Programming Assignment: Network Connection

Chapter 7: *Linear Algebra and Probability for Computer Science Applications*

Consider the following situation: There is a network of links between nodes — that is, an undirected graph whose vertices are the nodes and whose edges are the links. Each links fails with probability P. You wish to know what is the probability that K pairs of nodes are connected, for K between 0 and $N(N-1)/2$ where N is the number of nodes.

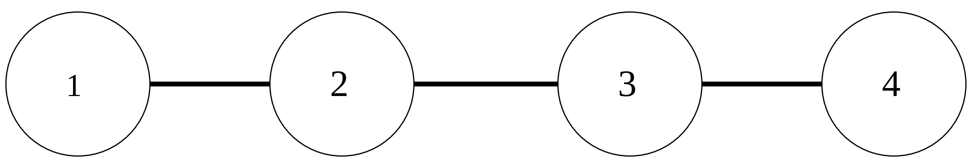
Part A

Write a program `NumConnectedPairs(E,P)` that takes as input a symmetric array `E` of 1's and 0's corresponding to the network and a probability `P` that any given link will fail. It should return a probability distribution `D` such that `D[I]` is the probability that exactly `I` pairs of cities are connected. This omits the probability that no cities are connected, which is always just $\mathtt{P}^{\mathtt{L}}$ where L is the number of links.

There is no polynomial-time solution to this problem, so you should just use exhaustive enumeration; consider all possible subsets of the links, and for each, compute their probability and the number of pairs of nodes connected, and add up the total probability for each.

For a given subnetwork, the number of pairs connected can be computed as follows: Use a depth-first search to divide the network into connected components. Then each connected component of size K connects $K(K-1)/2$ pairs of nodes.

For instance, consider the simple network of three cities shown below and suppose that each link can fail with probability 0.2.



The corresponding function call would be

```
E=[0,1,0,0; 1,0,1,0; 0,1,0,1; 0,0,1,0];
P=0.2;
NumConnectedPairs(E,P)
```

Since there are three links, there are 8 possible sub-networks. These are shown in the table below.

Case	Active links	Pairs connected	Probability
1.	1-2, 2-3, 3-4	6	0.512
2.	1-2, 2-3	3	0.128
3.	1-2, 3-4	2	0.128
4.	1-2	1	0.032
5.	2-3, 3-4	3	0.128
6.	2-3	1	0.032
7.	3-4	1	0.032
8.	∅	0	0.008

Therefore the probability is 0.096 that 1 pair of cities is connected; 0.128 that 2 pairs are connected; 0.256 that 3 pairs are connected; and 0.512 that 6 pairs are connected. So the function should return [0.096, 0.128, 0.256, 0, 0, 0.512]

Part B

In the same situation as part A, write a function `ProbConnected(E,P,PairA)` where `E` and `P` are the same as in part A, and `PairA` is a pair of nodes. For instance, in the above example with the same values of `E` and `P`, `ProbConnected(E,P,[1,2])` should return 0.8 and `ProbConnected(E,P,[1,4])` should return 0.512.

In this case the probability is easily computed, but that will not be the case in general; you should carry out the same enumeration of cases as in part A.

Part C

In the same situation as part A, write a function `CondProbConnected(E,P,PairA,PairB,BConn)` where `E` and `P` are the same as in part A; and `PairA` and `PairB` are each a pair of nodes. If `BConn=1` then the function returns the conditional probability that `PairA` is connected given that `PairB` is connected; if `BConn=0` then the function returns the conditional probability that `PairA` is connected given that `PairB` is *not* connected.

For instance, in the above example

`CondProbConnected(E,P,[1,2],[1,4],1)` should return 1.
`CondProbConnected(E,P,[1,2],[1,4],0)` should return 0.5902 (Cases 2,3,4 out of cases 2-8).
`CondProbConnected(E,P,[1,4],[1,2],1)` should return 0.64.
`CondProbConnected(E,P,[1,4],[1,2],0)` should return 0.
`CondProbConnected(E,P,[2,4],[1,3],1)` should return 0.8.
`CondProbConnected(E,P,[2,4],[1,3],0)` should return 0.3556 (Case 5 out of cases 3-8).

For all three parts of this problem it is OK to write an exponential space solution (i.e. one that essentially generates the table above as a data structure), but it is certainly better to write code that requires only reasonable amounts of memory, though an exponential amount of time.