Lab #7

Contents

1 Edge lists warm-up

1.1 Storing edge lists

To store an edge list you can use an $n \times 2$ matrix (or $2 \times n$), where each row (or column) consists of two numbers: u and v of the $e = (u, v)$ edge. For example, the following code shows how you can store edge list [(1, 2), (2, 3), (2, 1), (3, 4)] in an $n \times 2$ matrix.

```
1 edge_list = [1, 2; 2, 3; 2, 1; 3, 4];
```

1.2 Adding new edges

In order to add a new edge, we need to use concatenation (see also Lab. 5 and MIT slides).

If we want to add a new edge (1, 4) to the edge list from the previous example, we can use the following code:

```
new_edge = [1, 4];
edge_list = [edge_list; new_edge];
```

2 Plotting graphs

You can plot a graph using `graph` or `digraph` and `plot` functions. Functions `graph` and `digraph` take adjacency matrix as an input. Command `figure` creates a new window with a figure to plot in it. Function `plot` draws a visual representation of the graph in the last opened figure window. Command `close all` closes all open figure windows and is often used in the beginning of a script along with `clear` command.

Try the following example:

```
clear; close all;

A = [
    0 1 0 1;
    1 0 1 0;
    0 1 0 1;
    1 0 1 0
    ];

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

Or with a directed graph:

```
clear; close all;

A = [
    0 1 0 1;
    0 0 1 0;
    0 1 0 0;
    0 1 1 0
    ];

```

```
figure;
G = digraph(A);
plot(G);
```

2.1 Plotting several graphs

If more than one `plot` command is called for one figure window, by default Matlab will display only the result of the last `plot` call. `figure` can be used to create more than one figure window.

For example:

```
clear; close all;

A = [
    0 1 0 1;
    0 0 1 0;
    0 1 0 0;
    0 1 1 0
    ];

figure;
G = digraph(A);
plot(G);


figure; % create a second window for the second graph
G2 = digraph(A'); % digraph of the transposed
   adjacency matrix
plot(G2);
```

Alternatively, a `pause` command can be used to plot graphs in the same window one by one. The `pause` command pauses the execution of the program until the user continues it by pressing Enter in the command window. For example:

```
clear; close all;

A = [
    0 1 0 1;
```

```
    0 0 1 0;
    0 1 0 0;
    0 1 1 0
    ];

figure;
G = digraph(A);
plot(G);

pause; % pause after plotting the first graph

G2 = digraph(A'); % digraph of the transposed
   adjacency matrix
plot(G2);
```

3 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));
```

4 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)

```
1  Function is_independent_set (A, I)
2      t := true;
3      if length(I) > 1 then
4          for v_index = 1 TO length(I)−1 do
5              v := I[v_index];
6              for w_index = v_index + 1 TO length(I) do
7                  w := I[w_index];
8                  if A[v][w] == 1 then
9                      t := false;
10                 end
11             end
12         end
13     end
14     return t;
15 end
```