UNIVERSITÄT
BERN

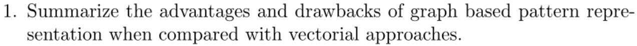
TA Session 1

Anthony Gillioz
Institute of Computer Science – University of Bern, Switzerland

Contact: anthony.gillioz@unibe.ch

Theoretical Tasks

Task 1

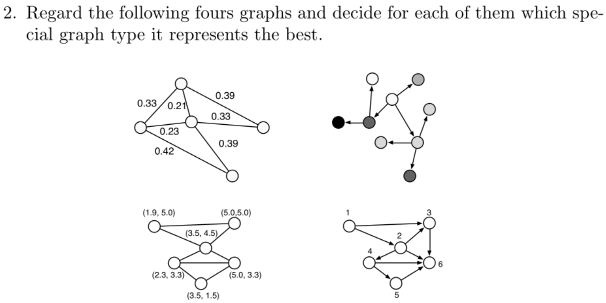


1. Summarize the advantages and drawbacks of graph based pattern representation when compared with vectorial approaches.

- You have to write a two-column table.
- The first column should contain at least two advantages, while the second column should contain at least two disadvantages.

Theoretical Tasks

Task 2

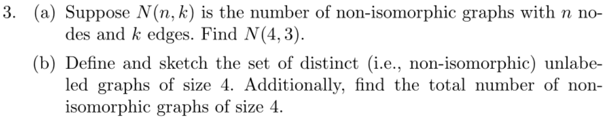


2. Regard the following fours graphs and decide for each of them which special graph type it represents the best.

- Use Fig. 1.9 (in the lecture notes) to find which graph type (labeled, unlabeled, directed, ...) corresponds to or would suit the most the following graphs.

Theoretical Tasks

Task 3

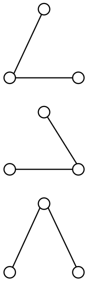


3. (a) Suppose $N(n, k)$ is the number of non-isomorphic graphs with n nodes and k edges. Find $N(4, 3)$.

   (b) Define and sketch the set of distinct (i.e., non-isomorphic) unlabeled graphs of size 4. Additionally, find the total number of non-isomorphic graphs of size 4.

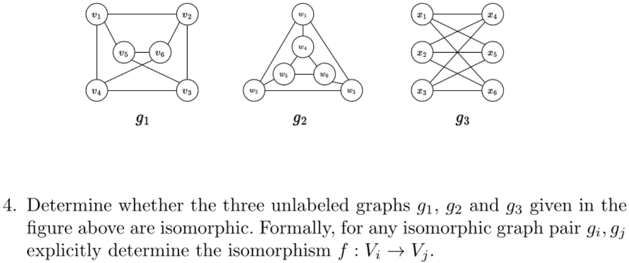
Example with n=3 and k=2

$N(3, 2) = 1$



Theoretical Tasks

Task 4



4. Determine whether the three unlabeled graphs g_1, g_2 and g_3 given in the figure above are isomorphic. Formally, for any isomorphic graph pair g_i, g_j explicitly determine the isomorphism $f : V_i \rightarrow V_j$.

- A 3×3 matrix with 1 in the entry if two graphs are isomorphic 0 otherwise.
- example of explicit isomorphism

$$f(v_1) = w_3$$

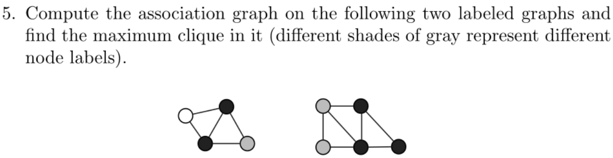
$$f(v_2) = w_5$$

$$\dots$$

$$f(v_6) = \dots$$

Theoretical Tasks

Task 5

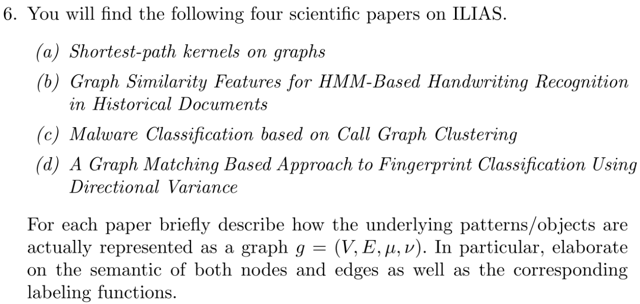


5. Compute the association graph on the following two labeled graphs and find the maximum clique in it (different shades of gray represent different node labels).

- Each color is a label.
- Define an ID for each node.
- Use Def. 14 (in the lecture notes) to create the association graph.
- Hint: Don't forget that the edge structure is still maintained if there is no edge from u_1 to v_1 in g_1 and no edge from u_2 to v_2 in $g2$. Thus there is an edge between (u_1, u_2) and (v_1, v_2) in the association graph.

Theoretical Tasks

Task 6



6. You will find the following four scientific papers on ILIAS.

   (a) *Shortest-path kernels on graphs*

   (b) *Graph Similarity Features for HMM-Based Handwriting Recognition in Historical Documents*

   (c) *Malware Classification based on Call Graph Clustering*

   (d) *A Graph Matching Based Approach to Fingerprint Classification Using Directional Variance*

   For each paper briefly describe how the underlying patterns/objects are actually represented as a graph $g = (V, E, \mu, \nu)$. In particular, elaborate on the semantic of both nodes and edges as well as the corresponding labeling functions.

- You do not need to read the complete paper
- Find the relevant parts, where the authors describe the graph extraction and/or graph representation
- A suitable way to present your submission could be in the form of a table or a similar format.

Implementation Tasks

In this implementation task, the goal is to implement Ullman's subgraph isomorphism test presented in the lecture notes (Alg. 2).

Remarks:

- You have the option to either implement Ullman's algorithm as presented in the lecture notes or the version presented on this page[1] for this exercise.

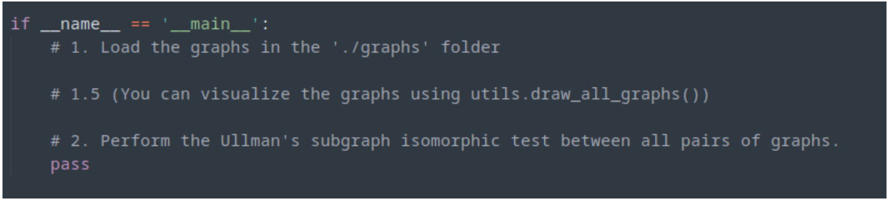
[1] https://adriann.github.io/Ullman%20subgraph%20isomorphism.html

Implementation Tasks

Where to code

Remarks

- The entire code must be contained within the file `PR_lecture/Exercise_1/ex1.py`
- You are allowed to modify the code as much as you want, including changing function signatures, creating new functions or classes, and so on.



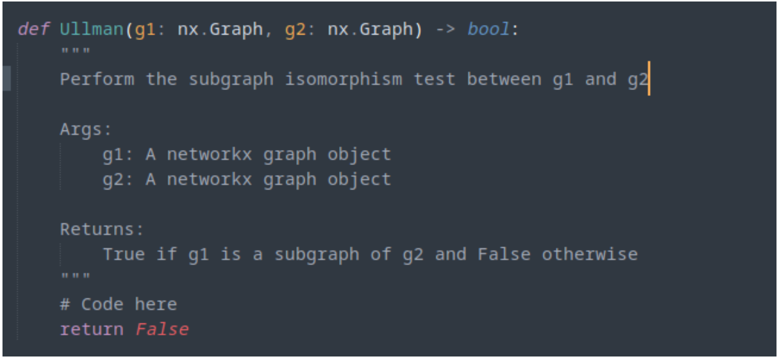
```python
if __name__ == '__main__':
    # 1. Load the graphs in the './graphs' folder

    # 1.5 (You can visualize the graphs using utils.draw_all_graphs())

    # 2. Perform the Ullman's subgraph isomorphic test between all pairs of graphs.
    pass
```

Implementation Tasks

Idea of code structure



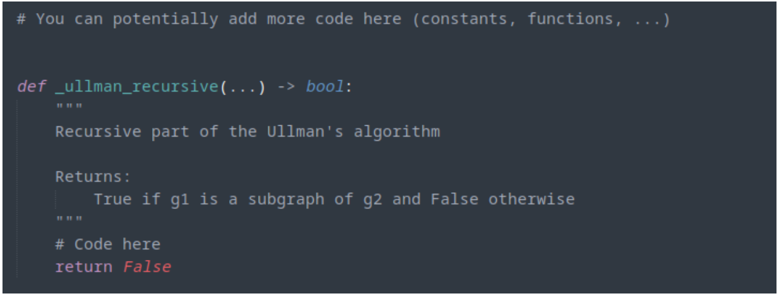
```
def Ullman(g1: nx.Graph, g2: nx.Graph) -> bool:
    """
    Perform the subgraph isomorphism test between g1 and g2

    Args:
        g1: A networkx graph object
        g2: A networkx graph object

    Returns:
        True if g1 is a subgraph of g2 and False otherwise
    """
    # Code here
    return False
```

Implementation Tasks

Idea of code structure



```python
# You can potentially add more code here (constants, functions, ...)


def _ullman_recursive(...) -> bool:
    """
    Recursive part of the Ullman's algorithm

    Returns:
        True if g1 is a subgraph of g2 and False otherwise
    """
    # Code here
    return False
```

Final remarks

- The file you upload on ILIAS as a `.zip` should be formatted in the following way: `exercise_1_firstname_lastname.zip`.