AcWing-4726：步行者

约翰参加了一场步行比赛。

比赛为期
N 天，参赛者共
M 人（包括约翰）。

参赛者编号为
$1 \sim M$，其中约翰的编号为
P。

每个参赛者的每日步数都将被赛事方记录并公布。

每日步数最多的参赛者是当日的日冠军（可以有并列冠军）。

如果一名参赛者可以连续 N 天成为日冠军，那么他将成为创造历史的传奇冠军，这正是约翰的最终目标。

在比赛结束后，约翰拿到了所有选手的全部成绩，并试图分析自己在实现目标方面还差了多少步。

对于第 i 天，如果约翰是当日的冠军，那么他就会对当日的发挥表示满意，即当日不需要额外多走步数，如果约翰不是当日的冠军，那么他就会计算当日若要夺冠，还需要额外走的最小步数。

请你计算并输出，为了实现目标，约翰在这 N 天中需要额外走的最小总步数。

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三个整数 M, N, P。

接下来
M 行，每行包含
N 个整数，其中第
i 行第
j 个数
$S_{i,j}$ 表示第
i 个参赛者第
j 天的行走步数。

时/空限制：	1s / 64MB
总通过数：	1088
总尝试数：	2375
来源：	Google Kickstart2022 Round G Problem A
算法标签	

输入样例1:

```
1
2 3 1
1000 2000 3000
1500 1500 3000
```

输出样例1:

```
Case #1: 500
```

样例1解释

在此样例中，比赛为期
3 天，共
2 人参赛，约翰是
1 号选手。

第
1 天，约翰若想夺冠，至少还需额外走
500 步。

后两天，约翰均为日冠军，无需额外多走步数。

所以，他一共需要额外走 500 步。

输入样例2:

```
2
3 2 3
1000 2000
1500 4000
500 4000
3 3 2
1000 2000 1000
1500 2000 1000
500 4000 1500
```

该题是个很简单的求差`模拟`问题，很方便的解决方法是存储两个数组，`maxstep`存储每日最高步数，`john`存储每日john这个家伙的步数，最后做差即可。

时间复杂度最高的地方在于输入，达到`O(MN)`，后续处理复杂度较低，空间复杂度较高，达到`O(N)`，实际上是`2N`。

Code:

```cpp
//
//  main.cpp
//  4726-步行者
//
//  Created by MacBook Pro on 2023/7/21.
//

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

int john[35];       //JOHN每日步数
int maxstep[35];    //每日最高步数

int main() {
    int T;
```

```
    scanf("%d",&T);
    for(int cases=1;cases<=T;cases++){
        memset(maxstep,0,sizeof(maxstep));
        int M,N,P;
        long long int ans=0;
        scanf("%d%d%d",&M,&N,&P);
        for(int id=1;id<=M;id++){
            for(int days=1;days<=N;days++){
                int x;
                scanf("%d",&x);
                if(id==P){
                    //是john
                    john[days]=x;
                }
                maxstep[days]=max(maxstep[days],x);
            }
        }
        //cal bias
        for(int i=1;i<=N;i++){
            ans+=maxstep[i]-john[i];
        }
        printf("Case #%d: %lld\n",cases,ans);
    }
}
```

⚠️ 需要注意的是每一组数据处理前一定要**清空**`maxstep`数组！