AcWing-3326：最大硬币数

题目描述

Mike 有一个
N 行
N 列的方格矩阵。

位于第
i 行第
j 列的方格的位置坐标表示为
(i,j)。

矩阵左上角方格的坐标即为 $(1,1)$。

每个方格中都包含一定数量的硬币，Mike 只有到达一个方格内时，方可收集方格中的硬币。

$C_{i,j}$ 表示第 i 行第 j 列的方格中的硬币数量。

当 Mike 处于方格
(i,j) 时，他可以选择移动至方格
$(i-1,j-1)$ 或方格
$(i+1,j+1)$ 中，前提是所选择的方格位于矩阵边界内，且之前没有到达过。

Mike 可以选择从任意方格开始移动，也可以选择在移动至任意方格时结束移动。

Mike 希望尽可能多的收集硬币。

请帮助他确定他可以收集的最大硬币数量。

输入格式

第一行包含整数
T，表示共有
T 组测试数据。

每组数据第一行包含整数 N。

接下来
N 行，每行包含
N 个整数，其中第
i 行第
j 列的整数表示
$C_{i,j}$。

输出格式

每组数据输出一个结果，每个结果占一行。

结果表示为 `Case #x: y`，其中
x 为组别编号（从
1 开始），
y 为可以收集的最大硬币数量。

数据范围

60% 的数据满足，$1 \le T \le 100$，$1 \le N \le 100$。
另外 40% 的数据满足，$1 \le T \le 10$，$1 \le N \le 1000$。
100% 的数据满足，$0 \le C_{i,j} \le 10^7$。

时/空限制：	5s / 256MB
总通过数：	876
总尝试数：	1720
来源：	Google Kickstart2020 Round G Problem B
算法标签	

输入样例：

```
2
3
1 2 5
3 6 1
12 2 7
5
0 0 0 0 0
1 1 1 1 0
2 2 2 8 0
1 1 1 0 0
0 0 0 0 0
```

输出样例：

```
Case #1: 14
Case #2: 9
```

样例解释

对于测试数据 1，Mike 可以选择的行进路径为 (1,1)→(2,2)→(3,3)，可收集到的最大硬币数量为 14。

对于测试数据 2，Mike 可以选择的行进路径为 (2,3)→(3,4)，可收集到的最大硬币数量为 9。

算法求解

题意可以理解为，求最长的“斜边和”，且这个斜边是从左上角到右下角的。

这个“求和”的过程可以在数据输入时进行。每一条从左上到右下角的斜边都有特征——其斜边上每一个点`row-column`值相等。那么可以通过这个行列位置的差值作为和数组的下标，由于会产生负下标，可以选择使用`map`或所有差值都加上总行/列值来解决（以下代码采用后者）。

Code：

```cpp
//
//  main.cpp
//  3326-最大硬币数
//
//  Created by MacBook Pro on 2023/8/12.
//

#include <iostream>
#include <cmath>
using namespace std;

int main() {
```

```
    int T;
    scanf("%d",&T);
    for(int cases=1;cases<=T;cases++){
        long long int nums[3000]={0};
        int N;
        scanf("%d",&N);
        for(int i=1;i<=N;i++){
            for(int j=1;j<=N;j++){
                int x; scanf("%d",&x);
                nums[i-j+N]+=x;

            }
        }
        long long int ans = 0;
        for(int i=1;i<=2*N-1;i++){
            ans = max(ans,(long long)(nums[i]));
        }
        printf("Case #%d: %lld\n",cases,ans);
    }
    return 0;
}
```

需要注意的是，和数组和结果都应该用`long long int`表示。