GRAPH MATCHING IN STRUCTURAL PATTERN RECOGNITION

BIPARTITE ALGORITHM IMPLEMENTATION

- Before implementing the algorithm, the following points must be covered:
    - Create necessary structures to store a graph (vertices, edges and their attributes)
    - Create a parser to extract graph related information from gxl files
    - Compute vertices and edges cost matrices

You will be provided with the graph database (gxl files) and the cost functions

Validation:

- ✓ To validate your parser, load few instances of graph into objects and then re-write them into files. Compare the files and make sure they look the same
- ✓ Verify manually or using a function the computed costs for some vertices and edges

- BP algorithm: Refer to the course (p. 50-54) to understand the workflow of BP.

You will need to use the Hungarian algorithm to solve the LSAP problem, you can see this version or any other appropriate implementation.

Validation:

- ✓ Validate the assignment computed by the Hungarian algorithm:
    - The sum of all values on a line (and column) must be equal to 1

- Transform the solution to a GED solution.
    - Determine edges edit operations based on the obtained vertices operations
    - Compute the actual cost of the GED solution
- Save all the necessary information (e.g. graph instance, time spent, cost, edit operations, …)

Validation:

- ✓ Make sure that a vertex is assigned to only one vertex (or deleted/inserted)
- ✓ Make sure that an edge is assigned to only one edge (or deleted/inserted)

TEST BP OVER A DATABASE

- Use your algorithm and run it over all instances of the given database.
    - CMUHouse DB is provided
    - A file indicating the pair of graphs to be compared is provided
- Save all the results in a format of your choice

Validation:

- ✓ Make sure the total number of compared instances is 660 and,
- ✓ The same pair of graphs are selected as indicated in the provided file.

VISUALIZING THE MATCHING

- Knowing the ground-truth matching, you need to compare your results/matching with them

    - Please refer to this page to get an idea of the expected results
- Steps to get there:
    - You will be using ImageJ library (provided)
    - Display images side to side
    - Draw the graphs over the images
    - Display the matching
    - Display the distances (real one, computed one, time, …)
    - Save the results
- Display the obtained images sequentially.

COMMENTS AND OBSERVATIONS

- Compare your output with the online one, any comments?
- Repeat the test on the randomized version of the database and visualize the matching. What are your observations?

Validation:

- ✓ To be discussed!