1. 题目:

取 $A=\begin{pmatrix}1 & 2\\ 0 & 5\end{pmatrix}$ ，用 A 加（1）密 meet，再求其逆矩阵，并对其解密。

2. 假设:

在对 meet 进行加密时，我们需将从左至右的英语字母逐个转换为对应的数字，这样才能与矩阵 A 进行后续处理，故假设英语字母按字母表的顺序依次转换为数字 1～26，且后续运算均为 26 进制运算。

3. 符号说明:

A^{-1} :解密矩阵，也是A矩阵的逆矩阵

4. 解答:

对 meet 进行加密:

（1）将 meet 转换成对（2）应 det(A) 数字串, 并划分为两个元素一组表示为向量:

$$\begin{pmatrix}13\\ 5\end{pmatrix}\quad\begin{pmatrix}5\\ 20\end{pmatrix}$$

（2）用矩阵 A 左乘各向量加密（关于 26 取余）得:

$$A\times\begin{pmatrix}13\\ 5\end{pmatrix}=\begin{pmatrix}1 & 2\\ 0 & 5\end{pmatrix}\times\begin{pmatrix}13\\ 5\end{pmatrix}=\begin{pmatrix}23\\ 25\end{pmatrix}$$

$$A\times\begin{pmatrix}5\\ 20\end{pmatrix}=\begin{pmatrix}1 & 2\\ 0 & 5\end{pmatrix}\times\begin{pmatrix}5\\ 20\end{pmatrix}=\begin{pmatrix}19\\ 22\end{pmatrix}$$

得到密文 $wysv$ ，完成了对 $meet$ 的加密。

对 $wysv$ 进行解密:

（1）根据初等行变换求矩阵 A 的逆矩阵 A^{-1}，得:

$$A^{-1}=\begin{pmatrix}1 & 10\\ 0 & 21\end{pmatrix}$$

（2）用 A^{-1} 左乘各密文向量解密（关于 26 取余）得:

$$A^{-1}\times\begin{pmatrix}23\\25\end{pmatrix}=\begin{pmatrix}1&10\\0&21\end{pmatrix}\times\begin{pmatrix}23\\25\end{pmatrix}=\begin{pmatrix}13\\5\end{pmatrix}$$

$$A^{-1}\times\begin{pmatrix}19\\22\end{pmatrix}=\begin{pmatrix}1&10\\0&21\end{pmatrix}\times\begin{pmatrix}19\\22\end{pmatrix}=\begin{pmatrix}5\\20\end{pmatrix}$$

此时明文为 $meet$，完成了对密文的解密。

5. 结论:

（1）用矩阵 $A=\begin{pmatrix}1&2\\0&5\end{pmatrix}$ 对 meet 进行加密，得到的密文为 $wysv$；

（2）用矩阵 $A^{-1}=\begin{pmatrix}1&10\\0&21\end{pmatrix}$ 对 $wysv$ 进行解密，得到单词 meet。

附录（完整代码）:

```
A = [[1,2],[0,5]]  #加密矩阵
#将单词转换为两行的数字向量
def turnToNum(word):
    #初始化向量
    vector = []
    #填充向量
    for i, char in enumerate(word):
        #将字母转换为数字（'a'为1，'b'为2，依此类推）
        num = ord(char) - ord('a') + 1
        if i % 2 == 0:
            vector.append([num])
        else:
            vector[-1].append(num)
    #如果单词长度为奇数，则在第二行添加一个0
    if len(word) % 2 == 1:
        vector[1].append(0)
    return vector
B = turnToNum("meet")

#计算
result1 = [sum(a*b for a, b in zip(A_row, B[0])) % 26 for A_row in A]
result2 = [sum(a*b for a, b in zip(A_row, B[1])) % 26 for A_row in A]
print("加密前的矩阵为:",B[0],B[1])
print("加密后的矩阵为:",result1,result2)

#将数字向量转换为字母
```

```
def turnToWord(vector):
    words = ""
    #转换操作
    for i in range(len(vector)):
        for j in range(len(vector[i])):
            word = chr(vector[i][j] + ord('a') - 1)
            words += word
    return words

#输出结果
encryptedWord = turnToWord([result1, result2])
print("对 meet 加密的结果:",encryptedWord)

#计算 A 的逆矩阵
def mod26_inverse(x):
    for i in range(26):
        if (x * i) % 26 == 1:
            return i
    return None
#计算行列式
det_A = (A[0][0] * A[1][1] - A[0][1] * A[1][0]) % 26
#检查行列式是否可逆
if det_A == 0:
    A_inv = "矩阵不可逆"
else:
    #计算行列式的逆元
    det_A_inv = mod26_inverse(det_A)
    #计算逆矩阵
    A_inv = [
        [(A[1][1] * det_A_inv) % 26, (-A[0][1] * det_A_inv) % 26],
        [(-A[1][0] * det_A_inv) % 26, (A[0][0] * det_A_inv) % 26]
    ]
print("A 矩阵的逆矩阵为:",A_inv)

#对密文解密
C = turnToNum(encryptedWord)
#计算
result3 = [sum(a*b for a, b in zip(A_inv_row, C[0])) % 26 for A_inv_row
in A_inv]
result4 = [sum(a*b for a, b in zip(A_inv_row, C[1])) % 26 for A_inv_row
in A_inv]
print(encryptedWord,"解密后的矩阵为:",result3,result4)
print("解密后的结果为:",turnToWord([result3,result4]))
```

代码部分运行结果展示：

```
加密前的矩阵为: [13, 5] [5, 20]
加密后的矩阵为: [23, 25] [19, 22]
对meet加密的结果: wysv
A矩阵的逆矩阵为: [[1, 10], [0, 21]]
wysv 解密后的矩阵为: [13, 5] [5, 20]
解密后的结果为: meet
```