LARSON—MATH 750—SAGE WORKSHEET 06
Mobius Functions and Mobius Inversion

1. Create a Sage/Cocalc account.
   (a) Start the Chrome browser.
   (b) Go to http://cocalc.com
   (c) Login. You should see an existing Project for our class. Click on that.
   (d) Click “New”, then “Worksheets”, then call it s06.

The goal of today’s lab is to investigate the built-in mobius function poset methods in Sage, and run some experiments on Mobius inversion.

We have 3 posets now to experiment with: $P_1$, the poset $(\mathcal{P}(\{3\}), \subseteq)$ of the subsets of $\{3\}$ with inclusion; $P_2$, the poset $(\{10\}, |)$ of the integers $\{10\}$ with the divisibility relation; and $P_3$, the poset of all connected subgraphs of the cycle on 5 vertices.

2. Instead of regenerating these I’ve put the initialization code in the file posets.sage in your Handouts folder. Copy that to your main folder (the Handouts version will change when I change it. The version in your main directory will only change when you change it).

3. In your s06 worksheet, run: load('posets.sage').

4. Run: $P_1$ to check that you have a poset object with that name. Repeat for $P_2$ and $P_3$.

5. For any poset $P$ we can not only view the poset in CoCalc but we can also get the latex code to put this diagram in our homework, research papers, etc. Run: \texttt{latex(P)} for each of our posets. Copy the output into a LaTeX document. You will need to add the line

\usepackage{tikz}

\begin{tikzpicture}
\end{tikzpicture}

to the header of your LaTeX source.

6. This file also contains definitions for $L_1$, $L_2$ and $L_3$, the default linear extensions of these posets. Run $L_1$, $L_2$, and $L_3$ to confirm.

7. Last week we defined and tested the following code to calculate the mobius function of a pair of elements $x$ and $y$ of a poset $P$. Test it for our 3 posets.

\begin{verbatim}
def mobius(P, x, y):
    return P.moebius_function(x, y)
\end{verbatim}
8. We proved that the mobius function of a power set with inclusion is given by a simple formula. Try it for a few sets to see how it works then use the following script that will test it for ever subset in \( P_1 \).

```python
def mobius_power_set_theorem(A, B):
    if not A.issubset(B):
        return 0
    else:
        return (-1)^(B.cardinality()-A.cardinality())

for A in L1:
    for B in L1:
        print A, B, conjectured_mobius(A,B), mobius(P1, A, B)
```

9. Our goal is to harness the power of mobius inversion to find estimates of the independence number of a graph in terms of subgraphs of various types. Here’s our independence number function. Try it for a few graphs. Following that is a function that will calculate the independence number of a subgraph of a graph \( g \) defined by specific edges.

```python
def independence_number(g):
    return g.independent_set(value_only = True)

def alpha(g, x):
    return independence_number(g.subgraph(vertices=vertices(x),edges=x))
```

10. Here is a function that calculates the \( G \) defined in class.

```python
def mobius_G(g, P, F, x):
    return sum([F(g,y) for y in P if P.is_lequal(y,x)])

Run mobius_G(c5,P3, alpha, Set([(3,4)])) to try it. Test it on other edge sets.

11. Here is a function that calculates the mobius inversion of a given function \( F \), for a given graph \( g \) and graph poset \( P \) and set of edges \( x \), followed by code that computes this for our \( \alpha \) function on (the connected subgraphs defined by) every subset of edges of \( c_5 \).

```python
def graph_mobius_inversion(g, P, F, x):
    return sum([P.moebius_function(y,x)*mobius_G(g, P, F, y) for y in P])

for x in P3:
    print x, alpha(c5,x),graph_mobius_inversion(c5,P3,alpha,x)
```

12. Now experiment with these functions for other (small) graph. You’ll have to define the graph \( g \) and corresponding poset yourself, imitating what we’ve done.