该题就是在一个矩阵中找到一个子矩阵，该子矩阵和最大！！输出最大和即可。。

思路：

首先你可能会想到穷举的方法，但当n很大时，显然是不可取的。。这个题应该是用dp来解决的。

让我们先来看另外的一个问题(最大子段和问题)： 给定一个长度为n的一维数组a，请找出此数组的一个子数组,使得此子数组的和sum＝a[i]+a[i+1]+……+a[j]最大，其中i>=0,i<n,j>=i,j<n，例如 31 -41 59 26 -53  58 97 -93 -23 84     子矩阵59+26-53+58+97=187为所求的最大子数组。这个题会写吧，简单的dp。

该题就是在一个矩阵中找到一个子矩阵，该子矩阵和最大！！输出最大和即可。。

思路：

首先你可能会想到穷举的方法，但当n很大时，显然是不可取的。。这个题应该是用dp来解决的。

让我们先来看另外的一个问题(最大子段和问题)： 给定一个长度为n的一维数组a，请找出此数组的一个子数组,使得此子数组的和sum＝a[i]+a[i+1]+……+a[j]最大，其中i>=0,i<n,j>=i,j<n，例如 31 -41 59 26 -53  58 97 -93 -23 84     子矩阵59+26-53+58+97=187为所求的最大子数组。这个题会写吧，简单的dp。

联系本题，二者区别呢？一个是一维的，一个是多维的，那么把多维转化为一维不就可以ac了。。

 假设最大子矩阵的结果为从第r行到k行、从第i列到j列的子矩阵，如下所示(ari表示a[r][i],假设数组下标从1开始)：  
  | a11 …… a1i ……a1j ……a1n |  
  | a21 …… a2i ……a2j ……a2n |  
  |  .     .     .    .    .     .    .   |  
  |  .     .     .    .    .     .    .   |  
  | ar1 …… ari ……arj ……arn |  
  |  .     .     .    .    .     .    .   |  
  |  .     .     .    .    .     .    .   |  
  | ak1 …… aki ……akj ……akn |  
  |  .     .     .    .    .     .    .   |  
  | an1 …… ani ……anj ……ann |

 那么我们将从第r行到第k行的每一行中相同列的加起来，可以得到一个一维数组如下：  
 (ar1+……+ak1, ar2+……+ak2, ……,arn+……+akn)  
 由此我们可以看出最后所求的就是此一维数组的最大子断和问题，到此我们已经将问题转化为上面的已经解决了的问题了。

收获：又认识了下最大连续子段和

在一维情况下最大连续子段和的求法是从左到有顺序扫描数据，以0为边界，当累加和小于0时则重置为0.动态规划的状态转移方程为

s=max{si-1+ai,ai}，该方程和前面的描述是等价的。本题是对一维最大子段和的扩展，思路是从上到下找出所有的连续行(如第i行到第j行)，然后计算每列从第i行到第j行的和，之后对这n个列的和进行一维最大子段和的计算，并找出最大的值。

#include <iostream>

#include <cstdio>

#include <cstring>

#include <cmath>

#include <algorithm>

#define maxn 110

using namespace std;

int a[maxn][maxn];

int dp[maxn][maxn];

int main()

{

//freopen("input.txt","r",stdin);

int n;

scanf("%d",&n);

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

for(int j=1;j<=n;j++)

{

scanf("%d",&a[i][j-1]);

dp[i][j]=dp[i][j-1]+a[i][j-1];

}

int ans=0;int sum;

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

for(int j=i;j<=n;j++)//从第i行到第j行

{

sum=0;

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

{

sum+=dp[k][j]-dp[k][i];//每列的和

if(sum<0)sum=0;

ans=max(ans,sum);

}

}

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

return 0;

}