heap

package | standard library

Version: go1.20.1 Latest | Published: Feb 14, 2023 | License: BSD-3-Clause | Imports: 1 | Imported by: 12,905

Details Valid go.mod file Redistributable license Tagged version Stable version

Learn more

Repository cs.opensource.google/go/go

Links Report a Vulnerability

Documentation

Documentation

Overview

Package heap provides heap operations for any type that implements heap.Interface. A heap is a tree with the property that each node is the minimum-valued node in its subtree.

The minimum element in the tree is the root, at index 0.

A heap is a common way to implement a priority queue. To build a priority queue, implement the Heap interface with the (negative) priority as the ordering for the Less method, so Push adds items while Pop removes the highest-priority item from the queue. The Examples include such an implementation; the file example_pq_test.go has the complete source.

▶ Example (IntHeap)

▶ Example (PriorityQueue)

Index

Examples

Package (IntHeap)
Package (PriorityQueue)

Constants

This section is empty.

Variables

This section is empty.

Functions

func Fix

added in go1.2

```
func Fix(h Interface, i int)
```

Fix re-establishes the heap ordering after the element at index i has changed its value. Changing the value of the element at index i and then calling Fix is equivalent to, but less expensive than, calling Remove(h, i) followed by a Push of the new value. The complexity is O(log n) where n = h.Len().

func Init

```
func Init(h Interface)
```

Init establishes the heap invariants required by the other routines in this package. Init is idempotent with respect to the heap invariants and may be called whenever the heap invariants may have been invalidated. The complexity is O(n) where n = h.Len().

func Pop

```
func Pop(h Interface) any
```

Pop removes and returns the minimum element (according to Less) from the heap. The complexity is O(log n) where n = h.Len(). Pop is equivalent to Remove(h, 0).

func Push

```
func Push(h Interface, x any)
```

Push pushes the element x onto the heap. The complexity is O(log n) where n = h.Len().

func Remove

```
func Remove(h Interface, i int) any
```

Remove removes and returns the element at index i from the heap. The complexity is O(log n) where n = h.Len().

Types

type Interface

```
type Interface interface {
    sort.Interface
    Push(x any) // add x as element Len()
    Pop() any   // remove and return element Len() - 1.
}
```

The Interface type describes the requirements for a type using the routines in this package. Any type that implements it may be used as a min-heap with the following invariants (established after Init has been called or if the data is empty or sorted):

```
!h.Less(j, i) for 0 <= i < h.Len() and 2*i+1 <= j <= 2*i+2 and j < h.Len()
```

Note that Push and Pop in this interface are for package heap's implementation to call. To add and remove things from the heap, use heap.Push and heap.Pop.

Source Files

View all

heap.go