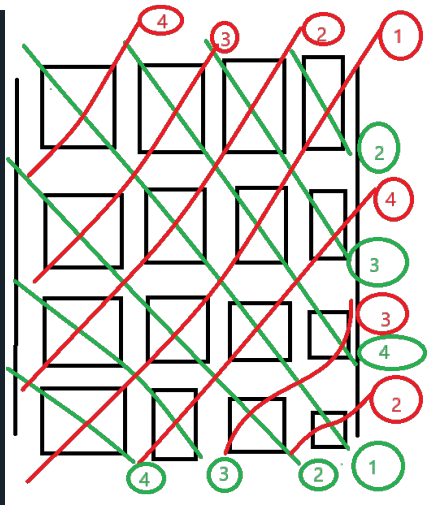
```python
def det(A):#運算行列式
    S=0
    if(len(A)==1):#特例：1X1矩陣
        S=A[0][0]
    elif(len(A)==2):#特例：2X2矩陣
        S=A[0][0]*A[1][1]-A[0][1]*A[1][0]
    else:
        for x in range(0,len(A),1):#偏移迴圈(Shift)
            EP=1#正半週
            ED=1#負半週
            for y in range(0,len(A),1):#主迴圈
                EP*=A[(y+x)%len(A)][y]
                ED*=A[(len(A)-y+x)%len(A)][y]
            S+=EP
            S-=ED
    return S
```

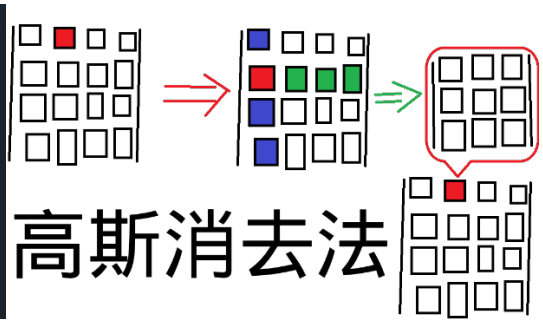


※紅色是正半週，綠色是負半週

```python
def inverse_multiplication(A):
    if (len(A)!=len(A[0])):
        return'error-The inverse matrix does not exist(this column and this row are differant size)'#沒有反矩陣(行與列的大小不同)
    else:
        if(det(A)==0):
            return'error-The inverse matrix does not exist(det==0)'#沒有反矩陣(det==0)
        elif(len(A)==1):
            S=A
        elif(len(A)==2):#特例:size==2
            Base=det(A)
            P=[[A[1][1],-A[0][1]],[-A[1][0],A[0][0]]]
            S=int_multiplication(Base**-1,P)
        else:
            Base=det(A)
            S=list(list(0 for i in range(0,len(A))) for j in range(0,len(A)))
            Q=list(list(0 for i in range(len(A))) for j in range(len(A)))
            for i in range(0,len(A)):
                for j in range(0,len(A)):
                    Q[j][i]=delate_pop_C(delate_pop_R(A,i),j)#回傳值為一個(n-1)X(n-1)的矩陣(其最後一個位置的未知數表述式為Q[n][n][n-1][n-1])
                    Q[j][i]=det(Q[j][i])#將Q[j][i]內部的矩陣用行列式算法換成數值
            S=int_multiplication(Base**-1,Q)
            S=PN(S)
            S=check(S)
        return S
```

```python
def delate_pop_C(A,y):#delate a column
    R=list((i*1) for i in A)
    for j in range(0, len(A)):
        R[j].pop(y)
    return R

def delate_pop_R(A,x):#delate a row
    R=list((i*1) for i in A)
    R.pop(x)
    return R
```



※def delate_pop_C 代表刪掉藍色加上紅色區域

※def delate_pop_R 代表刪掉綠色加上紅色區域