3.1

(a) suppose, $P(X_t=j|X_1=i)=[U^{t-1}]_{ij}$ is true for $t\geq 2$

Then:

$$P(X_{t+1}=j|X_1=i)=\sum_k P(X_{t+1}=j|X_t=k,X_1=i)\cdot P(X_t=k|X_1=i)$$
$$=\sum_k P(X_{t+1}=j|X_t=k)\cdot P(X_t=k|X_1=i)$$
$$=\sum_k U_{kj}\cdot[U^{t-1}]_{ik}$$
$$=\sum_k [U^{t-1}]_{ik}\,U_{kj}=[U^t]_{ij}$$

And when $t=1$

$P(X_2=j|X_1=i)=U_{12}$, ∴ also true

Hence,

$P(X_{t+1}=j|X_1=i)=[U^t]_{ij}$ is true.

(b) from the part (a), we could see:

$$[U^t]_{ij}=\sum_k [U^{t-1}]_{ik}\,U_{kj},$$

which means, in fact, we only need the jth column of U_{kj} to compute. And for $[U^1]$ to $[U^t]$, we need do the multiplication t times.

∴ Time complexity is $O(n^2t)$

Algorithm.

```
nx1 matrix: Col = Uj;  // the jth colum of U
for k=1:t
    col= U·col
end for loop
return col
```

(c). when we see $\log_2 t$, we think of use the matrix U^t, where t's the power of 2.

Hence, we just do:

$$A^t = (A^{\frac{t}{2}})^2 = ((A^{\frac{t}{4}})^2)^2 \cdots$$

And for any matrix $n \times n$ multiplication, the time complexity is $O(n^3)$

$\therefore$ the time complexity in our case is $O(n^3 \log_2 t)$

Algorithm:

```
Assume m is a identity matrix
while (t > 0)
    if t % 2 = 1
        m = m × U
    U = U × U
    t = t / 2;
end while
return m;
```

(d). Recall $[U^t]_{ij} = \sum [U^{t-1}]_{ik} U_{kj}$, again we only need to know jth column of U. In our case, since $m \ll n$, we could open an list to store the non-zero elements.

Hence, we just need muliply the matrix with the list.

$\therefore$ The time complexity is $O(mnt)$

3.2

(a). $P(Y_1|x_1) = \sum_{x_0} P(Y_1, x_0 | x_1)$

$= \sum_{x_0} P(Y_1|x_1, x_0) \cdot P(x_0|x_1)$

$= \sum_{x_0} P(Y_1|x_1, x_0) \cdot P(x_0)$

from CPT, we know $P(Y_1 | x_0, x_1)$, $P(x_0)$

$\therefore$ we get $P(Y_1|x_1)$

(b) $P(Y_1) = \sum_{x_1}\sum_{x_0} P(Y_1, x_0, x_1)$

$= \sum_{x_1}\sum_{x_0} \left(P(Y_1|x_0, x_1) \cdot P(x_0|x_1) \cdot P(x_1) \right)$

$= \sum_{x_1}\sum_{x_0} \left(P(Y_1|x_0, x_1) \cdot P(x_0) \cdot P(x_1) \right)$

$\therefore$ we get $P(Y_1)$

(c). $P(X_n | Y_1, \dots Y_{n-1})$

$= P(X_n)$

(d) $P(Y_n | X_n, Y_1, \dots Y_{n-1}) = \sum_{X_{n-1}} P(Y_n, X_{n-1} | X_n, \dots Y_{n-1})$

$= \sum_{X_{n-1}} \dfrac{P(Y_n, X_n, X_{n-1} | Y_1 \dots Y_{n-1})}{P(X_n | Y_1 \dots Y_{n-1})}$

$= \sum_{X_{n-1}} \dfrac{P(X_{n-1}|Y_1 \dots Y_{n-1}) \cancel{P(X_n|Y_1 \dots Y_{n-1} X_{n-1})} P(Y_n | Y_1 \dots Y_{n-1}, X_n, X_{n-1})}{\cancel{P(X_n|Y_1 \dots Y_{n-1})}}$

(independe from (c))

$= \sum_{X_{n-1}} P(Y_n|Y_1 \dots Y_{n-1}, X_n, X_{n-1}) P(X_{n-1} | Y_1 \dots Y_{n-1})$

$= \sum_{X_{n-1}} P(X_{n-1}|Y_1 \dots Y_{n-1}) P(Y_n | X_n, X_{n-1})$

(e) $P(Y_n \mid Y_1, \ldots Y_{n-1})$

$$= \sum_{Y_{n-1}} P(Y_n, X_{n-1} \mid Y_1, \ldots Y_{n-1})$$

$$= \sum_{X_{n-1}} P(X_{n-1} \mid Y_1 \ldots Y_n) \cdot P(Y_n \mid X_{n-1}, \ldots Y_1, Y_n)$$

$$= \sum_{X_{n-1}} P(X_{n-1} \mid Y_1 \ldots Y_n) P(Y_n \mid X_{n-1})$$

$$= \sum_{X_{n-1}} P(X_{n-1} \mid Y_1 \ldots Y_n) \cdot \sum_{X_n} P(Y_n \mid X_n, X_{n-1}) \cdot P(X_n \mid X_{n-1})$$

$$= \sum_{X_{n-1}} P(X_{n-1} \mid Y_1 \ldots Y_n) \sum_{X_n} P(Y_n \mid X_n, X_{n-1}) \cdot P(X_n)$$

3.3.

(a) $\sum_{z} P(Z=z \mid B_1 \dots B_n) = 1$

$\therefore \sum_{z=-\infty}^{+\infty} \left(\frac{1-\alpha}{1+\alpha}\right) \alpha^{|z-f(B)|}$

$\because z = -\infty$ to $+\infty$

$\therefore z + f(B)$ still from $-\infty$ to $+\infty$

$\therefore \sum_{z=-\infty+f(B)}^{+\infty+f(B)} \left(\frac{1-\alpha}{1+\alpha}\right) \alpha^{|z+f(B)-f(B)|}$

$= \sum_{z=-\infty}^{\infty} \left(\frac{1-\alpha}{1+\alpha}\right) \alpha^{|z|} = \left(\frac{1-\alpha}{1+\alpha}\right) \sum_{z=-\infty}^{\infty} \alpha^{|z|}$

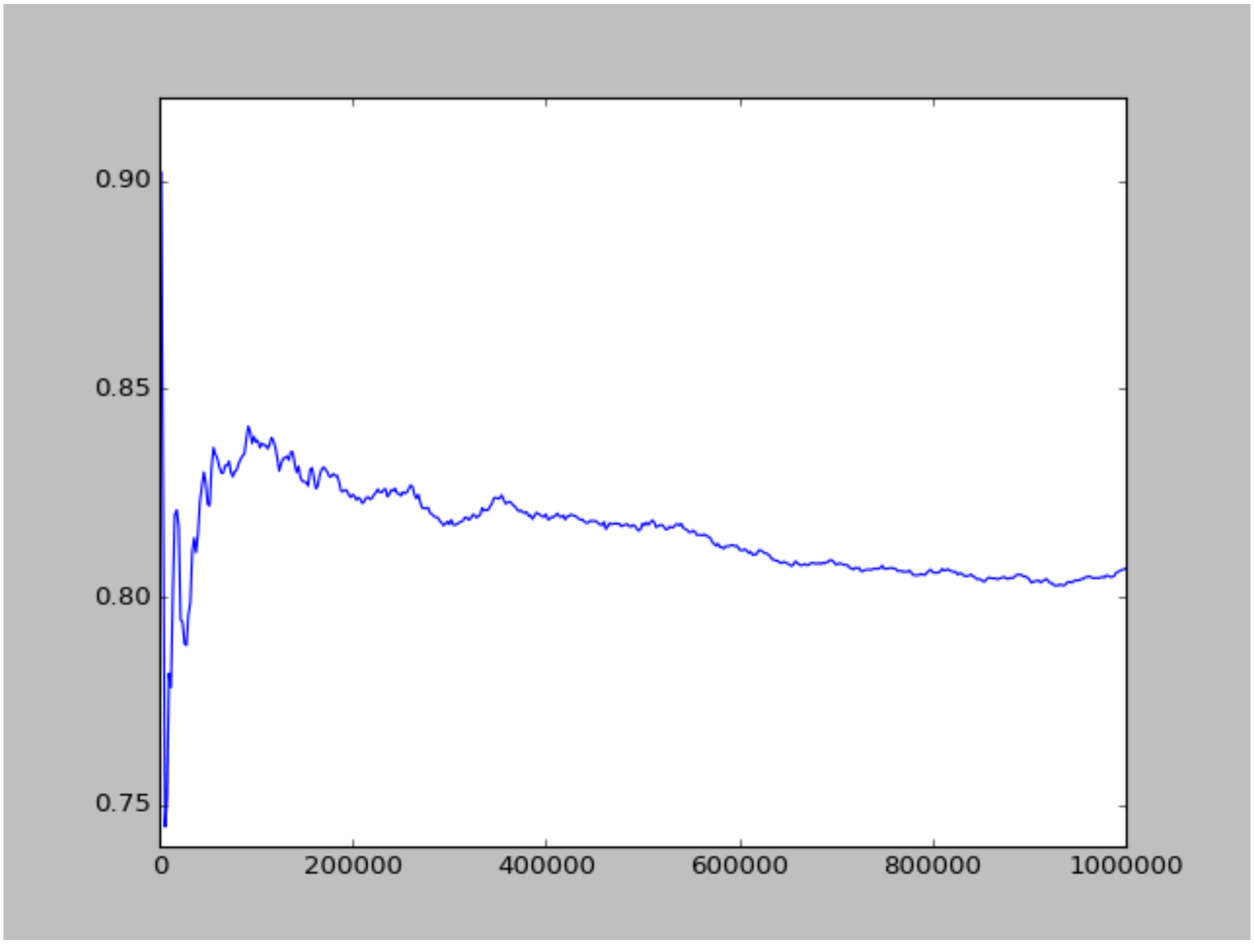
$= \left(\frac{1-\alpha}{1+\alpha}\right) \left(\alpha^0 + 2\sum_{z=1}^{\infty} \alpha^z\right) = \left(\frac{1-\alpha}{1+\alpha}\right)\left(1 + 2\left(\frac{\alpha}{1-\alpha}\right)\right)$

$= \left(\frac{1-\alpha}{1+\alpha}\right)\left(\frac{1+\alpha}{1-\alpha}\right) = 1$

$\therefore$ the conversion is normalized when $z \in [-\infty, +\infty]$.

(b)

3.3
(b) 0.80
(c)



(d)

```
from random import randint
from math import pow
import matplotlib.pyplot as plt

def convert(B):
   f = 0
   base = 1
   for i in range(0, len(B)):
      f += base * B[i]
      base *= 2
   return f

def genRandom(n):
   b=[]
   for i in range(0,10):
      b.append(randint(0,1))
   return b

n = 10
alpha = 0.25
numerator = 0
denominator = 0
Z = 128
p = 1
l =[]
t = []
for i in range(0, 1000000):
   B = genRandom(n)
   f = convert(B)
   pf = (1-alpha)/(1+alpha) * pow(alpha, abs(Z - f))
   denominator += pf
   if B[7] == 1:
      numerator += pf
   if denominator > 0:
      p = numerator / denominator
   if (i+1) % 2000 == 0:
      #print p
      t.append(i)
      l.append(p)

plt.plot(t,l)
plt.show()
```

(a) Show that the conditional distribution for binary to decimal conversion is normalized; namely, that $\sum_z P(Z=z|B_1, B_2, \ldots, B_n) = 1$, where the sum is over all integers $z \in [-\infty, +\infty]$.

0.7

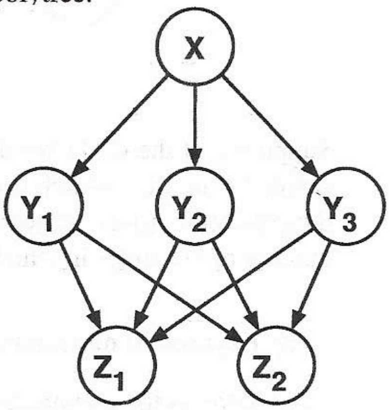
(b) Use the method of *likelihood weighting* to estimate the probability $P(B_8=1|Z=128)$ for a network with $n=10$ bits and noise level $\alpha=0.25$.

(c) Plot your estimate in part (b) *as a function of the number of samples*. You should be confident from the plot that your estimate has converged to a good degree of precision (say, at least two significant digits).

(d) Submit your source code (electronically). You may program in the language of your choice, and you may use any program at your disposal to plot the results.

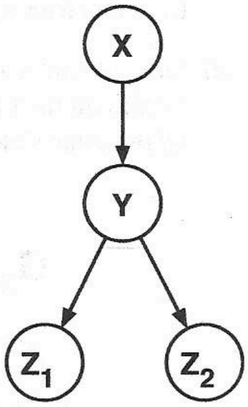
3.4 Node clustering

Consider the belief network shown below over binary variables X, Y_1, Y_2, Y_3, Z_1, and Z_2. The network can be transformed into a polytree by clustering the nodes Y_1, Y_2, and Y_3 into a single node Y. From the CPTs in the original belief network, fill in the missing elements of the CPTs for the polytree.



X	$P(Y_1=1\|X)$	$P(Y_2=1\|X)$	$P(Y_3=1\|X)$
0	0.1	0.3	0.5
1	0.8	0.6	0.4

Y_1	Y_2	Y_3	$P(Z_1=1\|Y_1,Y_2,Y_3)$	$P(Z_2=1\|Y_1,Y_2,Y_3)$
0	0	0	0.1	0.9
1	0	0	0.2	0.8
0	1	0	0.3	0.7
0	0	1	0.4	0.6
1	1	0	0.5	0.5
1	0	1	0.6	0.4
0	1	1	0.7	0.3
1	1	1	0.8	0.2



Y_1	Y_2	Y_3	Y	$P(Y\|X=0)$	$P(Y\|X=1)$	$P(Z_1=1\|Y)$	$P(Z_2=1\|Y)$
0	0	0	1	0.315	0.112	0.1	0.9
1	0	0	2	0.035	0.192	0.2	0.8
0	1	0	3	0.135	0.072	0.3	0.7
0	0	1	4	0.315	0.032	0.4	0.6
1	1	0	5	0.015	0.288	0.5	0.5
1	0	1	6	0.035	0.128	0.6	0.4
0	1	1	7	0.135	0.048	0.7	0.3
1	1	1	8	0.015	0.192	0.8	0.2

3.5

(a). the pa of (X_i) is $X_i - 1$

$\therefore$ for $Pa(X_{t+1}) = X_t$.

According to Markov model in class.

we have

$$P_{ML} = P(X_{t+1} = x' \mid X_t = x)$$

$$= \frac{P(X_{t+1} = x', X_t = x)}{P(X_t = x)} = \frac{COUNT_t(x, x')}{COUNT_t(x)}$$

(b) $P_{ML} = P(X_t = x \mid X_{t+1} = x')$

$$= \frac{COUNT_t(x, x')}{COUNT_{t+1}(x')}$$

(c)

$P(X_1 = x_1, X_2 = x_2 \ldots X_n = x_n)$

$= \prod_{i=1}^{n} P(X_i = x_i \mid Pa(X_i) = a)$

from G_1, where $Pa(X_i) = X_{i-1}$

$\because Pa(X_1) = X_0$, which doesn't exist

$\therefore = P(X_1 = x_1) \prod_{i=2}^{n} P(X_i = x_i \mid X_{i-1} = x_{i-1})$

$= P(X_1 = x_1) \prod_{i=1}^{n-1} P(X_{i+1} = x_{i+1} \mid X_i = x_i)$

① $= \dfrac{COUNT(x_1)}{data} \prod_{i=1}^{n-1} \dfrac{COUNT(x_i, x_{i+1})}{COUNT(x_i)}$

from G_2, where $Pa(X_i) = X_{i+1}$, doesn't exist

$\therefore = P(X_n = x_n) \prod_{i=1}^{n-1} P(X_i = x_i \mid X_{i+1} = x_{i+1})$

② $= \dfrac{count(x_n)}{data} \prod_{i=1}^{n-1} \dfrac{COUNT(x_i, x_{i+1})}{COUNT(x_{i+1})}$

for ①

$= \dfrac{\prod_{i=1}^{n-1} COUNT(x_i, x_{i+1})}{data \prod_{i=2}^{n-1} COUNT(x_i)}$

for ②

$= \dfrac{count(x_n)}{data} \cdot \dfrac{\prod_{i=1}^{n-1} COUNT(x_i, x_{i+1})}{\prod_{i=2}^{n} COUNT(x_i)}$

$= \dfrac{\prod_{i=1}^{n-1} COUNT(x_i, x_{i+1})}{data \cdot \prod_{i=2}^{n-1} COUNT(x_i)}$

$\therefore$ ① = ②

(d) Yes,

e.g. for

$$\begin{tikzcd} & x_2 \arrow[dl] \arrow[dr] & \\ x_1 & & x_3 \end{tikzcd}$$

$P_{ML}(x_i \mid Pa)$

$\therefore P(x_1, x_2, x_3) = P(x_1 \mid x_2) \cdot P(x_2) \cdot P(x_3 \mid x_2)$

which share the similar pattern as G_1/G_2.

(a)

```
for word BY,      the probality is 0.004180
for word BE,      the probality is 0.003370
for word BUT,     the probality is 0.002994
for word BEEN,    the probality is 0.001658
for word BECAUSE, the probality is 0.000902
for word BILLION, the probality is 0.000822
for word B.,      the probality is 0.000684
for word BEFORE,  the probality is 0.000666
for word BUSINESS,the probality is 0.000541
for word BUSH,    the probality is 0.000516
for word BANK,    the probality is 0.000464
for word BETWEEN, the probality is 0.000458
for word BEING,   the probality is 0.000453
for word BACK,    the probality is 0.000448
for word BASED,   the probality is 0.000424
for word BOTH,    the probality is 0.000400
for word BIG,     the probality is 0.000340
for word BOARD,   the probality is 0.000334
for word BEGAN,   the probality is 0.000276
for word BILL,    the probality is 0.000267
for word BLACK,   the probality is 0.000239
for word BONDS,   the probality is 0.000213
for word BEST,    the probality is 0.000212
for word BETTER,  the probality is 0.000212
for word BUY,     the probality is 0.000209
for word BUDGET,  the probality is 0.000205
for word BANKS,   the probality is 0.000201
```

(b)

```
The word HUNDRED  has probality 0.209061
The word <UNK>    has probality 0.124304
The word .POINT   has probality 0.099952
The word OF       has probality 0.073947
The word THOUSAND has probality 0.068654
The word MILLION  has probality 0.031832
The word ,COMMA   has probality 0.031622
The word -HYPHEN  has probality 0.030479
The word HALF     has probality 0.029139
The word .PERIOD  has probality 0.024376
```

(c)

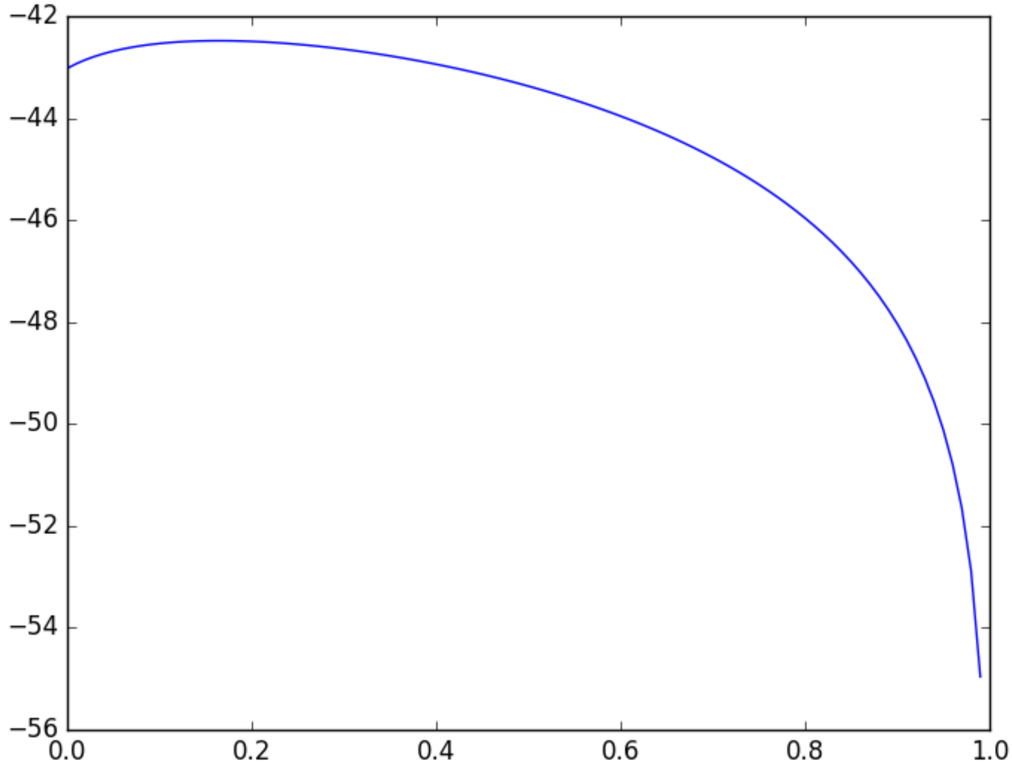
```
Lu = -64.509
Lb = -40.918
The Lb is higher than Lu
```

(d)

```
Lu = -44.2364299857
Lb = undefined,
no FOURTEEN followed by OFFICIALS
```

So the missing adjacent pair will cause the undefined problem in Lb

(e)



The highest should around 0.2

Codefrom math import log
import matplotlib.pyplot as plt

```
def lu(judge):
	token = judge.upper().strip('\n').split(' ')
	p = 1.0
	for i in range(0,len(token)):
		p *= cal1(token[i])
	return log(p*1.0)

def cal1(val1):
	return tokenDict[val1]*1.0/ totalCount

def lb(judge):
	token = judge.upper().strip('\n').split(' ')
```

```
    pa = token[0]
    p = cal2('<s>',pa)
    for i in range(1, len(token)):
        if not token[i] in orderDict[token[i-1]]:
            print 'no %s followed by %s' % (token[i-1], token[i])
            break
        else:
            p *= cal2(token[i-1],token[i])
    return log(p*1.0)

def cal2(val1, val2):
    return orderDict[val1][val2]/tokenDict[val1];

def lm(judge, r):
    token = judge.upper().strip('\n').split(' ')
    pa = token[0]
    p = cal2('<s>',pa)*(1-r)
    for i in range(1,len(token)):
        if not token[i] in orderDict[token[i-1]]:
            p *= (1-r)*cal1(token[i])
        else:
            p *= (1-r)*cal1(token[i]) + r*cal2(token[i-1],token[i])
    return log(p*1.0)



voc = open('vocab.txt','r')
tokenDict = {}
tokenList = []
for token in voc.readlines():
    token = token.strip('\n');
    tokenList.append(token)
    tokenDict[token] = 0

uni = open('unigram.txt','r')
totalCount = 0
index = 0
for count in uni.readlines():
    tokenDict[tokenList[index]] = int(count)
    totalCount += int(count)
    index += 1

# (a)
for token in tokenList:
```

```
        if token[0] == 'B':
                print 'for word %s, the probality is %f' % (token, tokenDict[token]*1.0 /
totalCount)


bi = open('bigram.txt', 'r')
orderDict = {}
for line in bi.readlines():
        line = line.split('\t')
        i1 = int(line[0]) - 1
        i2 = int(line[1]) - 1
        count2 = float(line[2])
        if not tokenList[i1] in orderDict.keys():
                orderDict[tokenList[i1]] = {}
        orderDict[tokenList[i1]].update({tokenList[i2]:count2})

#(b)
b = sorted(orderDict['ONE'].items(), key = lambda x: x[1], reverse = True)
for i in range(0,10):
  print ('The word %s has probality %f') % (b[i][0], b[i][1]*1.0/tokenDict['ONE'])

#(c)
print lu('The stock market fell by one hundred points last week')
print lb('The stock market fell by one hundred points last week')
#(d)
print lu('The fourteen officials sold fire insurance')
print lb('The fourteen officials sold fire insurance')

#(e)
pp = []
rr = []
for r in range(0,100,10):
        pp.append(lm('The fourteen officials sold fire insurance',r*1.0/100))
        rr.append(r*1.0/100)

print pp
print rr

plt.plot(rr,pp)
plt.show()
```