```
%%% Q.5: Is any graph-node's presence in a path steering the path to end with a specific FP, always?
%%% In other words, are there any graph-nodes that are biased towards a specific FP?

% See if any path node's presence is directing the path towards a specific FP
FPs = [20, 45, 54]; % All fixed points row vector

% Extract all paths ending in FPs 20, 45, and 54
pathCell20 = {}; % Initialize cell to store all paths ending in FP 20

pathCell45 = {}; % Initialize cell to store all paths ending in FP 45

pathCell54 = {}; % Initialize cell to store all paths ending in FP 54

numStates = 64; % Number of possible initial states

for inAllPaths = 1:numStates % Iterate through each first node

    if inAllPaths ~= 1 % For all first nodes except node 1

        forNode = inAllPaths - 1; % Term in product to reach current node's paths in all paths cell

        pathCell20{inAllPaths, 1} = allPaths{1, (forNode*64) + 20}; % Extract current node's paths which end in FP node 20

        pathCell45{inAllPaths, 1} = allPaths{1, (forNode*64) + 45}; % Extract current node's paths which end in FP node 45

        pathCell54{inAllPaths, 1} = allPaths{1, (forNode*64) + 54}; % Extract current node's paths which end in FP node 54

    else % For first node = node 1

        pathCell20{1, 1} = allPaths{1, 20}; % Extract paths which end in FP node 20

        pathCell45{1, 1} = allPaths{1, 45}; % Extract paths which end in FP node 45

        pathCell54{1, 1} = allPaths{1, 54}; % Extract paths which end in FP node 54

    end

end

% Get frequency of nodes in paths ending in FPs 20, 45, & 54, respectively
nodeFrequencyWFP20 = zeros(1, 64); % Initialize row vector to store frequency of all nodes of occuring before FP 20
```

```
nodeFrequencyWFP45 = zeros(1, 64); % Initialize row vector to store frequency of
all nodes of occuring before FP 45

nodeFrequencyWFP54 = zeros(1, 64); % Initialize row vector to store frequency of
all nodes of occuring before FP 54

for inList = 1:length(FPs) % Iterate through all FPs

    if FPs(inList) == 20 % For current FP is 20

        for inPathCell = 1:length(pathCell20) % Iterate through all cells
storing paths ending in FP 20

            if ~isempty(pathCell20{inPathCell}) % For cells that are not empty

                for amongPaths = 1:length(pathCell20{inPathCell}) % Iterate
through all paths within current cell

                    currentPath = pathCell20{inPathCell}(amongPaths); % Extract
current path cell

                    lengthCurrentPath = length(currentPath{1}); % Extract
current path length

                    for inInnerPath = 1:lengthCurrentPath % Iterate through all
nodes in current path

                        nodeInPath = currentPath{1}(inInnerPath); % Extract
current node

                        nodeFrequencyWFP20(nodeInPath) =
nodeFrequencyWFP20(nodeInPath) + 1; % Update frequency of current node in
frequencies vector

                    end

                end

            end

        end

    elseif FPs(inList) == 45 % For current FP is 45

        for inPathCell = 1:length(pathCell45) % Iterate through all cells
storing paths ending in FP 45

            if ~isempty(pathCell45{inPathCell}) % For cells that are not empty

                for amongPaths = 1:length(pathCell45{inPathCell}) % Iterate
through all paths within current cell

                    currentPath = pathCell45{inPathCell}(amongPaths); % Extract
current path cell
```

```
                    lengthCurrentPath = length(currentPath{1}); % Extract
current path length

                    for inInnerPath = 1:lengthCurrentPath % Iterate through all
nodes in current path

                        nodeInPath = currentPath{1}(inInnerPath); % Extract
current node

                        nodeFrequencyWFP45(nodeInPath) =
nodeFrequencyWFP45(nodeInPath) + 1; % Update frequency of current node in
frequencies vector

                    end

                end

            end

        end

    elseif FPs(inList) == 54 % For current FP is 54

        for inPathCell = 1:length(pathCell54) % Iterate through all cells
storing paths ending in FP 54

            if ~isempty(pathCell54{inPathCell}) % For cells that are not empty

                for amongPaths = 1:length(pathCell54{inPathCell}) % Iterate
through all paths within current cell

                    currentPath = pathCell54{inPathCell}(amongPaths); % Extract
current path cell

                    lengthCurrentPath = length(currentPath{1}); % Extract
current path length

                    for inInnerPath = 1:lengthCurrentPath % Iterate through all
nodes in current path

                        nodeInPath = currentPath{1}(inInnerPath); % Extract
current node

                        nodeFrequencyWFP54(nodeInPath) =
nodeFrequencyWFP54(nodeInPath) + 1; % Update frequency of current node in
frequencies vector

                    end

                end

            end
        end

    end

end

% Create table using frequencies of path ending in respective FPs
Node = indices'; % Column vector of graph-nodes

FP20 = nodeFrequencyWFP20'; % Column vector of graph-nodes corresponding
frequencies for path ending in FP 20

FP45 = nodeFrequencyWFP45'; % Column vector of graph-nodes corresponding
frequencies for path ending in FP 45

FP54 = nodeFrequencyWFP54'; % Column vector of graph-nodes corresponding
frequencies for path ending in FP 54

nodeFrequencyTable = table(Node, FP20, FP45, FP54); % Create table

% Display table
disp('Node and the number of times it ends in FPs 20, 45, & 54, respectively:')
disp(nodeFrequencyTable);

%

% Create matrix containing rows from node frequencies table which are biased
towards a specific FP
biasedNodeMatrix = []; % Initialize matrix

for inNodeFrequencyTable = 1:height(nodeFrequencyTable) % Iterate through each
row in frequencies table

    currentNodeFreqArray = nodeFrequencyTable{inNodeFrequencyTable, 2:end}; %
Extract frequencies of current node appearing in paths ending in FPs 20, 45,
& 54, respectively

    presenceNon0 = currentNodeFreqArray(currentNodeFreqArray ~= 0); % Extract
frequency if it's not 0

    if length(presenceNon0) == 1 % For two out of three frequencies is 0

        biasedNodeMatrix(end + 1, :) =
nodeFrequencyTable{inNodeFrequencyTable, :}; % Add row to matrix

    end

end

% Create table of nodes, their network-states, and the FP they are biased to end
a path in
biasedNodes = zeros(1, size(biasedNodeMatrix, 1)); % Initialize vector to store
biased nodes
```

```
biasedNodeStates = strings(1, size(biasedNodeMatrix, 1)); % Initialize matrix to
store biased node network-states

FPbias = zeros(1, size(biasedNodeMatrix, 1)); % Initialize vector to store FPs
nodes are biased towards

for inBiasedNodeMatrix = 1:size(biasedNodeMatrix, 1) % Iterate through all rows
of frequencies matrix

    frequencyData = biasedNodeMatrix(inBiasedNodeMatrix, :); % Extract
frequencies data for current row

    biasedNodes(inBiasedNodeMatrix) = frequencyData(1); % Store current biased
node

    biasedNodeStates(inBiasedNodeMatrix) =
binaryStatesChar(frequencyData(1), :); % Store current biased node network-state

    for inFreqData = 2:length(frequencyData(:)) % Iterate through frequencies of
biased node-including path ending in FPs

        if frequencyData(inFreqData) ~= 0 % For frequency is greater than 0

            if inFreqData == 2 % For frequency is non-zero for ending in FP 20

                FPbias(inBiasedNodeMatrix) = 20; % Store FP in which path
carrying current biased node ends

            elseif inFreqData == 3 % For frequency is non-zero for ending in FP
45

                FPbias(inBiasedNodeMatrix) = 45; % Store FP in which path
carrying current biased node ends

            elseif inFreqData == 4 % For frequency is non-zero for ending in FP
54

                FPbias(inBiasedNodeMatrix) = 54; % Store FP in which path
carrying current biased node ends

            end

        end

    end

end

biasedNodes = biasedNodes'; % Row vector transpose for table

biasedNodeStates = biasedNodeStates'; % Row vector transpose for table

FPbias = FPbias'; % Row vector transpose for table
```

```
biasedNodesTable = table(biasedNodes, biasedNodeStates, FPbias); % Create table

% Display table
disp('All biased nodes and their FB biases:')
disp(biasedNodesTable);

%

% FP 20 is desirable (TGFb node is inactivated); finding biased nodes leading
invariably to FP 20
FP20bias = biasedNodesTable(biasedNodesTable.FPbias == 20, :); % Extract rows
where FPbias is 20

disp('Nodes biased towards FP 20:'); % Display table label

disp(FP20bias); % Display table

disp('Number of nodes biased towards FP 20:'); % Display number of nodes biased
towards FP 20
disp(height(FP20bias));

%

% In other cases, FP 45 is desirable (TGFb node is activated); finding biased
nodes leading invariably to FP 45
FP45bias = biasedNodesTable(biasedNodesTable.FPbias == 45, :); % Extract rows
where FPbias is 45

disp('Nodes biased towards FP 45:'); % Display table label

disp(FP45bias); % Display table

disp('Number of nodes biased towards FP 45:'); % Display number of nodes biased
towards FP 20
disp(height(FP45bias));
```

Output:

```
Node and the number of times it ends in FPs 20, 45, & 54, respectively:

Node    FP20    FP45    FP54

____    ____    ____    ____


1        7       6       8

2        2       0       1

3       35      20      25

4       10       0       0

5        0      10       5
```

```
6       0       0       5
7       5       2       2
8       2       0       2
9      22      31      35
10      17      28      28
11      22      31      35
12      17      29      30
13       0      40      32
14       0       5       4
15      15      11      11
16      15      11      11
17       3       4       5
18       2       0       1
19       7       0       0
20     309       0       0
21       0      18       9
22       0       0      72
23       5      10      10
24       9       0      18
25       3       8       8
26       2       6       4
27      13       0       0
28      32       0       0
29       0      55      44
30       0       5       4
31      21      17      18
32      21      17      18
33       8       6       8
34       2       0       1
35      40      20      25
36      46       0       0
37       0      64      32
```

38	0	0	86
39	35	35	30
40	18	0	18
41	12	6	15
42	6	3	8
43	12	4	14
44	6	2	9
45	0	323	0
46	0	21	42
47	0	21	0
48	68	17	51
49	4	4	5
50	2	0	1
51	6	0	0
52	94	0	0
53	0	64	32
54	0	0	379
55	15	60	50
56	9	0	18
57	4	4	6
58	2	2	2
59	14	0	0
60	27	0	0
61	0	107	0
62	0	9	18
63	13	14	13
64	14	14	15

All biased nodes and their FB biases:

biasedNodes	**biasedNodeStates**	**FPbias**

```
4             "000011"          20

6             "000101"          54

19             "010010"          20

20             "010011"          20

22             "010101"          54

27             "011010"          20

28             "011011"          20

36             "100011"          20

38             "100101"          54

45             "101100"          45

47             "101110"          45

51             "110010"          20

52             "110011"          20

54             "110101"          54

59             "111010"          20

60             "111011"          20

61             "111100"          45
```

```
Nodes biased towards FP 20:

biasedNodes    biasedNodeStates    FPbias

───────────    ────────────────    ──────


4             "000011"          20

19             "010010"          20

20             "010011"          20

27             "011010"          20

28             "011011"          20

36             "100011"          20

51             "110010"          20

52             "110011"          20

59             "111010"          20

60             "111011"          20
```

```
Number of nodes biased towards FP 20:

10


Nodes biased towards FP 45:

biasedNodes    biasedNodeStates    FPbias

___________    ________________    ______


45             "101100"          45

47             "101110"          45

61             "111100"          45


Number of nodes biased towards FP 45:

3
```