```
%%% Q.6: Are there any cyclic paths in the STG that are cyclic (except those
starting from the FPs)?

% Find all acyclic paths and cycles occuring in STG
function [currentNodeAcyclicPaths, currentNodeCyclicPaths] =
findCurrentNodePaths(G, whichNode, numberAcyclicPaths, currentNodeAcyclicPaths,
currentNodeCyclicPaths) % Define recursive function to build current source
graph-node complete and cyclic paths

% Create current source graph-node path and initialize and reset return
variables and temporary variables
if numberAcyclicPaths == 0 % For no paths added to current source graph-node

    currentNodeAcyclicPaths = {}; % Initialize cell to store all FP-ending paths
starting from current source graph-node

    currentNodeCyclicPaths = {}; % Initialize cell to store all cyclic paths
starting from current source graph-node

    currentNodeAcyclicPaths{1} = whichNode; % Add node ID of current source
graph-node as path #1

    numberAcyclicPaths = numberAcyclicPaths + 1; % Update number of acyclic
paths (numberAcyclicPaths = 1)

end

newCurrentNodeAcyclicPaths = {}; % Initialize temporary cell to store all
acyclic paths starting from current source graph-node

% Prevent further recursion if the last node of a path appears as a successor
% and flag cyclic paths that have subpaths appearing at least twice
sameSuccessorArray = zeros(1, numberAcyclicPaths); % Initialize and reset array
to store logical value for same successor presence

isCyclicPathArray = zeros(1, numberAcyclicPaths); % Initialize and reset array
to store logical values for whether path is cyclic path

for inAllPaths = 1:numberAcyclicPaths % Iterate through all paths

    currentPath = currentNodeAcyclicPaths{inAllPaths}; % Extract current path

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

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

    if numberNodeSuccessors == 1 && successorNodes(1) == currentPath(end) % For
the only successor matching with last node (acyclic complete path)

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

```
    else % For 1 or more successors (path last node is not FP)

        % Check if a certain path has subpath(s) that appear more than once
        for inSuccessors = 1:numberNodeSuccessors % Iterate through all
successors of current path last node

            currentSuccessor = successorNodes(inSuccessors); % Extract current
successor

            successorCount = sum(currentPath == currentSuccessor); % Count the
number of times current successor appears in current path

            subPaths = {}; % Initialize and reset cell to store subpaths between
current successor in full path

            foundMatch = false; % Flag to track subpath matches

            if successorCount >= 2 % For successor appears for 3rd time or more
and already present in path at least 2 times

                disp(['Successor node ', num2str(currentSuccessor), ' appeared
', num2str(successorCount), ' times in a path.']); % Display number of times a
successor appears in a path
                %%% The above display line is never executed, showing that no
paths enter repeating subpaths

                currentSuccessorIndices = find(currentPath == currentSuccessor);
% Store indices of current successor in path

                numberCurrentSuccessor = length(currentSuccessorIndices); %
Store number of times successor appears in path

                % Create subpaths with graph-nodes between 2 occurences of
current successor node
                for inCurrentSuccessorIndices = 1:numberCurrentSuccessor %
Iterate number of times successor appears in path

                    currentIndex =
currentSuccessorIndices(inCurrentSuccessorIndices); % Extract index of ith
appearance

                    if inCurrentSuccessorIndices ~= numberCurrentSuccessor % For
current successor appearance is its last in path

                        nextIndex =
currentSuccessorIndices(inCurrentSuccessorIndices + 1); % Extract next index of
current successor

                        subPaths{inCurrentSuccessorIndices} =
currentPath(currentIndex : nextIndex - 1); % Store subpath from current index
of current successor to its next index

                    elseif inCurrentSuccessorIndices == numberCurrentSuccessor %
For current successor appearance is its last in path
```

```
                        subPaths{inCurrentSuccessorIndices} =
currentPath(currentIndex : end); % Store subpath from last index of current
successor to path end

                    end

                end

                numberSubPaths = numel(subPaths); % Number of subpaths

                % Find subpath matches
                for refPath = 1 : numberSubPaths - 1 % Iterate through all
subpaths to be used as reference to search for match

                    for testPath = refPath + 1 : numberSubPaths % Iterate
through all subpaths to be tested for match

                        if isequal(subPaths{refPath}, subPaths{testPath}) % For
compared subpaths are same

                            foundMatch = true; % Update flag to positive for
match

                            break; % Exit inner loop if match found

                        end

                    end

                    if foundMatch % For flag positive for match

                        break; % Exit outer loop if match found

                    end

                end

            end

            if foundMatch % For flag positive for match for current successor

                isCyclicPathArray(inAllPaths) = 1; % Update cycle-causing
successor status for current path in array

                break % Go to next path

            end

        end

    end

end
```

```
indicesCyclicPaths = find(isCyclicPathArray == 1); % Store indices of all paths
that are cyclic

if all(sameSuccessorArray) % For all cyclic paths starting from current graph-
node have been moved to cyclic paths cell & only 1 successor--same as path's
last node--present

    return % Return control outside recursive function (to invoking function)--
go to next source graph-node

end

% Add successors to current source graph-node paths
% and separate cyclic paths from acyclic FP-ending ones
for inAllPaths = 1:numberAcyclicPaths % Iterate through all paths starting from
current source graph-node

    if ismember(inAllPaths, indicesCyclicPaths) % Current path index matches
cyclic path's index (is cyclic path)

        currentNodeCyclicPaths{end + 1} = currentNodeAcyclicPaths{inAllPaths}; %
Add path to cyclic paths array

        continue % Move to next path

    end

    currentPath = currentNodeAcyclicPaths{inAllPaths}; % Extract current path

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

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

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

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

        if currentPath(end) == currentSuccessorNode % For successor is same as
last node in current path

            newCurrentNodeAcyclicPaths{end + 1} = currentPath; % No successor
added because last node is FP and add path to temporary cell

        elseif currentPath(end) ~= currentSuccessorNode % For successor is
unique or present at position other than path end

            newCurrentNodeAcyclicPaths{end + 1} = [currentPath,
currentSuccessorNode]; % Add successor to current path and add updated path to
temporary cell

        end
```

```
    end

end

if ~isempty(newCurrentNodeAcyclicPaths) % For new successors added to paths,
variable will not be empty

    currentNodeAcyclicPaths = {}; % Empty cell to repopulate it

    currentNodeAcyclicPaths = newCurrentNodeAcyclicPaths; % Repopulate cell

end

numberAcyclicPaths = length(currentNodeAcyclicPaths); % Assign updated number of
acyclic paths to include as argument in recursive function

% Recursive function call to find successors of current paths (cyclic and
acyclic)
[currentNodeAcyclicPaths, currentNodeCyclicPaths] = findCurrentNodePaths(G,
whichNode, numberAcyclicPaths, currentNodeAcyclicPaths, currentNodeCyclicPaths);
% Add new path-end successors recursively

end

%

% Function to store all complete paths and cyclic paths from each source graph-
node
function [acyclicPaths, cyclicPaths] = findCompleteAndCyclicPaths(G) % Define
function

nodeIDs = 1:numnodes(G); % Initialize graph-nodes vector

acyclicPaths = {}; % Initialize and reset variable to store acyclic paths

cyclicPaths = {}; % Initialize and reset variable to store cyclic paths

numberAcyclicPaths = 0; % Initialize and reset variable to store number of
paths--initially, only path comprises only source node

% Paths starting from each graph-node
for whichNode = nodeIDs % Iterate through all graph-nodes

    % Invoke current graph-node paths function
    [acyclicPaths{whichNode, 1}, cyclicPaths{whichNode, 1}] =
findCurrentNodePaths(G, whichNode, numberAcyclicPaths, acyclicPaths,
cyclicPaths);

end

end

%
```

```
% Store and display all complete paths and cyclic paths starting from every
graph-node
tic % Start timer

[acyclicPaths, cyclicPaths] = findCompleteAndCyclicPaths(G); % Store all
complete and cyclic paths

disp('All complete (acyclic) paths:'); % Describe output

for inAcyclicPaths = 1:length(acyclicPaths) % Iterate through every row in
complete paths cell

    disp(acyclicPaths{inAcyclicPaths}); % Extract and display all complete paths
starting from current row source graph-node

end

disp('All incomplete (not having an FP at end or cyclic) paths:'); % Describe
output

for inCyclicPaths = 1:length(cyclicPaths) % Iterate through every row in cyclic
paths cell

    disp(cyclicPaths{inCyclicPaths}); % Extract and display all cyclic paths
starting from current row source graph-node

end

toc % End timer

%%% No cyclic paths were found in throughtout the STG
```

**Output:**

```
All complete (acyclic) paths:

{[1 20]}    {[1 45]}    {[1 46 38 54]}    {[1 46 45]}    {[1 46 54]}    {[1 47 45]}
{[1 48 20]}    {[1 48 22 54]}    {[1 48 36 20]}    {[1 48 36 52 20]}    {[1 48 38 54]}
{[1 48 45]}    {[1 48 52 20]}    {[1 48 54]}    {[1 52 20]}    {[1 54]}    {[1 60 20]}
{[1 61 45]}    {[1 62 38 54]}    {[1 62 45]}    {[1 62 54]}


{[2 20]}    {[2 52 20]}    {[2 54]}


{[3 20]}    {[3 45]}    {[3 46 38 54]}    {[3 46 45]}    {[3 46 54]}    {[3 47 45]}
{[3 48 20]}    {[3 48 22 54]}    {[3 48 36 20]}    {[3 48 36 52 20]}    {[3 48 38 54]}
{[3 48 45]}    {[3 48 52 20]}    {[3 48 54]}    {[3 52 20]}    {[3 60 20]}
```

{[4 20]}  {[4 52 20]}

{[5 45]}  {[5 54]}  {[5 61 45]}

{[6 54]}

{[7 20]}  {[7 22 54]}  {[7 45]}  {[7 52 20]}  {[7 54]}  {[7 59 20]}  {[7 59 28 20]}  {[7 60 20]}  {[7 61 45]}

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

{[9 20]}  {[9 23 20]}  {[9 23 22 54]}  {[9 23 45]}  {[9 23 54]}  {[9 23 61 45]}  {[9 24 20]}  {[9 24 22 54]}  {[9 24 54]}  {[9 37 45]}  {[9 37 54]}  {[9 37 61 45]}  {[9 38 54]}  {[9 39 20]}  {[9 39 22 54]}  {[9 39 27 20]}  {[9 39 28 20]}  {[9 39 29 37 45]}  {[9 39 29 37 54]}  {[9 39 29 37 61 45]}  {[9 39 29 38 54]}  {[9 39 29 45]}  {[9 39 29 53 45]}  {[9 39 29 53 54]}  {[9 39 29 53 61 45]}  {[9 39 29 54]}  {[9 39 45]}  {[9 39 52 20]}  {[9 39 54]}  {[9 39 59 20]}  {[9 39 59 28 20]}  {[9 39 60 20]}  {[9 39 61 45]}  {[9 40 20]}  {[9 40 22 54]}  {[9 40 52 20]}  {[9 40 54]}  {[9 45]}  {[9 46 38 54]}  {[9 46 45]}  {[9 46 54]}  {[9 47 45]}  {[9 48 20]}  {[9 48 22 54]}  {[9 48 36 20]}  {[9 48 36 52 20]}  {[9 48 38 54]}  {[9 48 45]}  {[9 48 52 20]}  {[9 48 54]}  {[9 52 20]}  {[9 53 45]}  {[9 53 54]}  {[9 53 61 45]}  {[9 54]}  {[9 55 13 37 45]}  {[9 55 13 37 54]}  {[9 55 13 37 61 45]}  {[9 55 13 38 54]}  {[9 55 13 45]}  {[9 55 13 53 45]}  {[9 55 13 53 54]}  {[9 55 13 53 61 45]}  {[9 55 13 54]}  {[9 55 20]}  {[9 55 22 54]}  {[9 55 27 20]}  {[9 55 28 20]}  {[9 55 29 37 45]}  {[9 55 29 37 54]}  {[9 55 29 37 61 45]}  {[9 55 29 38 54]}  {[9 55 29 45]}  {[9 55 29 53 45]}  {[9 55 29 53 54]}  {[9 55 29 53 61 45]}  {[9 55 29 54]}  {[9 55 45]}  {[9 55 54]}  {[9 55 61 45]}  {[9 56 20]}  {[9 56 22 54]}  {[9 56 54]}  {[9 60 20]}  {[9 61 45]}  {[9 62 38 54]}  {[9 62 45]}  {[9 62 54]}

{[10 20]}  {[10 23 20]}  {[10 23 22 54]}  {[10 23 45]}  {[10 23 54]}  {[10 23 61 45]}  {[10 24 20]}  {[10 24 22 54]}  {[10 24 54]}  {[10 37 45]}  {[10 37 54]}  {[10 37 61 45]}  {[10 38 54]}  {[10 39 20]}  {[10 39 22 54]}  {[10 39 27 20]}  {[10 39 28 20]}  {[10 39 29 37 45]}  {[10 39 29 37 54]}  {[10 39 29 37 61 45]}  {[10 39 29 38 54]}  {[10 39 29 45]}  {[10 39 29 53 45]}  {[10 39 29 53 54]}  {[10 39 29 53 61 45]}  {[10 39 29 54]}  {[10 39 45]}  {[10 39 52 20]}  {[10 39 54]}  {[10 39 59 20]}  {[10 39 59 28 20]}  {[10 39 60 20]}  {[10 39 61 45]}  {[10 40 20]}  {[10 40 22 54]}  {[10 40 52 20]}  {[10 40 54]}  {[10 45]}  {[10 47 45]}  {[10 52 20]}  {[10 53 45]}  {[10 53 54]}  {[10 53 61 45]}  {[10 54]}  {[10 55 13 37 45]}  {[10 55 13 37 54]}  {[10 55 13 37 61 45]}  {[10 55 13 38 54]}  {[10 55 13 45]}  {[10 55 13 53 45]}  {[10 55 13 53 54]}  {[10

55 13 53 61 45]}  {[10 55 13 54]}  {[10 55 20]}  {[10 55 22 54]}  {[10 55 27 20]}  {[10 55 28 20]}  {[10 55 29 37 45]}  {[10 55 29 37 54]}  {[10 55 29 37 61 45]}  {[10 55 29 38 54]}  {[10 55 29 45]}  {[10 55 29 53 45]}  {[10 55 29 53 54]}  {[10 55 29 53 61 45]}  {[10 55 29 54]}  {[10 55 45]}  {[10 55 54]}  {[10 55 61 45]}  {[10 56 20]}  {[10 56 22 54]}  {[10 56 54]}  {[10 61 45]}

{[11 20]}  {[11 21 45]}  {[11 21 54]}  {[11 21 61 45]}  {[11 22 54]}  {[11 23 20]}  {[11 23 22 54]}  {[11 23 45]}  {[11 23 54]}  {[11 23 61 45]}  {[11 24 20]}  {[11 24 22 54]}  {[11 24 54]}  {[11 37 45]}  {[11 37 54]}  {[11 37 61 45]}  {[11 38 54]}  {[11 39 20]}  {[11 39 22 54]}  {[11 39 27 20]}  {[11 39 28 20]}  {[11 39 29 37 45]}  {[11 39 29 37 54]}  {[11 39 29 37 61 45]}  {[11 39 29 38 54]}  {[11 39 29 45]}  {[11 39 29 53 45]}  {[11 39 29 53 54]}  {[11 39 29 53 61 45]}  {[11 39 29 54]}  {[11 39 45]}  {[11 39 52 20]}  {[11 39 54]}  {[11 39 59 20]}  {[11 39 59 28 20]}  {[11 39 60 20]}  {[11 39 61 45]}  {[11 40 20]}  {[11 40 22 54]}  {[11 40 52 20]}  {[11 40 54]}  {[11 45]}  {[11 46 38 54]}  {[11 46 45]}  {[11 46 54]}  {[11 47 45]}  {[11 48 20]}  {[11 48 22 54]}  {[11 48 36 20]}  {[11 48 36 52 20]}  {[11 48 38 54]}  {[11 48 45]}  {[11 48 52 20]}  {[11 48 54]}  {[11 52 20]}  {[11 53 45]}  {[11 53 54]}  {[11 53 61 45]}  {[11 54]}  {[11 55 13 37 45]}  {[11 55 13 37 54]}  {[11 55 13 37 61 45]}  {[11 55 13 38 54]}  {[11 55 13 45]}  {[11 55 13 53 45]}  {[11 55 13 53 54]}  {[11 55 13 53 61 45]}  {[11 55 13 54]}  {[11 55 20]}  {[11 55 22 54]}  {[11 55 27 20]}  {[11 55 28 20]}  {[11 55 29 37 45]}  {[11 55 29 37 54]}  {[11 55 29 37 61 45]}  {[11 55 29 38 54]}  {[11 55 29 45]}  {[11 55 29 53 45]}  {[11 55 29 53 54]}  {[11 55 29 53 61 45]}  {[11 55 29 54]}  {[11 55 45]}  {[11 55 54]}  {[11 55 61 45]}  {[11 56 20]}  {[11 56 22 54]}  {[11 56 54]}  {[11 60 20]}

{[12 20]}  {[12 21 45]}  {[12 21 54]}  {[12 21 61 45]}  {[12 22 54]}  {[12 23 20]}  {[12 23 22 54]}  {[12 23 45]}  {[12 23 54]}  {[12 23 61 45]}  {[12 24 20]}  {[12 24 22 54]}  {[12 24 54]}  {[12 37 45]}  {[12 37 54]}  {[12 37 61 45]}  {[12 38 54]}  {[12 39 20]}  {[12 39 22 54]}  {[12 39 27 20]}  {[12 39 28 20]}  {[12 39 29 37 45]}  {[12 39 29 37 54]}  {[12 39 29 37 61 45]}  {[12 39 29 38 54]}  {[12 39 29 45]}  {[12 39 29 53 45]}  {[12 39 29 53 54]}  {[12 39 29 53 61 45]}  {[12 39 29 54]}  {[12 39 45]}  {[12 39 52 20]}  {[12 39 54]}  {[12 39 59 20]}  {[12 39 59 28 20]}  {[12 39 60 20]}  {[12 39 61 45]}  {[12 40 20]}  {[12 40 22 54]}  {[12 40 52 20]}  {[12 40 54]}  {[12 45]}  {[12 47 45]}  {[12 52 20]}  {[12 53 45]}  {[12 53 54]}  {[12 53 61 45]}  {[12 54]}  {[12 55 13 37 45]}  {[12 55 13 37 54]}  {[12 55 13 37 61 45]}  {[12 55 13 38 54]}  {[12 55 13 45]}  {[12 55 13 53 45]}  {[12 55 13 53 54]}  {[12 55 13 53 61 45]}  {[12 55 13 54]}  {[12 55 20]}  {[12 55 22 54]}  {[12 55 27 20]}  {[12 55 28 20]}  {[12 55 29 37 45]}  {[12 55 29 37 54]}  {[12 55 29 37 61 45]}  {[12 55 29 38 54]}  {[12 55 29 45]}  {[12 55 29 53 45]}  {[12 55 29 53 54]}  {[12 55 29 53 61 45]}  {[12 55 29 54]}  {[12 55 45]}  {[12 55 54]}  {[12 55 61 45]}  {[12 56 20]}  {[12 56 22 54]}  {[12 56 54]}

{[13 37 45]}  {[13 37 54]}  {[13 37 61 45]}  {[13 38 54]}  {[13 45]}  {[13 53 45]}  {[13 53 54]}  {[13 53 61 45]}  {[13 54]}

{[14 37 45]}  {[14 37 54]}  {[14 37 61 45]}  {[14 38 54]}  {[14 45]}  {[14 53 45]}  {[14 53 54]}  {[14 53 61 45]}  {[14 54]}

{[15 19 20]}  {[15 20]}  {[15 21 45]}  {[15 21 54]}  {[15 21 61 45]}  {[15 22 54]}  {[15 35 20]}  {[15 35 28 20]}  {[15 35 45]}  {[15 35 46 38 54]}  {[15 35 46 45]}  {[15 35 46 54]}  {[15 35 47 45]}  {[15 35 48 20]}  {[15 35 48 22 54]}  {[15 35 48 36 20]}  {[15 35 48 36 52 20]}  {[15 35 48 38 54]}  {[15 35 48 45]}  {[15 35 48 52 20]}  {[15 35 48 54]}  {[15 35 52 20]}  {[15 35 60 20]}  {[15 36 20]}  {[15 36 52 20]}  {[15 37 45]}  {[15 37 54]}  {[15 37 61 45]}  {[15 38 54]}  {[15 45]}  {[15 51 20]}  {[15 51 28 20]}  {[15 52 20]}  {[15 53 45]}  {[15 53 54]}  {[15 53 61 45]}  {[15 54]}

{[16 19 20]}  {[16 20]}  {[16 21 45]}  {[16 21 54]}  {[16 21 61 45]}  {[16 22 54]}  {[16 35 20]}  {[16 35 28 20]}  {[16 35 45]}  {[16 35 46 38 54]}  {[16 35 46 45]}  {[16 35 46 54]}  {[16 35 47 45]}  {[16 35 48 20]}  {[16 35 48 22 54]}  {[16 35 48 36 20]}  {[16 35 48 36 52 20]}  {[16 35 48 38 54]}  {[16 35 48 45]}  {[16 35 48 52 20]}  {[16 35 48 54]}  {[16 35 52 20]}  {[16 35 60 20]}  {[16 36 20]}  {[16 36 52 20]}  {[16 37 45]}  {[16 37 54]}  {[16 37 61 45]}  {[16 38 54]}  {[16 45]}  {[16 51 20]}  {[16 51 28 20]}  {[16 52 20]}  {[16 53 45]}  {[16 53 54]}  {[16 53 61 45]}  {[16 54]}

{[17 20]}  {[17 45]}  {[17 46 38 54]}  {[17 46 45]}  {[17 46 54]}  {[17 52 20]}  {[17 54]}  {[17 60 20]}  {[17 61 45]}  {[17 62 38 54]}  {[17 62 45]}  {[17 62 54]}

{[18 20]}  {[18 52 20]}  {[18 54]}

{[19 20]}

{[20]}

{[21 45]}  {[21 54]}  {[21 61 45]}

{[22 54]}

{[23 20]}  {[23 22 54]}  {[23 45]}  {[23 54]}  {[23 61 45]}

{[24 20]}  {[24 22 54]}  {[24 54]}

{[25 20]}  {[25 37 45]}  {[25 37 54]}  {[25 37 61 45]}  {[25 38 54]}  {[25 45]}  {[25 46 38 54]}  {[25 46 45]}  {[25 46 54]}  {[25 52 20]}  {[25 53 45]} {[25 53 54]}  {[25 53 61 45]}  {[25 54]}  {[25 60 20]}  {[25 61 45]}  {[25 62 38 54]}  {[25 62 45]}  {[25 62 54]}

{[26 20]}  {[26 37 45]}  {[26 37 54]}  {[26 37 61 45]}  {[26 38 54]}  {[26 45]}  {[26 52 20]}  {[26 53 45]}  {[26 53 54]}  {[26 53 61 45]}  {[26 54]} {[26 61 45]}

{[27 20]}

{[28 20]}

{[29 37 45]}  {[29 37 54]}  {[29 37 61 45]}  {[29 38 54]}  {[29 45]}  {[29 53 45]}  {[29 53 54]}  {[29 53 61 45]}  {[29 54]}

{[30 37 45]}  {[30 37 54]}  {[30 37 61 45]}  {[30 38 54]}  {[30 45]}  {[30 53 45]}  {[30 53 54]}  {[30 53 61 45]}  {[30 54]}

{[31 3 20]}  {[31 3 45]}  {[31 3 46 38 54]}  {[31 3 46 45]}  {[31 3 46 54]} {[31 3 47 45]}  {[31 3 48 20]}  {[31 3 48 22 54]}  {[31 3 48 36 20]}  {[31 3 48 36 52 20]}  {[31 3 48 38 54]}  {[31 3 48 45]}  {[31 3 48 52 20]}  {[31 3 48 54]}  {[31 3 52 20]}  {[31 3 60 20]}  {[31 4 20]}  {[31 4 52 20]}  {[31 5 45]}  {[31 5 54]}  {[31 5 61 45]}  {[31 6 54]}  {[31 19 20]}  {[31 20]} {[31 21 45]}  {[31 21 54]}  {[31 21 61 45]}  {[31 22 54]}  {[31 35 20]}  {[31 35 28 20]}  {[31 35 45]}  {[31 35 46 38 54]}  {[31 35 46 45]}  {[31 35 46 54]} {[31 35 47 45]}  {[31 35 48 20]}  {[31 35 48 22 54]}  {[31 35 48 36 20]}  {[31 35 48 36 52 20]}  {[31 35 48 38 54]}  {[31 35 48 45]}  {[31 35 48 52 20]}  {[31 35 48 54]}  {[31 35 52 20]}  {[31 35 60 20]}  {[31 36 20]}  {[31 36 52 20]} {[31 37 45]}  {[31 37 54]}  {[31 37 61 45]}  {[31 38 54]}  {[31 45]}  {[31 53 45]}  {[31 53 54]}  {[31 53 61 45]}  {[31 54]}

{[32 3 20]}  {[32 3 45]}  {[32 3 46 38 54]}  {[32 3 46 45]}  {[32 3 46 54]} {[32 3 47 45]}  {[32 3 48 20]}  {[32 3 48 22 54]}  {[32 3 48 36 20]}  {[32 3

48 36 52 20]}  {[32 3 48 38 54]}  {[32 3 48 45]}  {[32 3 48 52 20]}  {[32 3 48 54]}  {[32 3 52 20]}  {[32 3 60 20]}  {[32 4 20]}  {[32 4 52 20]}  {[32 5 45]}  {[32 5 54]}  {[32 5 61 45]}  {[32 6 54]}  {[32 19 20]}  {[32 20]}  {[32 21 45]}  {[32 21 54]}  {[32 21 61 45]}  {[32 22 54]}  {[32 35 20]}  {[32 35 28 20]}  {[32 35 45]}  {[32 35 46 38 54]}  {[32 35 46 45]}  {[32 35 46 54]}  {[32 35 47 45]}  {[32 35 48 20]}  {[32 35 48 22 54]}  {[32 35 48 36 20]}  {[32 35 48 36 52 20]}  {[32 35 48 38 54]}  {[32 35 48 45]}  {[32 35 48 52 20]}  {[32 35 48 54]}  {[32 35 52 20]}  {[32 35 60 20]}  {[32 36 20]}  {[32 36 52 20]}  {[32 37 45]}  {[32 37 54]}  {[32 37 61 45]}  {[32 38 54]}  {[32 45]}  {[32 53 45]}  {[32 53 54]}  {[32 53 61 45]}  {[32 54]}

{[33 20]}  {[33 28 20]}  {[33 45]}  {[33 46 38 54]}  {[33 46 45]}  {[33 46 54]}  {[33 47 45]}  {[33 48 20]}  {[33 48 22 54]}  {[33 48 36 20]}  {[33 48 36 52 20]}  {[33 48 38 54]}  {[33 48 45]}  {[33 48 52 20]}  {[33 48 54]}  {[33 52 20]}  {[33 54]}  {[33 60 20]}  {[33 61 45]}  {[33 62 38 54]}  {[33 62 45]}  {[33 62 54]}

{[34 20]}  {[34 52 20]}  {[34 54]}

{[35 20]}  {[35 28 20]}  {[35 45]}  {[35 46 38 54]}  {[35 46 45]}  {[35 46 54]}  {[35 47 45]}  {[35 48 20]}  {[35 48 22 54]}  {[35 48 36 20]}  {[35 48 36 52 20]}  {[35 48 38 54]}  {[35 48 45]}  {[35 48 52 20]}  {[35 48 54]}  {[35 52 20]}  {[35 60 20]}

{[36 20]}  {[36 52 20]}

{[37 45]}  {[37 54]}  {[37 61 45]}

{[38 54]}

{[39 20]}  {[39 22 54]}  {[39 27 20]}  {[39 28 20]}  {[39 29 37 45]}  {[39 29 37 54]}  {[39 29 37 61 45]}  {[39 29 38 54]}  {[39 29 45]}  {[39 29 53 45]}  {[39 29 53 54]}  {[39 29 53 61 45]}  {[39 29 54]}  {[39 45]}  {[39 52 20]}  {[39 54]}  {[39 59 20]}  {[39 59 28 20]}  {[39 60 20]}  {[39 61 45]}

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

{[41 20]} {[41 24 20]} {[41 24 22 54]} {[41 24 54]} {[41 28 20]} {[41 38 54]} {[41 40 20]} {[41 40 22 54]} {[41 40 52 20]} {[41 40 54]} {[41 45]} {[41 46 38 54]} {[41 46 45]} {[41 46 54]} {[41 47 45]} {[41 48 20]} {[41 48 22 54]} {[41 48 36 20]} {[41 48 36 52 20]} {[41 48 38 54]} {[41 48 45]} {[41 48 52 20]} {[41 48 54]} {[41 52 20]} {[41 54]} {[41 56 20]} {[41 56 22 54]} {[41 56 54]} {[41 60 20]} {[41 61 45]} {[41 62 38 54]} {[41 62 45]} {[41 62 54]}

{[42 20]} {[42 24 20]} {[42 24 22 54]} {[42 24 54]} {[42 38 54]} {[42 40 20]} {[42 40 22 54]} {[42 40 52 20]} {[42 40 54]} {[42 45]} {[42 47 45]} {[42 52 20]} {[42 54]} {[42 56 20]} {[42 56 22 54]} {[42 56 54]} {[42 61 45]}

{[43 20]} {[43 22 54]} {[43 24 20]} {[43 24 22 54]} {[43 24 54]} {[43 28 20]} {[43 38 54]} {[43 40 20]} {[43 40 22 54]} {[43 40 52 20]} {[43 40 54]} {[43 45]} {[43 46 38 54]} {[43 46 45]} {[43 46 54]} {[43 47 45]} {[43 48 20]} {[43 48 22 54]} {[43 48 36 20]} {[43 48 36 52 20]} {[43 48 38 54]} {[43 48 45]} {[43 48 52 20]} {[43 48 54]} {[43 52 20]} {[43 54]} {[43 56 20]} {[43 56 22 54]} {[43 56 54]} {[43 60 20]}

{[44 20]} {[44 22 54]} {[44 24 20]} {[44 24 22 54]} {[44 24 54]} {[44 38 54]} {[44 40 20]} {[44 40 22 54]} {[44 40 52 20]} {[44 40 54]} {[44 45]} {[44 47 45]} {[44 52 20]} {[44 54]} {[44 56 20]} {[44 56 22 54]} {[44 56 54]}

{[45]}

{[46 38 54]} {[46 45]} {[46 54]}

{[47 45]}

{[48 20]} {[48 22 54]} {[48 36 20]} {[48 36 52 20]} {[48 38 54]} {[48 45]} {[48 52 20]} {[48 54]}

{[49 20]} {[49 28 20]} {[49 45]} {[49 46 38 54]} {[49 46 45]} {[49 46 54]} {[49 52 20]} {[49 54]} {[49 60 20]} {[49 61 45]} {[49 62 38 54]} {[49 62 45]} {[49 62 54]}

{[50 20]}  {[50 52 20]}  {[50 54]}

{[51 20]}  {[51 28 20]}

{[52 20]}

{[53 45]}  {[53 54]}  {[53 61 45]}

{[54]}

{[55 13 37 45]}  {[55 13 37 54]}  {[55 13 37 61 45]}  {[55 13 38 54]}  {[55 13 45]}  {[55 13 53 45]}  {[55 13 53 54]}  {[55 13 53 61 45]}  {[55 13 54]} {[55 20]}  {[55 22 54]}  {[55 27 20]}  {[55 28 20]}  {[55 29 37 45]}  {[55 29 37 54]}  {[55 29 37 61 45]}  {[55 29 38 54]}  {[55 29 45]}  {[55 29 53 45]} {[55 29 53 54]}  {[55 29 53 61 45]}  {[55 29 54]}  {[55 45]}  {[55 54]}  {[55 61 45]}

{[56 20]}  {[56 22 54]}  {[56 54]}

{[57 20]}  {[57 28 20]}  {[57 38 54]}  {[57 45]}  {[57 46 38 54]}  {[57 46 45]}  {[57 46 54]}  {[57 52 20]}  {[57 54]}  {[57 60 20]}  {[57 61 45]} {[57 62 38 54]}  {[57 62 45]}  {[57 62 54]}

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

{[59 20]}  {[59 28 20]}

{[60 20]}

{[61 45]}

{[62 38 54]}  {[62 45]}  {[62 54]}

{[63 3 20]}  {[63 3 45]}  {[63 3 46 38 54]}  {[63 3 46 45]}  {[63 3 46 54]} {[63 3 47 45]}  {[63 3 48 20]}  {[63 3 48 22 54]}  {[63 3 48 36 20]}  {[63 3

48 36 52 20]}  {[63 3 48 38 54]}  {[63 3 48 45]}  {[63 3 48 52 20]}  {[63 3 48 54]}  {[63 3 52 20]}  {[63 3 60 20]}  {[63 4 20]}  {[63 4 52 20]}  {[63 5 45]}  {[63 5 54]}  {[63 5 61 45]}  {[63 6 54]}  {[63 13 37 45]}  {[63 13 37 54]}  {[63 13 37 61 45]}  {[63 13 38 54]}  {[63 13 45]}  {[63 13 53 45]}  {[63 13 53 54]}  {[63 13 53 61 45]}  {[63 13 54]}  {[63 19 20]}  {[63 20]}  {[63 21 45]}  {[63 21 54]}  {[63 21 61 45]}  {[63 22 54]}  {[63 27 20]}  {[63 28 20]}  {[63 45]}

{[64 3 20]}  {[64 3 45]}  {[64 3 46 38 54]}  {[64 3 46 45]}  {[64 3 46 54]}  {[64 3 47 45]}  {[64 3 48 20]}  {[64 3 48 22 54]}  {[64 3 48 36 20]}  {[64 3 48 36 52 20]}  {[64 3 48 38 54]}  {[64 3 48 45]}  {[64 3 48 52 20]}  {[64 3 48 54]}  {[64 3 52 20]}  {[64 3 60 20]}  {[64 4 20]}  {[64 4 52 20]}  {[64 5 45]}  {[64 5 54]}  {[64 5 61 45]}  {[64 6 54]}  {[64 13 37 45]}  {[64 13 37 54]}  {[64 13 37 61 45]}  {[64 13 38 54]}  {[64 13 45]}  {[64 13 53 45]}  {[64 13 53 54]}  {[64 13 53 61 45]}  {[64 13 54]}  {[64 19 20]}  {[64 20]}  {[64 21 45]}  {[64 21 54]}  {[64 21 61 45]}  {[64 22 54]}  {[64 27 20]}  {[64 36 20]}  {[64 36 52 20]}  {[64 38 54]}  {[64 45]}  {[64 54]}

All incomplete (not having an FP at end or cyclic) paths:

<No paths>

Elapsed time is 0.420618 seconds.