Solution to Problem 12

Problem Statement:

A scorpion on n vertices is a graph that has a vertex of degree 1 (the tail), connected to a vertex of degree 2 (the body), connected to a vertex of degree n-2 (the head). The other n-3 vertices (the feet) can be a rbitrarily interconnected. Give an O(n) algorithm for deciding whether or not an arbitrary graph is a scorpion, assuming that the graph is represented by an adjacency matrix.

Algorithm:

```
Boolean Scorpion(A: array [1..n,1..n] of (0,1))
{
v,x: integer in [1..n]
//refer to nodes
con_v: list of integer;
//{i| A(v,i)=1}, containing the nodes connected to node v.
discon_v: list of integer;
//{j|A(v,j)=0 and j<>v}, containing the nodes not connected to node v.
If n<4 then print_return("Not a Scorpion");
Else
If n=4 then {
Compute degree of 4 nodes, and sort the 4 degree, if we get (1,1,2,2) then
print_return(" It is a Scorpion");
Else print_return(" Not a Scorpion")
}
Else
If n>4 then {
randomly select v from 1..n, compute con_v and discon_v;
case degree(v) of :
//case 0
0, n-1,n: print_return ("Not a Scorpion");
//end of case 0, case n-1, and case n
//case 1; tail or foot
1: {
L2:             find the only node x connected to node v;
compute con_x and discon_x;
if degree(x)=2 then {
find another node y connected to x besides v;
if degree(y)=n-2 then print_return("It is a Scorpion");
else print_return ("Not a Scorpion");
                }
                else if degree(x)=n-2 then {
L1:                     find only one node y which is not connected to x;
                        if degree(y)<>1 print_return("Not a Scorpion");
                        else { find the only one node z connected to y;
                                if degree(z)=2 print_return("It is a Scorpi
on");
                                else print_return("Not a Scorpion");
```

```
                }
                else print_return ("Not a Scorpion");
}// end of case 1


//case 2; body or foot
2: {
        find the other 2 nodes x,y connected to v;
        if degree(x)=n-2 and degree(y)<>n-2 then GOTO L1;
        else if degree(y)=n-2 and degree(x)<>n-2 then {
                x:=y; GOTO L1;
        }
        else print_return("Not a Scorpion");
}
//end of case 2
//case n-2; head
n-2:{
        x:=v; GOTO L1;
}
//end of case n-2;
//then the degree of v is between 3 to n-3;  foot
otherwise: {
        while !(|con_v|=0 or |discon_v|=0) do{
                choose first node from con_v as x;
                choose first node from discon_v as y;
                if (A[x,y]=1) then  delete y from discon_v;
                else  delete x from con_v;
        }
        if (|discon_v|=0) then print_return("Not a Scorpion");
        else choose the first node of con_v as v; GOTO L2;

      }//end of otherwise;
   }//end of if
}//end of algorithm;



Time Complexity:

If n < 5, we can check the degree of all nodes and decide whether the graph

is a scorpion in O(n). The more complex and general case occurs when the nu
mber of nodes is >= 5.  To solve this case we randomly select a node v, and
 if the degree of v is 0,1,2 and n-2, we find the set of connected vertices
, con_v, and disconnected vert
ices,
discon_v, 4 times and make some constant time comparision, resulting in a t
otal running
time is O(n);
If the degree of the node v is, 3<= degree(v) <= (n-3), the
inner while loop is executed, and in the loop, each time we delete a node f
rom con_v
or discon_v.  Since the total number of nodes in con_v and discon_v is n-1,
```

the running time of whole loop is at most O(n). Thus the running time of t
he algorithm is O(n).

Proof of Correction:

To check whether a graph is a scorpion, we need to confirm that only one
node has a degree of n-2 and this node is the head. From the head, we can u
se the set of disconnected vertices, discon_v,
to decide whether there is a node with degree of 1.
This node should to be the tail. From the tail, we can choose the node
connected to it which should be the body with degree of 2.
If we happen to select a node of degree between 3 and n-3, that node
should be foot . For con_v and discon_v of this node, the head should be in

the set of connected vertices and the tail in the set of disconnected verti
ces. Also, the head should
connect to all nodes except the tail. As a result, we can compare these tw
o sets
to find the tail. Once the tail is found, we can determine whether there i
s a
body and a head. After we find a tail, a body, and a head, we can correctl
y answer
whether the graph is a scorpion.