1. Using the number of vowels to detect ciphertext rectangles
   (In English approximately 40% of plaintext consists of vowels).

總共有 77 個字母，可以拆成 7*11 和 11*7 的 rectangle
然後分別計算兩個 rectangle 的 variance (每一橫行理應有 40%為母音)
如下圖，可以發現 11*7 的 average difference 比較小
可以推出 rectangle 應該為 11*7

```
rectangle 7*11
E R H N E R C R N E C   difference: 1.4
O N H G T A S W Y Y E   difference: 1.4
E P E N O E E K A I I   difference: 3.6
Y S T D C M U D R C E   difference: 2.4
E E Y D R H S R A N T   difference: 1.4
G C H D A T I I N T U   difference: 0.4
T E S D E E A R U T S   difference: 0.6
average difference: 1.6

rectangle 11*7
E E G A E R C   difference: 1.2
O C N E U N N   difference: 0.2
E E D R S Y T   difference: 0.8
Y H D A I A T   difference: 0.2
E H D E A R C   difference: 0.2
G E D M R A E   difference: 0.2
T T E H W N I   difference: 0.8
R Y T T K U E   difference: 0.8
N H O E D E T   difference: 0.2
P S C C R Y U   difference: 1.8
S N R S I I S   difference: 0.8
average difference: 0.65454545454545
```

2. Using plaintext bigrams and trigrams to calculate conditional probabilities for Markov decision processing (MDP).

我將作業提供的 plaintext 用來訓練
分別以 bigrams 和 trigrams 的形式記錄各個次數(也就是兩個一數或三個一數)
將這些次數用來下一步計算條件機率的參考資料(conditional probabilities for MDP)

```
11 def train(msg, trigram, bigram):
12     for i in range(len(msg)-2):
13         if (msg[i:i+3] not in trigram):
14             trigram.update({msg[i:i+3]:1})
15         else:
16             tmp = trigram[msg[i:i+3]]
17             trigram.update({msg[i:i+3]:tmp+1})
18         if (msg[i:i+2] not in bigram):
19             bigram.update({msg[i:i+2]:1})
20         else:
21             tmp = bigram[msg[i:i+2]]
22             bigram.update({msg[i:i+2]:tmp+1})
```

3. Using MDP to recover columnar transposition ciphers.

接著上述得到了 11*7 的 rectangle

根據提示前面 3 個字為 GRE

所以可以先將開頭為 G、R 兩直欄換到第一、二欄

```
107  # training
108  for i in range(col-2):
109      #test column
110      change=i+2
111      Pr=0
112      for k in range(i+2, col):
113          prob=0
114          for j in range(row):
115              word2=ansRect[i][j]+ansRect[i+1][j]
116              word3=word2+ansRect[k][j]
117              if ((word2 not in bigram) or (word3 not in trigram)):
118                  prob += 0
119              else:
120                  prob += math.log(26*(trigram[word3]/bigram[word2]))
121          if(prob>Pr):
122              Pr = prob
123              change = k
124      #swap column
125      for j in range(row):
126          ansRect[i+2][j], ansRect[change][j] = ansRect[change][j], ansRect[i+2][j]
```

如上，接著利用計算條件機率的公式：log 26 * (#trigram / #bigram)

也就是看到 bigram 的兩個字母時，三個連續字母是 trigram 的機率是多少

接著把各欄條件機率加總起來

看哪一直欄的加總條件機率最高，就能得出下一直欄應為哪一欄

往後做到最後一欄可以得出答案，如下

```
G R E E C E A
N N O U N C E
D Y E S T E R
D A Y I T H A
D R E A C H E
D A G R E E M
E N T W I T H
T U R K E Y T
O E N D T H E
C Y P R U S C
R I S I S N S
```