Planarity Algorithm for Hamiltonian Graphs

Toby Mallon, Sam Wisnoski, Daniel Theunissen

Our Goal:

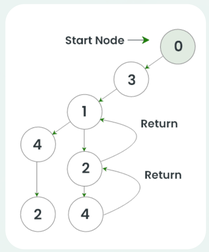
Determine if a graph has a Hamiltonian Circuit, and if so, if the graph is planar.

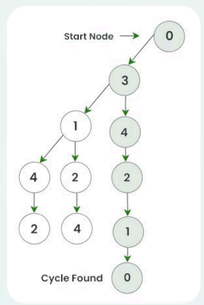
Introduction and Potential Applications

Determining whether a graph is planar or not is useful in several applications such as finding optimal circuit layouts, creating transportation networks, or visualizing data. In this project, we are showcasing an algorithm that can find the planarity of a graph if it contains a Hamiltonian cycle.

Step One: Finding a Hamiltonian Circuit

Our approach to finding a Hamiltonian cycle in a graph starts by attempting to build a path that visits each vertex exactly once, returning to the starting point to complete the cycle. The algorithm places vertices sequentially into the path, checking each time if the current vertex can be safely added without revisiting or breaking adjacency. If adding a vertex leads to a dead end, it back-tracks to try different possibilities.





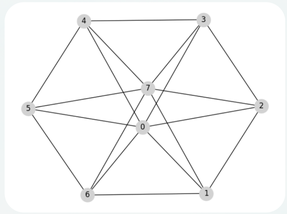
The process continues recursively until either a complete Hamiltonian cycle is formed by connecting the final vertex back to the start, or it exhausts all options, in which case no cycle exists.

Step Two: Checking Planarity

We can use a simple planarity algorithm to check if a graph with a Hamiltonian circuit is planar. We outline the steps below with visuals created by NetworkX for an example planar graph with 8 nodes.

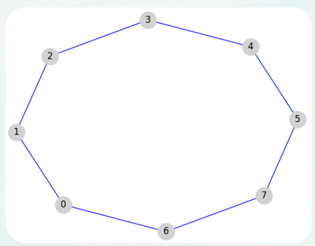
Step One: Identify all edges and label them by the two vertices they connect.

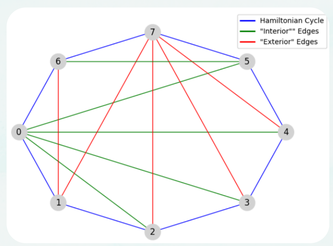
```
ADJACENCY_MATRIX = [
    [0, 0, 0, 1, 1, 0, 1, 0],
    [0, 0, 1, 0, 1, 0, 1, 0],
    [0, 1, 0, 1, 0, 0, 1, 1],
    [1, 0, 1, 0, 1, 0, 1, 0],
    [1, 1, 0, 1, 0, 1, 0, 0],
    [0, 0, 0, 0, 1, 0, 1, 1],
    [1, 1, 1, 1, 0, 1, 0, 1],
    [0, 0, 1, 0, 0, 1, 1, 0],
]
```



Step Two: Identify the Hamiltonian circuit and draw the edges connecting the vertices such that the circuit forms a cycle graph.

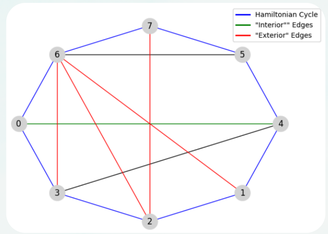
Step Three: Draw each of the other edges in the graph. Choose an unlabeled edge and label it to be an “interior” edge. Label all edges intersecting your “interior” edges to be “exterior” edges. If any of the exterior edges intersect each other, the graph is non-planar. Repeat for each other unlabeled edge until all edges are labeled or the graph is found to be nonplanar.





Results: If the edges are labeled correctly, interior edges should not intersect other interior edges, and exterior edges should not intersect other exterior edges. If this holds, the graph is planar.

If any interior edges intersect other interior edges, or any exterior edges intersect other exterior edges, the graph is not planar. An example nonplanar graph is shown below. Once the algorithm detects that two “exterior” edges intersect, it realizes that the graph is non-planar and leaves the rest of the edges unlabeled.

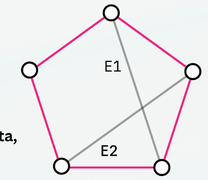


What is the Math Behind This?

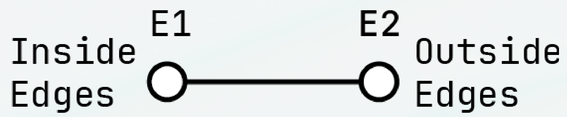
Let's consider a graph G and a Hamiltonian cycle within that graph C. We can draw this cycle as a polygon with each vertex connected to the subsequent vertex in the cycle. This then means that every remaining edge has to lie inside or outside of the polygon.

Graph G

Graph C highlighted in magenta, remaining edges shown

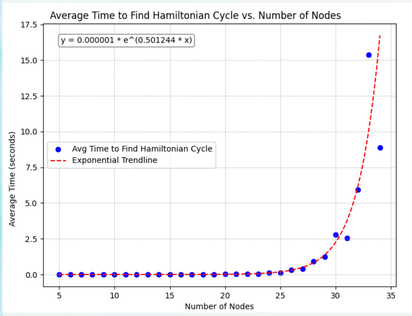


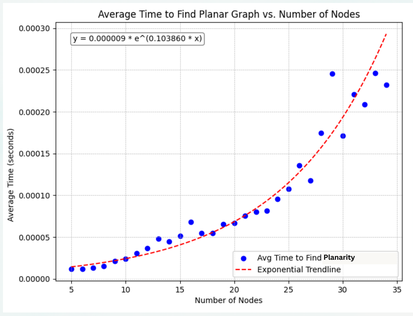
We can define a new graph cross(G,C) which has the remaining edges as vertices and an edge between them if they cross.



Adjacency between two points on this graph means that the two edges can't be drawn on the same side of the polygon without crossing. With this in mind, we can say that if cross(G,C) is bipartite, then G is planar.

Time Complexity





Sources:

Hamiltonian Circuits:

1. https://www.geeksforgeeks.org/hamiltonian-cycle/

Planarity Algorithm:

1. https://www.savemyexams.com/a-level/further-maths_decision-maths-1/edexcel/17/revision-notes/algorithms-and-graph-theory/graphs/planarity-algorithm/
2. https://ptwiddle.github.io/Graph-Theory-Notes/s_graphs_on_surfaces_planarity_algorithm.html

Line intersection:

1. https://www.geeksforgeeks.org/check-if-two-given-line-segments-intersect/