二维前缀和

给定一个N行M列的非负整数矩阵，求一个最大的正方形子矩阵，该矩阵满足：

矩阵中每一个元素权值都大于0

在满足上述条件的前提下，矩阵面积最大

在满足上述条件的前提下，选择元素和最小的

//一开始预处理前缀和，然后二分正方形边长，枚举正方形的位置即可

int n,m,a[1005][1005],ans1;

long long sum[1005][1005],sum0[1005][1005],ans2=1LL<<60;

bool checkzero(int x,int y,int len){

int newx = x+len-1;

int newy = y+len-1;

//cout << x << ' ' << y << ' ' << newx << ' ' << newy << ' ' << len << endl;

// 右下角面积 == 4 - 3 - 2 + 1

return (sum0[newx][newy]-sum0[x-1][newy]

-sum0[newx][y-1]+sum0[x-1][y-1] )==0; // 巧妙}

bool check(int len){

int i,j;

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

for(j=1;j+len-1<=m;j++)// 枚举每个矩形

if(checkzero(i,j,len)) return true;

return false;}

int main(){

int i,j,nowsum0,l,r,mid,nowi,nowj;

long long nowsum;

scanf("%d%d",&n,&m);

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

for(j=1;j<=m;j++)

scanf("%d",&a[i][j]);/// 读入

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

{

nowsum=0LL;

nowsum0=0;

for(j=1;j<=m;j++)

{

nowsum += a[i][j];

nowsum0 += (a[i][j]==0);

sum[i][j]=sum[i-1][j]+nowsum;

sum0[i][j]=sum0[i-1][j]+nowsum0; // 巧妙

//cout << i << ' ' << j << ' ' << sum0[i][j] << endl;

}

}

// nowsum:当前列的前缀和

// sum[i][j]:矩阵i j的前缀和

// nowsum: 当前列的前缀和有为0的存在

// sum0[i][j] 矩阵i j的 0的前缀和

/\*

0 0 0 1 1 1 1

0 0 0 2 2 2 2

0 0 0 3 3 3 3\*/

l=1;// ans矩阵的边长的最小值

r=n<m?n:m; // ans矩阵的边长的最大值

while(l<=r) // 二分求得边长

{

mid=(l+r)>>1;

if(check(mid))

{

ans1=mid;

l=mid+1;

}

else r=mid-1;

}

printf("%d ",ans1\*ans1);// 边长乘以边长: 求得满足条件的面积

for(i=1;i+ans1-1<=n;i++)

for(j=1;j+ans1-1<=m;j++)

if(checkzero(i,j,ans1)) // 再次枚举是否矩阵包含0

{

nowi=i+ans1-1;

nowj=j+ans1-1;

ans2=min(ans2,sum[nowi][nowj]-sum[i-1][nowj]

-sum[nowi][j-1]+sum[i-1][j-1]);

}

printf("%lld",ans2);

return 0;

}