3.1

(a) Let $P(X_t = k \mid X_1 = i) = [A^{t-1}]_{ik}$, $X_t \in \{0, 1, \cdots k\}$ (previous term of X_{t+1})

Then $P(X_{t+1} = j \mid X_1 = i) = \sum_k P(X_{t+1} = j, X_t \mid X_1 = i)$

$= \sum_k P(X_{t+1} = j \mid X_t, X_1 = i) \cdot P(X_t \mid X_1 = i)$

$= \sum_k P(X_{t+1} = j \mid X_t) \cdot P(X_t \mid X_1 = i)$

$= \sum_k A_{kj} \cdot [A^{t-1}]_{ik}$

$= [A^t]_{ij}$

Therefore through induction, $P(X_{t+1} = j \mid X_1 = i) = [A^t]_{ij}$

(b) A matrix-vector multiplication takes $O(m^2)$ given the matrix is at most $(m \times m)$ and vector is at most size of m. From Qa, we can see the inference is done by square a matrix t times to get to $P(X_{t+1} = j \mid X_1 = i)$. But to get $[A^t]_{ij}$, instead of times A into A^{t-1} every times, we can times a vector A_i into A for t times. Like $\sum_k A_{kj} \cdot [A^{t-1}]_{ik}$ in a. The algorithm will work as:

$m=2$, $A=\begin{bmatrix} a & b \\ c & d \end{bmatrix}$

$i=1$, $j=1$ $\quad A_i = [a, b]$

$P(X_3=1 \mid X_1=1)$

$T=3$: $[a, b] \times \begin{bmatrix} a & b \\ c & d \end{bmatrix} = [a', b']$

$P(X_4=1 \mid X_1=1)$

$T=3$ $[a', b'] \times \begin{bmatrix} a & b \\ c & d \end{bmatrix} = [a'', b'']$

$T=t+1$

⋮

from above algorithm, the run time to get

$P(X_{t+1}=i \mid X_1=j) = O(m^2 t)$

Over T times

vector x matrix complexity

(c) From a, we know that $P(X_{t+1}=i \mid X_1=j)$ can be get from multiple the same matrix A t times.

We know that matrix times matrix operation is $O(m^3)$ give it is a $m \times m$ matrix. If we do the multiplication in the following way:

$O(m^3)$ $\quad O(m^3)$ $\quad O(m^3)$

$A \to A \times A\,(A^2) \to A^2 \times A^2\,(A^4) \to A^4 \times A^4\,(A^8) \cdots$

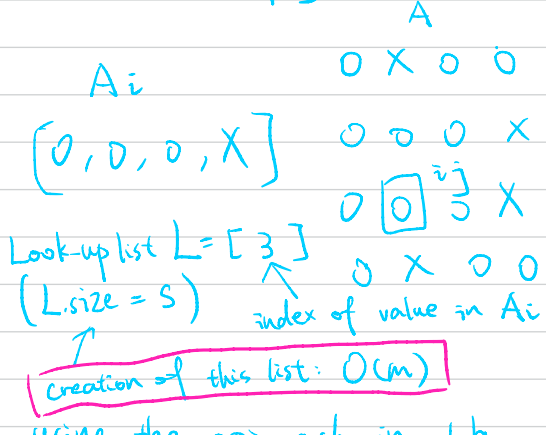
log t times of multiplication

To get to $[A^t]_{ij}$, only log t # of matrix multiplication needed.

Therefore the runtime is $O(m^3 \log t)$

(d) A sparse matrix A with at most s items per row ($s \ll m$), we can trasform that matrix A from

[0 denotes empty, X denotes value]



psudo code: O(sm)

```
for j in m:
  res = 0
  for l in L:
    res += Ai[l] * A[l][j]
  Aij = res
```

vector matrix

using the approach in 1.b, using $A_i \cdot A$ t times get the final result of $P(X_{t+1} = j \mid X_1 = i)$, but to find the non-zero value in Ai, we use a look-up list that is the size of s. It will track the index of non-zero item in Ai from left to right.

Then, there is at most s multiplications done to compute the value of Aij [rest fill with zero] from $\sum_k A_{kj} \cdot [A^{t-1}]_{ik}$

Do the operation t times, the runtime will be $O(m) + O(smt) = O(smt)$

$O(m)$: Look-up list creation

(e) $P(X_1 = i \mid X_{T+1} = j) = \dfrac{P(X_{T+1} = j \mid X_1 = i) \cdot P(X_1 = i)}{P(X_{T+1} = j)}$

$P(X_{T+1} = j) = \sum_k P(X_1, X_{T+1} = j)$

$= \sum_k P(X_{T+1} = j \mid X_1) \times P(X_1)$

Therefore: $P(X_1 = i \mid X_{T+1} = j) = \dfrac{P(X_{T+1} = j \mid X_1 = i) \cdot P(X_1 = i)}{\sum_k P(X_{T+1} = j \mid X_1) \cdot P(X_1)}$

$= \dfrac{[A^T]^{ij} \cdot P(X_1 = i)}{\sum_k [A^T]^{kj} \cdot P(X_1 = k)}$

3.2

(a) $P(Y_1 \mid X_1) = \sum_{X_0} P(X_0, Y_1 \mid X_1)$

$$= \frac{\sum_{X_0} P(Y_1 \mid X_0, X_1) \cdot P(X_0, X_1)}{P(X_1)}$$

$$= \frac{\sum_{X_0} P(Y \mid X_0, X_1) \cdot P(X_0) \cdot \cancel{P(X_1)}}{\cancel{P(X_1)}} = \sum_{X_0} P(Y \mid X_0 X_1) \cdot P(X_0)$$

(b) $P(Y_1) = \sum_{X_0} \sum_{X_1} P(X_0, X_1, Y_1)$

$$= \sum_{X_0} \sum_{X_1} P(Y_1 \mid X_0, X_1) \cdot P(X_0) \cdot P(X_1)$$

(c) $P(X_n \mid Y_1, Y_2 \cdots Y_{n-1})$, Y_n not in evidence set, therefore X_n, X_{n-1} independent, X_n and $Y_1, Y_2 \cdots Y_{n-1}$ independent, $P(X_n \mid Y_1, Y_2 \cdots Y_{n-1}) = P(X_n)$

(d) $P(Y_n \mid X_n, Y_1 \cdots Y_{n-1})$

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

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

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

(e) $P(Y_n \mid Y_1, Y_2 \cdots Y_{n-1})$

$= \sum_{X_{n-1}} \sum_{X_n} P(Y_n, X_{n-1}, X_n \mid Y_1, Y_2 \cdots Y_{n-1})$

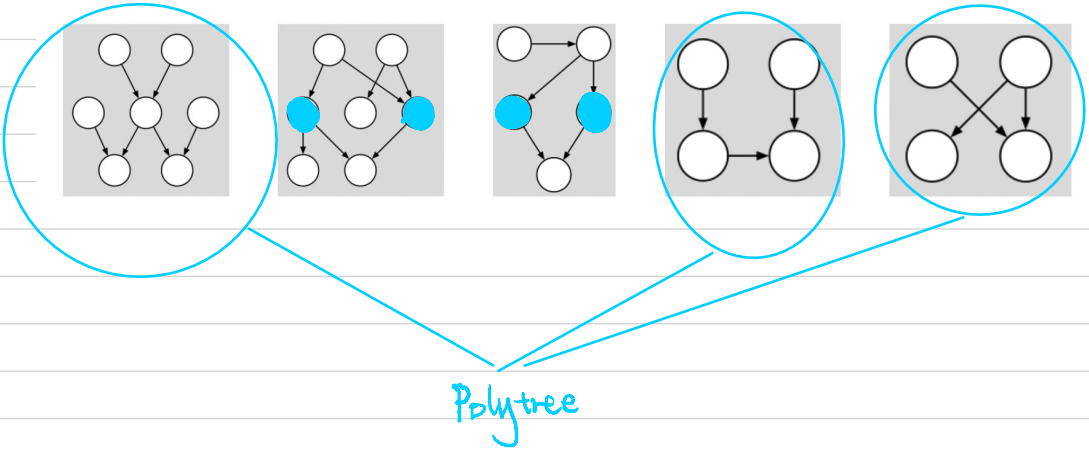
$= \sum_{X_{n-1}} \sum_{X_n} P(Y_n \mid Y_1 \cdots Y_{n-1}, X_{n-1}, X_n) P(X_{n-1}, X_n \mid Y_1, Y_2 \cdots Y_{n-1})$

$= \sum_{X_{n-1}} \sum_{X_n} P(Y_n \mid X_{n-1}, X_n) P(X_{n-1} \mid Y_1, Y_2 \cdots Y_{n-1}) P(X_n \mid Y_1, Y_2 \cdots Y_{n-1})$

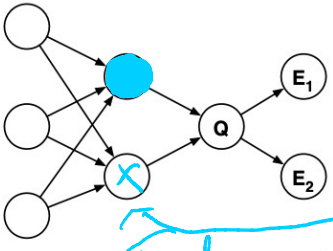
$= \sum_{X_{n-1}} \sum_{X_n} P(Y_n \mid X_{n-1}, X_n) P(X_{n-1} \mid Y_1, Y_2 \cdots Y_{n-1}) P(X_n)$

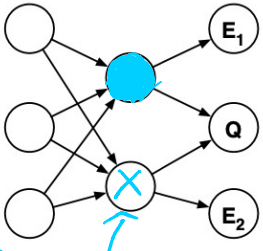
3.3

In the figure below, *circle* the DAGs that are polytrees. In the other DAGs, shade **two** nodes that could be *clustered* so that the resulting DAG is a polytree.

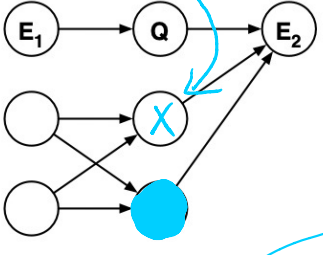


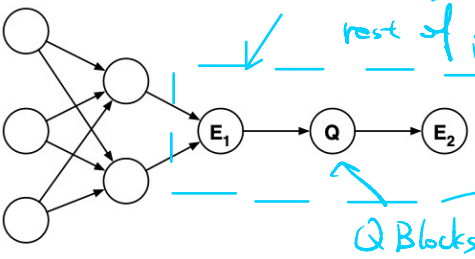
$P(Q, X \mid E_1, E_2)$





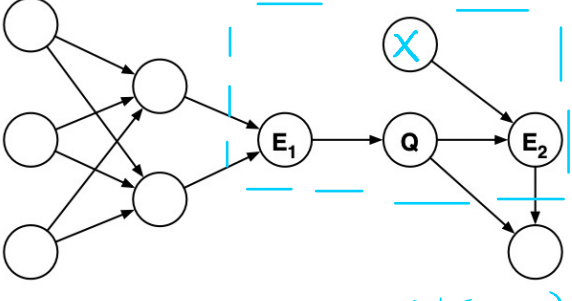
need to introduce new node to marginalization





E_1 Blocks rest of graph

Q Blocks E_1, E_2



$$P(Q \mid E_1, E_2) = \frac{P(E_2 \mid Q) \cdot P(Q \mid E_1)}{P(E_2)}$$

$$P(Q \mid E_1, E_2) = \frac{P(E_2 \mid Q) \cdot P(Q \mid E_1)}{\sum_x P(E_2 \mid x) P(x)}$$

3.5

(a) $\sum_z P(Z=z \mid B_1, B_2, B_3 \cdots, B_n)$

$= \sum_z \left(\frac{1-\alpha}{1+\alpha}\right) \alpha^{|z-f(B)|} \quad z \in \{-\infty, +\infty\}$

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

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

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

Take these two terms:

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

$= \frac{\alpha^{-f(B)} - \alpha^{\infty+1}}{1-\alpha}$

Through the same process, we have:

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

→ continue next page

Back to the original formula, we have:

$$\sum_z P(Z=z \mid B_1, B_2, B_3 \cdots, B_n) = \left(\frac{1-\alpha}{1+\alpha}\right)\left(\alpha^{f(B)}\frac{\alpha^{-f(B)}-\alpha^{\infty+1}}{1-\alpha} + \alpha^{-f(B)}\frac{\alpha^{f(B)+1}-\alpha^{\infty+1}}{1-\alpha}\right)$$

give $0 < \alpha < 1$, $\alpha^{\infty+1} \simeq 0$

$$\sum_z P(Z=z \mid B_1, B_2, B_3 \cdots, B_n) = \frac{1-\alpha}{1+\alpha}\left(\frac{1}{1-\alpha} + \frac{\alpha}{1-\alpha}\right) = 1$$

(b)

$P(B_2=1 \mid Z=128) = 0.098472\,18635$

$P(B_5=1 \mid Z=128) = 0.09061998$

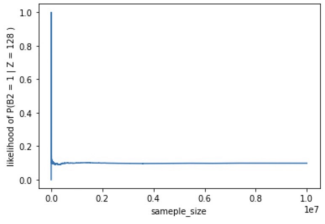
$P(B_8=1 \mid Z=128) = 0.908653954$

$P(B_{10}=1 \mid Z=128) = 0.0$

(C)

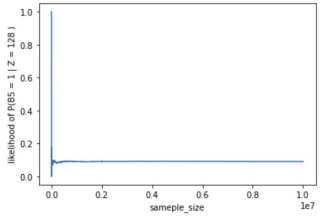
$P(B_2=1 \mid Z=128)$

[0.09847218633146385, 0.09847218633146385, 0.09847218633146385, 0.09847218633146385, 0.09847218633146385, 0.09847218633904911, 0.09847218633904911, 0.09847218633904911, 0.09847218633904911]



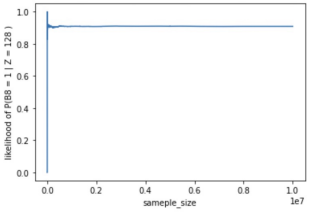
$P(B_5=1 \mid Z=128)$

[0.09061998647019387, 0.09061998647019387, 0.09061998647019387, 0.09061998647019387, 0.09061998723751934, 0.09061998723751934, 0.09061998723751934, 0.09061998723751934, 0.09061998723751934]



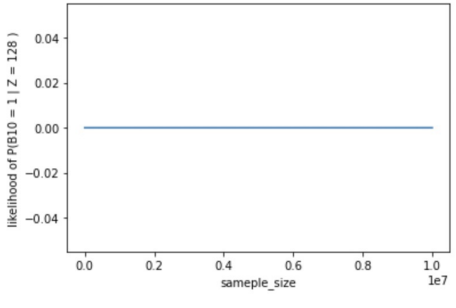
$P(B_8=1 \mid Z=128)$

[0.9086539549018728, 0.9086539549018728, 0.9086539549018728, 0.9086539549018728, 0.9086539549018728, 0.90865395490187 28, 0.9086539549018728, 0.9086539549018728, 0.9086539549018728]



$P(B_{10}=1 \mid Z=128)$

[0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0]



(d)

In [226]:
```
import random
def Bi():
    a = random.random()
    return 1 if a >= 0.5 else 0
```

In [227]:
```
def B(n):
    return [Bi() for i in range(n)]
```

In [228]:
```
def fB(B,n):
    return sum([2**(i-1)*B[i-1] for i in range(1,n+1)])
```

In [229]:
```
import numpy as np
def PZ_B(n, alpha, Z, B): # a*b^n = e^(log(a)+ log(b)*n)
    constant = np.log((1-alpha)/(1+alpha))
    res = constant
    res += np.log(alpha)*abs(Z-fB(B,n))
    return np.exp(res)
```

In [230]:
```
def PZ_B_old(n, alpha, Z, B):
    return ((1-alpha)/(1+alpha)) * (alpha **abs(Z-fB(B,n)))
```

In [231]:
```
b = B(10)
print(PZ_B(10, 0.1, 128, b), PZ_B_old(10, 0.1, 128, b))
```

```
0.0 0.0
```

```
def likelihood_weighting(N, n, alpha, Z, target_bi):
    numerator = 0.0
    denominator = 0.0
    estimates = []

    for sample in range(N):
        b_lst = B(n)
        pz_b = PZ_B(n, alpha, Z, b_lst)

        numerator += b_lst[target_bi] * pz_b # I(bi, bi')* P(Z|B1, B2, B3...B10)
        denominator += pz_b # P(Z|B1, B2, B3...B10)

        if denominator == 0: # due to pz_b too small
            estimates.append(0)
        else:
            estimates.append(numerator/denominator)
    return estimates
```

```
import matplotlib.pyplot as plt

target_bis = [1,4,7,9] # index of [2, 5, 8, 10]
sample_size = 10000000 # 10^7
Bi_size = 10  #n
noise_level = 0.1 # alpha
Z = 128

for target_bi in target_bis:
    result = likelihood_weighting(sample_size, Bi_size, noise_level, Z, target_bi)
    print(result[-10:-1])
    plt.plot(result)
    plt.xlabel('sameple_size')
    plt.ylabel(f'likelihood of P(B{target_bi+1} = 1 | Z = 128 )')
    plt.show()
```

3.6

(a) $P(B|A,C,D)$

$$= \frac{P(D|A,B,C) \cdot P(B|A,C)}{P(D|A,C)}$$

$$= \frac{P(D|B,C) \cdot P(B|A)}{\sum_B P(D|B,C) P(B|A)}$$

(b) $P(B|A,C,D,E,F)$

$$= P(B|A,C,D)$$

$$= \frac{P(D|B,C) \cdot P(B|A)}{\sum_B P(D|B,C) P(B|A)}$$

(c) $P(B,E,F|A,C,D)$

$$= P(B|A,C,D) \cdot P(E|A,C,D) \cdot P(F|A,C,D)$$

$$= P(B|A,C,D) \cdot P(E|C) \cdot P(F|A)$$

$$= \frac{P(D|B,C) \cdot P(B|A)}{\sum_B P(D|B,C) P(B|A)} \cdot P(E|C) \cdot P(F|A)$$

3.7

(a) $P(Q=q \mid E, e)$

$$= \frac{\sum_{i}^{I} I(q, q_i') P(E=e \mid Y=y_i, Z=z_i)}{\sum_{i}^{I} P(E=e \mid Y=y_i, Z=z_i)}$$

(b) $P(Q_1=q_1, Q_2=q_2 \mid E_1=e_1, E_2=e_2)$

$$= \frac{\sum_{i}^{I} I(q_1, q_i') P(E_1=e_1 \mid X=x_i, Q=q_1) I(q_2, q_i') P(E_2=e_2 \mid E_1=e_1, Z=z_i)}{\sum_{i}^{I} P(E_1=e_1 \mid X=x_i, Q=q_i) P(E_2=e_2 \mid E_1=e_1, Z=z_i)}$$