CS 4820, Spring 2019	**Homework 2, Problem 2**
Name: Kevin Klaben	
NetID: kek228	
Collaborators: Quinn Lui	

(2) *(15 points)* Consider the following simplified model of how a law enforcement organization such as the FBI apprehends the members of an organized crime ring. The crime ring has n members, denoted by $x_1, \ldots, x_n$. A *law enforcement plan* is a sequence of n actions, each of which is either:

- apprehending a member x_j directly: this succeeds with probability q_j; or
- apprehending a member x_j using another member, x_i, as a decoy: this action can only be taken if x_i was already apprehended in a previous step. The probability of success is p_{ij}.

Let's assume that, if an attempt to apprehend x_j fails, then x_j will go into hiding in a country that doesn't allow extradition, and hence x_j can never be apprehended after a failed attempt. Therefore, a law enforcement plan is only considered *valid* if for each of the crime ring's n members, the plan contains only one attempt to apprehend him or her.

Assume we are given an input that specifies the number of members in the crime ring, n, and the probabilities q_j and p_{ij} for each $i \neq j$. You can assume these numbers are strictly positive and that $p_{ij} = p_{ji}$ for all $i \neq j$.

(2a) *(5 points)* Design an algorithm to compute a valid law enforcement plan that maximizes the probability of apprehending all of the crime ring's members, *in the order* $x_1, x_2, \ldots, x_n$. In other words, for this part of the problem you should assume that the j^{th} action in the sequence should be an attempt to apprehend x_j, and the only thing your algorithm needs to decide is whether to apprehend x_j directly or to use one of the earlier members as a decoy, and if so, which decoy to use.

(2b) *(10 points)* Design an algorithm to compute a valid law enforcement plan that maximizes the probability of apprehending all of the crime ring's members, *in any order*. In other words, for this part of the question, your algorithm must decide on the order in which to apprehend the crime ring's members *and* the sequence of operations to use to apprehend them in that order.

```
i=1
while i is less than or equal to n do
  j=1
  store i
  z=x_i
  for each j¡i do
    if p_j i¿z then
      store j
    end if
  end for
```

if the stored valued is equal to i **then**
then no decoy will be used to person i
else
person j will be used as a decoy to apprehend person i
end if
Add 1 to i
end while

Runtime Analysis
The while loop will run exactly n times and each time the while loop runs the inner for each loop will run a maximum of n times with each time making simple comparisons and stored that will run in constant time. Outside the for each loop there is also an if statement will run in constant time as well. Thus, the overall runtime will be O(n^2).

Proof of Correctness Claim: The algorithm generates a valid law enforcement plan which maximizes the probability of apprehending all members of the crime ring going from person 1 in order to person n.

Proof: Suppose for the sake of contradiction that there exists a higher probability law enforcement plan to apprehend every member of the crime ring. The probability is calculated by multiplying the probabilities of each apprehension. Thus, for the overall probability to be higher at least one of the apprehensions in the higher probability plan must be of higher probability then the corresponding probability in the original plan. In such a case this within the for each loop when the comparison was reached this higher probability would have replaced the probability that ended up being in the actual plan. In this case this is a contradiction as the original algorithm could not have passed this point without swapping these probabilities.

(2b) The basic outline for part b is essentially Prim's algorithm using a dummy root node. This situation can be represented by a graph in which each crime gang member is a node as well as there is an additional start node x_0. The edges connecting each of these nodes x_i and x_j have cost 1-(the probability of apprehending member x_i given x_j has already been apprehended). In the case of the start node x_0, it is connected to each other node x_i with weight 1-q_i.

begin with a a pairing T of a set of vertices which initially contains only node x_0 and an edge set that starts empty
while There is at least one node in the graph which is not in the vertex set of T **do**
Find the min-cost edge from the set of vertices in T to the set of vertices not in T
As this edge connects one vertex not in T to a vertex in T, this vertex not in T should be added to T, and this edge not in the edge list of T should be added maintaining order in which it was added to the edge list of T.
end while The order of the edges added to the edge set of T is the order in which apprehensions should be made of the crime gang members starting with the first edge added to the edge set of T.

Runtime Analysis
The pre-processing step will insert tuples of 0 (referencing the node to be used as a decoy in this case the dummy start node) and 1-(the probability of apprehending the ith crime member with no decoy) in the ith position of an array list. This will require time O(n) as there are n

crime members and thus n insertions of constant time. When the algorithm begins the while loop will run a maximum of n times as n crime members must be apprehended. In each iteration of the while loop a for loop will search through the array list looking for the tuple with the smallest value for the probability component. This will take O(n) time. When this minimum probability is found this tuple will be inserted into an apprehension array list in constant time. Next the tuple that was just determined to be the minimum is swapped for a tuple of the same first number and the second number is replaced with 1.01. Then loop over the numbers from 1 to n. Accessing the p_ij where i is this number and j is the number that was just found to be the minimum. Compare 1- (this number) to each number in the same slot of the array list. If it is lower and the second number stored in that tuple is less than or equal to 1 then replaced with a tuple of the number that was just placed in the final apprehension list, and the lower probability. All of these operations are simply comparisons, computations, and accesses. Thus the loop inside takes O(n) time. Overall it takes time complexity O(n^2)

Proof of Correctness Claim: If all probabilities are distinct then the apprehension list returned by the algorithm is the unique most likely apprehension list for the see of probabilities.

Proof: Assume for the sake of contradiction that the apprehension list is not the highest probability list of capturing all criminals. Consider when the apprehension list chose an action that was not optimal and thus was not in the highest probability apprehension list. Let S be the set of criminals (or dummy criminal to indicate no decoy) that have already been apprehended and let the action e be the apprehension of v using decoy w where w is in set S. We claim that e is the highest probability apprehension using one of the criminals already captured or no decoy. If there had been a higher probability apprehension e' in this iteration it would have been considered and selected over e as it is higher probability. However, e was selected implying that e would be selected in every highest probability apprehension list. This is a contradiction as the assumption says that e was not a part of the highest probability apprehension list thus it must be the case that the algorithm generates the highest probability apprehension list.