CMI Math Circle

Synopsis for August 09 2026

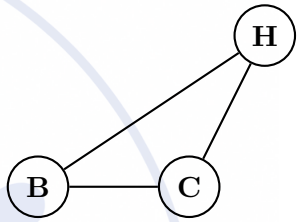
Junior Batch

We started the session by recalling the "No Crossing" game we played during our last session on August 2, 2026. They showed how to predict the winner of the game based on the number of points chosen, without having to play the entire game. (For more details, please see the synopsis from August 2, 2026.)

Next, we began our new topic by recalling the definition of graphs.

A graph is a collection of dots called **Nodes** (or Vertices) and lines connecting them called **Edges**.

Imagine the nodes are cities like Chennai, Bengaluru, and Hyderabad, and the edges are the highways connecting them!



We **colour** the nodes of the graph by following a very strict rule.

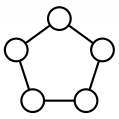
No two nodes that are connected by an edge can have the same colour!

That is, if we colour Chennai RED, we **cannot** colour Bengaluru RED as they are connected by a highway(edge). We must choose a different colour, like BLUE or GREEN.

The main goal of this session was to understand and find the chromatic number of different graphs. The chromatic number is the smallest number of colours needed to completely colour a graph without breaking the main rule.

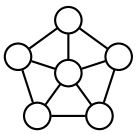
Students practiced colouring various graph examples of different difficulty levels and then calculated their chromatic numbers. Then, students examined different types of graphs to discover patterns.

- **Cycle Graphs:** These look like rings as shown below. For example, here is a cycle graph with 5 outer vertices.



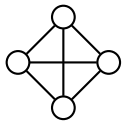
  The students found that rings with an even number of nodes need 2 colours. Rings with an odd number of nodes need 3 colours.

- **Wheel Graphs:** These have a center node connected to a ring. For example, here is a wheel graph with 5 outer vertices.



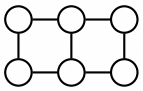
Because the center node touches everything, it needs its own colour. This means a wheel graph needs one more colour than a cycle graph.

- **Complete Graphs:** In these graphs, every node is connected to every other node. For example, here is a complete graph with 4 vertices.



The students saw that a complete graph with 5 nodes needs 5 colours, and a complete graph with 10 nodes needs 10 colours.

- **Grid Graphs:** These graphs look like a chessboard. For example, here is a 2×3 grid graph.



The students learned that grid graphs only ever need 2 colours, no matter how big they are.

Next, we talked about how computers solve graph colouring problems. Computers cannot look at a picture to plan ahead; they need simple, step-by-step instructions called an algorithm. The students acted like computers by following the **Alphabet algorithm**.

Step 1: Line up the colours in a strict order (for example: 1st = Red, 2nd = Blue, 3rd = Green).

Step 2: Put all the nodes in alphabetical order (A, B, C, D...).

Step 3: Go to the first node on the list and give it the first colour.

Step 4: Move to the next node. Look at the nodes connected to it. Give it the first colour from your list that its neighbours are not using.

Step 5: Repeat this process until every node is coloured.

The students found a problem with this method. When the computer simply follows an alphabetical list, it sometimes gets trapped and uses extra colours. The computer followed the colouring rules perfectly, but it did not find the true chromatic number. This showed the students that the final answer changes depending on the **order of the list**.

To fix this problem, the students learned how to make a smarter list of vertices for the computer to follow.

1. Count the number of connections for each node. This number is called the **degree**.
2. Find the nodes with the most connections.
3. Write a new list. Put the nodes with the highest degree first, and the nodes with the lowest degree last.

When the students ran the **Alphabet Algorithm** using this new list, the computer coloured the most connected nodes first. This simple change helped the computer find the lowest possible number of colours most of the time. However, the students also learned that even this smart list is not perfect, and it will not work for every single graph! The session concluded on an open-ended note, as there is no universal rule for ordering vertices that will work optimally for all graphs.