```
function [visitedNodes, queue, timeElapsed] =
 breadth_first_search(goalState, mode, prevQueue, prevVisitedNodes)
% This function realizes Breadth First Search algorithm.

% goalState" is a column vector respresenting goal configuration.

% "mode": can take values of 'single_step' or 'complete'
    ...'single_step': take one step and returns
    ...'complete': tries to solve the puzzle completely.

% "prevQueue": is the last snapshot of the queue.

% "prevVisitedNodes": is the last snapshot of the visitedNodes.

% INITIALIZE VARIABLES
visitedNodes = prevVisitedNodes; % It will be used to store visited
 nodes
queue = prevQueue;
numTiles = length(goalState); % Total number of tiles in the puzzle
timeElapsed = 0;

%Find the ID number to be assigned
if isempty(visitedNodes)
    idAssignedLast = max(queue(numTiles+1,:));
else
    idAssignedLast = max([visitedNodes(numTiles+1, :) queue(numTiles
+1,:)]);
end
idTobeAssigned = idAssignedLast + 1; % Update the id to be assigned to
 the next node


% MAIN LOOP
% Loop until the queue is empty
% Note also that when the goal state is discovered, the loop will be
 terminated (by an if-statement)
tic;
iIteration = 0;
while (~isempty(queue))

    % If the mode is 'single_step', then stop search after one
 iteration
    if strcmp(mode, 'single_step') && (iIteration == 1)
        return;
    end

    % Dequeue parentNode
    parentNode = queue(:,1);
    queue(:,1) = [];

    % Add parentNode into visitedNodes
    visitedNodes = [visitedNodes parentNode];
```

```
    % Investigate details of the parent node
    parentID = parentNode(numTiles+1);
    parentCost = parentNode(numTiles+3);

    % When the algorithm reaches the goal state, return
    if all(parentNode(1:numTiles) == goalState)
        timeElapsed = toc;
        return;
    end

    % Find successors of the parent
    successorStates = successors(parentNode(1:numTiles));

    % Add unvisited successors to the queue
    for iSuccessor = 1:size(successorStates,2)
        curSuccState = successorStates(:, iSuccessor);

        % If the successor has already been labelled, skip it.
        if any(ismember(curSuccState',
 visitedNodes(1:numTiles, :)', 'rows'))
            continue;
        end

        % If the successor is not in queue add it into the queue.
        if ~any(ismember(curSuccState',
 queue(1:numTiles, :)', 'rows'))
            curSuccNode = [curSuccState; idTobeAssigned; parentID;
 parentCost+1];
            idTobeAssigned = idTobeAssigned + 1; % Update the id to be
 assigned to the next node

            queue = [queue curSuccNode]; % Enqueue curSuccNode
            % Note that, there is no ordering operation on the queue
 since
            % it is quaranteed that each successor has a cost value
 that is
            % bigger than that of all previously visited nodes.

        end
    end
    iIteration = iIteration + 1;
end

% Issues an error since the queue is empty and a solution could not be
 obtained
error("The BFS algorithm could not find a solution.");
end
```

Published with MATLAB® R2017b