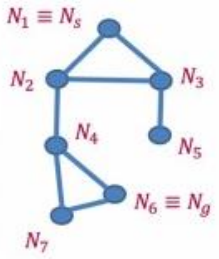
HOMEWORK n. 3



1. Given the graph above, implement a software program to search the graph with the bread-first algorithm (odd matriculation number) or with the depth-first algorithm (even matriculation number) from N_s to N_g. Briefly describe the obtained result.

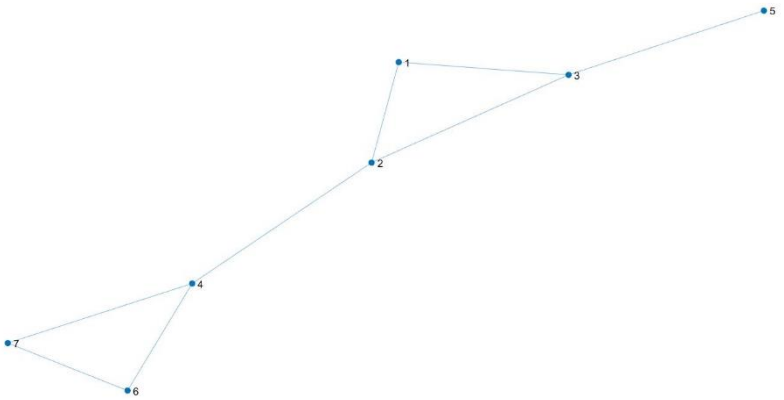
Depth-first search (DFS) is a search algorithm for tree or graph structures. The algorithm starts at the root node and explores as far as possible along each branch before backtracking.

```
clc
clear all
Matrix = [0 1 1 0 0 0 0
          1 0 1 1 0 0 0
          1 1 0 0 1 0 0
          0 1 0 0 0 1 1
          0 0 1 0 0 0 0
          0 0 0 1 0 0 1
          0 0 0 1 0 1 0];

Graph = graph(Matrix);
plot (Graph);
```

Adjacency Matrix of the graph

With these commands is possible to print the graph



```
OPEN = [1];
sequence = [];
node = 1;
father = [];
son = [];
j = 1;
i=1;
found=0;
pre(1) = 1;
t = 1;
```

OPEN initially contains only the node N_s and it is marked as visited

DFS uses the LIFO strategy through a stack called OPEN.

At the beginning, OPEN contains the node N_s only and it is marked as visited, and the other nodes are marked as unvisited. At each iteration, the last node in OPEN is extracted and all the connect nodes marked as unvisited are inserted into OPEN as visited

The search terminates once N_g has been found, or OPEN is empty (failure).

The explanation of the single instructions is shown in the figure.

```
while(isempty(OPEN)==false & found==0)%The algorithm ends when Ng has been found or when there are no more nodes to explore
    sequence(i) = OPEN(end);

    OPEN(end) = [];%this instruction deletes the last element of the stack
    for neighbor = 1:7
        if (Matrix(node,neighbor) == 1) %this instruction checks if the node and the neighbor are connected
            if visited(neighbor) == 0 %this instruction checks if the neighbor has already been visited
                OPEN = [OPEN neighbor];%if the neighbor has not yet been visited, the neighbor is added to OPEN
                visited(neighbor) = 1;%it indicates that the node has been visited
                father(j) = node;%father and son nodes are updated
                son(j)= neighbor;
                if (son(j)==goal_node)%if Ng has been reached, the sequence vector is updated and found is set equal to 1 so as to exit the while loop
                    found=1;
                    sequence(i+1)=son(j)
                     for w = 2:length(sequence)%with these 2 for loops I create the vector pre that is useful for the creation of the final graph
                        for p = 1:length(son)
                            if sequence(w) == son(p);
                                pre(t+1) = father(p);
                                t = t+1;
                            end
                        end
                    end

                    break
                end
                j = j+1;
            end
        else
            fprintf ('The %d neighbor has already been visited***\n', neighbor)
        end

    end
    if (isempty(OPEN)==false)
        node = OPEN(end) %node takes on the last value of the OPEN vector
        fprintf('\n The extracted node is %d   \n',OPEN(end))
    end

    i=i+1;
end
```

At the end, a message is printed to inform if the node N_g has been reached or not

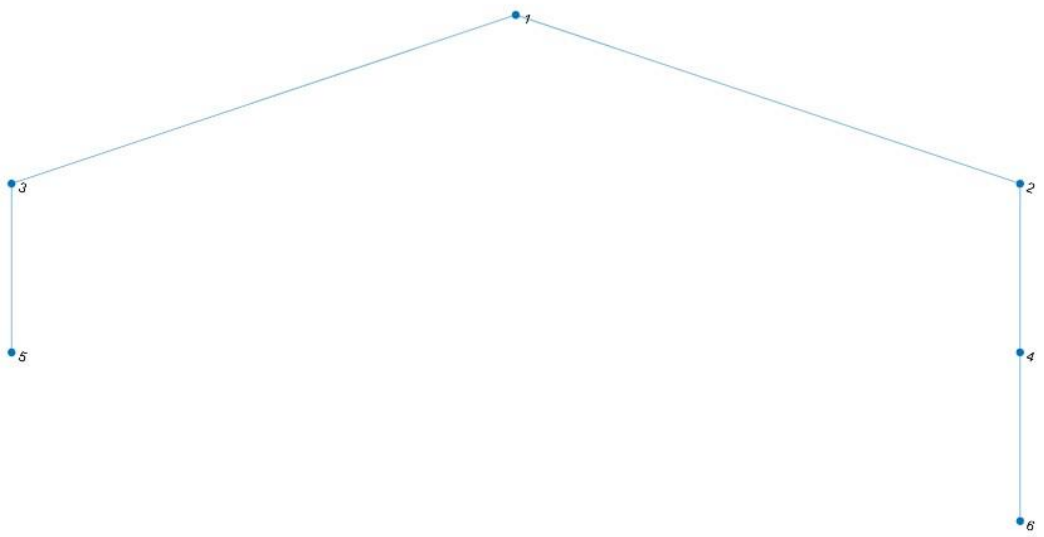
```
if(isempty(OPEN)==false)
    fprintf('The sequence of the elements is: \n')
    sequence
    DFS_graph = graph(sequence,pre,'omitselfloops');
    figure(2)
    plot(DFS_graph)
else
    fprintf('There is not any sequence to reach the final node')
end
```

```
The sequence of the elements is:

sequence =

     1     3     5     2     4     6
```

A graphical representation is shown below. The graph initially executes the left branch and then the right one.



The file containing the algorithm is called: "bread-first algorithm.m"