Package 'collections'

November 2, 2019

Type Package

Title High Performance Container Data Types

Version 0.2.3

Date 2019-10-31

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.

License MIT + file LICENSE

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

Depends R (>= 3.4.0)

Imports xptr

Suggests testthat

ByteCompile yes

Encoding UTF-8

LazyData true

NeedsCompilation yes

RoxygenNote 6.1.1

Author Randy Lai [aut, cre],
Andrea Mazzoleni [cph] (tommy hash table library)

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

Repository CRAN

Date/Publication 2019-11-02 08:10:02 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>

Other contributors:

- Andrea Mazzoleni (tommy hash table library) [copyright holder]

See Also

Useful links:

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

`Deque` *Double Ended Queue*

Description

The `Deque` function creates a double ended queue.

Usage

```
Deque(items = NULL)
```

Arguments

items	a list of items

Details

Following methods are exposed:

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

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

See Also

DequeL

Examples

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

q <- Deque(list("foo", "bar"))
q$push("baz")$pushleft("bla")
```

DequeL *Double Ended Queue (R implementation)*

Description

The DequeL function creates a double ended queue. Pure R implementation for benchmarking.

Usage

```
DequeL(items = NULL)
```

Arguments

items	a list of items

Details

Following methods are exposed:

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

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

See Also

Deque

Examples

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

q <- DequeL(list("foo", "bar"))
q$push("baz")$pushleft("bla")
```

Dict *Dictionary*

Description

The `Dict` function creates an ordinary (unordered) dictionary (a.k.a. hash).

Usage

```
Dict(items = NULL)
```

Arguments

`items`	a list of items

Details

Following methods are exposed:

```
.$set(key, value)
.$get(key, default)
.$remove(key)
.$pop(key, default)
.$has(key)
.$keys()
.$values()
.$update(d)
.$clear()
.$size()
.$as_list()
.$print()
```

- `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(list(apple = 5, orange = 10))
d$set("banana", 3)
d$get("apple")
d$as_list()  # unordered
d$pop("orange")
d$as_list()  # "orange" is removed
d$set("orange", 3)$set("pear", 7)  # chain methods
```

DictL *Dictionary (R implementation)*

Description

The DictL function creates an ordinary (unordered) dictionary (a.k.a. hash). Pure R implementation for benchmarking.

Usage

```
DictL(items = NULL)
```

Arguments

items	a list of items

Details

Following methods are exposed:

```
.$set(key, value)
.$get(key, default)
.$remove(key)
.$pop(key, default)
.$has(key)
.$keys()
.$values()
.$update(d)
.$clear()
.$size()
.$as_list()
.$print()
```

- 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 <- DictL(list(apple = 5, orange = 10))
d$set("banana", 3)
d$get("apple")
d$as_list()  # unordered
d$pop("orange")
```

```
d$as_list()  # "orange" is removed
d$set("orange", 3)$set("pear", 7)  # chain methods
```

`OrderedDict` *Ordered Dictionary*

Description

The `OrderedDict` function creates an ordered dictionary.

Usage

```
OrderedDict(items = NULL)
```

Arguments

`items`	a list of items

Details

Following methods are exposed:

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

- `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
- `d`: an OrderedDict or OrderedDictL

See Also

Dict and OrderedDictL

Examples

```
d <- OrderedDict(list(apple = 5, orange = 10))
d$set("banana", 3)
d$get("apple")
d$as_list()  # the order the item is preserved
d$pop("orange")
d$as_list()  # "orange" is removed
d$set("orange", 3)$set("pear", 7)  # chain methods
```

OrderedDictL *Ordered Dictionary (R implementation)*

Description

The OrderedDictL function creates an ordered dictionary. Pure R implementation for benchmarking.

Usage

```
OrderedDictL(items = NULL)
```

Arguments

items	a list of items

Details

Following methods are exposed:

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

- 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
- d: an OrderedDict or OrderedDictL

See Also

Dict and OrderedDict

Examples

```
d <- OrderedDictL(list(apple = 5, orange = 10))
d$set("banana", 3)
d$get("apple")
d$as_list()  # the order the item is preserved
d$pop("orange")
d$as_list()  # "orange" is removed
d$set("orange", 3)$set("pear", 7)  # chain methods
```

PriorityQueue *Priority Queue*

Description

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

Usage

```
PriorityQueue(items = NULL, priorities = rep(0, length(items)))
```

Arguments

items	a list of items
priorities	a vector of interger valued priorities

Details

Following methods are exposed:

```
.$push(item, priority = 0)
.$pop()
.$clear()
.$size()
.$as_list()
.$print()
```

- item: any R object
- priority: a real number, item with larger priority pops first

Examples

```
q <- PriorityQueue()
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

q <- PriorityQueue(list("not_urgent", "urgent"), c(0, 2))
q$push("not_as_urgent", 1)$push("not_urgent2")
```

Queue *Queue*

Description

The Queue function creates a queue.

Usage

```
Queue(items = NULL)
```

Arguments

items	a list of items

Details

Following methods are exposed:

```
.$push(item)
.$pop()
.$peek()
.$clear()
.$size()
.$as_list()
.$print()
```

- item: any R object

See Also

QueueL

Examples

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

q <- Queue(list("foo", "bar"))
q$push("baz")$push("bla")
```

QueueL *QueueL (R implementation)*

Description

The QueueL function creates a queue. Pure R implementation for benchmarking.

Usage

```
QueueL(items = NULL)
```

Arguments

items	a list of items

Details

Following methods are exposed:

```
.$push(item)
.$pop()
.$peek()
.$clear()
.$size()
.$as_list()
.$print()
```

- item: any R object

See Also

Queue

Examples

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

q <- QueueL(list("foo", "bar"))
q$push("baz")$push("bla")
```

`Stack` *Stack*

Description

The `Stack` function creates a stack.

Usage

```
Stack(items = NULL)
```

Arguments

`items`	a list of items

Details

Following methods are exposed:

```
.$push(item)
.$pop()
.$peek()
.$clear()
.$size()
.$as_list()
.$print()
```

- `item`: any R object

See Also

StackL

Examples

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

s <- Stack(list("foo", "bar"))
s$push("baz")$push("bla")
```

StackL *StackL (R implementation)*

Description

The StackL function creates a stack. Pure R implementation for benchmarking.

Usage

```
StackL(items = NULL)
```

Arguments

items	a list of items

Details

Following methods are exposed:

```
.$push(item)
.$pop()
.$peek()
.$clear()
.$size()
.$as_list()
.$print()
```

- item: any R object

See Also

Stack

Examples

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

s <- StackL(list("foo", "bar"))
s$push("baz")$push("bla")
```

Index