*//Written By CmYkRgB123*

#include <iostream>

#include <ctime>

#define MAX 100

using namespace std;

typedef struct

{

int l,r,key,fix;

}node;

class treap

{

public:

node p[MAX];

int size,root;

treap()

{

srand(time(0));

size=-1;

root=-1;

}

void rot\_l(int &x)

{

int y=p[x].r;

p[x].r=p[y].l;

p[y].l=x;

x=y;

}

void rot\_r(int &x)

{

int y=p[x].l;

p[x].l=p[y].r;

p[y].r=x;

x=y;

}

void insert(int &k,int tkey)

{

if (k==-1)

{

k=++size;

p[k].l=p[k].r=-1;

p[k].key=tkey;

p[k].fix=rand();

}

else

if (tkey<p[k].key)

{

insert(p[k].l,tkey);

if (p[ p[k].l ].fix>p[k].fix)

rot\_r(k);

}

else

{

insert(p[k].r,tkey);

if (p[ p[k].r ].fix>p[k].fix)

rot\_l(k);

}

}

void remove(int &k,int tkey)

{

if (k==-1) return;

if (tkey<p[k].key)

remove(p[k].l,tkey);

else if (tkey>p[k].key)

remove(p[k].r,tkey);

else

{

if (p[k].l==-1 && p[k].r==-1)

k=-1;

else if (p[k].l==-1)

k=p[k].r;

else if (p[k].r==-1)

k=p[k].l;

else

if (p[ p[k].l ].fix < p[ p[k].r ].fix)

{

rot\_l(k);

remove(p[k].l,tkey);

}

else

{

rot\_r(k);

remove(p[k].r,tkey);

}

}

}

void print(int k)

{

if (p[k].l!=-1)

print(p[k].l);

[cout](http://www.opengroup.org/onlinepubs/009695399/functions/cout.html) << p[k].key << " : " << p[k].fix << endl;

if (p[k].r!=-1)

print(p[k].r);

}

};

treap T;

int main()

{

int i;

for (i=3;i>=1;i--)

T.insert(T.root,i);

T.print(T.root);

for (i=3;i>=1;i--)

{

[cout](http://www.opengroup.org/onlinepubs/009695399/functions/cout.html) << endl;

T.remove(T.root,i);

T.print(T.root);

}

return 0;

}

*//使用指针的treap by zjf*

#include <time.h>

#include <stdlib.h>

#include <iostream>

using namespace std;

typedef struct treapnode;

typedef struct treapnode \*treap;

struct treapnode

{

int key,fix;

treap left,right;

};

treap nullnode,root;

void initialize()

{

nullnode = new treapnode;

nullnode->left = nullnode->right = nullnode;

root=nullnode;

}

void sigrotl(treap& k1)

{

treap k2;

k2 = k1->right;

k1->right = k2->left;

k2->left = k1;

k1 = k2;

}

void sigrotr(treap& k1)

{

treap k2;

k2 = k1->left;

k1->left = k2->right;

k2->right = k1;

k1 = k2;

}

void insert(treap& t,int x)

{

if(t == nullnode)

{

t = new treapnode;

t->left = t->right = nullnode;

t->key = x;

t->fix = rand();

}

else

if(t->key == x)return;

else

if(x < t->key)

{

insert(t->left,x);

if(t->left->fix > t->fix)sigrotr(t);

}

else

{

insert(t->right,x);

if(t->right->fix > t->fix)sigrotl(t);

}

}

void remove(treap& t,int x)

{

if(t == nullnode)return;

if(x > t->key)remove(t->right,x);

else if(x < t->key)remove(t->left,x);

else

{

if(t->left == nullnode && t->right == nullnode)

{

delete t;

t=nullnode;

}

else if(t->left == nullnode)

{

treap tmp = t;

t = t->right;

delete tmp;

}

else if(t->right == nullnode)

{

treap tmp = t;

t = t->left;

delete tmp;

}

else

if(t->left->fix < t->right->fix)

{

sigrotl(t);

remove(t->left,x);

}

else

{

sigrotr(t);

remove(t->right,x);

}

}

}

void list(treap t)

{

if(t == nullnode)return;cout << t->key << endl;

list(t->left);

list(t->right);

}

int main()

{

initialize();

insert(root,1);

insert(root,2);

insert(root,3);

insert(root,4);

insert(root,5);

list(root);

system("pause");

}

发一个我的。主要是运用了孩子数组c来简化代码。并且把null节点的优先级设为inf，方便删除 By WJMZBMR..题目是SPOJ的ORDERSET。。

#include<cstdio>

#include<cstdlib>

using namespace std;

const int inf=~0U>>1;

class treap

{

struct node

{

int value,key,size;

node(int v,node\*n):value(v)

{c[0]=c[1]=n;size=1;key=rand()-1;}

void rz(){size=c[0]->size+c[1]->size+1;}

node\*c[2];

}\*root,\***null**;

void rot(node\*&t,bool d)

{

node\*c=t->c[d];

t->c[d]=c->c[!d];

c->c[!d]=t;

t->rz();c->rz();

t=c;

}

void insert(node\*&t,int x)

{

if(t==**null**) {t=new node(x,**null**);return;}

if(x==t->value) return;

bool d=x>t->value;

insert(t->c[d],x);

if(t->c[d]->key<t->key)

rot(t,d);

else

t->rz();

}

void Delete(node\*&t,int x)

{

if(t==**null**) return;

if(t->value==x)

{

bool d=t->c[1]->key<t->c[0]->key;

if(t->c[d]==**null**)

{

delete t;

t=**null**;

return;

}

rot(t,d);

Delete(t->c[!d],x);

}

else

{

bool d=x>t->value;

Delete(t->c[d],x);

}

t->rz();

}

int select(node\*t,int k)

{

int r=t->c[0]->size;

if(k==r) return t->value;

if(k<r) return select(t->c[0],k);

return select(t->c[1],k-r-1);

}

int rank(node\*t,int x)

{

if(t==**null**) return 0;

int r=t->c[0]->size;

if(x==t->value) return r;

if(x<t->value) return rank(t->c[0],x);

return r+1+rank(t->c[1],x);

}

public:

treap()

{

**null**=new node(0,0);null->size=0;null->key=inf;

root=**null**;

}

void ins(int x)

{

insert(root,x);

}

int sel(int k)

{

if(k>root->size) return -inf;

return select(root,k-1);

}

int ran(int x)

{

return rank(root,x);

}

void del(int x)

{

Delete(root,x);

}

}T;

int main()

{

*//freopen("in","r",stdin);*

int m;scanf("%d**\n**",&m);

char t;int x,tmp;

while(m--)

{

scanf("%c %d**\n**",&t,&x);

switch(t)

{

case 'I':T.ins(x);break;

case 'D':T.del(x);break;

case 'K':tmp=T.sel(x);if(tmp==-inf)[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("invalid**\n**");else [printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("%d**\n**",tmp);break;

case 'C':[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("%d**\n**",T.ran(x));break;

}

}

}

*/\**

*Author: lqhl*

*Problem: HNOI 2004 day1 pet 宠物收养所*

*\*/*

#include <cstdlib>

#include <fstream>

#include <iostream>

using namespace std;

const int INF = 1<<30;

const int DIVISOR = 1000000;

template <class DType>

class Treap {

private:

typedef struct node;

typedef struct node \*pnode;

struct node {

DType key;

int count, total;

pnode left, right;

int priority;

};

pnode **null**, root;

void left\_rotate(pnode &cur) {

pnode tmp;

tmp = cur->left;

cur->left = tmp->right;

tmp->right = cur;

tmp->total = cur->total;

cur->total = cur->left->total + cur->right->total + cur->count;

cur = tmp;

}

void right\_rotate(pnode &cur) {

pnode tmp;

tmp = cur->right;

cur->right = tmp->left;

tmp->left = cur;

tmp->total = cur->total;

cur->total = cur->left->total + cur->right->total + cur->count;

cur = tmp;

}

void insert\_node(pnode &cur, DType key) {

if (cur == **null**) {

cur = new struct node;

cur->key = key;

cur->count = 1;

cur->total = 1;

cur->left = **null**;

cur->right = **null**;

cur->priority = rand();

}

else if (key < cur->key) {

insert\_node(cur->left, key);

if (cur->left->priority < cur->priority)

left\_rotate(cur);

}

else if (key > cur->key) {

insert\_node(cur->right, key);

if (cur->right->priority < cur->priority)

right\_rotate(cur);

}

else

cur->count++;

cur->total = cur->left->total + cur->right->total + cur->count;

}

void delete\_node (pnode &cur, DType key) {

if (cur != **null**)

if (key < cur->key)

delete\_node(cur->left, key);

else if (key > cur->key)

delete\_node(cur->right, key);

else if (cur->count > 1)

cur->count--;

else if ((cur->left == **null**) && (cur->right == **null**)) {

delete cur;

cur = **null**;

}

else {

if (cur->left->priority < cur->right->priority)

left\_rotate(cur);

else

right\_rotate(cur);

delete\_node(cur, key);

}

cur->total = cur->left->total + cur->right->total + cur->count;

}

void erase(pnode cur) {

if (cur->left != **null**)

erase(cur->left);

if (cur->right != **null**)

erase(cur->right);

delete cur;

}

public:

Treap () {

**null** = new struct node;

null->key = INF;

null->count = 0;

null->total = 0;

null->left = **null**;

null->right = **null**;

null->priority = INF;

root = **null**;

}

~Treap () {

erase(root);

delete **null**;

delete root;

}

void insert(DType key) {

insert\_node(root, key);

}

void del(DType key) {

delete\_node(root, key);

}

DType find\_nearest(DType key) {

DType left, right;

pnode cur;

cur = root;

left = -INF;

right = INF;

while (cur != **null**) {

if (key < cur->key) {

if (cur->key < right)

right = cur->key;

cur = cur->left;

}

else {

if (cur->key > left)

left = cur->key;

cur = cur->right;

}

}

if (key - left <= right - key)

return left;

else

return right;

}

};

Treap <int> bst;

int main() {

FILE \*fin = fopen("pet.in", "r");

FILE \*fout = fopen("pet.out", "w");

int n,

kind = 0, ty, num,

ans = 0, tmp;

fscanf(fin, "%d**\n**", &n);

for (int i = 0; i < n; i++) {

fscanf(fin, "%d %d**\n**", &ty, &num);

if (ty) {

if (kind >= 0)

bst.insert(num);

else {

tmp = bst.find\_nearest(num);

ans += abs(num - tmp);

ans %= DIVISOR;

bst.del(tmp);

}

kind++;

}

else {

if (kind <= 0)

bst.insert(num);

else {

tmp = bst.find\_nearest(num);

ans += abs(num - tmp);

ans %= DIVISOR;

bst.del(tmp);

}

kind--;

}

}

fprintf(fout, "%d**\n**", ans);

}