AIMS DMG Exercises 14
AIMS 2013-14: Designs, Matroids and Graphs

Rob Beezer
University of Puget Sound

Nancy Neudauer
Pacific University

January 14, 2014

Exercise 1. The Petersen graph is not planar. Use the Sage `.is_planar()` method to verify this. Now use Sage to assist you with an application of Kuratowski's Theorem.

1. Create a subgraph of the Petersen graph using Sage commands to delete edges and vertices.
2. Use a list comprehension to create a list of all the vertices of this subgraph which have degree 2.
3. Merge each degree 2 vertex of the subgraph with one of its neighbors using the `.merge_vertices()` method. Use `.neighbors()` to locate a neighbor.
4. Check that your result is isomorphic to $K_{3,3}$ using the Sage `.is_isomorpic()` method.
5. Start over with a fresh copy of the Petersen graph and demonstrate that the Petersen graph is contractible to K_5. You should be able to do this in one cell with a minimum of code.

Automate as much of the work as you can with Sage, paying special attention to Steps 2 and 3 above. In other words, some of the steps can be done simultaneously with commands, though you will still need to do some things by hand. Put Steps 1 through 5 each in their own Sage cell and do not use more than five cells for your solution. Plot your subgraph as the final command for Steps 1 and 3.

Full marks for a successful creation and verification of a subgraph isomorphic to $K_{3,3}$ and verification of being contractible to K_5. Some marks will be reserved for clean code that makes the best use of Sage and Python.

Exercise 2. Create the planar Circular Ladder graph on 10 vertices with `G = graphs.CircularLadder(5)`. The dual graph has vertices that are the faces of the planar graph, and edges join faces that are adjacent, and will thus be planar also.

Run the cell below to create a function that will build the dual graph. Do not edit this code. Note that the function returns two things — the dual graph, plus a Python dictionary which associates each vertex of the dual with a face of the original.

```
def dual_graph(G):
    r"""Returns a pair with (1) the dual graph, and
    (2) a dictionary of vertices of the dual giving faces of original"""
    if not(G.is_planar(set_embedding=True, set_pos=True)):
        raise ValueError( "input graph is not planar" )
    faces = G.trace_faces(G.get_embedding())
```

```
    edges = G.edges(labels=False)
    dual_edges = []
    for e in edges:
        right_face = [f for f in faces if e in f][0]
        # oriented the other way
        flipped = (e[1],e[0])
        left_face = [f for f in faces if flipped in f][0]graph.
        dual_edges.append((faces.index(left_face), faces.index(right_face)))
    D = Graph(dual_edges)
    if not(D.is_planar(set_embedding=True, set_pos=True)):
        raise RuntimeError('the dual of a planar graph is not planar.  Hmm...')
    # translation dictionary
    translation = {}
    for f in faces:
        circuit = [e[0] for e in f]
        translation[faces.index(f)] = circuit
    return D, translation
```

1. Compute some cycles in `G` using the `.cycle_basis()` method. Each edge of a cycle will have a face on either side, and so will correspond with an edge in the dual. Describe the set of such edges in the dual for a cycle of the orginal. You can create more cycles in the original graph by combining two or more elements of the cycle basis.

2. Test your description from the previous step by using Sage functions to obtain the objects you have described for the dual graph and exploring their corresponding objects in the original graph.

3. Create the cycle matroid of the original graph and ask for its circuits. What do you observe?

4. Create the dual of this matroid and again ask for its circuits. What do you observe?

This problem is very vague, because to say more would spoil the fun. Explore!