```
%%% Get longest paths starting from each graph-node and get insights thereof

% Recursive function to build longest paths
function longestPaths = recursive(G, longestPaths, numLongestPaths)

% Find if successors present of current longest path(s)
successorsLogical = zeros(1, numLongestPaths); % Initialize array containing
logical values for presence of successors

for inLongestPaths = 1:numLongestPaths % Iterate through every longest path

    endNode = longestPaths{inLongestPaths}(end); % Extract last node in current
longest path

    successorsList = successors(G, endNode); % Extract all successors of last
node

    haveSuccessor = ~isempty(successorsList); % Check if successors of last node
present

    successorsLogical(inLongestPaths) = haveSuccessor; % Update status of
successor(s) presence

end

pathsToUpdate = find(successorsLogical == 1); % Logical 1 for longest paths with
end node successors

% Add successors to current longest paths
if any(successorsLogical) % For successor(s) present for any longest path

    % Prevent further recursion if each last node of every path has only one
successor that's same as last node
    sameSuccessorArray = zeros(1, length(longestPaths)); % Initialize array to
store logical value for same successor

    for inLongestPaths = 1:length(longestPaths) % Iterate through every longest
path

        path = longestPaths{inLongestPaths}; % Extract current longest path

        successorNodes = successors(G, path(end)); % Extract successor(s) of the
last node

        numNodeSuccessors = length(successorNodes); % Store number of successors

        if numNodeSuccessors == 1 & successorNodes(1) == path(end) % For the
only successor matching with last node

            sameSuccessorArray(inLongestPaths) = 1; % Update same successor
status for current path in array

        end
```

```
    end

    if all(sameSuccessorArray) % For all longest paths having a single successor
same as path's last node

        return % Return control outside recursive function (to invoking function)

    else

        % Do nothing

    end

    newLongestPaths = {}; % Initialize cell to store updated longest paths

    for inPathsToUpdate = 1:length(pathsToUpdate) % Iterate through each longest
path having end node successor(s)

        pathIndex = pathsToUpdate(inPathsToUpdate); % Extract index of path
having successor(s) to last node

        path = longestPaths{pathIndex}; % Extract path with successor

        pathEnd = path(end); % Extract last node in path

        nodeSuccessors = successors(G, pathEnd); % Extract successors of last
node

        for inNodeSuccessors = 1:length(nodeSuccessors) % Iterate through array
of end node-successors

            currentSuccessorNode = nodeSuccessors(inNodeSuccessors); % Extract
current successor among all successors

            if ~ismember(currentSuccessorNode, path) % For current successor not
already present in path (to avoid network loops)

                newLongestPaths{end + 1} = [path, currentSuccessorNode]; % Add
successor to current path and add updated path to new longest paths array

            else

                continue % Choose next successor node in successors array

            end

        end

    end

    if ~isempty(newLongestPaths) % For new successor not already present in
path, variable will not be empty
```

```
        longestPaths = {}; % Empty longest paths array to repopulate it

        longestPaths = newLongestPaths; % Refill longest paths array

    end

    numLongestPaths = length(longestPaths); % Assign updated number of longest
paths to include as argument in recursive function

    % Recursive function call to find successors of current longest path(s)
    longestPaths = recursive(G, longestPaths, numLongestPaths); % Add new path-
end successors recursively

elseif ~any(successorsLogical) % For no successor present for any longest path

    disp(['returning control because no successors for', num2str(longestPaths{1}
(1))]);
    return % Return control outside recursive function (to invoking function)

end

end

% Number of initial states (64)
numStates = numnodes(G);

% Function to find longest paths from each node
function longestPathsCell = findLongestPaths(G, numStates)

% Initialize nodes array
allNodes = 1:numStates;

% Initialize variable to store number of longest paths
numLongestPaths = 1; % Only longest path comprises only source node (initially)

% Display description
disp('Longest paths starting from each graph-node:');

% Paths for each initial state (node)
for whichNode = 1:numStates % Iterate through all nodes

    % Extract node
    node = allNodes(whichNode);

    % Reset longest paths variable for current node
    longestPaths = {node};

    % Invoke recursive function to find longest path
    longestPathsCell{node} = recursive(G, longestPaths, numLongestPaths);

end

end
```

```
% Measure time to execute block
tic % Start time-keeping

% Store longest paths starting from each initial state
longestPathsCell = findLongestPaths(G, numStates); % Generate cell array to
store longest paths

for inLongestPathsCell = 1:length(longestPathsCell) % Iterate through longest
paths cells for every initial state

    disp(longestPathsCell{inLongestPathsCell}); % Extract and display all
longest paths of current initial state

end

disp('Time taken to obtain longest paths:'); % Display measured time

toc % End time-keeping

%%

% Number of occurences of each graph-node across all longest paths
counts = zeros(1, numStates); % Values in dictionary to store number of node
occurences

countsDictionary = dictionary(1:numStates, counts); % Initialize dictionary to
store number of node occurences

for inLongestPathsCell = 1:length(longestPathsCell) % Iterate through longest
paths cells for every node

    longestPaths = longestPathsCell{inLongestPathsCell}; % Extract longest paths
corresponding to current node

    for inEachColumn = 1:length(longestPaths) % Iterate through longest paths
corresponding to current node

        currentPath = longestPaths{inEachColumn}; % Extract current longest path
for current node

        FPofPath = currentPath(end); % Extract FP at the end of path

        for inCurrentPath = 1:length(currentPath) % Iterate through every node
in current longest path

            element = currentPath(inCurrentPath); % Extract current node ID

            countsDictionary(element) = countsDictionary(element) + 1; % Update
number of occurences corresponding to current node ID

        end

    end
```

```
end

% Display dictionary storing number of occurences corresponding to each node
disp('Graph-nodes and their corresponding number of occurences across all
longest paths:');
disp(countsDictionary);

%%

% Across longest paths, visualize frequency of number of graph-nodes vs. number
of occurences
counts = values(countsDictionary); % Extract number of occurences from
dictionary storing counts

edges = 1:max(counts); % Define bin edges for histogram

% Plot histogram
disp('How many graph-nodes vs. how many times they occur:');
histogram_plot = histogram(counts, edges);
ylabel("# graph-nodes in bin", "FontName", "Helvetica", "FontAngle", "normal",
"FontWeight", "normal");
xlabel("# graph-node occurences", "FontName", "Helvetica", "FontAngle",
"normal", "FontWeight", "normal");
```

Output:

```
Longest paths starting from each graph-node:

{[1 48 36 52 20]}


{[2 52 20]}


{[3 48 36 52 20]}


{[4 52 20]}


{[5 61 45]}


{[6 54]}


{[7 59 28 20]}


{[8 22 54]}    {[8 52 20]}
```

{[9 39 29 37 61 45]} {[9 39 29 53 61 45]} {[9 55 13 37 61 45]} {[9 55 13 53 61 45]} {[9 55 29 37 61 45]} {[9 55 29 53 61 45]}

{[10 39 29 37 61 45]} {[10 39 29 53 61 45]} {[10 55 13 37 61 45]} {[10 55 13 53 61 45]} {[10 55 29 37 61 45]} {[10 55 29 53 61 45]}

{[11 39 29 37 61 45]} {[11 39 29 53 61 45]} {[11 55 13 37 61 45]} {[11 55 13 53 61 45]} {[11 55 29 37 61 45]} {[11 55 29 53 61 45]}

{[12 39 29 37 61 45]} {[12 39 29 53 61 45]} {[12 55 13 37 61 45]} {[12 55 13 53 61 45]} {[12 55 29 37 61 45]} {[12 55 29 53 61 45]}

{[13 37 61 45]} {[13 53 61 45]}

{[14 37 61 45]} {[14 53 61 45]}

{[15 35 48 36 52 20]}

{[16 35 48 36 52 20]}

{[17 46 38 54]} {[17 62 38 54]}

{[18 52 20]}

{[19 20]}

{[20]}

{[21 61 45]}

{[22 54]}

{[23 22 54]} {[23 61 45]}

{[24 22 54]}

{[25 37 61 45]} {[25 46 38 54]} {[25 53 61 45]} {[25 62 38 54]}

{[26 37 61 45]} {[26 53 61 45]}

{[27 20]}

{[28 20]}

{[29 37 61 45]} {[29 53 61 45]}

{[30 37 61 45]} {[30 53 61 45]}

{[31 3 48 36 52 20]} {[31 35 48 36 52 20]}

{[32 3 48 36 52 20]} {[32 35 48 36 52 20]}

{[33 48 36 52 20]}

{[34 52 20]}

{[35 48 36 52 20]}

{[36 52 20]}

{[37 61 45]}

{[38 54]}

{[39 29 37 61 45]} {[39 29 53 61 45]}

{[40 22 54]} {[40 52 20]}

{[41 48 36 52 20]}

{[42 24 22 54]} {[42 40 22 54]} {[42 40 52 20]} {[42 56 22 54]}

{[43 48 36 52 20]}

{[44 24 22 54]} {[44 40 22 54]} {[44 40 52 20]} {[44 56 22 54]}

{[45]}

{[46 38 54]}

{[47 45]}

{[48 36 52 20]}

{[49 46 38 54]} {[49 62 38 54]}

{[50 52 20]}

{[51 28 20]}

{[52 20]}

{[53 61 45]}

{[54]}

{[55 13 37 61 45]} {[55 13 53 61 45]} {[55 29 37 61 45]} {[55 29 53 61 45]}

```
{[56 22 54]}

{[57 46 38 54]}    {[57 62 38 54]}

{[58 38 54]}    {[58 52 20]}    {[58 61 45]}

{[59 28 20]}

{[60 20]}

{[61 45]}

{[62 38 54]}

{[63 3 48 36 52 20]}

{[64 3 48 36 52 20]}

Time taken to obtain longest paths:
Elapsed time is 0.126948 seconds.

Graph-nodes and their corresponding number of occurences across all longest paths:
1  ⇒ 1
2  ⇒ 1
3  ⇒ 5
4  ⇒ 1
5  ⇒ 1
6  ⇒ 1
7  ⇒ 1
8  ⇒ 2
9  ⇒ 6
```

10 □ 6

11 □ 6

12 □ 6

13 □ 12

14 □ 2

15 □ 1

16 □ 1

17 □ 2

18 □ 1

19 □ 1

20 □ 35

21 □ 1

22 □ 12

23 □ 2

24 □ 3

25 □ 4

26 □ 2

27 □ 1

28 □ 4

29 □ 22

30 □ 2

31 □ 2

32 □ 2

33 □ 1

34 □ 1

35 □ 5

36 □ 16

37 □ 22

38 □ 12

39 □ 10

40 □ 6

41 □ 1

42 □ 4

43 □ 1

44 □ 4

45 □ 51

46 □ 5

47 □ 1

48 □ 15

49 □ 2

50 □ 1

51 □ 1

52 □ 27

53 □ 22

54 □ 26

55 □ 20

56 □ 3

57 □ 2

58 □ 3

59 □ 2

60 □ 1

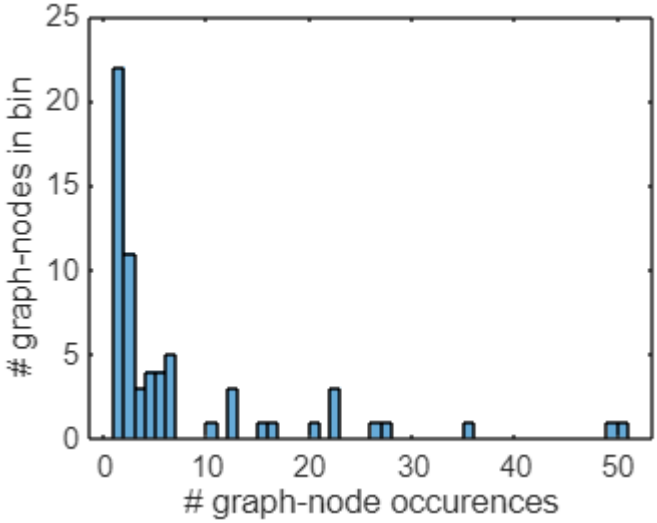
61 □ 49

62 □ 5

63 □ 1

64 □ 1

How many graph-nodes vs. how many times they occur:


graph-nodes in bin
graph-node occurences
0
5
10
15
20
25
0
10
20
30
40
50