**算法训练 逆序对**

时间限制：1.0s   内存限制：256.0MB

问题描述

Alice是一个让人非常愉跃的人!他总是去学习一些他不懂的问题，然后再想出许多稀奇古怪的题目。这几天，Alice又沉浸在逆序对的快乐当中，他已近学会了如何求逆序对对数，动态维护逆序对对数等等题目，他认为把这些题让你做简直是太没追求了，于是，经过一天的思考和完善，Alice终于拿出了一道他认为差不多的题目：

有一颗2n-1个节点的二叉树，它有恰好n个叶子节点，每个节点上写了一个整数。如果将这棵树的所有叶子节点上的数从左到右写下来，便得到一个序列a[1]…a[n]。现在想让这个序列中的逆序对数量最少，但唯一的操作就是选树上一个非叶子节点，将它的左右两颗子树交换。他可以做任意多次这个操作。求在最优方案下，该序列的逆序对数最少有多少。

Alice自己已近想出了题目的正解，他打算拿来和你分享，他要求你在最短的时间内完成。

输入格式

第一行一个整数n。

下面每行，一个数x。

如果x=0，表示这个节点非叶子节点，递归地向下读入其左孩子和右孩子的信息，如果x≠0，表示这个节点是叶子节点，权值为x。

输出格式

输出一个整数，表示最少有多少逆序对。

样例输入

3  
0  
0  
3  
1  
2

样例输出

1

数据规模与约定

对于20%的数据，n <= 5000。

对于100%的数据，1 <= n <= 200000，0 <= a[i]<2^31。

锦囊1

使用平衡树。

锦囊2

从叶子到根依次处理，每次把结点个数少的树中的结点依次添加到结点个数多的里面，并顺便计算逆序对个数和两棵子树交换后的逆序对个数。

本题的C++参考代码如下：

#include<stdio.h>

#include<iostream>

using namespace std;

#define ForD(i,n) for(int i=n;i;i--)

#define F (100000007)

#define MAXN (2\*200000+10)

long long mul(long long a,long long b){return (a\*b)%F;}

long long add(long long a,long long b){return (a+b)%F;}

long long sub(long long a,long long b){return (a-b+(a-b)/F\*F+F)%F;}

int n,root=0;

struct node

{

int fa,ch[2],size,c;

node():size(0),c(0){ch[0]=ch[1]=fa=0;}

}a[MAXN];

void update(int x){a[x].size=a[a[x].ch[0]].size+a[a[x].ch[1]].size+(a[x].c>0);}

int tail=0;

void pushdown(int x){a[a[x].ch[0]].fa=a[a[x].ch[1]].fa=x;}

void build(int &x)

{

if (!x) x=++tail;

scanf("%d",&a[x].c);

if (a[x].c==0)

{

build(a[x].ch[0]);

build(a[x].ch[1]);

update(x);pushdown(x);

}else a[x].size=1;

}

void rotate(int x)

{

int y=a[x].fa,z=a[y].fa;

bool p=a[y].ch[0]==x;

if (z)

{

if (a[z].ch[0]==y) a[z].ch[0]=x;

else a[z].ch[1]=x;

}

a[x].fa=z,a[y].fa=x;

if (a[x].ch[p]) a[a[x].ch[p]].fa=y;

a[y].ch[p^1]=a[x].ch[p];

a[x].ch[p]=y;

update(y);

}

void splay(int x)

{

while (a[x].fa)

{

int y=a[x].fa,z=a[y].fa;

if (z)

if ((a[y].ch[0]==x)^(a[z].ch[0]==y)) rotate(x);

else rotate(y);

rotate(x);

}

update(x);

}

void ins(long long &tot,int x,int y)

{

a[x].size++;

if (a[y].c<=a[x].c)

{

if (a[x].ch[0]) ins(tot,a[x].ch[0],y);

else a[y].fa=x,splay(a[x].ch[0]=y);

}

else

{

tot+=a[a[x].ch[0]].size+(a[x].c>0);

if (a[x].ch[1]) ins(tot,a[x].ch[1],y);

else a[y].fa=x,splay(a[x].ch[1]=y);

}

}

int q[MAXN],size;

void clac(int x,int y)

{

if (a[y].ch[0]) clac(x,a[y].ch[0]);

if (a[y].c) q[++size]=y;

if (a[y].ch[1]) clac(x,a[y].ch[1]);

}

long long merge(bool &lor,int z)

{

int x=a[z].ch[0],y=a[z].ch[1];

if (a[x].size<a[y].size) swap(x,y);

a[x].fa=0;a[y].fa=0;q[1]=y;

size=0;clac(x,y);

long long tot=0;

ForD(i,size)

{

int now=q[i];

a[now].ch[0]=a[now].ch[1]=a[now].fa=0;a[now].size=1;

ins(tot,x,now);

x=now;

}

a[x].fa=z;

a[z].ch[0]=0,a[z].ch[1]=x;

return tot;

}

long long qur(int &x)

{

if (a[x].c) return 0;

else

{

long long lson=a[a[x].ch[0]].size,rson=a[a[x].ch[1]].size,ls=qur(a[x].ch[0]),rs=qur(a[x].ch[1]);

bool lor=0;

long long ms=merge(lor,x);

return ls+rs+min(lson\*rson-ms,ms);

}

}

int main()

{

scanf("%d",&n);

build(root);

cout<<qur(root)<<endl;

return 0;

}

本题的C参考代码如下：

// 蓝桥杯 逆序对 c语言实现

#include<stdio.h>

#include<stdlib.h>

#define MAXN 200005

int n,m;

long long ans;

long long min(long long a,long long b){

if(a>b) return b;

return a;

}

int S[MAXN],Left[MAXN],Right[MAXN];

unsigned int Key[MAXN];

int Left\_Rotate(int rt){

int k=Right[rt];

Right[rt]=Left[k];

Left[k]=rt;

S[k]=S[rt];

S[rt]=S[Left[rt]]+S[Right[rt]]+1;

return rt=k;

}

int Right\_Rotate(int rt){

int k=Left[rt];

Left[rt]=Right[k];

Right[k]=rt;

S[k]=S[rt];

S[rt]=S[Left[rt]]+S[Right[rt]]+1;

return rt=k;

}

int Maintain(int rt,int flag){

if(flag){

if(S[Left[Right[rt]]]>S[Left[rt]] || S[Right[Right[rt]]]>S[Left[rt]]){

if(S[Left[Right[rt]]]>S[Left[rt]])Right[rt]=Right\_Rotate(Right[rt]);

return Left\_Rotate(rt);

}

}

else{

if(S[Right[Left[rt]]]>S[Right[rt]] || S[Left[Left[rt]]]>S[Right[rt]]){

if(S[Right[Left[rt]]]>S[Right[rt]])Left[rt]=Left\_Rotate(Left[rt]);

return Right\_Rotate(rt);

}

}

return rt;

}

int Insert(int rt,int RT){

S[rt]+=1;

if(Key[RT]>Key[rt]){

if(Right[rt]>0)Right[rt]=Insert(Right[rt],RT);

else{

Right[rt]=RT;

}

}

else{

if(Left[rt]>0)Left[rt]=Insert(Left[rt],RT);

else{

Left[rt]=RT;

}

}

return rt=Maintain(rt,Key[RT]>Key[rt]);

}

int Rank(int rt,int v){

if(v>Key[rt]){

if(Right[rt]==0)return S[rt];

return S[rt]-S[Right[rt]]+Rank(Right[rt],v);

}

if(Left[rt]==0)return 0;

return Rank(Left[rt],v);

}

int Merge(int rt,int Start,int End){

int i;

long long lans=0,rans=0;

for(i=Start;i<=End;i++){

lans+=Rank(rt,Key[i]);

rans+=S[rt]-Rank(rt,Key[i]+1);

}

for(i=Start;i<=End;i++){

S[i]=1;

Left[i]=Right[i]=0;

rt=Insert(rt,i);

}

ans+=min(lans,rans);

return rt;

}

int Init(){

int v;

scanf("%d",&v);

if(v){

Key[++m]=v;

S[m]=1;

Left[m]=Right[m]=0;

return m;

}

int TL,TR,Ls,Le,Rs,Re;

Ls=m+1;

TL=Init();

Le=m;

Rs=m+1;

TR=Init();

Re=m;

if(S[TL]<S[TR])return Merge(TR,Ls,Le);

return Merge(TL,Rs,Re);

}

int main()

{

scanf("%d",&n);

Init();

printf("%I64d\n",ans);

system("pause");

return 0;

}

本题的Java参考代码如下：

无