

Heap di Fibonacci

Generated by Doxygen 1.9.1

1 Class Index	1
1.1 Class List	1
2 File Index	3
2.1 File List	3
3 Class Documentation	5
3.1 fibonacci_heap< T, CMP > Struct Template Reference	5
3.1.1 Detailed Description	5
3.1.2 Constructor & Destructor Documentation	6
3.1.2.1 fibonacci_heap()	6
3.1.3 Member Function Documentation	6
3.1.3.1 decrease_key()	6
3.1.3.2 extract_min()	6
3.1.3.3 heap_union()	7
3.1.3.4 insert()	7
3.1.3.5 print_children()	7
3.1.3.6 print_roots()	8
3.1.4 Member Data Documentation	8
3.1.4.1 cmp	8
3.1.4.2 head	8
3.1.4.3 min	8
3.1.4.4 n_nodes	8
3.1.4.5 n_tree	9
3.2 Node< T > Struct Template Reference	9
3.2.1 Detailed Description	9
3.2.2 Constructor & Destructor Documentation	10
3.2.2.1 Node()	10
3.2.3 Member Data Documentation	10
3.2.3.1 child	10
3.2.3.2 degree	10
3.2.3.3 key	10
3.2.3.4 left	10
3.2.3.5 mark	11
3.2.3.6 parent	11
3.2.3.7 right	11
3.3 NodeComparator< T > Struct Template Reference	11
3.3.1 Member Function Documentation	11
3.3.1.1 operator>()	11
4 File Documentation	13
4.1 fibonacciheap.hpp File Reference	13
4.2 test.cpp File Reference	13

4.2.1 Function Documentation	13
4.2.1.1 main()	13
Index	15

Chapter 1

Class Index

1.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

fibonacci_heap< T, CMP >	
Heap di Fibonacci	5
Node< T >	
Struct di un nodo	9
NodeComparator< T >	11

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

fibonacciheap.hpp	13
test.cpp	13

Chapter 3

Class Documentation

3.1 fibonacci_heap< T, CMP > Struct Template Reference

Heap di Fibonacci.

```
#include <fibonacciheap.hpp>
```

Public Member Functions

- [fibonacci_heap](#) ()
costruttore della Heap.
- void [insert](#) (const T x)
Funzione di inserimento.
- void [heap_union](#) ([fibonacci_heap](#)< T, CMP > heap)
funzione di unione.
- void [print_roots](#) ()
stampa radici.
- void [print_children](#) (const [Node](#)< T > *a)
stampa figli.
- T [extract_min](#) ()
estrazione minimo.
- void [decrease_key](#) ([Node](#)< T > *x, T new_key)
decremento chiave.

Public Attributes

- [Node](#)< T > * [head](#)
- [Node](#)< T > * [min](#)
- size_t [n_nodes](#)
- size_t [n_tree](#)
- CMP [cmp](#)

3.1.1 Detailed Description

```
template<typename T, typename CMP>  
struct fibonacci_heap< T, CMP >
```

Heap di Fibonacci.

Template Parameters

<i>T</i>	tipo delle chiavi dei nodi della Heap
<i>CMP</i>	classe del comparatore

3.1.2 Constructor & Destructor Documentation

3.1.2.1 fibonacci_heap()

```
template<typename T , typename CMP >
fibonacci_heap< T, CMP >::fibonacci_heap ( ) [inline]
```

costruttore della Heap.

Inizializza una Heap vuota

3.1.3 Member Function Documentation

3.1.3.1 decrease_key()

```
template<typename T , typename CMP >
void fibonacci_heap< T, CMP >::decrease_key (
    Node< T > * x,
    T new_key ) [inline]
```

decremento chiave.

Decrementa il valore della chiave di un nodo della Heap e se necessario lo sposta nella lista di radici

Parameters

<i>x</i>	nodo da decrementare
<i>new_key</i>	nuovo valore del nodo

3.1.3.2 extract_min()

```
template<typename T , typename CMP >
T fibonacci_heap< T, CMP >::extract_min ( ) [inline]
```

estrazione minimo.

Funzione per estrarre il minimo dalla Heap. Dopo aver estratto il minimo chiama la funzione di consolidazione.

Returns

valore minimo della Heap estratto

3.1.3.3 heap_union()

```
template<typename T , typename CMP >
void fibonacci_heap< T, CMP >::heap_union (
    fibonacci_heap< T, CMP > heap ) [inline]
```

funzione di unione.

Unione alla heap corrente di un'altra heap. In particolare si aggiunge in coda alla lista di radici corrente quella nuova. Il nodo di testa rimane dunque invariato ma si aggiornano se necessario il minimo, il numero totale di nodi e degli alberi.

Parameters

<i>heap</i>	Heap da unire
-------------	---------------

3.1.3.4 insert()

```
template<typename T , typename CMP >
void fibonacci_heap< T, CMP >::insert (
    const T x ) [inline]
```

Funzione di inserimento.

Parameters

<i>x</i>	valore del nodo da inserire
----------	-----------------------------

3.1.3.5 print_children()

```
template<typename T , typename CMP >
void fibonacci_heap< T, CMP >::print_children (
    const Node< T > * a ) [inline]
```

stampa figli.

Dato un nodo, semplice funzione per stampare la lista dei suoi figli

Parameters

<i>a</i>	nodo di cui stampare i figli
----------	------------------------------

3.1.3.6 print_roots()

```
template<typename T , typename CMP >
void fibonacci_heap< T, CMP >::print_roots ( ) [inline]
```

stampa radici.

Semplice funzione per stampare la lista di radici

3.1.4 Member Data Documentation

3.1.4.1 cmp

```
template<typename T , typename CMP >
CMP fibonacci_heap< T, CMP >::cmp
```

comparatore

3.1.4.2 head

```
template<typename T , typename CMP >
Node<T>* fibonacci_heap< T, CMP >::head
```

puntatore alla testa della lista di radici

3.1.4.3 min

```
template<typename T , typename CMP >
Node<T>* fibonacci_heap< T, CMP >::min
```

puntatore alla radice con chiave minima

3.1.4.4 n_nodes

```
template<typename T , typename CMP >
size_t fibonacci_heap< T, CMP >::n_nodes
```

numero di nodi nella Heap

3.1.4.5 n_tree

```
template<typename T , typename CMP >
size_t fibonacci_heap< T, CMP >::n_tree
```

numero di alberi nella Heap

The documentation for this struct was generated from the following file:

- [fibonacciheap.hpp](#)

3.2 Node< T > Struct Template Reference

Struct di un nodo.

```
#include <fibonacciheap.hpp>
```

Public Member Functions

- [Node](#) ()
costruttore.

Public Attributes

- T [key](#)
- [Node](#)< T > * [child](#)
- [Node](#)< T > * [left](#)
- [Node](#)< T > * [right](#)
- [Node](#)< T > * [parent](#)
- int [degree](#)
- bool [mark](#)

3.2.1 Detailed Description

```
template<typename T>
struct Node< T >
```

Struct di un nodo.

Tutti i nodi, che siano radici o figli di altri nodi, sono in liste doppiamente concatenate circolari, dunque la testa della lista raggiunge ,e a sua volta è raggiunta, dalla coda della lista.

Template Parameters

<i>T</i>	tipo della chiave del nodo
----------	----------------------------

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Node()

```
template<typename T >  
Node< T >::Node ( ) [inline]
```

costruttore.

< vale 1 se il nodo è marcato, 0 altrimenti

Imposta come valore della chiave 0

3.2.3 Member Data Documentation

3.2.3.1 child

```
template<typename T >  
Node<T>* Node< T >::child
```

< valore della chiave

3.2.3.2 degree

```
template<typename T >  
int Node< T >::degree
```

< puntatore al nodo genitore

3.2.3.3 key

```
template<typename T >  
T Node< T >::key
```

3.2.3.4 left

```
template<typename T >  
Node<T>* Node< T >::left
```

< puntatore alla testa della lista di figli del nodo

3.2.3.5 mark

```
template<typename T >
bool Node< T >::mark
```

< numero di figli del nodo

3.2.3.6 parent

```
template<typename T >
Node<T>* Node< T >::parent
```

< puntatore al nodo a destra

3.2.3.7 right

```
template<typename T >
Node<T>* Node< T >::right
```

< puntatore al nodo a sinistra

The documentation for this struct was generated from the following file:

- [fibonacciheap.hpp](#)

3.3 NodeComparator< T > Struct Template Reference

Public Member Functions

- bool [operator\(\)](#) (const [Node](#)< T > *n1, const [Node](#)< T > *n2)

3.3.1 Member Function Documentation

3.3.1.1 operator>()

```
template<typename T >
bool NodeComparator< T >::operator() (
    const Node< T > * n1,
    const Node< T > * n2 ) [inline]
```

The documentation for this struct was generated from the following file:

- [test.cpp](#)

Chapter 4

File Documentation

4.1 fibonacciheap.hpp File Reference

```
#include <iostream>
#include <stdexcept>
Include dependency graph for fibonacciheap.hpp:
```

4.2 test.cpp File Reference

```
#include "fibonacciheap.hpp"
Include dependency graph for test.cpp:
```

Classes

- struct [NodeComparator< T >](#)

Functions

- int [main](#) ()

4.2.1 Function Documentation

4.2.1.1 main()

```
int main ( )
```


Index

- child
 - Node< T >, [10](#)
- cmp
 - fibonacci_heap< T, CMP >, [8](#)
- decrease_key
 - fibonacci_heap< T, CMP >, [6](#)
- degree
 - Node< T >, [10](#)
- extract_min
 - fibonacci_heap< T, CMP >, [6](#)
- fibonacci_heap
 - fibonacci_heap< T, CMP >, [6](#)
- fibonacci_heap< T, CMP >, [5](#)
 - cmp, [8](#)
 - decrease_key, [6](#)
 - extract_min, [6](#)
 - fibonacci_heap, [6](#)
 - head, [8](#)
 - heap_union, [7](#)
 - insert, [7](#)
 - min, [8](#)
 - n_nodes, [8](#)
 - n_tree, [8](#)
 - print_children, [7](#)
 - print_roots, [8](#)
- fibonacciheap.hpp, [13](#)
- head
 - fibonacci_heap< T, CMP >, [8](#)
- heap_union
 - fibonacci_heap< T, CMP >, [7](#)
- insert
 - fibonacci_heap< T, CMP >, [7](#)
- key
 - Node< T >, [10](#)
- left
 - Node< T >, [10](#)
- main
 - test.cpp, [13](#)
- mark
 - Node< T >, [10](#)
- min
 - fibonacci_heap< T, CMP >, [8](#)
- n_nodes
 - fibonacci_heap< T, CMP >, [8](#)
- n_tree
 - fibonacci_heap< T, CMP >, [8](#)
- Node
 - Node< T >, [10](#)
- Node< T >, [9](#)
 - child, [10](#)
 - degree, [10](#)
 - key, [10](#)
 - left, [10](#)
 - mark, [10](#)
 - Node, [10](#)
 - parent, [11](#)
 - right, [11](#)
- NodeComparator< T >, [11](#)
 - operator(), [11](#)
- operator()
 - NodeComparator< T >, [11](#)
- parent
 - Node< T >, [11](#)
- print_children
 - fibonacci_heap< T, CMP >, [7](#)
- print_roots
 - fibonacci_heap< T, CMP >, [8](#)
- right
 - Node< T >, [11](#)
- test.cpp, [13](#)
 - main, [13](#)