ACM模板 for LTF

**稀疏二叉树：**

用于空间不够时，由于指针以及内存操作很多，尽量不用…

typedef struct TNode //表示一个二叉树节点

{

TNode()

{

reset();

}

void reset()

{

val=-1;

left=NULL;

right=NULL;

}

int val;

struct TNode \*left,\*right;

} Node;

void addNode(Node\* tree,char\* r) //根据char\*r中储存的L和R插入一个节点

{

int v;

char a;

sscanf(r,"%c%d",&a,&v);

r=strchr(r,',')+1;

int n=strlen(r);

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

{

switch(r[i])

{

case 'L':

if(tree->left==NULL) tree->left=new Node;

tree=tree->left;

break;

case 'R':

if(tree->right==NULL) tree->right=new Node;

tree=tree->right;

break;

}

}

else tree->val=v;

}

Node\* dui[maxn]= {0};// 宽搜输出

int out[maxn];

void bfs(Node \*root)

{

int front=0,rear=1;

bool fault=0;

dui[front]=root;

if(dui[front]==NULL)

return;

while(front!=rear)

{

if(dui[front]==NULL) break;

if(dui[front]->val==-1)

{

fault=1;

break;

}

if(dui[front]->left!=NULL)

{

dui[rear++]=dui[front]->left;

}

if(dui[front]->right!=NULL)

{

dui[rear++]=dui[front]->right;

}

out[front]=dui[front]->val;

++front;

}

if(fault)

cout<<-1<<endl;

else

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

cout<<out[i]<<" ";

cout<<endl;

}

void reset(Node \*root) //删除子树

{

for(int i=1; i<front; ++i)

delete dui[i];

root->left=root->right=NULL;

}

**迭代加深搜索：（搜分解分数式）：**

注意剪枝….

bool ids(int d,double sum,int now) //深度，和，现在的值

{

if(d==maxd)

{

if(abs(sum-val)<dev)

{

return 1;

}

else return 0;

}

int k=(maxd-d)/abs(val-sum)+1; //迭代上限，还剩下maxd-d个数,到val的值为val-sum，为能到达val现在可以填的最小的数即1/k

for(int i=now;i<=k;++i)

{

if(ids(d+1,sum+1.0/i,i+1))

{

cout<<"1/"<<i<<" ";

return 1;

}

}

return 0;

}

Int main()

{

While(ids(0,0,1))

deep++;

}

**用回溯法解题的一般步骤：**

　　（1）针对所给问题，定义问题的解空间；

　　（2）确定易于搜索的解空间结构；

（3）以深度优先方式搜索解空间，并在搜索过程中用剪枝函数避免无效搜索。

**链式前向星：**

struct Node

{

int to,next;

};

Node edge[MAXN];

int box[MAXN];

int ent=1;

void makeMap(int from,int to)

{

edge[ent].to=to;

edge[ent].next=box[from];

box[from]=ent++;

}

void edgefrom(int n) //输出以n为开头的所有边

{

for(int j=0; j<10; ++j)

for(int i=box[j]; i; i=edge[i].next)

{

cout<<j<<"to"<<edge[i].to<<endl;

}

}

**素数表：**

//如果只需要判断一个数是否为素数的话可以直接删去sushu数组

bool isn[100000000]={0};

int sushu[100000];

void printpri(int n)

{

int k=0,i;

int sqrtn=sqrt(n);

for(i=4;i<n;i+=2)

isn[i]=1;

sushu[k++]=2;

for(i=3;i<sqrtn;i+=2)

if(isn[i]==0)

{

sushu[k++]=i;

for(int s=2\*i,j=i\*i;j<n;j+=s)

isn[j]=1;

}

for(;i<n;i++)

if(isn[i]==0)

sushu[k++]=i;

for(i=0;i!=k;++i)

cout<<sushu[i]<<" ";

}

**堆排序：**

void maxsort(float a[],int count);

void maxbuild(float a[],int count);

void heapify(float a[],int start,int count);

int main()

{

freopen("in.txt","r",stdin);

float a[100];

float \*p=a+1;

int count=0;

printf("please input the data,end up by 0\n");

scanf("%f",p);

while(\*p++)

{

scanf("%f",p);

count++;

}

maxbuild(a,count);

maxsort(a,count);

p=a+1;

for(;count>0;count--)

printf("%f\t",\*p++);

return 0;

}

void heapify(float a[],int start,int count)

{

int m=start;

int k,max;

float t;

while(1)

{

k=m\*2;

if(k>count)

break;

else if(k+1>count)

{

if(a[k]>a[m])

{

t=a[k];

a[k]=a[m];

a[m]=t;

}

break;

}

else

{

if(a[k]>a[k+1])

max=k;

else max=k+1;

if(a[max]>a[m])

{

t=a[max];

a[max]=a[m];

a[m]=t;

}

}

m=max;

}

}

void maxbuild(float a[],int count)

{

int i;

for(i=count/2;i>0;i--)

heapify(a,i,count);

}

void maxsort(float a[],int count)

{

float t;

while(count>1)

{

t=a[count];

a[count]=a[1];

a[1]=t;

heapify(a,1,--count);

}

}

**线段树：**

Poj2777的模板，其实此题可以去掉lazy标记的，判断color是否只有一个就行

pushDowm函数和pushUp函数尽量不写，直接加代码

注意dfs的剪枝

const int maxn=100000+100;

struct Node

{

int l,r;

int color;

bool lazy;

};

Node tree[4\*maxn];

void buildtree(int l,int r,int n)

{

tree[n].r=r;

tree[n].l=l;

tree[n].color=1;

tree[n].lazy=0;

if(r==l) {return;}

int m=(l+r)>>1;

buildtree(l,m,n<<1);

buildtree(m+1,r,n<<1|1);

}

Inline void pushDown(int n)

{

if(tree[n].lazy)

{

tree[n<<1].color=tree[n<<1|1].color=tree[n].color;

tree[n].lazy=0; tree[n<<1].lazy=tree[n<<1|1].lazy=1;

}

}

Inline void pushUp(int n)

{

tree[n].color=tree[n<<1].color|tree[n<<1|1].color;

}

void change(int l,int r,int col,int n)

{

if(tree[n].l==l && tree[n].r==r)

{

tree[n].color=col;

if(l!=r)

tree[n].lazy=1;

return;

}

if(tree[n].color==col) return;

pushDown(n);

int m=(tree[n].l+tree[n].r)>>1;

if(r<=m) change(l,r,col,n<<1);

else if(l<=m && r>m)

{

change(l,m,col,n<<1);

change(m+1,r,col,n<<1|1);

}

else change(l,r,col,n<<1|1);

pushup(n);

}

int query(int l,int r,int n)

{

int res=0;

if(tree[n].l==l&&tree[n].r==r)

return tree[n].color;

pushDown(n);

int m=(tree[n].l+tree[n].r)>>1;

if(r<=m) res=res|query(l,r,n<<1);

else if(l<=m && r>m)

{

res=res|query(l,m,n<<1);

res=res|query(m+1,r,n<<1|1);

}

else res=res|query(l,r,n<<1|1);

return res;

}

**Rank测试数据：**

floor(static\_cast<double>(rand())\*(maxn-1)/RAND\_MAX+0.5)+1

WA篇：

一、是不是把boy的数据安到girl上了，n的值赋给m了……特别是样例中这一对数据还是相等的时候……

二、还是输入数据。如果一个测试点里有多组数据，有没有可能比方说sum = 0的时候结果一定为0，而无论后面的数据是什么？这样的话你可能会一个continue转到下一组数据。可是数据流才不会continue，后面没读完的数据理所当然的冲到了最前方，你的程序就这样WA了……

三、初始化。你的初始化正确么？边界条件是0，1，还是kn？

四、循环变量。for的循环变量如果是k，就看看后边用的是不是i。

五、你的智商够不够高，是否考虑到所有的问题了？你可以写一份暴力的代码，里面写什么连看都不看，之后编写一个“测试数据制造机”，数大，数据多，情况就复杂，就可能出现你没想到的状况，拿两个程序对一下答案就知道了。

RE篇：

一、数组是不是开小了？实在不行在Memory Limit允许的前提下开到数据范围的10倍以上。

二、除0错误。有多少高人在这个地方摔跤，甚至一蹶不振！全面检查一下 和 % 运算符的右边有没有可能等于 0。

三、数组起始位置。如果定义a[MAX]，就千万不要使用a[MAX]这个值。事实上，这样一般都引发WA。

四、死循环的话一般不报TLE，特别是循环中有数组操作的情况。因为这种情况下通常会出现a[-6569]一类的囧物……RE是理所应当的。死循环的话，就看看循环结束的条件是不是正确。