给定两个长度为n的整数列A和B，每次你可以从A数列的左端或右端取走一个数。假设第i次取走的数为ax，则第i次取走的数的价值vi=bi⋅ax，现在希望你求出∑vi的最大值。

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

const ll MOD=1e9+7;

ll a[2100],b[2100],f[2100];

int main()

{

int T,n;

cin>>T;

while(T--)

{

cin>>n;

for(int i=1;i<=n;i++) scanf("%lld",&a[i]);

for(int i=1;i<=n;i++) scanf("%lld",&b[i]);

memset(f,0,sizeof(f));

//f[i][j]表示第i次取数后右面共取j个的最大值

//f[i][j]=max(f[i-1][j-1]+i\*a[n-j+1],f[i-1][j]+i\*a[i-j]);(j<=i)

f[0]=b[1]\*a[1];f[1]=b[1]\*a[n];

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

{

for(int j=i;j>=0;j--)

{

if(j) f[j]=max(f[j-1]+b[i]\*a[n-j+1],f[j]+b[i]\*a[i-j]);

else f[j]+=b[i]\*a[i];

}

}

ll \_max=0;

for(int i=0;i<=n;i++) if(f[i]>\_max) \_max=f[i];

printf("%lld\n",\_max);

}

return 0;

}

图形分形

当n=0时输出  
O  
当n=1时输出  
\*O  
O\*O  
\*O  
当n=2时输出  
\*\*\*\*O  
\*\*\*O\*O  
\*\*\*\*O  
\*O\*\*\*\*\*O  
O\*O\*\*\*O\*O  
\*O\*\*\*\*\*O  
\*\*\*\*O  
\*\*\*O\*O  
\*\*\*\*O

#include<bits/stdc++.h>

using namespace std;

char f[10][2500][2500];

int a[10],g[10][2500];

void fil(int n,int x,int y,int m){

x--,y--;

for(int i=1;i<=a[m];i++){

for(int j=1;j<=a[m];j++)

f[n][x+i][y+j]=f[m][i][j];

g[n][x+i]=max(g[n][x+i],g[m][i]+y); 计算换行位置

}

}

int main()

{

a[0]=g[0][1]=f[0][1][1]=1;

for(int i=1;i<=7;i++){

a[i]=a[i-1]\*3;

fil(i,a[i-1]+1,1,i-1); //以左上为原点 填充四个区域块

fil(i,a[i-1]+1,2\*a[i-1]+1,i-1);

fil(i,1,a[i-1]+1,i-1);

fil(i,2\*a[i-1]+1,a[i-1]+1,i-1);

}

int n;

int T;

cin>>T;

while(T--)

{

cin>>n;

for(int i=1;i<=a[n];i++){

for(int j=1;j<=g[n][i];j++)

putchar(f[n][i][j]?'O':' ');

puts("");

}

}

return 0;

}

树的重心

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

vector<int>tree[1005];

const int inf=10000000;

int num[1005];

int n;

int minNode,minbalance;

void bfs(int u,int v){

int minson=0;

num[u]=1;

for(int i=0;i<tree[u].size();i++){

int son=tree[u][i];

if(son!=v){

bfs(son,u);

num[u]+=num[son];

minson=max(minson,num[son]);

}

}

minson=max(minson,n-num[u]);

if(minson<minbalance){

minbalance=minson;

minNode=u;

}

else if(minson==minbalance)

minNode=min(minNode,u);

}

int main()

{

while(cin>>n){

for(int i=0;i<1005;i++) tree[i].clear();

for(int i=0;i<n-1;i++){

int u,v;

scanf("%d%d",&u,&v);

tree[u].push\_back(v);

tree[v].push\_back(u);

}

minNode=0;

minbalance=inf;

bfs(1,0);

cout<<minNode<<' '<<minbalance<<endl;

}

return 0;

}

线段树终极版

#include<bits/stdc++.h>

#define MAXN 100010

#define inf 0x3f3f3f3f

using namespace std;

struct node{

int l,r;//区间[l,r]

int add;//区间的延时标记

int sum;//区间和

int mx; //区间最大值

int mn; //区间最小值

}tree[MAXN<<2];//一定要开到4倍多的空间

void pushup(int index){

tree[index].sum = tree[index<<1].sum+tree[index<<1|1].sum;

// tree[index].mx = max(tree[index<<1].mx,tree[index<<1|1].mx);

// tree[index].mn = min(tree[index<<1].mn,tree[index<<1|1].mn);

}

void pushdown(int index){

//说明该区间之前更新过

//要想更新该区间下面的子区间，就要把上次更新该区间的值向下更新

if(tree[index].add > 0){

//替换原来的值

/\*

tree[index<<1].sum = (tree[index<<1].r-tree[index<<1].l+1)\*tree[index].add;

tree[index<<1|1].sum = (tree[index<<1|1].r-tree[index<<1|1].l+1)\*tree[index].add;

tree[index<<1].mx = tree[index].add;

tree[index<<1|1].mx = tree[index].add;

tree[index<<1].mn = tree[index].add;

tree[index<<1|1].mn = tree[index].add;

tree[index<<1].add = tree[index].add;

tree[index<<1|1].add = tree[index].add;

tree[index].add = 0;\*/

//在原来的值的基础上加上val

tree[index<<1].sum += (tree[index<<1].r-tree[index<<1].l+1)\*tree[index].add;

tree[index<<1|1].sum +=(tree[index<<1|1].r-tree[index<<1|1].l+1)\*tree[index].add;

// tree[index<<1].mx += tree[index].add;

// tree[index<<1|1].mx += tree[index].add;

// tree[index<<1].mn += tree[index].add;

// tree[index<<1|1].mn += tree[index].add;

tree[index<<1].add += tree[index].add;

tree[index<<1|1].add += tree[index].add;

tree[index].add = 0;

}

}

void build(int l,int r,int index){

tree[index].l = l;

tree[index].r = r;

tree[index].add = 0;//刚开始一定要清0

if(l == r){

scanf("%d",&tree[index].sum);

tree[index].mn = tree[index].mx = tree[index].sum;

return ;

}

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

build(l,mid,index<<1);

build(mid+1,r,index<<1|1);

pushup(index);

}

void updata(int l,int r,int index,int val){

if(l <= tree[index].l && r >= tree[index].r){

/\*把原来的值替换成val,因为该区间有tree[index].r-tree[index].l+1

个数，所以区间和 以及 最值为：

\*/

/\*tree[index].sum = (tree[index].r-tree[index].l+1)\*val;

tree[index].mn = val;

tree[index].mx = val;

tree[index].add = val;//延时标记\*/

//在原来的值的基础上加上val,因为该区间有tree[index].r-tree[index].l+1

//个数，所以区间和 以及 最值为：

tree[index].sum += (tree[index].r-tree[index].l+1)\*val;

// tree[index].mn += val;

// tree[index].mx += val;

tree[index].add += val;//延时标记

return ;

}

pushdown(index);

int mid = (tree[index].l+tree[index].r)>>1;

if(l <= mid){

updata(l,r,index<<1,val);

}

if(r > mid){

updata(l,r,index<<1|1,val);

}

pushup(index);

}

int query(int l,int r,int index){

if(l <= tree[index].l && r >= tree[index].r){

return tree[index].sum;

// return tree[index].mx;

//return tree[index].mn;

}

pushdown(index);

int mid = (tree[index].l+tree[index].r)>>1;

int ans = 0;

int Max = 0;

int Min = inf;

if(l <= mid){

ans += query(l,r,index<<1);

// Max = max(query(l,r,index<<1),Max);

// Min = min(query(l,r,index<<1),Min);

}

if(r > mid){

ans += query(l,r,index<<1|1);

// Max = max(query(l,r,index<<1|1),Max);

// Min = min(query(l,r,index<<1|1),Min);

}

return ans;

// return Max;

//return Min;

}

int main()

{

int n,m,q,x,y,z;

cin>>n>>m;

build(1,n,1);

while(m--){

scanf("%d",&q);

if(q == 2){

// cout<<"查询:(x,y)"<<endl;

scanf("%d %d",&x,&y);

// cout<<query(x,y,1)<<endl;

printf("%d\n",query(x,y,1));

}

else{

// cout<<"更新(x,y)为z："<<endl;

scanf("%d %d",&x,&y);

updata(x,x,1,y);

// for(int i = 1; i <= n; ++i){

// printf("a[%d] = %d\n",i,query(i,i,1));

// }

}

}

return 0;

}