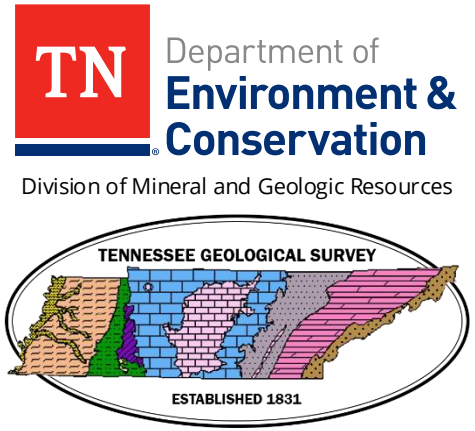
DIGITAL MAPPING TECHNIQUES 2025

The following was presented at DMT'25
May 18 - 21, 2025

The contents of this document are provisional

See Presentations and Proceedings
from the DMT Meetings (1997-2025)
http://ngmdb.usgs.gov/info/dmt/



Graph Theory: A Method for Visualizing Complex Stratigraphic Relationships

Leveraging *NetworkX* in Python to effectively sort map units

Andrew L. Wunderlich

DMT'25 Norman, OK 5/21/2025

Presentation Overview

- Define the problem
  - Stratigraphic unit sorting for a few (or a few hundred!) maps
- Graph theory
  - What is it and how can we apply it to this problem?
- Input data
  - Creating the needed dataset for a script to process
- Python script
  - Leveraging components *Pandas*, *NetworkX*, and *pydot*
- Interpret results
  - Analyze output, make corrections, iterate the process
- Conclusions

Defining the Problem

- Collection of newly created GeMS databases; mostly complete
  - Pros: DMU tables include MapUnit, Description, and Age
  - Cons: **NO** HierarchyKey so tables can't be sorted stratigraphically; relationships not preserved

- Inventory of map units compiled from (almost all) published maps
  - Pros: Comprehensive and Hierarchically ordered (see Cons)
  - Cons: Issues with logical consistency, 1000s of entries = errors exist

How can units be sorted CONSISTENTLY, across ALL databases?

Defining the Problem

Quad Name	Abbreviation	Group, Formation or Member	Age - Period
Adairville	Qal	Alluvium	Quaternary
Adairville	Msg	Ste. Genevieve Limestone	Upper Mississippian
Adairville	Msl	St. Louis Limestone	Upper Mississippian
Adams	Msg	Ste. Genevieve Limestone	Upper Mississippian
Adams	Msl	St. Louis Limestone	Upper Mississippian
Adams	Mw	Warsaw Formation	Upper Mississippian
Adolphus	Qal	Alluvium	Quaternary
Adolphus	Msl	St. Louis Limestone	Upper Mississippian
Adolphus	Msw	Warsaw	Upper Mississippian
Adolphus	Mfp	Fort Payne	Lower Mississippian
Adolphus	as	Argillaceous Siltstone	Lower Mississippian
Adolphus	Dc	Chattanooga Shale	Upper Devonian
Albany	Qal	Alluvium	Quaternary
Albany	Plsh	Shale member of the Lee Formation	Lower Pennsylvanian
Albany	Mp	Pennington Formation	Mississippian
Albany	Mb	Bangor Limestone	Mississippian
Albany	Mh	Hartselle Sandstone	Mississippian
Albany	Mmu	Mounteagle Limestone Upper limestone member	Mississippian
Albany	Mmg	Mounteagle Limestone Ste. Genevieve limestone member	Mississippian
Albany	Msl	St. Louis Limestone	Mississippian
Albany	Msw	Salem and Warsaw Formation	Mississippian
Albany	Mfp	Fort Payne Formation	Mississippian
Albany	rl	reef limestone	Mississippian
Alexandria	Mfp	Fort Payne Formation and Chattanooga Shale	Lower Mississippian and Upper Devonian
Alexandria	Olcy	Leipers and Catheys Formations	Upper Ordovician
Alexandria	Obc	Bigby-Cannon Limestone	Ordovician
Alexandria	Oh	Hermitage Formation	Ordovician
Alexandria	Oc	Carters Limestone	Ordovician
Alexandria	Olb	Lebanon Limestone	Middle Ordovician
Allensville	Qal	Alluvium	Quaternary
Allensville	Msg	Ste. Genevieve Limestone	Mississippian
Allensville	Msl	St. Louis Limestone	Mississippian
Alpine	Pf	Fentress Formation	Pennsylvanian
Alpine	Mp	Pennington Formation	Mississippian
Alpine	Mb	Bangor Limestone	Mississippian
Alpine	Mh	Hartselle Formation	Mississippian
Alpine	Mm	Monteagle Limestone	Mississippian
Alpine	Msl	St. Louis Limestone	Mississippian
Alpine	Mw	Warsaw Formation	Mississippian
Alpine	Mfp	Fort Payne Formation	Mississippian
Altamont	Qal	Alluvium	Quaternary
Altamont	Pw	Whitwell Shale	Lower Pennsylvanian
Altamont	Ps	Sewanee Conglomerate	Lower Pennsylvanian
Altamont	Pwp	Warren Point Sandstone	Lower Pennsylvanian
Altamont	Pra	Raccoon Mountain Formation	Lower Pennsylvanian
Altamont	Mpu	Pennington Formation Upper Member	Mississippian
Altamont	Mpl	Pennington Formation Lower Member	Mississippian
Altamont	Mb	Bangor Limestone	Upper Mississippian
Altamont	Mh	Hartselle Formation	Upper Mississippian
Altamont	Mm	Monteagle Limestone	Upper Mississippian

MapUnits:

Qal	Msg	Qal	Qal	Mfp	Qal	Pf	Qal
Msg	Msl	Msl	Plsh	Olcy	Msg	Mp	Pw
Msl	Mw	Msw	Mp	Obc	Msl	Mb	Ps
		Mfp	Mb	Oh		Mh	Pwp
		as	Mh	Oc		Mm	Pra
		Dc	Mmu	Olb		Msl	Mpu
			Mmg			Mw	Mpl
			Msl			Mfp	Mb
			Msw				Mh
			Mfp				Mm
			rl				

MapUnits "stratified":

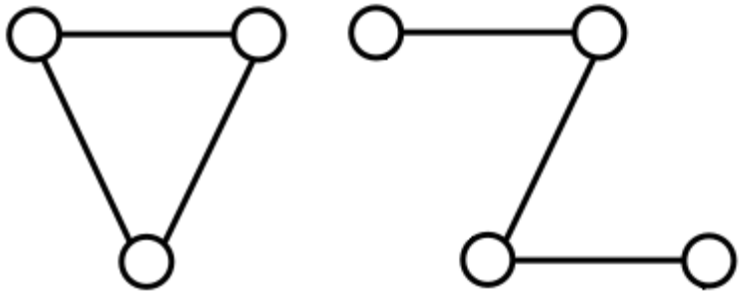
Qal		Qal	Qal		Qal		Qal
			Plsh			Pf	Pw
							Ps
							Pwp
							Pra
			Mp			Mp	Mpu
							Mpl
			Mb			Mb	Mb
			Mh			Mh	Mh
						Mm	Mm
			Mmu				
			Mmg				
Msg	Msg				Msg		
Msl	Msl	Msl	Msl		Msl	Msl	
		Msw	Msw				
	Mw					Mw	
		Mfp	Mfp	Mfp		Mfp	
		as	rl				
		Dc					
				Olcy			
				Obc			
				Oh			
				Oc			
				Olb			

Ordered list:

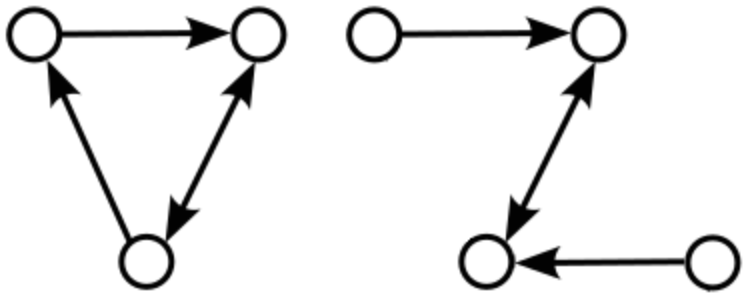
Qal
Plsh
Pf
Pw
Ps
Pwp
Pra
Mp
Mpu
Mpl
Mb
Mh
Mm
Mmu
Mmg
Msg
Msw
Mw
Mfp
as
rl
Dc
Olcy
Obc
Oh
Oc
Olb

Graph Theory: Definitions

- In computer science/mathematics, graph theory is the study of structures (graphs) "used to model pairwise relations between objects."
- Graphs are made up of *vertices* connected by *edges*
  - Represented by an ordered pair: $G = (V, E)$
- There are two broad categories of graphs:
  - *Undirected*: edges link two vertices symmetrically



  - *Directed*: edges link two vertices asymmetrically



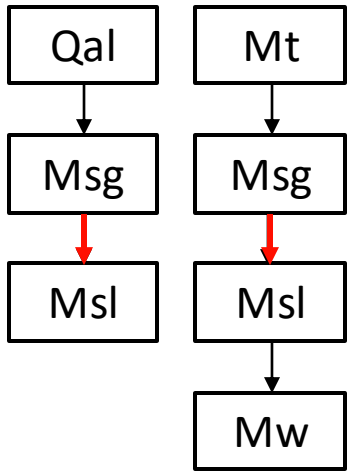
- Graphs that have paths (a set of edges) that start and end at the same node contain a *cycle*

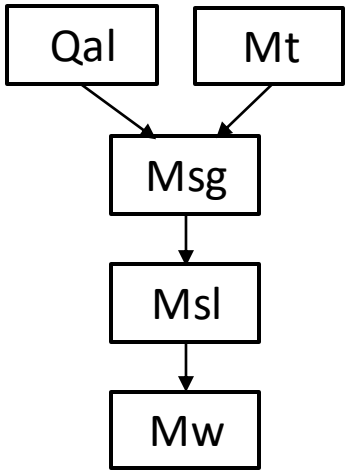
A network that helps define and visualize relationships between components

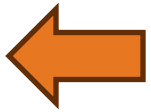
Graph Theory: Visualization

- Apply the concept of a “directed graph” to lists of map units
- Connect units (*nodes*) from youngest to oldest
- Remove duplicate connections (*edges*) and simplify

Qal	Mt
Msg	Msg
Msl	Msl
	Mw







Hey, it’s a graph!

Input data: What do we need?

- Clean up the map unit compilation table
  - Apply data hygiene techniques
    - Leading and trailing spaces, erroneous punctuation, etc.
    - Misspellings
    - Consistent formatting of Age
- Group units by age
  - Create “HierarchyKeyAge” and apply to each unit
  - Makes QC of sorted lists **MUCH** easier
- Define the necessary attributes:
  - Unit, Name, Age
  - **RELATIONSHIPS:**
    - Unit above, Unit below

Simple table to be processed in Python

ID	Unit	Group	UnitAbove	UnitBelow	HierarchyKeyAge	
1	Qal	Alluvium		Msg	hk01-01-01	
2	Msg	Ste. Genevieve Limestone	Qal	Msl	hk01-03-02-03-03	
3	Msl	St. Louis Limestone	Msg		hk01-03-02-03-03	
4	Msg	Ste. Genevieve Limestone		Msl	hk01-03-02-03-03	
5	Msl	St. Louis Limestone	Msg	Mw	hk01-03-02-03-03	
6	Mw	Warsaw Formation	Msl		hk01-03-02-03-03	
7	Qal	Alluvium		Msl	hk01-01-01	
8	Msl	St. Louis Limestone	Qal	Msw	hk01-03-02-03-03	
9	Msw	Salem and Warsaw Limestones	Msl	Mfp	hk01-03-02-03-03	
10	Mfp	Fort Payne Formation	Msw	Mfps	hk01-03-02-03-04	
11	Mfps	Fort Payne Formation - Argillaceous Siltstone	Mfp	MDc	hk01-03-02-03-04	
12	MDc	Chattanooga Shale	Mfps		hk01-03-03-01-01	
13	Qal	Alluvium		Plsh	hk01-01-01	
14	Plsh	Lee Formation - Shale Member	Qal	Mp	hk01-03-02-01-04	
15	Mp	Pennington Formation	Plsh	Mb	hk01-03-02-03-01	
16	Mb	Bangor Limestone	Mp	Mh	hk01-03-02-03-01	
17	Mh	Hartselle Sandstone	Mb	Mmu	hk01-03-02-03-03	
18	Mmu	Monteagle Limestone - Upper limestone Member	Mh	Mmg	hk01-03-02-03-03	
19	Mmg	Monteagle Limestone - Ste. Genevieve limestone Member	Mmu	Msl	hk01-03-02-03-03	
20	Msl	St. Louis Limestone	Mmg	Msw	hk01-03-02-03-03	
21	Msw	Salem and Warsaw Formations	Msl	Mfp	hk01-03-02-03-03	
22	Mfp	Fort Payne Formation	Msw	Mfpr	hk01-03-02-03-04	
23	Mfpr	Fort Payne Formation - Reef Limestone	Mfp		hk01-03-02-03-04	
24	Mfp	Fort Payne Formation and Chattanooga Shale		Olcy	hk01-03-03-01-01	d Upper Devonian
25	Olcy	Leipers and Catheys Formations	Mfp	Obc	hk01-03-08-01-01	
26	Obc	Bigby-Cannon Limestone	Olcy	Oh	hk01-03-08-01-01	
27	Oh	Hermitage Formation	Obc	Oca	hk01-03-08-01-01	
28	Oca	Carters Limestone	Oh	Olb	hk01-03-08-01-01	
29	Olb	Lebanon Limestone	Oca		hk01-03-08-01-01	
30	Qal	Alluvium		Msg	hk01-01-01	
31	Msg	Ste. Genevieve Limestone	Qal	Msl	hk01-03-02-03-03	
32	Msl	St. Louis Limestone	Msg		hk01-03-02-03-03	

Python script

- Utilize the following Python packages:
  - **Pandas**: read data table
  - **networkx**: create directed graphs
  - **pydot**: plot the relationships as flowchart drawings

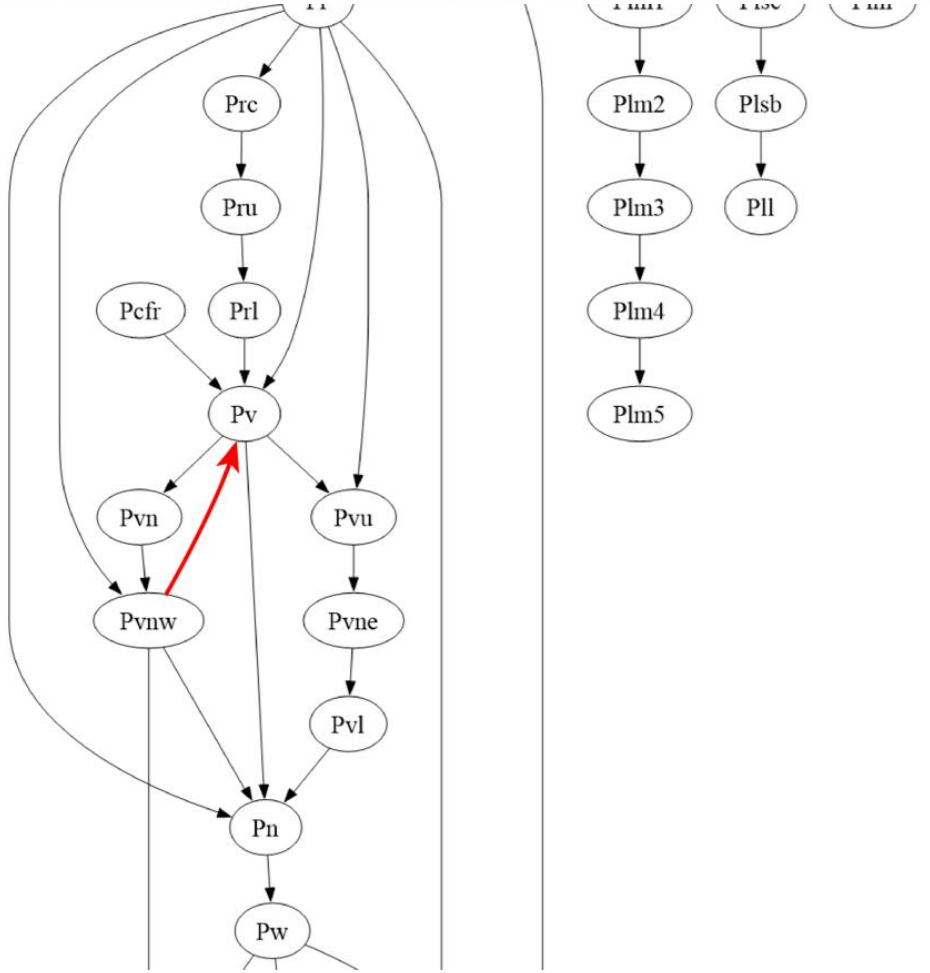
- What the script does:
  - Groups rows from input table by “HierarchyKeyAge” and stores them in a Pandas DataFrame
  - Creates a graph for each DataFrame with “Unit” added as a node (unique values)
  - Creates edges between each “Unit” and its “UnitAbove” and “UnitBelow” (skips blanks)
  - Performs a “topological sort” of the edges:
    - “A topological sort is a nonunique permutation of the nodes of a directed graph such that an edge from *u* to *v* implies that *u* appears before *v* in the topological sort order. *This ordering is valid only if the graph has no directed cycles.*”
  - Plots the sorted graph and saves images (PNGs) for review
  - Writes a tab-delimited table of the sorted units

Python script

```python
# Sort the entire table by HierarchyKey first (this ensures primary grouping)
df = df.sort_values(by="HierarchyKeyAge", key=lambda x: x.str.split("-"))

# Dictionary to store sorted results
sorted_units = []

# Process each HierarchyKey independently
for hk, group_df in df.groupby( by: "HierarchyKeyAge", sort=False):
    # Create a directed graph from the MapUnits in the HierarchyKey
    graph = nx.DiGraph()

    # Add nodes and edges
    for _, row in group_df.iterrows():
        #unitGroup = row["Unit"] + " (" + row["Group"] + ")"
        group = row["Group"]
        unit = row["Unit"]
        above = row["UnitAbove"] if pd.notna(row["UnitAbove"]) else None
        below = row["UnitBelow"] if pd.notna(row["UnitBelow"]) else None
        unitId = row["SORT"]

        # Add each unit as a node in the graph
        graph.add_node(unit)

        # Create edges between nodes based on unit above and unit below
        if above and above in group_df["Unit"].values:
            graph.add_edge(above, unit)  # UnitAbove → Unit edge
        if below and below in group_df["Unit"].values:
            graph.add_edge(unit, below)  # Unit → UnitBelow edge


    try:
        # Perform topological sorting within the HierarchyKey group
        sorted_list = list(nx.topological_sort(graph))
        sorted_units.extend([(unit, hk) for unit in sorted_list])
        # Draw a flowchart explaining the relationships between the units in the topologically sorted list
        nx.drawing.nx_agraph.write_dot(graph, path: f"{folder}Images\\{ver}\\UnitSort test {ver} graph {hk}.dot")
        (plot,) = pydot.graph_from_dot_file(f"{folder}Images\\{ver}\\UnitSort test {ver} graph {hk}.dot")
        plot.write_png(f"{folder}Images\\{ver}\\UnitSort test {ver} graph {hk}.png")
        os.remove(f"{folder}Images\\{ver}\\UnitSort test {ver} graph {hk}.dot")
        print(f"See 'UnitSort test {ver} graph {hk}.png' for diagram of sorted units in {hk}.")
    except nx.NetworkXUnfeasible:
        print(f"Cycle detected in HierarchyKey {hk}; Unable to resolve directed graph, "
              f"items in this key will not be added to the final sorted list. "
              f"\n  Check the following for help resolving the error:")
        # Reports the units involved in the cycle error
        print(f"  Error(s) found in the sequence of the following units: {list(nx.chordless_cycles(graph))}")
        print(f"  cycle_basis for {hk}: {list(nx.simple_cycles(graph))}")
        # Find all simple cycles for error reporting
        errorCycles = list(nx.simple_cycles(graph))
        # Extract closing edges
        closing_edges = []
        for cycle in errorCycles:
            if len(cycle) > 1:
                # The edge from last node to the first closes the cycle
                closing_edge = (cycle[-1], cycle[0])
                closing_edges.append(closing_edge)
        # Print results
        print(f"  Detected cycle(s):")
        for cycle in errorCycles:
            print("   ", cycle)
        print(f"  Closing edge of each detected cycle(s):")
        for edge in closing_edges:
            print("   ", edge)

        # Draw a flowchart explaining the relationships (cycle) between the units causing the error
        nx.drawing.nx_agraph.write_dot(graph, path: f"{folder}Images\\{ver}\\UnitSort test {ver} graph error {hk}.dot")
        (plot,) = pydot.graph_from_dot_file(f"{folder}Images\\{ver}\\UnitSort test {ver} graph error {hk}.dot")
        plot.write_png(f"{folder}Images\\{ver}\\UnitSort test {ver} graph error {hk}.png")
        # Don't remove the dot file, so it can be edited to show error in different color
        #os.remove(f"{folder}Images\\UnitSort test {ver} cycle {unitId}.dot")
        try:
            print(f"  Check 'UnitSort test {ver} graph error {hk}.png' for diagram of graph error.")
            #print(f"  find_cycle: {nx.find_cycle(graph, orientation='original')}")
        except nx.NetworkXNoCycle:
            #print("  networkx.exception.NetworkXNoCycle: No cycle found.")
            continue


# Convert to DataFrame and output sorted data
sorted_df = pd.DataFrame(sorted_units, columns=["Unit", "HierarchyKeyAge"])
print(sorted_df)
sorted_df.to_csv( path_or_buf: f"{folder}UnitSort test {ver} RESULTS NoCoal.txt", sep='\t', index=False)
```

Interpreting script output

```
C:\Users\awund\AppData\Local\ESRI\conda\envs\arcgispro-py3-clone\python.exe "G:\Other computers\My Work
See 'UnitSort test 28 graph hk01-01-01.png' for diagram of sorted units in hk01-01-01.
See 'UnitSort test 28 graph hk01-01-01-01-01.png' for diagram of sorted units in hk01-01-01-01-01.
See 'UnitSort test 28 graph hk01-01-02.png' for diagram of sorted units in hk01-01-02.
See 'UnitSort test 28 graph hk01-01-03-02-02-02.png' for diagram of sorted units in hk01-01-03-02-02-02
See 'UnitSort test 28 graph hk01-01-03-02-02-03.png' for diagram of sorted units in hk01-01-03-02-02-03
See 'UnitSort test 28 graph hk01-01-03-02-02-04.png' for diagram of sorted units in hk01-01-03-02-02-04
See 'UnitSort test 28 graph hk01-01-03-02-03.png' for diagram of sorted units in hk01-01-03-02-03.
See 'UnitSort test 28 graph hk01-01-04-01-01.png' for diagram of sorted units in hk01-01-04-01-01.
See 'UnitSort test 28 graph hk01-02-01-01-01.png' for diagram of sorted units in hk01-02-01-01-01.
See 'UnitSort test 28 graph hk01-03.png' for diagram of sorted units in hk01-03.
See 'UnitSort test 28 graph hk01-03-02-01-02.png' for diagram of sorted units in hk01-03-02-01-02.
See 'UnitSort test 28 graph hk01-03-02-01-03.png' for diagram of sorted units in hk01-03-02-01-03.
Cycle detected in HierarchyKey hk01-03-02-01-04; Unable to resolve directed graph, items in this key wi
  Check the following for help resolving the error:
  Error(s) found in the sequence of the following units: [['Pvnw', 'Pv', 'Pvn'], ['Pgl', 'Pg', 'Pgu', '
  cycle_basis for hk01-03-02-01-04: [['Pv', 'Pvn', 'Pvnw'], ['Pg', 'Pgu', 'Pgum', 'Pgml', 'Pgl'], ['Pg'
  Detected cycle(s):
    ['Pv', 'Pvn', 'Pvnw']
    ['Pg', 'Pgu', 'Pgum', 'Pgml', 'Pgl']
    ['Pg', 'Pgu', 'Pgml', 'Pgl']
  Closing edge of each detected cycle(s):
    ('Pvnw', 'Pv')
    ('Pgl', 'Pg')
    ('Pgl', 'Pg')
  Check 'UnitSort test 28 graph error hk01-03-02-01-04.png' for diagram of graph error.
See 'UnitSort test 28 graph hk01-03-02-02-01.png' for diagram of sorted units in hk01-03-02-02-01.
Cycle detected in HierarchyKey hk01-03-02-03-01; Unable to resolve directed graph, items in this key wi
  Check the following for help resolving the error:
  Error(s) found in the sequence of the following units: [['Mh', 'Mnl', 'Mb']]
  cycle_basis for hk01-03-02-03-01: [['Mnl', 'Mb', 'Mh']]
  Detected cycle(s):
    ['Mnl', 'Mb', 'Mh']
  Closing edge of each detected cycle(s):
    ('Mh', 'Mnl')
  Check 'UnitSort test 28 graph error hk01-03-02-03-01.png' for diagram of graph error.
See 'UnitSort test 28 graph hk01-03-02-03-02.png' for diagram of sorted units in hk01-03-02-03-02.
See 'UnitSort test 28 graph hk01-03-02-03-03.png' for diagram of sorted units in hk01-03-02-03-03.
```



Conclusions

- **Graph theory** is a powerful, algorithm-based tool for topological sorting with wide application
- In this case, having the original lists of map units from each quadrangle was **vital**
- Python modules **significantly** reduce the amount of code and script complexity (and use ChatGPT to help with code snippets)
- Adapting this script to work on well-sorted GeMS DMU tables could be very helpful for compilation maps or heads-up web display

Additional Information

- Graph theory:
  - Wikipedia: https://en.wikipedia.org/wiki/Graph_theory
  - YouTube: https://www.youtube.com/watch?v=LFKZLXVO-Dg

- Python modules:
  - Pandas DataFrame: https://pandas.pydata.org/docs/user_guide/10min.html
  - NetworkX: https://networkx.org/
  - pydot: https://pypi.org/project/pydot/
    - Graphvis: https://graphviz.org/; PyGraphvis: https://pygraphviz.github.io/

Department of Environment & Conservation

Division of Mineral and Geologic Resources

TENNESSEE GEOLOGICAL SURVEY

ESTABLISHED 1831

Thank you!

Questions?

andrew.wunderlich@tn.gov