1282：最大子矩阵

题目描述

已知矩阵的大小定义为矩阵中所有元素的和。给定一个矩阵，你的任务是找到最大的非空(大小至少是1 × 1)子矩阵。

比如，如下4 × 4的矩阵

```
0   -2 -7   0
9   2 -6   2
-4   1 -4   1
-1   8   0 -2
```

的最大子矩阵是

```
 9 2
-4 1
-1 8
```

这个子矩阵的大小是15。

输入

输入是一个N×N的矩阵。输入的第一行给出N(0<N≤100)。再后面的若干行中，依次(首先从左到右给出第一行的N个整数，再从左到右给出第二行的N个整数……)给出矩阵中的N2个整数，整数之间由空白字符分隔(空格或者空行)。已知矩阵中整数的范围都在[-127,127]。

输出

输出最大子矩阵的大小。

输入样例

```
4
0 -2 -7   0
9   2 -6   2
-4   1 -4   1
-1   8   0 -2
```

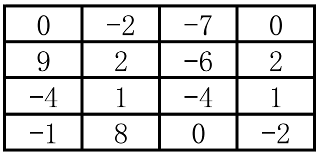
输出样例

```
15
```

解析

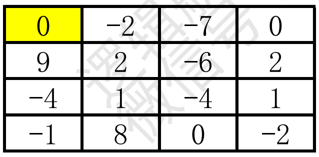
求子矩阵的最大和没有什么好的办法，只能枚举各种组合的结果，直至比较出最大值。在算法上，我们所能做的就是减少计算量，比较示意如下。

原数组



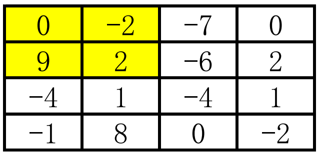
0	-2	-7	0
9	2	-6	2
-4	1	-4	1
-1	8	0	-2

原数组



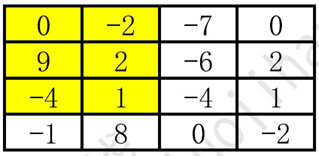
0	-2	-7	0
9	2	-6	2
-4	1	-4	1
-1	8	0	-2

原数组



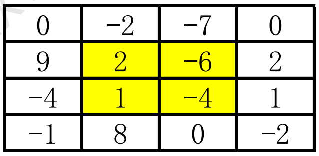
0	-2	-7	0
9	2	-6	2
-4	1	-4	1
-1	8	0	-2

原数组



0	-2	-7	0
9	2	-6	2
-4	1	-4	1
-1	8	0	-2

原数组

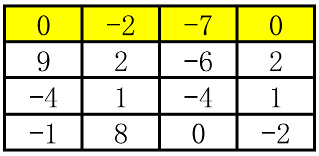


0	-2	-7	0
9	2	-6	2
-4	1	-4	1
-1	8	0	-2

遇到数组和的问题，我们首先要想到的就是前缀和，它能够大大的简化我们的计算。

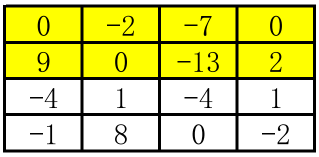
对于最大子矩阵，我们的计算过程是这样：

1、从第一行开始，依次计算每一列的前缀和。行数依次后移，如下图所示：



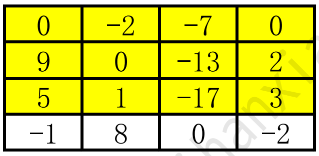
0	-2	-7	0
9	2	-6	2
-4	1	-4	1
-1	8	0	-2

第一行各列的前缀和就是他本身



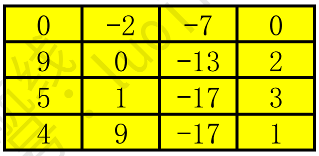
0	-2	-7	0
9	0	-13	2
-4	1	-4	1
-1	8	0	-2

计算前两行各列的前缀和



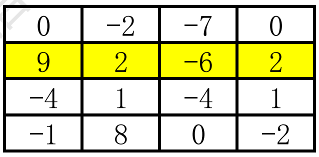
0	-2	-7	0
9	0	-13	2
5	1	-17	3
-1	8	0	-2

计算前三行各列的前缀和



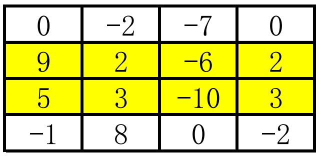
0	-2	-7	0
9	0	-13	2
5	1	-17	3
4	9	-17	1

计算前四行各列的前缀和



0	-2	-7	0
9	2	-6	2
-4	1	-4	1
-1	8	0	-2

计算以第二行为起始行的各列的前缀和

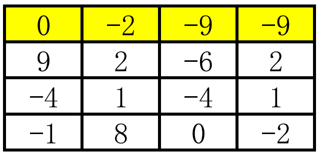


0	-2	-7	0
9	2	-6	2
5	3	-10	3
-1	8	0	-2

计算第二、三行的各列的前缀和

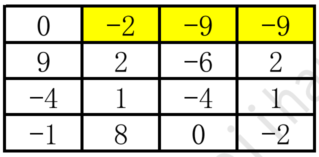
以此类推，直到计算到最后一行。

2、计算矩阵的最大值，每当我们计算完几个列的前缀和时，我们就可以把这些列加起来，求整个矩阵的最大值，当然，这个列数也是从左往右依次进行移动的，如下所示：



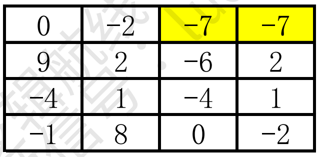
0	-2	-9	-9
9	2	-6	2
-4	1	-4	1
-1	8	0	-2

当前起始列为1，最大值为0



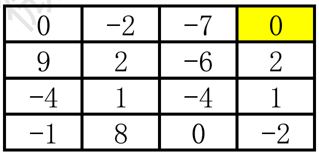
0	-2	-9	-9
9	2	-6	2
-4	1	-4	1
-1	8	0	-2

当前起始列为2，最大值为-2



0	-2	-7	-7
9	2	-6	2
-4	1	-4	1
-1	8	0	-2

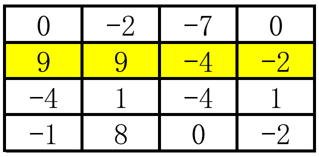
当前起始列为3，最大值为-7



0	-2	-7	0
9	2	-6	2
-4	1	-4	1
-1	8	0	-2

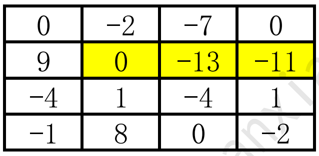
当前起始列为4，最大值为0

接下来，我们求前两行的矩阵最大值：



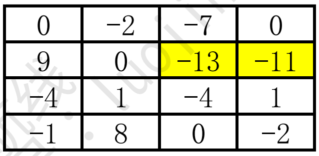
0	-2	-7	0
9	9	-4	-2
-4	1	-4	1
-1	8	0	-2

直接累加本行数据即可，很明显最大值为9



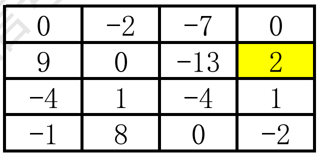
0	-2	-7	0
9	0	-13	-11
-4	1	-4	1
-1	8	0	-2

当前起始列为2，最大值为0



0	-2	-7	0
9	0	-13	-11
-4	1	-4	1
-1	8	0	-2

当前起始列为3，最大值为-11



0	-2	-7	0
9	0	-13	2
-4	1	-4	1
-1	8	0	-2

当前起始列为4，最大值为2

以此类推，直至遍历到最后一个格子。我们在遍历过程中不断的与全局最大值作比较，即可求出答案。

编码

```cpp
#include<bits/stdc++.h>

using namespace std;

const int N = 101;
int a[N][N];  //原始数组
int dp[N];  //当前矩阵的最大值
int f[N];   //列前缀和
int n;

//全局最大值，注意此处必须定义极小值，因为子矩阵和可能为负数
int mx = -INT_MAX;

//求出最大子矩阵
void solve() {
    //每次计算都要清理
    memset(dp, 0, sizeof(dp));
    //开始遍历每一列，将其想加
    for (int i = 1; i <= n; i++) {
        //计算当前列的最大值
        dp[i] = max(f[i], dp[i - 1] + f[i]);
        //用当前列的最大值和全局最大值作比较
        mx = max(mx, dp[i]);
    }
}

int main() {
    //读入数据
    scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        for (int j = 1; j <= n; j++) {
            scanf("%d", &a[i][j]);
        }
    }
    //开始遍历起始行数
    for (int i = 1; i <= n; i++) {
        //注意，每次重新计算前缀和时都要进行清理
        memset(f, 0, sizeof(f));
        //遍历第i行至最后一行
        for (int j = i; j <= n; j++) {
            //计算每一列的前缀和
            for (int k = 1; k <= n; k++) {
                f[k] += a[j][k];
            }
            //开始计算矩阵的最大值
            solve();
        }
    }
    printf("%d\n", mx);
    return 0;
}
```

逻辑航线培优教育，信息学奥赛培训专家。

扫码添加作者获取更多内容。

数组和

0	-2	-7	0
9	0	-13	2
5	1	-17	3
4	9	-17	1

原数组

0		#		#		0	
9		2		#		2	
#		1		#		1	
#		8		0		#	

原数组

0		#		#		0	
9		2		#		2	
#		1		#		1	
#		8		0		#	