# Lab #9

## Contents

## 1 Highlighting vertices in a graph

When plotting a graph, it is possible to color certain vertices with another color. This can be achieved with `highlight` function as presented in the following example:

```
clear; close all;

A = [
    0 1 0 1;
    1 0 1 0;
    0 1 0 1;
    1 0 1 0
    ]; % Adjacency matrix

C = [1, 3]; % Vertices to highlight

figure;
G = graph(A);
h = plot(G);
```

```
highlight(h, C, 'NodeColor', 'r'); % 'r' for red, 'g'
    for green, 'b' for blue, etc.
```

## 2 Task #15

- Write a program which creates an edge list representation of an adjacency matrix. Test the program on several graphs from Homework 1.
- In the same program write code which converts the resulting edge list from part 1 of the task back to an adjacency matrix. Check that the original adjacency matrix and the resulting one are the same. Plot the graphs.

**Hint:** if you want to create a new matrix of the same size as an existing one and fill it with zeros, you can use the following code, where `A` is the existing matrix.

```
new_A = zeros(size(A));
```

## 3 Task #16

Write a program which for a given subset of vertices $I$ of the graph $G = (V, E)$ checks whether $I$ is an independent set of $G$. Plot the graph and highlight the vertices which are in the IS.

You can base your code on the following pseudocode that uses an adjancency matrix representation of $G$:

```
Algorithm 1: Test for independence
  Input  : Adjacency matrix $A$, set of vertices $I$
  Output: Boolean value $t$ (true if $I$ is an independent set and false
          otherwise)
Function is_independent_set ($A, I$)
    $t :=$ true;
    if $length(I) > 1$ then
        for $v_{index} = 1$ TO $length(I) - 1$ do
            $v := I[v_{index}]$;
            for $w_{index} = v_{index} + 1$ TO $length(I)$ do
                $w := I[w_{index}]$;
                if $A[v][w] == 1$ then
                    $t :=$ false;
                end
            end
        end
    end
    return $t$;
end
```

## 4 Task #17

Write a program which for a given graph $G = (V, E)$ finds a maximal Independent Set in it.

You can implement a simple greedy algorithm which is based on the idea of starting from an empty set $I$, picking vertices of $G$ one by one and adding them to $I$ if $I$ continues to be an independent set. The algorithm is described in more details in the following pseudocode:

**Algorithm 2:** Maximal Independent Set

```
Input  : Adjacency matrix $A$
Output: A subset of vertices $I$
1 $P$ := a random permutation of vertices $V$;
2 $I := \varnothing$;
3 for $i$ = 1 TO length(P) do
4     $T := I \cup P(i)$;
5     if $T$ is an Independent Set then
6         $I := I \cup P(i)$;
7     end
8 end
```

**Hint 1:** To find a random permutation of vertices of $G$ you can use the following command: `P = randperm(size(A, 1));`

**Hint 2:** You can use the routine from the task 16 to check whether a subset of vertices $T$ is an independent set

## 5 Task #18

Write a program that for a given undirected graph $G = (V, E)$ as an edge list and a subset of vertices $C$ of $V$ checks whether $C$ is a vertex cover of $G$. You may use the code from the task 15 to convert an adjacency matrix to an edge list or just write an edge list directly.

Plot the graph and highlight the vertices of the VC.

Example of an input is:

```
E = [
    1, 2;
    2, 3;
    4, 5;
    6, 7;
]; % edge list
C = [1, 2, 3, 4, 5, 6, 7]; % Subset of vertices of
   the vertex cover
```

To plot the graph using an edge list and color vertices from $C$ with a different color, you can use the following code:

```
G = graph(E(:, 1), E(:, 2));
```

```
h = plot(G);
highlight(h, C, 'NodeColor', 'g'); % 'g' - green, 'r'
    - red, 'b' - blue, etc
```

You can use the following pseudocode:

**Algorithm 3:** Test for a vertex cover

```
Input   : Edge list $E$, subset of vertices $C$
Output: True if $C$ is a vertex cover of $G$, false otherwise
1  $is\_vc$ := true;
2  for $i$ = 1 TO number of rows in $E$ do
3      u := E(i, 1);
4      v := E(i, 2);
5      covered := false;
6      for $j$ = 1 TO length(C) do
7          if $C(j) == u$ or $C(j) == v$ then
8              covered := true;
9          end
10     end
11     if not covered then
12         $is\_vc$ := false;
13     end
14 end
15 print($is\_vc$);
```