**矩阵翻转**

问题描述

Ciel有一个N\*N的矩阵，每个格子里都有一个整数。

N是一个奇数，设X = (N+1)/2。Ciel每次都可以做这样的一次操作：他从矩阵选出一个X\*X的子矩阵，并将这个子矩阵中的所有整数都乘以-1。

现在问你经过一些操作之后，矩阵中所有数的和最大可以为多少。

输入格式

第一行为一个正整数N。

接下来N行每行有N个整数，表示初始矩阵中的数字。每个数的绝对值不超过1000。

输出格式

输出一个整数，表示操作后矩阵中所有数之和的最大值。

样例输入

3  
-1 -1 1  
-1 1 -1  
1 -1 -1

样例输出

9

数据规模与约定

1 <= N <= 33，且N为奇数。

锦囊1

使用贪心。

锦囊2

令dp[i][j]表示矩阵中(i, j)位置的数是否要乘以-1。

注意dp[i][j]^dp[i][x]^dp[i][j+x] == 0,

dp[i][j]^dp[x][j]^dp[i+x][j] == 0 ;

所以只要确定了dp[x][j], dp[i][x], 就能将dp[i][j] 与 dp[i+x][j], dp[i][j+x]绑定起来了。

首先我们可以枚举dp[x][j]--可以通过枚举使用了哪些以(x, j)为左上角的矩形，那么这样，我们可以同时将dp[x][j+x]确定下来。

然后我们依次dp[j][x](j<x)--我们可以发现，对于每一个dp[j][x],其影响的只有dp[j][t](t<x)与dp[j][t+x]的关系以及dp[j+x][x]，所以任两个dp[j][x]是线性无关的。

那么我们就可以一个个枚举dp[j][x]是否为1，然后运用贪心。

C++代码：

#include<stdio.h>

#include<cmath>

using namespace std;

int x[33][33],ans,N;

void fun1(int n)

{

int i,j,lin=0,aa,bb;

for(j=0;j<N;j++)

lin+=x[n-1][j];

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

{

aa=-1000000000;

bb=x[i][n-1]+x[i+n][n-1];

for(j=0;j<n-1;j++)

bb+=abs(x[i][j]+x[i+n][j]+x[i][j+n]+x[i+n][j+n]);

aa=aa>bb?aa:bb;

bb=-x[i][n-1]-x[i+n][n-1];

for(j=0;j<n-1;j++)

bb+=abs(-x[i][j]-x[i+n][j]+x[i][j+n]+x[i+n][j+n]);

aa=aa>bb?aa:bb;

lin+=aa;

}

ans=ans>lin?ans:lin;

}

void fun(int n)

{

int i,j,k;

for(k=0;k<(1<<n-1);k++)

{

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

if((k&(1<<i))!=0)

for(j=0;j<n;j++)

{

x[j][i]\*=-1;

x[j][i+n]\*=-1;

}

fun1(n);

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

if((k&(1<<i))!=0)

for(j=0;j<n;j++)

{

x[j][i]\*=-1;

x[j][i+n]\*=-1;

}

}

}

int main(void)

{

int i,j,k;

scanf("%d",&N);

for(i=0;i<N;i++)

for(j=0;j<N;j++)

scanf("%d",&x[i][j]);

k=(N+1)/2;

ans=-1000000000;

fun(k);

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

for(j=0;j<k;j++)

x[i][j]=-x[i][j];

fun(k);

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

return 0;

}

C代码：

#include<stdio.h>

int x[33][33],ans,N;

void fun1(int n)

{

int i,j,lin=0,aa,bb;

for(j=0;j<N;j++)

lin+=x[n-1][j];

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

{

aa=-1000000000;

bb=x[i][n-1]+x[i+n][n-1];

for(j=0;j<n-1;j++)

bb+=abs(x[i][j]+x[i+n][j]+x[i][j+n]+x[i+n][j+n]);

aa=aa>bb?aa:bb;

bb=-x[i][n-1]-x[i+n][n-1];

for(j=0;j<n-1;j++)

bb+=abs(-x[i][j]-x[i+n][j]+x[i][j+n]+x[i+n][j+n]);

aa=aa>bb?aa:bb;

lin+=aa;

}

ans=ans>lin?ans:lin;

}

void fun(int n)

{

int i,j,k;

for(k=0;k<(1<<n-1);k++)

{

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

if((k&(1<<i))!=0)

for(j=0;j<n;j++)

{

x[j][i]\*=-1;

x[j][i+n]\*=-1;

}

fun1(n);

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

if((k&(1<<i))!=0)

for(j=0;j<n;j++)

{

x[j][i]\*=-1;

x[j][i+n]\*=-1;

}

}

}

int main(void)

{

int i,j,k;

scanf("%d",&N);

for(i=0;i<N;i++)

for(j=0;j<N;j++)

scanf("%d",&x[i][j]);

k=(N+1)/2;

ans=-1000000000;

fun(k);

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

for(j=0;j<k;j++)

x[i][j]=-x[i][j];

fun(k);

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

return 0;

}

JAVA代码：

import java.io.IOException;

import java.io.InputStream;

public class Main

{

private static class MyScanner

{

private InputStream is = System.in;

public int nextInt() throws IOException

{

int i;

while ((i=is.read())<45||i>57)

{

}

int mark = 1, temp = 0;

if (i==45)

{

mark = -1;

i = is.read();

}

while (i>47&&i<58)

{

temp = temp\*10+i-48;

i = is.read();

}

return temp\*mark;

}

}

private static int x;

private static int[][] map, symbol;

private static int[][][][][] dp;

private static int calculate(int j)

{

int sum = 0;

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

{

if (dp[symbol[i][x]+1][symbol[x][j]+1][symbol[x][x]+1][i][j]>0)

{

sum += dp[symbol[i][x]+1][symbol[x][j]+1][symbol[x][x]+1][i][j];

continue;

}

int i2 = x+i+1;

int j2 = x+j+1;

int temp = map[i][j];

temp += map[i][j2]\*symbol[i][x];

temp += map[i2][j]\*symbol[x][j];

temp += map[i2][j2]\*symbol[i2][x]\*symbol[x][j2];

sum += Math.abs(temp);

dp[symbol[i][x]+1][symbol[x][j]+1][symbol[x][x]+1][i][j] = Math.abs(temp);

}

return sum;

}

public static void main(String[] args) throws IOException

{

MyScanner sc = new MyScanner();

int n = sc.nextInt();

map = new int[n][n];

symbol = new int[n][n];

dp = new int[3][3][3][n][n];

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

{

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

{

map[i][j] = sc.nextInt();

symbol[i][j] = 1;

}

}

x = n/2;

int maxSum = Integer.MIN\_VALUE;

for (int count = (int) Math.pow(2, x+1); count >= 0; --count)

{

int k = count;

int center = (k&1)>0?1:-1;

k >>= 1;

symbol[x][x] = center;

int sum = map[x][x]\*center;

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

{

int t = (k&1)>0?1:-1;

symbol[j][x] = t;

symbol[x+j+1][x] = t\*center;

sum += map[j][x]\*t;

sum += map[x+j+1][x]\*t\*center;

k >>= 1;

}

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

{

int j2 = x+j+1;

symbol[x][j] = 1;

symbol[x][j2] = center;

int temp = calculate(j)+map[x][j]+map[x][j2]\*center;

symbol[x][j] = -1;

symbol[x][j2] = -1\*center;

sum += Math.max(temp, calculate(j)-map[x][j]+map[x][j2]\*-1\*center);

}

maxSum = Math.max(maxSum, sum);

}

System.out.println(maxSum);

}

}