerator of length 24.

Usage

```
PriorityQueue$new()
PriorityQueue$push(item, priority = 0)
PriorityQueue$pop()
PriorityQueue$clear()
PriorityQueue$size()
PriorityQueue$as_list()
```

Argument

- item: any R object
- priority: non-negative interger, item with larger priority pops first

Examples

```
q <- PriorityQueue$new()
q$push("not_urgent")
q$push("urgent", priority = 2)
q$push("not_as_urgent", priority = 1)
q$pop()  # urgent
q$pop()  # not_as_urgent
q$pop()  # not_urgent
```

Queue *Queue*

Description

The Queue class creates a queue with pairlist backend.

Usage

```
Queue
```

Format

An object of class `R6ClassGenerator` of length 24.

Usage

```
Queue$new()
Queue$push(item)
Queue$pop()
Queue$peek()
Queue$clear()
Queue$size()
Queue$as_list()
```

Argument

- `item`: any R object

See Also

QueueL

Examples

```
q <- Queue$new()
q$push("first")
q$push("second")
q$pop()  # first
q$pop()  # second
```

QueueL *Queue (list based)*

Description

The QueueL class creates a queue with list backend. Pure R implementation, mainly for benchmark.

Usage

```
QueueL
```

Format

An object of class R6ClassGenerator of length 24.

Usage

```
QueueL$new()
QueueL$push(item)
QueueL$pop()
QueueL$peek()
QueueL$clear()
QueueL$size()
QueueL$as_list()
```

Argument

- item: any R object

See Also

Queue

Examples

```
q <- QueueL$new()
q$push("first")
q$push("second")
q$pop()  # first
q$pop()  # second
```

`Stack` *Stack*

Description

The `Stack` class creates a stack with pairlist backend.

Usage

```
Stack
```

Format

An object of class `R6ClassGenerator` of length 24.

Usage

```
Stack$new()
Stack$push(item)
Stack$pop()
Stack$peek()
Stack$clear()
Stack$size()
Stack$as_list()
```

Argument

- `item`: any R object

See Also

StackL

Examples

```
s <- Stack$new()
s$push("first")
s$push("second")
s$pop()  # second
s$pop()  # first
```

StackL *Stack (list based)*

Description

The StackL class returns a stack with list backend. Pure R implementation, mainly for benchmark.

Usage

```
StackL
```

Format

An object of class R6ClassGenerator of length 24.

Usage

```
StackL$new()
StackL$push(item)
StackL$pop()
StackL$peek()
StackL$clear()
StackL$size()
StackL$as_list()
```

Argument

- item: any R object

See Also

StackL

Examples

```
s <- StackL$new()
s$push("first")
s$push("second")
s$pop()  # second
s$pop()  # first
```

Index