Code:SBT C++

**来自"NOCOW"**

跳转到: [导航](http://www.nocow.cn/index.php/Code:SBT_C%2B%2B#column-one), [搜索](http://www.nocow.cn/index.php/Code:SBT_C%2B%2B#searchInput)

仅供参考。。。

*/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\**

*\* 程序分类： 高级数据结构实践*

*\* 子分类： SBT[Size Blanced Tree]*

*\* 程序名称： myavl.c*

*\* 程序功能： 实现SBT的各种操作*

*\* 程序作者： oopos@oopos.com*

*\* 开始时间： 2008/04/13 21:46 pm.*

*\* 完成时间： 2008/04/13 pm.*

*\* 程序感想：*

*\**

*\**

*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**

*\*\*/*

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct tree

{

int data;

int size;

struct tree \*left,\*right;

}SBT;

void LeftRotate(SBT \*\*t)

{

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("left...**\n**");

SBT \*p = (\*t)->right ;

(\*t)->right = p->left ;

p->left = \*t ;

p->size = (\*t)->size ;

(\*t)->size = 1 ;

if( (\*t)->left )

(\*t)->size += (\*t)->left->size ;

if( (\*t)->right )

(\*t)->size += (\*t)->right->size;

\*t = p ;

}

void RightRotate(SBT \*\*t)

{

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("right...**\n**");

SBT \*p = (\*t)->left ;

(\*t)->left = p->right ;

p->right = \*t ;

p->size = (\*t)->size ;

(\*t)->size = 1 ;

if( (\*t)->left )

(\*t)->size += (\*t)->left->size ;

if( (\*t)->right )

(\*t)->size += (\*t)->right->size;

\*t = p ;

}

void Maintain(SBT \*\*root,int flag)

{

if( !(\*root) )

return ;

if( !(\*root)->left && !(\*root)->right )

return ;

if( 1 == flag && (\*root)->left)

{

if( (\*root)->left->left || (\*root)->left->right )

{

if( !(\*root)->right )

{

if( (\*root)->left->left )

RightRotate(root);

else

{

LeftRotate(&(\*root)->left);

RightRotate(root);

}

}

else

{

if( (\*root)->left->left && (\*root)->left->left->size > (\*root)->right->size )

{

RightRotate(root);

}

else if( (\*root)->left->right && (\*root)->left->right->size > (\*root)->right->size )

{

LeftRotate(&(\*root)->left);

RightRotate(root);

}

else

return ;

}

}

else

return ;

}

else if( 0 == flag && (\*root)->right )

{

if( (\*root)->right->left || (\*root)->right->right )

{

if( !(\*root)->left )

{

if( (\*root)->right->right )

LeftRotate(root);

else

{

RightRotate(&(\*root)->right);

LeftRotate(root);

}

}

else

{

if( (\*root)->right->right && (\*root)->right->right->size > (\*root)->left->size )

{

LeftRotate(root);

}

else if( (\*root)->right->left && (\*root)->right->left->size > (\*root)->left->size )

{

RightRotate(&(\*root)->right);

LeftRotate(root);

}

else

return ;

}

}

else

return ;

}

else

return ;

Maintain(&(\*root)->left,1);

Maintain(&(\*root)->right,0);

Maintain(root,1);

Maintain(root,0);

}

void SbtInsert(SBT \*\*root,int key)

{

if( !(\*root) )

{

\*root = (SBT \*) malloc (sizeof(SBT)) ;

(\*root)->data = key ;

(\*root)->size = 1 ;

(\*root)->left = (\*root)->right = 0 ;

}

else if( key < (\*root)->data )

{

(\*root)->size += 1 ;

SbtInsert(&(\*root)->left,key);

Maintain(root,1);

}

else

{

(\*root)->size += 1 ;

SbtInsert(&(\*root)->right,key);

Maintain(root,0);

}

}

int SbtSelectMax(SBT \*root,int num)

{

if( !root )

return 0;

if( root->right )

{

if( root->right->size+1 == num )

return root->data;

else if( root->right->size+1 > num )

return SbtSelectMax(root->right,num);

else

return SbtSelectMax(root->left,num-root->right->size-1);

}

else if(num == 1)

return root->data;

else if( root->left )

return SbtSelectMax(root->left,num-1);

else

return -1;

}

int SbtSelectMin(SBT \*root,int num)

{

if( !root )

return 0;

if( root->left )

{

if( root->left->size+1 == num )

return root->data;

else if( root->left->size+1 > num )

return SbtSelectMin(root->left,num);

else

return SbtSelectMin(root->right,num-root->left->size-1);

}

else if(num == 1)

return root->data;

else if( root->right )

return SbtSelectMin(root->right,num-1);

else

return -1;

}

void MaxOutput(SBT \*root)

{

if( root )

{

MaxOutput(root->right);

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("%d ",root->data);

MaxOutput(root->left);

}

}

void MinOutput(SBT \*root)

{

if( root )

{

MinOutput(root->left);

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("%d ",root->data);

MinOutput(root->right);

}

}

int SbtSelectAllMax(SBT \*root,int num)

{

if( !root )

return 0;

if( root->right )

{

if( root->right->size+1 == num )

{

MaxOutput(root->right);

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("%d ",root->data);

return 1;

}

else if( root->right->size+1 > num )

return SbtSelectAllMax(root->right,num);

else

{

MaxOutput(root->right);

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("%d ",root->data);

return SbtSelectAllMax(root->left,num-root->right->size-1);

}

}

else if( num == 1 )

{

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("%d ",root->data);

return 1;

}

else if( root->left )

{

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("%d ",root->data);

return SbtSelectAllMax(root->left,num-1);

}

else

return -1;

}

int SbtSelectAllMin(SBT \*root,int num)

{

if(!root)

return 0;

if( root->left )

{

if( root->left->size+1 == num )

{

MinOutput(root->left);

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("%d ",root->data);

return 1;

}

else if( root->left->size+1 > num )

return SbtSelectAllMin(root->left,num);

else

{

MinOutput(root->left);

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("%d ",root->data);

return SbtSelectAllMin(root->right,num-root->left->size-1);

}

}

else if( num == 1 )

{

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("%d ",root->data);

return 1;

}

else if( root->right )

{

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("%d ",root->data);

return SbtSelectAllMin(root->right,num-1);

}

else

return -1;

}

SbtOutput(SBT \*root,int flag)

{

if( root )

{

SbtOutput(root->left,flag);

if( 1 == flag )

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("[%d-%d] ",root->data,root->size);

SbtOutput(root->right,flag);

if( 0 == flag )

free(root);

}

}

void SbtRank(SBT \*root,int flag,int \*rank)

{

if(root)

{

if(flag)

SbtRank(root->right,flag,rank);

else

SbtRank(root->left,flag,rank);

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("Rank[%d] is : %d**\n**",++\*rank,root->data);

if(flag)

SbtRank(root->left,flag,rank);

else

SbtRank(root->right,flag,rank);

}

}

int ComputeRank(SBT \*root,int key,int rank)

{

if( !root )

return 0;

if( key < root->data)

{

rank += 1 ;

if( root->right )

rank += root->right->size ;

return ComputeRank(root->left,key,rank);

}

else if( key > root->data)

{

return ComputeRank(root->right,key,rank);

}

else if( key == root->data)

{

if( root->right )

rank += root->right->size ;

return rank+1;

}

}

void Begin()

{

char st[128]={'**\0**'};

int key,tmp;

SBT \*mySbt = 0 ;

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("**\n**Usage: [insert|delete|exit|smax|smin|allmax|allmin|rank|crank] [key] **\n**");

while( scanf("%s %d",st,&key) != EOF && strcmp(st,"exit") )

{

if(!strcmp(st,"insert") )

SbtInsert(&mySbt,key);

else if( !strcmp(st,"smax") )

{

tmp = SbtSelectMax(mySbt,key);

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("[###Message:[No.%d max is : %d]**\n**",key,tmp);

}

else if( !strcmp(st,"smin") )

{

tmp = SbtSelectMin(mySbt,key);

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("[###Message:[No.%d min is : %d]**\n**",key,tmp);

}

else if( !strcmp(st,"allmax") )

{

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("[###Message:%d maxs is : [",key,tmp);

SbtSelectAllMax(mySbt,key);

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("]**\n**");

}

else if( !strcmp(st,"allmin") )

{

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("###Message:%d mins is : [",key);

SbtSelectAllMin(mySbt,key);

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("]**\n**");

}

else if( !strcmp(st,"rank") )

{

if(key)

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("###Message: From max to min Data Rank ###**\n**");

else

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("###Message: From min to max Data Rank ###**\n**");

tmp = 0 ;

SbtRank(mySbt,key,&tmp);

}

else if( !strcmp(st,"crank") )

{

tmp = ComputeRank(mySbt,key,0);

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("###Message: [%d]'s rank is [%d] ###**\n**",key,tmp);

}

else if( !strcmp(st,"exit") )

{

exit(0);

}

else

{

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("###Message: No Operators....**\n**");

}

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("**\n\n**Now,the data is : **\n**");

SbtOutput(mySbt,1);

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("**\n\n**");

*/\*if( mySbt )*

*printf("root: [%d-%d] , root->left: [%d] , root->right: [%d]\n",mySbt->data,mySbt->size,(mySbt->left ? mySbt->left->data : 0),*

*(mySbt->right ? mySbt->right->data : 0));*

*printf("\nJust test: \n");*

*MinOutput(mySbt);*

*printf("\n");*

*\*/*

[printf](http://www.opengroup.org/onlinepubs/009695399/functions/printf.html)("**\n**--------------------------------------------------------**\n**");

}

SbtOutput(mySbt,0);

}

int main(int argc,char \*argv[])

{

Begin();

return 0;

}

也发一个我的。。速度不是很快。。 By WJMZBMR。。。

#include<cstdio>

#include<algorithm>

using namespace std;

const int inf=~0U>>1;

struct node

{

int key,size;

node\*c[2];

node(int \_key,int \_size,node\*\_c)

:key(\_key),size(\_size)

{c[0]=c[1]=\_c;}

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

}TNull(0,0,0),\***Null**=&TNull;

class SBT

{

node\*root;

int top;

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

{

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

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

p->c[!d]=t;

t->rz();p->rz();

t=p;

}

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

{

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

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

if(p->c[d]->size>t->c[!d]->size)

rot(t,d);

else if(p->c[!d]->size>t->c[!d]->size)

{

rot(p,!d);

rot(t,d);

}

else return;

maintain(t->c[0],0);

maintain(t->c[1],1);

maintain(t,0);

maintain(t,1);

}

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

{

if(t==**Null**)

{t=new node(x,1,**Null**);return;}

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

bool d=x>t->key;

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

maintain(t,d);

t->rz();

}

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

{

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

int d;

if(t->key==x)

{

if(t->c[1]==**Null**)

{delete t;t=t->c[0];return;}

if(t->c[0]==**Null**)

{delete t;t=t->c[1];return;}

node\*p=t->c[1];while(p->c[0]!=**Null**)p=p->c[0];

t->key=p->key;

remove(t->c[1],p->key);d=1;

}

else

{

d=x>t->key;

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

}

maintain(t,1-d);

t->rz();

}

int select(node\*t,int k)

{

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

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

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

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

}

int rank(node\*t,int x)

{

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

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

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

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

}

public:

SBT()

{

Null->c[0]=Null->c[1]=**Null**;

root=**Null**;

}

void Insert(int x)

{

insert(root,x);

}

void Remove(int x)

{

remove(root,x);

}

int Select(int k)

{

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

return select(root,k-1);

}

int Rank(int x)

{

return rank(root,x);

}

int size(){return root->size;}

};

int main()

{

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

char t;SBT\*T=new SBT;

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

{

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

switch(t)

{

case 'I':T->Insert(x);break;

case 'D':T->Remove(x);break;

case 'K':tmp=T->Select(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->Rank(x));break;

}

}

}