[rgb]0.2,0.4,0.6 adjustwidth2em0em minted[linenos]c // // Created by on 2018/10/29. //
include ¡stdio.h¿ include ¡stdlib.h¿
// //5 // //1. //2. //3. // //4. //5.
//x // // //5 //T //
define RED 1 define BLACK 0
typedef struct node struct node * left; struct node * right; struct node * parent; int value; int color; Node, *pNode;
struct node * nil;
pNode tree$_minimum(pNodex)//xif(x == NULL)returnNULL$;
while (x-¿left != nil) x = x-¿left; return x;
pNode rb$_find(pNodeT, intvalue)//xvalueT = T- > parent; //while(T! = nilT- > value! = value)if(T- > value$
return T;
pNode tree$_successor(pNodex)if(x == NULL)returnNULL$;
if (x-¿right) // return tree$_minimum(x- > right)$;
// // //xxyy
while (x-¿parent x==x-¿parent-¿right) //x x = x-¿parent;
return x-¿parent;
void left$_rotate(pNodex)pNodey = x- > right$;
x-¿right = y-¿left; if (y-¿left != nil) y-¿left-¿parent = x;
y-¿parent = x-¿parent; if (x-¿parent-¿parent == x) //x x-¿parent-¿parent = y; else if (x-¿parent-¿left == x) x-¿parent-¿left = y; else x-¿parent-¿right = y;
y-¿left = x; x-¿parent = y;
void right$_rotate(pNodex)pNodey = x- > left$;
x-¿left = y-¿right; if (y-¿right != nil) y-¿right-¿parent = x;
y-¿parent = x-¿parent; if (x-¿parent-¿parent == x) x-¿parent-¿parent = y; else if (x-¿parent-¿left == x) x-¿parent-¿left = y; else x-¿parent-¿right = y;
y-¿right = x; x-¿parent = y;
void rb$_insert_fixup(pNodeT, pNodex)//TT- > parentpNodey$;
while (x-¿parent-¿color == RED) //xx() if (x-¿parent-¿parent-¿left == x-¿parent) //x y = x-¿parent-¿parent-¿right; //y if (y != nil y-¿color == RED) //4 x-¿parent-¿color = BLACK; // y-¿color = BLACK; // x-¿parent-¿parent-¿color = RED; //4 x = x-¿parent-¿parent; // else if (x == x-¿parent-¿right) //()x x = x-¿parent; left$_rotate(x); else//x- > parent- > color = BLACK; x- > parent- > parent- > color = RED; right_rotate($
$parent- > color = BLACK; ////$
void rb$_insert(pNodeT, intval)pNodex, y, z$;
y = nil; x = T-¿parent; // z = (pNode)malloc(sizeof(Node)); z-¿color = RED; z-¿value = val; z-¿left = nil; z-¿right = nil;
while (x != nil) y = x; if (x-¿value ¡ val) x = x-¿right; else x = x-¿left;
z-¿parent = y;
if (y == nil) z-¿color = BLACK; T-¿parent = z; z-¿parent = T; return; else if (y-¿value ¡ z-¿value) y-¿right = z; else y-¿left = z; rb$_insert_fixup(T, z)$;
void rb$_transplant(pNodeu, pNodev)if(u- > parent- > parent == u)//uu- > parent- > parent = v; elseif(u- >$
/* * xy * xxnil (yynil)5 * x */
void rb$_delete_fixup(pNodeT, pNodex)/ * *xnilx * xnil5 * * /$
pNode w; while (x != T-¿parent x-¿color == BLACK) if (x == x-¿parent-¿left) //x w = x-¿parent-¿right; //wx //case1: if (w-¿color == RED)
/* * wwxwnil5 * case2 * w * wwwx() w() * wx(nil) w() */
w-¿parent-¿color = RED; // w-¿color = BLACK; left$_rotate(w- > parent); //w()x//wxw = x- > parent- > right; //wx//case2 : if(w- > left- > color == BLACKw- > right- > color == BLACK)$
/* * xcase1 * case1w(nil) */
w-¿color = RED; //w4 x = w-¿parent; //x else if (w-¿right-¿color == BLACK) //case3: /* * wwnilw (5) * w * w * ww * x (wx) * x(nil)w() (w) */
w-¿left-¿color = BLACK; //w w-¿color = RED; //w right$_rotate(w); //ww = x- > parent- > right; //wxw = w- > parent//case4 : / * *w * w()w()w()w() * w- > parentww- > rightww * www * wx * /w- >$