Lab. 09 - Graph Problems (1)

10.1 Graph Reading

Consider the undirected weighted graphs specified in the zip file **graphs.zip**, with the following DIMACS format

```
nn na
ni nj wij
```

where the first line indicates the number **nn** of nodes and the number **na** of arcs, and the subsequent lines specify all the **na** arcs, each by a triple `<ni, nj, wij>` where **ni** and **nj** are the node identifiers and **wij** the weight of the connecting arc. Note that the graphs are undirected, and so for any arc between nodes **i** and **j** there is an arc between nodes **j** and **i** with the same weight.

Specify a function with signature

```
def dimacs_read(filename)
```

that reads a graph with the above DIMACS format from a file with name filename, and returns the adjacency matrix A of the represented graph.

Note: The files format assume that the nodes are numbered from **1** no **nn**.

10.2 Graph Writing

Consider an undirected graph specified by its adjacency matrix A. Specify a function with signature

```
def dimacs_write(A, fname)
```

that prints the graph **M** in a file with the given **filename**, with the DIMACS format (see above). Test this function with the graphs obtained in the previous question.

10.3 Graph Encodings

a) Implement function with signature

```
def matrix_to_dicts_convert(A)
```

that, for a given **A**, the adjacency matrix encoding of a graph, returns its dictionary list encoding.

b) Implement function with signature

```
def dicts_to_matrix_convert(D)
```

that, for a given **D**, the dictionary list encoding of a graph, returns its adjacency matrix encoding.

Test your codes with the matrix shown in the slides of class 9.

10.4 Subgraph Projection

Consider an undirected graph specified by its adjacency matrix **A**. Implement a function with signature

```
def subgraph_projection(A, Nodes)
```

that returns the adjacency matrix **S** of the subgraph of **A** obtained by its projection to the list **Nodes.**

Note 1: Notice that matrix **S** of the subgraph should have the nodes numbered from **0**. If **A** encodes a graph with nodes **0..9**, and **Nodes** is **[2,5,7]** than **S** has **3** rows and columns and **Nodes** is a mapping from the new nodes to the original ones (i.e. nodes **0/1/2** of **S** are the original nodes **2/5/7** of **A**).

Note 2: When the **n** nodes of a graph have labels different from the natural numbers **0..n-1**, than the graph should be represented by a pair **G = (A,M)** where **A** is an adjacency matrix and **M** the mapping of the nodes to their intended labels.

10.5 Connected Components (1)

Consider an undirected graph specified by its adjacency matrix **A**. Specify a function with signature

```
def connected_with(k,A)
```

that returns the connected component **C** that includes k, i.e. a list of the nodes of the graph that are connected to node k (including k). Test this function with graphs G1 and G2 given in the theory class (slides 11 and 12)

Note 1: Adapt function `connected` from the slides of class 9 to obtain the nodes of the sub-graph that are connected to k.

10.6 Connected Components (2)

Consider an undirected graph specified by its adjacency matrix **A**. Specify a function with signature

```
def connected_components(A)
```

that returns a list, **Cs**, with is a partition of the nodes of **A** corresponding to its connected components.

Suggestion: Iterate the previous function, with nodes left out from previous connected components.