14.18 Consider the query P(Rain | Sprinkler =true, WetGrass =true) in Figure 14.12(a) (page 529) and how Gibbs sampling can answer it.

a. How many states does the Markov chain have?

b. Calculate the transition matrix Q containing q(y → y') for all y, y'.

c. What does Q^2 , the square of the transition matrix, represent?

d. What about Q^n as $n \to \infty$?

e. Explain how to do probabilistic inference in Bayesian networks, assuming that Q^n is available. Is this a practical way to do inference?

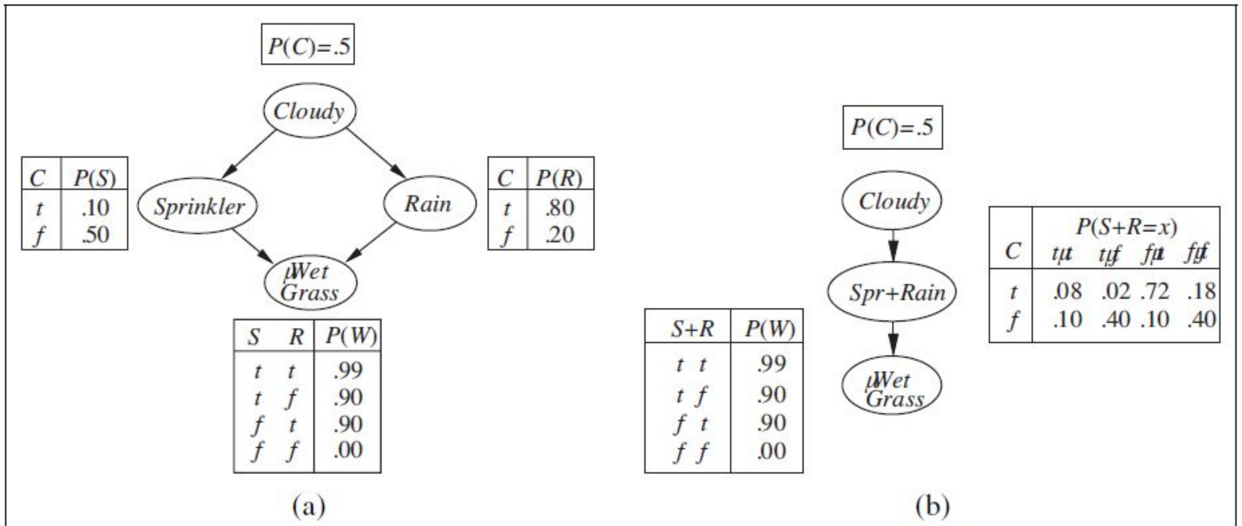

Figure 14.12 (a) A multiply connected network with conditional probability tables. (b) A clustered equivalent of the multiply connected network.

a. How many states does the Markov chain have?

function GIBBS-ASK(X, **e**, bn, N) **returns** an estimate of $\mathbf{P}(X|\mathbf{e})$

  local variables: **N**, a vector of counts for each value of X, initially zero

    Z, the nonevidence variables in bn

    x, the current state of the network, initially copied from **e**

  initialize **x** with random values for the variables in **Z**

  for $j = 1$ to N **do**

    for each Z_i in **Z do**

      set the value of Z_i in **x** by sampling from $\mathbf{P}(Z_i|mb(Z_i))$

      $\mathbf{N}[x] \leftarrow \mathbf{N}[x] + 1$ where x is the value of X in **x**

  return NORMALIZE(**N**)

Figure 14.16 The Gibbs sampling algorithm for approximate inference in Bayesian networks; this version cycles through the variables, but choosing variables at random also works.

Let see what happens when we query GIBBS-ASK to return the P(Rain | Sprinkler =true, WetGrass =true) distribution:

<table>
<tr><td>X = Rain</td><td></td></tr>
<tr><td>e = (Sprinkler=true, WetGrass=true)</td><td>evidence</td></tr>
<tr><td>bn: the Bayesian network</td><td></td></tr>
<tr><td>N = <0, 0></td><td><0 times Rain=True; 0 times Rain=False></td></tr>
<tr><td>Z = Cloudy, Rain</td><td>Nonevidence variables in bn</td></tr>
<tr><td>x = <C, S=True, W=True, R></td><td>current state of the network (filled with e)</td></tr>
<tr><td>x = <C=True, S=True, W=True, R=True></td><td>current state of the network (random for Zi)</td></tr>
<tr><td>j=1</td><td></td></tr>
<tr><td>x = <C=sampling_from(P(C|mb(C)), S=True, W=True, R=True></td><td>Suppose we obtain C=False</td></tr>
<tr><td>N[True] += 1</td><td>R is True in the current state x</td></tr>
<tr><td>x = <C=False, S=True, W=True, R=sampling_from(P(R|mb(R)))></td><td>Suppose we obtain R=True</td></tr>
<tr><td>N[True] += 1</td><td>R is True in x (comes from previous sample)</td></tr>
<tr><td>j=2</td><td></td></tr>
<tr><td>x = <C=sampling_from(P(C|mb(C))), S=True, W=True, R=True></td><td>Suppose we obtain C=True</td></tr>
<tr><td>N[True] += 1</td><td>R is True in the current state x</td></tr>
<tr><td>x = <C=True, S=True, W=True, R=sampling_from(P(R|mb(R)))></td><td></td></tr>
<tr><td>N[False] += 1</td><td>R is False in x (comes from previous sample)</td></tr>
<tr><td></td><td>N = <3, 1> after the loops</td></tr>
<tr><td>Return Normalized(N)</td><td><0.75; 0.25></td></tr>
</table>

Variables in evidence are fixed. Then the only variables that can present different values are Rain and Cloudy.
If the state representations is of the form <Cloudy; Sprinkler; Rain; WetGrass>, then fixing the values of Sprinkler and WetGrass we obtain the next 4 possible states:

- S1: <**True**, True, **True**, True> = (cloudy, rain)
- S2: <**False**, True, **True**, True> = (-cloudy, rain)
- S3: <**True**, True, **False**, True> = (cloudy, -rain)
- S4: <**False**, True, **False**, True> = (-cloudy, -rain)

So, we can say that the Markov Chain presents 4 possible states.

b. Calculate the transition matrix Q containing q(y → y') for all y, y'.

The four possible samples are taken from four possible distributions (values s and w do not change because they are evidence):

$\mathbf{P}(C \mid s, r)$, $\mathbf{P}(C \mid s, -r)$, $\mathbf{P}(R \mid c, s, w)$ and $\mathbf{P}(R \mid -c, s, w)$

Let calculate them one by one

$\mathbf{P}(C \mid s, r) = a * \mathbf{P}(C, s, r)$
$\mathbf{P}(C \mid s, r) = a * \mathbf{P}(C) * \mathbf{P}(s|parents(s)) * \mathbf{P}(r|parents(r))$
$\mathbf{P}(C \mid s, r) = a * \mathbf{P}(C) * \mathbf{P}(s|C) * \mathbf{P}(r|C)$
$\mathbf{P}(C \mid s, r) = a * \langle 0.50; 0.50\rangle * \langle 0.10; 0.50\rangle * \langle 0.80; 0.20\rangle$
$\mathbf{P}(C \mid s, r) = a * \langle 0.04; 0.05\rangle = \langle 0.04/0.09; 0.05/0.09\rangle$
$\mathbf{P}(C \mid s, r) = \boldsymbol{\langle 4/9; 5/9\rangle}$

P(C | s, -r) = a * **P**(C, s, -r)
P(C | s, -r) = a * **P**(C) * **P**(s|parents(s)) * **P**(-r|parents(r))
P(C | s, -r) = a * **P**(C) * **P**(s|C) * **P**(-r|C)
P(C | s, -r) = a * <0.50; 0.50> * <0.10; 0.50> * <0.20; 0.80>
P(C | s, -r) = a * <0.01; 0.20> = <0.01/0.21; 0.20/0.21>
P(C | s, -r) = **<1/21; 20/21>**

P(R | c, s, w) = a * **P**(R, c, s, w)
P(R | c, s, w) = a * **P**(R|parents(R)) * **P**(c|parents(c)) * **P**(s|parents(s)) * **P**(w|parents(w))
P(R | c, s, w) = a * **P**(R|c) * **P**(c) * **P**(s|c) * **P**(w|s,R)
P(R | c, s, w) = a * **P**(R|c) * <1> * <1> * **P**(w|s,R)
P(R | c, s, w) = a * **P**(R|c) * **P**(w|s,R)
P(R | c, s, w) = a * <0.80; 0.20> * <0.99; 0.90>
P(R | c, s, w) = a * <0.792; 0.180> = <0.792/0.972; 0.180/0.972>
P(R | c, s, w) = **<22/27; 5/27>**

P(R | -c, s, w) = a * **P**(R, -c, s, w)
P(R | -c, s, w) = a * **P**(R|parents(R)) * **P**(-c|parents(c)) * **P**(s|parents(s)) * **P**(w|parents(w))
P(R | -c, s, w) = a * **P**(R|-c) * **P**(-c) * **P**(s|-c) * **P**(w|s,R)
P(R | -c, s, w) = a * **P**(R|-c) * <1> * <1> * **P**(w|s,R)
P(R | -c, s, w) = a * **P**(R|-c) * **P**(w|s,R)
P(R | -c, s, w) = a * <0.20; 0.80> * <0.99; 0.90>
P(R | -c, s, w) = a * <0.198; 0.720> = <0.198/0.918; 0.720/0.918>
P(R | -c, s, w) = **<11/51; 40/51>**

Now we calculate the transitions probabilities.

S1: <Cloudy=true; Sprinkler=true; Rain=true; WetGrass=true> = (c, r)

There are just two moments per loop in the algorithm where Cloudy or Rain can change (make a transition). These moments happen when we take the sample from the distributions P(Cloudy | mb(Cloudy)) and P(Rain | mb(Rain)) respectively. Both samples are equally likely to be called (then we divide in two).

q(S1 -> S1) = q((c, r) -> (c, r)) = (P(c | s,r) + P(r | c,s,w)) / 2
q(S1 -> S1) = (4/9 + 22/27) / 2 = 34/(27*2)
q(S1 -> S1) = 17/27

q(S1 -> S2) = q((c, r) -> (-c, r)) = P(-c | s,r) / 2
q(S1 -> S2) = 5/9 / 2
q(S1 -> S2) = 5/18

q(S1 -> S3) = q((c, r) -> (c, -r)) = P(-r | c,s,w) / 2
q(S1 -> S3) = 5/27 / 2
q(S1 -> S3) = 5/54

q(S1 -> S4) = q((c, r) -> (-c, -r)) = there are no possibilities
q(S1 -> S4) = 0

S2: <Cloudy=false; Sprinkler=true; Rain=true; WetGrass=true> = (-c, r)

q(S2 -> S1) = q((-c, r) -> (c, r)) = P(c | s,r) / 2
q(S2 -> S1) = (4/9) / 2
q(S2 -> S1) = 2/9

q(S2 -> S2) = q((-c, r) -> (-c, r)) = (P(-c | s,r) + P(r | -c, s, w)) / 2
q(S2 -> S2) = (5/9 + 11/51) / 2 = (255 + 99) / (9*51*2) = 354/(2*9*51) = 59/3*51
q(S2 -> S2) = 59/153

q(S2 -> S3) = q((-c, r) -> (c, -r)) = two changes in one step. Impossible.
q(S2 -> S3) = 0

q(S2 -> S4) = q((-c, r) -> (-c, -r)) = P(-r | -c, s, w) / 2
q(S2 -> S4) = (40/51) / 2
q(S2 -> S4) = 20/51

S3: <Cloudy=true; Sprinkler=true; Rain=false; WetGrass=true> = (c, -r)

q(S3 -> S1) = q((c, -r) -> (c, r)) = P(r | c,s,w) / 2
q(S3 -> S1) = 22/27 / 2
q(S3 -> S1) = 11/27

q(S3 -> S2) = q((c, -r) -> (-c, r)) = two changes in one step. Impossible.
q(S3 -> S2) = 0

q(S3 -> S3) = q((c, -r) -> (c, -r)) = (P(c | s,-r) + P(-r | c, s, w)) / 2
q(S3 -> S3) = (1/21 + 5/27) / 2 = (27 + 105) / (2*21*27) = 22 / (7*27)
q(S3 -> S3) = 22/189

q(S3 -> S4) = q((c, -r) -> (-c, -r)) = P(-c | s, -r) / 2
q(S3 -> S4) = 20/21 / 2
q(S3 -> S4) = 10/21

S4: <Cloudy=false; Sprinkler=true; Rain=false; WetGrass=true> = (-c, -r)

q(S4 -> S1) = q((-c, -r) -> (c, r)) = two changes in one step. Impossible.
q(S4 -> S1) = 0

q(S4 -> S2) = q((-c, -r) -> (-c, r)) = P(r | -c, s, w) / 2
q(S4 -> S2) = 11/51 / 2
q(S4 -> S2) = 11/102

q(S4 -> S3) = q((-c, -r) -> (c, -r)) = P(c | s, -r) / 2
q(S4 -> S3) = 1/21 /2
q(S4 -> S3) = 1/42

q(S4 -> S4) = q((-c, -r) -> (-c, -r)) = (P(-c | s,-r) + P(-r | -c, s, w)) / 2

q(S4 -> S4) = (20/21 + 40/51) / 2 = (1020 + 840) / (2*21*51) = 930 / (21*51)
q(S4 -> S4) = 310/357

So, the Transition Matrix Q is:

	(cloudy, rain)	(-cloudy, rain)	(cloudy, -rain)	(-cloudy, -rain)
(cloudy, rain)	17/27	5/18	5/54	0
(-cloudy, rain)	2/9	59/153	0	20/51
(cloudy, -rain)	11/27	0	22/189	10/21
(-cloudy, -rain)	0	11/102	1/42	310/357

c. What does Q^2 , the square of the transition matrix, represent?

Every cell in *Q* presents values in the range [0, 1] (because they are probabilities) and we can check that the cells of every row sum to 1 (because every row represent all possible probability worlds (states) for the its row's state. This property makes this matrix to be also known as a *right stochastic matrix*.

The Transition Matrix Q gives the probabilities of "translating" from one state to another in just one step. The Transition Matrix Q^2 gives the probabilities of "translating" from one state to another in two steps.

Being **1** the 4x4 dimensional matrix of all 1s (all ones). Presents the same dimension of Q. Then we can see by definition of a right stochastic matrix that:

Q.**1** = **1**

And that the product of a right stochastic Q with another right stochastic matrix M is algo right stochastic. This means that if Q.**1** = **1** and M.**1** = **1** then Q.M is right stochastic:

(Q.M).**1** = Q.(M.**1**) = Q.**1** = **1**

Then $Q^2, Q^3, ..., Q^k$ and any other power of Q is right stochastic (can be proved by induction).

d. What about Q^n as $n \to \infty$?

The matrix Q^n represents the long-term probability of being in any possible state. And it is proved that for any original transition matrix (right stochastic one), the expected distribution of possible states for a big number of steps in the future is independent of the initial state. This means that all rows in Q^n will tend to be the same bigger the n.

https://en.wikipedia.org/wiki/Stochastic_matrix
https://en.wikipedia.org/wiki/Ergodic_theory
https://en.wikipedia.org/wiki/Perron%E2%80%93Frobenius_theorem
https://www.cs.mcgill.ca/~dprecup/courses/ML/Lectures/ml-lecture08.pdf

e. Explain how to do probabilistic inference in Bayesian networks, assuming that Q^n is available. Is this a practical way to do inference?

I understand by "Q^n *is available*" that it exists, so it can be calculated. So, given the evidence, we can calculate the transition matrix *Q* using the Bayesian network information, and after that we can look for the value of Q^n :

One way to calculate Q^n :

```
function distribution_for(Q, n) {
    R = Q
    for i in range(2, n) {
        R = R x Q
    }
    return first_row(R)
}
```

But the above function can be improved yet (for any n that is a power of 2):

```
function distribution_for(Q, n) {
    R = Q
    for i in range(2, sqrt(n)) {
        R = R x R
    }
    return first_row(R)
}
```

This will work for problems with low number of states. But simple Bayesian networks usually represents big joint distributions for our algorithm. A Bayesian Network with *m* boolean variables with *k* connections per variable represents a joint distribution with 2^m possible states. This means that the transition matrix Q will present $2^m \times 2^m = 2^{2m}$ cell values (transition probabilities) if we enumerate the possibilities with no evidence at all. And if we use the transition matrix with our best algorithm here to calculate the probability distribution expected for a big *n* then we can see that:

Q^2 will require $2^{2m} \times 2^m = 2^{3m}$ simple multiplication operations. And then:
Q^n will require $\sqrt{n} \times 2^{3m}$ simple mul-ops.

It doesn't look good for Bayesian networks with many boolean variables and queries with little evidence.