392013 Mandatory Exercises Algorithmic Cheminformatics

Submission deadline: June 15th, 2026, 23:55

1 Canonicalisation / Equitable Partitions

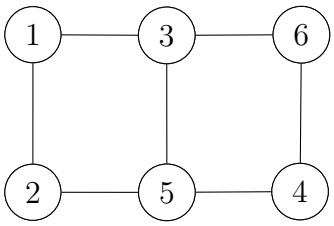
Given the following graphs, for each of them execute the equitable refinement procedure as decribed in

McKay's Canonical Graph Labeling Algorithm, Contemporary Mathematics, Stephen G. Hartke and A. J. Radcliffe, In Communicating Mathematics, volume 479 of Contemporary Mathematics, pages 99-111. American Mathematical Society, (2009).

The article can be found here.

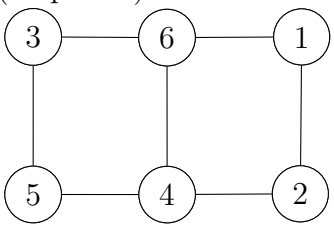
For each of the following (unlabelled) graphs give the resulting equitable partition. Note, the ids in the vertices are only representational ids, the graphs themselves are unlabelled.

a) (10 points)



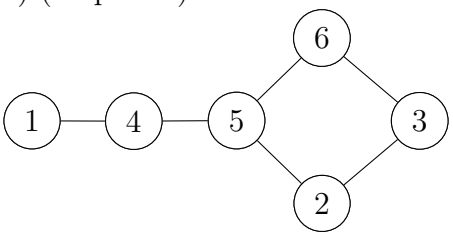
Your answer:

b) (10 points)



Your answer:

c) (10 points)



Your answer:

2 Ullmann algorithm

Given the following (unlabelled) graphs G_B and G_A.

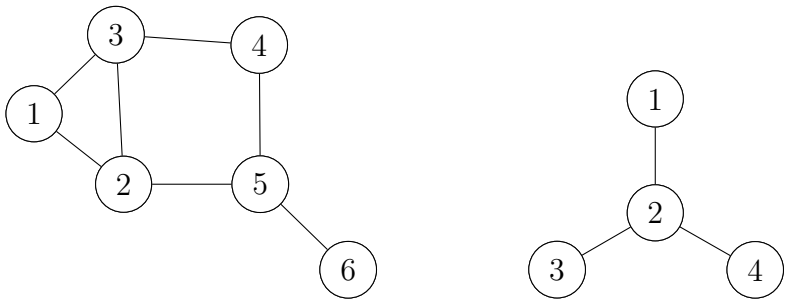


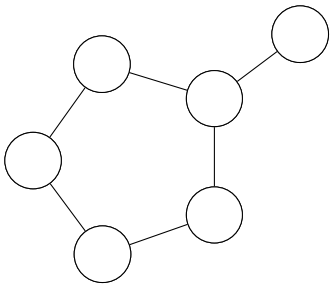
Figure 1: graph G_B (left) and graph G_A (right)

a) (10 points)
Give the adjacency matrices A and B for G_A and G_B.
Your answer:

b) (20 points)
b1) What dimension (rows x columns) will a matrix P (as described in the Ullmann algorithm) have, which acts as a certificate that G_A can be found as a subgraph isomorphism in G_B. b2) To which node in graph G_B will node 2 of graph G_A be mapped to? b3) How often can G_A be found as a subgraph isomorphism (i.e., as induced subgraph) in G_A? b4) How often can G_A be found as a monomorphism (i.e., a non-induced subgraph) in G_A?
Your answer:

3 Morgan algorithm

Given the following (unlabelled) graph G.



(20 points)
Execute Morgan's algorithm for at least 4 iterations: For each iteration draw the graph with integers in the nodes, and specify the size of the EC classes after each iteration).
Your answer:

4 Graph Transformations algorithm

From the first mandatory programming assignment, execute
`mod -f -i dgStrict.py` in order to create a chemical space for the pentose phosphate pathway. Note, the opion `-i` gives you an interactive shell before creation of the pdf.

a) (5 points)
How many vertices does the derivation graph have?
Your answer:

b) (5 points)
How many hyperedges does the derivation graph have?
Your answer:

c) (10 points)
Use the same derivation graph to find an optimal flow. Let `f` be an optimal flow. For each hyperedge having a non-zero flow, list in the below table the flow on the hyperedge. Hint: you can use the following code fragment to get the information (assuming `f` is your optimal flow object):

```
for e in dg.edges:
    flowOnEdge = f.eval(edge[e])
    if flowOnEdge != 0 : print(e, flowOnEdge)
```

which gives as an output (here, the first line only as an example):
`HyperEdge(DG(0), { 1 }, { 2 }) 2`

Your answer

Hyperedge	**Flow**
({ 1 }, { 2 })	2

Comment (not directly relevant for this exercise): Similarly to getting information on flows on hyperedges, you can get the information for flows on vertices as follows:

```
for v in dg.vertices:
    flowOnEdge = f.eval(vertex[v])
    if flowOnVertex != 0 : print(v, flowOnVertex)
```

resulting in

```
DGVertex(DG(0), 0) 1
DGVertex(DG(0), 1) 6
DGVertex(DG(0), 2) 2
DGVertex(DG(0), 4) 2
DGVertex(DG(0), 5) 4
DGVertex(DG(0), 7) 5
DGVertex(DG(0), 8) 2
DGVertex(DG(0), 28) 1
DGVertex(DG(0), 32) 1
DGVertex(DG(0), 34) 1
```