P3392. 涂国旗

方法一 n^3*m

```
#include <iostream>
using namespace std;
int n,m;
char c[100][100];
int main(){
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            cin>>c[i][j];//i行 j列 是颜色c
        }
    }
    int Min=3000;
    //n*n*n*m
    //50*50*50*50=6250000<1亿
    for(int l=2;l<=n-1;l++){//枚举蓝色的首行
        for(int r=l;r<=n-1;r++){//蓝色的末行
            //1到l-1 白色
            //l到r  蓝色
            //r+1到n  红色
            //统计1到n行所有需要变色的格子
            int sum=0;
            for(int i=1;i<=n;i++){
                for(int j=1;j<=m;j++){
                    if(i<=l-1&&c[i][j]!='W') sum++;
                    if(i>=l&&i<=r&&c[i][j]!='B') sum++;
                    if(i>r&&c[i][j]!='R') sum++;
                }
            }
            Min=min(Min,sum);
        }
    }
    cout<<Min;
}
```

方法二 n^3

```
#include <iostream>
using namespace std;
int n,m;
int W[100],R[100],B[100];
//W[i] 第i行 白色格子的数量
//R[i] 红色
//B[i] 蓝色
char c;
int main(){
    cin>>n>>m;
    for(int i=1;i<=n;i++){
```

```
        for(int j=1;j<=m;j++){
            cin>>c;//i行 j列 是颜色c
            if(c=='W')  W[i]++;
            if(c=='R')  R[i]++;
            if(c=='B')  B[i]++;
        }
    }
    int Min=3000;
    //n*n*n =50*50*50=125000
    for(int l=2;l<=n-1;l++){//枚举蓝色的首行
        for(int r=l;r<=n-1;r++){//蓝色的末行
            //1到l-1 白色
            //l到r  蓝色
            //r+1到n  红色
            //统计1到n行所有需要变色的格子
            int sum=0;
            for(int k=1;k<=n;k++){
                if(k<l) sum+=B[k]+R[k];//白色
                if(k>=l&&k<=r) sum+=W[k]+R[k];//蓝色
                if(k>r) sum+=B[k]+W[k];//红色
            }
            Min=min(sum,Min);
        }
    }
    cout<<Min;
}
```

方法三：前缀和优化 n^2

```
#include <iostream>
using namespace std;
int n,m;
int W[100],R[100],B[100];
int SW[100],SR[100],SB[100];
//W[i] 第i行 白色格子的数量
//R[i] 红色
//B[i] 蓝色
char c;
int main(){
    cin>>n>>m;
    for(int i=1;i<=n;i++){
        for(int j=1;j<=m;j++){
            cin>>c;//i行 j列 是颜色c
            if(c=='W')  W[i]++;
            if(c=='R')  R[i]++;
            if(c=='B')  B[i]++;
        }
    }
    for(int i=1;i<=n;i++){
        SW[i]=SW[i-1]+W[i];
        SR[i]=SR[i-1]+R[i];
        SB[i]=SB[i-1]+B[i];
    }
    //SW[1]=SW[0]+W[1]=W[1]
    //SW[2]=SW[1]+W[2]=W[1]+W[2]
```

```
    //SW[3]=SW[2]+W[3]=W[1]+W[2]+W[3]
    //SW[i]=W[1]+W[2]+...W[i] //前i行白色格子的数量的总和
    int Min=3000;
    //n*n*n =50*50*50=125000
    for(int l=2;l<=n-1;l++){//枚举蓝色的首行
        for(int r=l;r<=n-1;r++){//蓝色的末行
            //1到l-1 白色
            //l到r  蓝色
            //r+1到n  红色
            //统计1到n行所有需要变色的格子
            int sum=SB[l-1]+SR[l-1]+SW[r]-SW[l-1]+SR[r]-SR[l-1]+SB[n]-
SB[r]+SW[n]-SW[r];
//          //SB[l-1]  SR[l-1]
//          for(int k=1;k<=l-1;k++){
//              sum+=B[k]+R[k];//前l-1行 蓝色的格子数量总和 和 红色的总和
//          }
//          //SW[r]-SW[l-1]    SR[r]-SR[l-1]
//          for(int k=l;k<=r;k++){
//              sum+=W[k]+R[k];//l到r行  白色的数量总和 和 红色的总和
//          }
//          //SB[n]-SB[r]   SW[n]-SW[r]
//          for(int k=r+1;k<=n;k++){
//              sum+=B[k]+W[k];//l+1行n行 蓝色的总和 和 白色的总和
//          }
            Min=min(sum,Min);
        }
    }
    cout<<Min;
}
```

#P1028. [NOIP2001 普及组] 数的计算

方法一 n^2

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

int f[10000];
int n;

int main(){
    cin>>n;
    f[1]=1;
    for(int i=2;i<=n;i++){
        if(i%2==1) f[i]=f[i-1];
        else{
            //f[i]=f[1]+f[2]+.....f[i/2]+1
            for(int j=1;j<=i/2;j++){
                f[i]+=f[j];
            }
```

```
            f[i]=f[i]+1;
        }
    }
    cout<<f[n];
}
```

方法二 n

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

int f[10000],n;
int main(){
    cin>>n;
    f[1]=1;
    for(int i=2;i<=n;i++){
        if(i%2==1) f[i]=f[i-1];
        else f[i]=f[i/2]+f[i-1];
    }
    cout<<f[n];
}
```

高精度加

int类型 上限 2e9

long long 类型 上限 9e18

在大的数字一般采用字符串存储

```
string s;
cin>>s;//存一个字符串理论可以很长
```

```
#include <iostream>
using namespace std;
string s;

string plus_s(string a,string b){//字符串数字相加
    int x[100]={0};
    int y[100]={0};
    int z[100]={0};//存加法答案
    //字符串数字a 转化位整数数位同时倒过来 到x中
    int lena=a.size(),lenb=b.size();
    for(int i=lena-1,j=0;i>=0;i--,j++){//i是字符串a的遍历
        x[j]=a[i]-'0';
    }
    for(int i=lenb-1,j=0;i>=0;i--,j++){//i是字符串b的遍历
        y[j]=b[i]-'0';
    }
    int t=0;//当前进位量
    int len=max(lena,lenb);
    for(int i=0;i<=len;i++){
```

```
        z[i]=x[i]+y[i]+t;
        if(z[i]>=10){
            z[i]-=10;
            t=1;
        }else{
            t=0;
        }
    }
    string c="";
    int k;
    for(int i=len;i>=0;i--){
        if(z[i]!=0){
            k=i;//从第k位往下都是有效数字
            break;
        }
    }
    for(int i=k;i>=0;i--){
        c+=(z[i]+'0');
    }
    return c;
}

int main(){
//  cin>>s;//不在是一个数字而是由多个字符得到   一位代表一个字符
//字符串存储的数字没法进行数字运算
    string a,b,c;
    cin>>a>>b;
    c=plus_s(a,b);
    cout<<c;
//   c=a+b;//字符串拼接
//   cout<<c;
//  if(a>b){//字符串比较采用字典序
//  //字典序：按照最高位的大小决定顺序
//      cout<<a;
//  }else{
//      cout<<b;
//  }
}
//34858346583792873965348765928734926 5
```