

B-Tree ORI-UFSCar-2015

Generated by Doxygen 1.8.6

Thu Oct 29 2015 12:17:32

Contents

1	ori-ufscar-2015	1
2	Class Index	3
2.1	Class List	3
3	File Index	5
3.1	File List	5
4	Class Documentation	7
4.1	btree Struct Reference	7
4.1.1	Detailed Description	7
4.1.2	Member Data Documentation	7
4.1.2.1	order	7
4.1.2.2	root	7
4.2	btree_node Struct Reference	7
4.2.1	Detailed Description	8
4.2.2	Member Data Documentation	8
4.2.2.1	children	8
4.2.2.2	keys	8
4.2.2.3	leaf	8
4.2.2.4	number_of_keys	8
5	File Documentation	9
5.1	btree.c File Reference	9
5.1.1	Detailed Description	9
5.1.2	Function Documentation	10
5.1.2.1	allocate_node	10
5.1.2.2	btree_create	10
5.1.2.3	btree_search	10
5.1.2.4	delete_key	10
5.1.2.5	insert	10
5.1.2.6	insert_nonfull	11
5.1.2.7	print_node	11

5.1.2.8	print_post_order	11
5.1.2.9	remove_key_from_node	11
5.1.2.10	split_child	11
5.2	btree.h File Reference	12
5.2.1	Detailed Description	13
5.2.2	Typedef Documentation	13
5.2.2.1	btree	13
5.2.3	Function Documentation	13
5.2.3.1	allocate_node	13
5.2.3.2	btree_create	13
5.2.3.3	btree_search	13
5.2.3.4	delete_key	13
5.2.3.5	insert	14
5.2.3.6	insert_nonfull	14
5.2.3.7	print_node	14
5.2.3.8	print_post_order	14
5.2.3.9	remove_key_from_node	14
5.2.3.10	split_child	15
Index		16

Chapter 1

ori-ufscar-2015

A simple in-memory B-Tree implementation based on Introduction to Algorithms, Second Edition, Chapter 18 by Cormen et. al

Chapter 2

Class Index

2.1 Class List

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

btree	A B-Tree	7
btree_node	A B-Tree Node	7

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

btree.c	9
btree.h	12

Chapter 4

Class Documentation

4.1 btree Struct Reference

A B-Tree.

```
#include <btree.h>
```

Public Attributes

- [int order](#)
- [btree_node * root](#)

4.1.1 Detailed Description

A B-Tree.

A B-Tree that has at most $2 \times \text{order} - 1$ keys in its nodes

4.1.2 Member Data Documentation

4.1.2.1 `int btree::order`

Order of the tree

4.1.2.2 `btree_node* btree::root`

Root of the tree

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

- [btree.h](#)

4.2 btree_node Struct Reference

A B-Tree Node.

```
#include <btree.h>
```

Public Attributes

- int [number_of_keys](#)
- bool [leaf](#)
- int * [keys](#)
- struct [btree_node](#) ** [children](#)

4.2.1 Detailed Description

A B-Tree Node.

4.2.2 Member Data Documentation

4.2.2.1 struct [btree_node](#)** [btree_node::children](#)

Pointers to the children nodes

4.2.2.2 int* [btree_node::keys](#)

Array containing the keys

4.2.2.3 bool [btree_node::leaf](#)

Indicates if the node is a leaf

4.2.2.4 int [btree_node::number_of_keys](#)

Number keys currently stored in the node

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

- [btree.h](#)

Chapter 5

File Documentation

5.1 btree.c File Reference

```
#include <stdbool.h>
#include <stdio.h>
#include "btree.h"
#include "stdlib.h"
```

Functions

- `btree_node * allocate_node` (int order)
Allocates a B-Tree node.
- `btree * btree_create` (int order)
Creates an empty B-Tree.
- `btree_node * btree_search` (int key, `btree_node *node`)
Searches for a `key` in a tree with root in `node`.
- void `split_child` (`btree_node *parent`, int position, int order)
Splits a child node.
- `btree_node * insert_nonfull` (`btree_node *node`, int key, int order)
Inserts a key in a nonfull B-Tree node.
- `btree_node * insert` (`btree *tree`, int key)
Inserts a key in a B-Tree.
- void `print_post_order` (`btree_node *root`)
Print a B-Tree in-order First it prints the children and then the root.
- void `print_node` (`btree_node *node`)
Print a B-Tree node.
- `btree_node * remove_key_from_node` (`btree_node *node`, int key)
Removes a key from a node.
- `btree_node * delete_key` (`btree *tree`, `btree_node *node`, int key)
Removes a key from a B-Tree.

5.1.1 Detailed Description

Author

João Vitor Brandão

5.1.2 Function Documentation

5.1.2.1 `btree_node* allocate_node (int order)`

Allocates a B-Tree node.

Allocates a B-Tree node that has at most $2 \times \text{order} - 1$ keys

Parameters

<i>order</i>	Order of the B-Tree
--------------	---------------------

Returns

Returns a B-Tree node

5.1.2.2 `btree* btree_create (int order)`

Creates an empty B-Tree.

Parameters

<i>order</i>	Order of the B-Tree
--------------	---------------------

Returns

Returns an empty B-Tree

5.1.2.3 `btree_node* btree_search (int key, btree_node * node)`

Searches for a `key` in a tree with root in `node`.

Parameters

<i>key</i>	The key to be searched
<i>node</i>	Node that contains the root of the tree

Returns

If the key is found, returns the node containing the key

5.1.2.4 `btree_node* delete_key (btree * tree, btree_node * root, int key)`

Removes a key from a B-Tree.

Parameters

<i>tree</i>	Pointer to B-Tree
<i>root</i>	Pointer to the root node of the tree
<i>key</i>	Key to be removed

Returns

Node where the key was removed

5.1.2.5 `btree_node* insert (btree * tree, int key)`

Inserts a key in a B-Tree.

Parameters

<i>tree</i>	A B-Tree
<i>key</i>	Key to be inserted

Returns

Node where the key is inserted

5.1.2.6 btree_node* insert_nonfull (btree_node * node, int key, int order)

Inserts a key in a nonfull B-Tree node.

Parameters

<i>node</i>	A B-Tree node
<i>key</i>	Key to be inserted
<i>order</i>	Order of the tree

Returns

Node where the key is inserted

5.1.2.7 void print_node (btree_node * node)

Print a B-Tree node.

Parameters

<i>node</i>	A B-tree node
-------------	---------------

5.1.2.8 void print_post_order (btree_node * root)

Print a B-Tree in-order First it prints the children and then the root.

Parameters

<i>root</i>	The root of the B-Tree
-------------	------------------------

5.1.2.9 btree_node* remove_key_from_node (btree_node * node, int key)

Removes a key from a node.

Parameters

<i>node</i>	Node containing the key
<i>key</i>	Key to be removed

Returns

Node where the key was removed

5.1.2.10 void split_child (btree_node * parent, int position, int order)

Splits a child node.

Parameters

<i>node</i>	Parent node
<i>child</i>	Position of the child node to be splitted
<i>order</i>	Order of the B-Tree

5.2 btree.h File Reference

```
#include "stdbool.h"
```

Classes

- struct [btree_node](#)
A B-Tree Node.
- struct [btree](#)
A B-Tree.

Typedefs

- typedef struct [btree_node](#) [btree_node](#)
A B-Tree Node.
- typedef struct [btree](#) [btree](#)
A B-Tree.

Functions

- [btree_node](#) * [allocate_node](#) (int order)
Allocates a B-Tree node.
- [btree](#) * [btree_create](#) (int order)
Creates an empty B-Tree.
- [btree_node](#) * [btree_search](#) (int key, [btree_node](#) *node)
*Searches for a *key* in a tree with root in *node*.*
- void [split_child](#) ([btree_node](#) *parent, int position, int order)
Splits a child node.
- [btree_node](#) * [insert](#) ([btree](#) *tree, int key)
Inserts a key in a B-Tree.
- [btree_node](#) * [insert_nonfull](#) ([btree_node](#) *node, int key, int order)
Inserts a key in a nonfull B-Tree node.
- void [print_post_order](#) ([btree_node](#) *root)
Print a B-Tree in-order First it prints the children and then the root.
- void [print_node](#) ([btree_node](#) *node)
Print a B-Tree node.
- [btree_node](#) * [delete_key](#) ([btree](#) *tree, [btree_node](#) *root, int key)
Removes a key from a B-Tree.
- [btree_node](#) * [remove_key_from_node](#) ([btree_node](#) *node, int key)
Removes a key from a node.

5.2.1 Detailed Description

Author

João Vitor Brandão

5.2.2 Typedef Documentation

5.2.2.1 typedef struct btree btree

A B-Tree.

A B-Tree that has at most $2 \times \text{order} - 1$ keys in its nodes

5.2.3 Function Documentation

5.2.3.1 btree_node* allocate_node (int order)

Allocates a B-Tree node.

Allocates a B-Tree node that has at most $2 \times \text{order} - 1$ keys

Parameters

<i>order</i>	Order of the B-Tree
--------------	---------------------

Returns

Returns a B-Tree node

5.2.3.2 btree* btree_create (int order)

Creates an empty B-Tree.

Parameters

<i>order</i>	Order of the B-Tree
--------------	---------------------

Returns

Returns an empty B-Tree

5.2.3.3 btree_node* btree_search (int key, btree_node * node)

Searches for a *key* in a tree with root in *node*.

Parameters

<i>key</i>	The key to be searched
<i>node</i>	Node that contains the root of the tree

Returns

If the key is found, returns the node containing the key

5.2.3.4 btree_node* delete_key (btree * tree, btree_node * root, int key)

Removes a key from a B-Tree.

Parameters

<i>tree</i>	Pointer to B-Tree
<i>root</i>	Pointer to the root node of the tree
<i>key</i>	Key to be removed

Returns

Node where the key was removed

5.2.3.5 btree_node* insert (btree * tree, int key)

Inserts a key in a B-Tree.

Parameters

<i>tree</i>	A B-Tree
<i>key</i>	Key to be inserted

Returns

Node where the key is inserted

5.2.3.6 btree_node* insert_nonfull (btree_node * node, int key, int order)

Inserts a key in a nonfull B-Tree node.

Parameters

<i>node</i>	A B-Tree node
<i>key</i>	Key to be inserted
<i>order</i>	Order of the tree

Returns

Node where the key is inserted

5.2.3.7 void print_node (btree_node * node)

Print a B-Tree node.

Parameters

<i>node</i>	A B-tree node
-------------	---------------

5.2.3.8 void print_post_order (btree_node * root)

Print a B-Tree in-order First it prints the children and then the root.

Parameters

<i>root</i>	The root of the B-Tree
-------------	------------------------

5.2.3.9 btree_node* remove_key_from_node (btree_node * node, int key)

Removes a key from a node.

Parameters

<i>node</i>	Node containing the key
<i>key</i>	Key to be removed

Returns

Node where the key was removed

5.2.3.10 void split_child (btree_node * *parent*, int *position*, int *order*)

Splits a child node.

Parameters

<i>node</i>	Parent node
<i>child</i>	Position of the child node to be splitted
<i>order</i>	Order of the B-Tree

Index

- allocate_node
 - [btree.c](#), [10](#)
 - [btree.h](#), [13](#)
- btree, [7](#)
 - [btree.h](#), [13](#)
 - order, [7](#)
 - root, [7](#)
- btree.c, [9](#)
 - allocate_node, [10](#)
 - btree_create, [10](#)
 - btree_search, [10](#)
 - delete_key, [10](#)
 - insert, [10](#)
 - insert_nonfull, [11](#)
 - print_node, [11](#)
 - print_post_order, [11](#)
 - remove_key_from_node, [11](#)
 - split_child, [11](#)
- btree.h, [12](#)
 - allocate_node, [13](#)
 - btree, [13](#)
 - btree_create, [13](#)
 - btree_search, [13](#)
 - delete_key, [13](#)
 - insert, [14](#)
 - insert_nonfull, [14](#)
 - print_node, [14](#)
 - print_post_order, [14](#)
 - remove_key_from_node, [14](#)
 - split_child, [15](#)
- btree_create
 - [btree.c](#), [10](#)
 - [btree.h](#), [13](#)
- btree_node, [7](#)
 - children, [8](#)
 - keys, [8](#)
 - leaf, [8](#)
 - number_of_keys, [8](#)
- btree_search
 - [btree.c](#), [10](#)
 - [btree.h](#), [13](#)
- children
 - [btree_node](#), [8](#)
- delete_key
 - [btree.c](#), [10](#)
 - [btree.h](#), [13](#)
- insert
 - [btree.c](#), [10](#)
 - [btree.h](#), [14](#)
- insert_nonfull
 - [btree.c](#), [11](#)
 - [btree.h](#), [14](#)
- keys
 - [btree_node](#), [8](#)
- leaf
 - [btree_node](#), [8](#)
- number_of_keys
 - [btree_node](#), [8](#)
- order
 - [btree](#), [7](#)
- print_node
 - [btree.c](#), [11](#)
 - [btree.h](#), [14](#)
- print_post_order
 - [btree.c](#), [11](#)
 - [btree.h](#), [14](#)
- remove_key_from_node
 - [btree.c](#), [11](#)
 - [btree.h](#), [14](#)
- root
 - [btree](#), [7](#)
- split_child
 - [btree.c](#), [11](#)
 - [btree.h](#), [15](#)