Lab #8

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.

```
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. 6 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];
```

1.3 Plotting the graph

You can plot the graph using `digraph` and `plot` functions. The function `digraph` takes two vectors as an input. First vector consists of all u and the second vector consists of all v from the edge list. Function `plot` creates a visual representation of the graph. Try the following example:

```
clear;

u = [1 2 2 3 4];
v = [2 1 3 1 1];

G = digraph(u, v);
plot(G);
```

We can plot graph for the edge list from the previous examples using the following code:

```
G = digraph(edge_list(:, 1), edge_list(:, 2));
plot(G);
```

2 Task 12

- Write a program which creates an edge list representation of an adjacency matrix. Test the program on the graphs from Homework 1. Plot the graphs.
- In the same script, add code which creates a new adjacency matrix from the resulting edge list. Check that the original adjacency matrix and the new one are the same.

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 some matrix.

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

3 Task 12.5

Write a program which for given two numbers m and n constructs an $m \times n$ matrix, where for any row i and column j the element at i-th row and j-th column has the value of $i + j$. Try your code by creating a 200×100 matrix.

For example, for the input values $m = 4$ and $n = 3$ the resulting 4×3 matrix should be

$$\begin{bmatrix} (1+1) & (1+2) & (1+3) \\ (2+1) & (2+2) & (2+3) \\ (3+1) & (3+2) & (3+3) \\ (4+1) & (4+2) & (4+3) \end{bmatrix} = \begin{bmatrix} 2 & 3 & 4 \\ 3 & 4 & 5 \\ 4 & 5 & 6 \\ 5 & 6 & 7 \end{bmatrix} \tag{1}$$

Hint: you can start your program using the code below. Then you can use nested for loops to iterate through all rows and columns of the matrix to assign correct values one by one.

```
clear;

m = 200;
n = 100;
A = zeros(m, n);
```

4 Task 13

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.

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
```