Koishi 的地板

题意简述

给出一块 $n \times m$ 大小的矩形地板，求一块面积不为 0 的小矩形地板，这一块地板中的黑色瓷砖数量减去白色瓷砖数量是最大的。

题目解析

考虑黑色瓷砖和白色瓷砖对最终差值的贡献。多一块黑色瓷砖，最终答案 $+1$，多一块白色瓷砖，最终答案 -1。 因此我们把所有黑色瓷砖的位置替换为 1，把白色瓷砖的位置替换为 -1，构建出一个新的矩阵。

然后我们发现，如果直接枚举计算差值最大的子矩形，需要 $O((nm)^2)$ 的时间复杂度。考虑到 $1 \leq n, m \leq 400$，这样做是会超时的。

如何优化？我们发现我们要 $O((nm)^2)$ 的复杂度，是因为我们要枚举矩形的左上和右下两个顶点来确定矩形。那么有没有一种办法，能减少其中的某个枚举过程？

这时候我们可以尝试把矩形 **降维** 考虑。我们还是枚举矩形的左上角，然后枚举一行。**在枚举下一行的时候，把上一行的值累加到下一行。**从而把整个二维的矩形压缩成一维的。如此一来，时间复杂度就被降为了 $O(nm^2)$，是可以通过此题的。

对于每一行的处理，考虑记一个值 s，找最大子段和。每次把当前值加到 s 中，当 $s < 0$， 则把 s 重置为 0。在这之中更新最终答案 ans，最后输出 ans 的值即可。

代码

```
#include <stdio.h>
#include <string.h>
#define ll long long
#define rgt register int    //用于加速代码
#define qmx(a,b) a>b?a:b

int n,m,ans,s;
char mp[404][404];
int one_line[404];

inline void getmp(int line){
    char c=getchar();
    while(c!='0'&&c!='1')
        c=getchar();
    mp[line][1]=c;
    for(rgt i=2;i<=n;i++)
        mp[line][i]=getchar();
}   //用于快速读入一行的字符

int main(){
    scanf("%d%d",&m,&n);
    for(rgt i=1;i<=m;i++)
        getmp(i);
    for(rgt i=1;i<=m;i++){
        memset(one_line,0,sizeof(one_line));    //重置单行的累加值
        for(rgt j=i;j<=m;j++){
            for(rgt k=1;k<=n;k++)
```

```
                one_line[k]+=(mp[j][k]=='1'?1:-1);  //累加一行
            s=0;
            for(rgt k=1;k<=n;k++){
                s+=one_line[k];     //找最大子段和，更新答案
                ans=qmx(ans,s);
                if(s<0)
                    s=0;
            }
        }
    }
    printf("%d",ans);
    return 0;
}
```

补充

这题实际上是两道**动态规划**算法的经典例题最大子段和和最大加权矩形的结合体，建议可以先去完成这两题再回来看这题。