

The Four-Colour problem: How computers and mathematics intertwine

Research question: What is the Four-Colour problem, what is its significance, and how does it demonstrate the use and effects of computers in mathematics?

IB Mathematics Elongated Essay

Word Count: 3985

## Table of Contents

<b>Introduction</b>	<b>2</b>
What is the Four-Colour problem?	2
<b>The first major attempt by Alfred Bray Kempe</b>	<b>2</b>
The “proof”	2
Counterproof	9
Computerized equivalence and its benefits	10
<b>The first correct proof</b>	<b>12</b>
How it comes to be	12
The computer’s role and effects in the first correct proof	16
The final�	18
The after...math	18
<b>Conclusion</b>	<b>19</b>
Journey’s end	19
<b>Appendix</b>	<b>20</b>
<b>References</b>	<b>24</b>

## **Introduction**

### **What is the Four-Colour problem?**

The Four-Colour problem or the Four-Colour theorem imposes that given four colours, one can colour all the vertices on a planar graph in such a way that no two adjacent vertices or no two connected vertices share the same colour (“Four Color Theorem and Kuratowski’s Theorem in Discrete Mathematics” 2021). Another way to state this theorem is that using only four colours, it is possible to colour all regions of every map in such a way that no two adjacent regions or no two regions that share the same border have the same colour (“Four Color Theorem and Kuratowski’s Theorem in Discrete Mathematics” 2021). It is a small and simple theorem with big consequences and effects in mathematics and ways of knowing in mathematics. Thus, in this investigation, the significance of the Four-Colour problem, its solutions, and the effects of both the Four-Colour problem and the use of computers in mathematics as demonstrated through the Four-Colour problem would be explored.

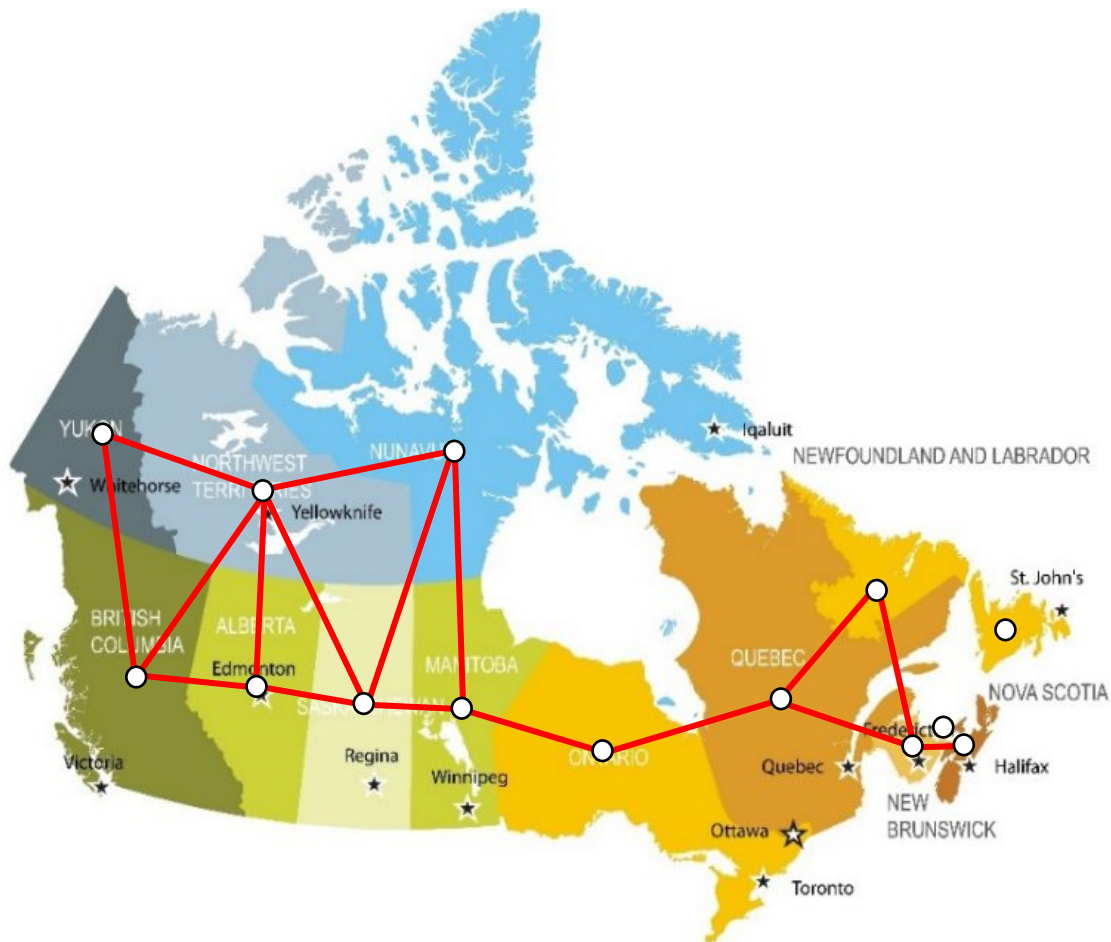
### **The first major attempt by Alfred Bray Kempe**

#### **The “proof”**

To start, due to the age of Kempe’s original paper detailing his “proof” (1879), his liberal use of prose and non-mathematical terms (such as “patch”, “countries”, etc.) (1879) to describe his proof and his use of non-standard, modified formulas (such as a modified version of Euler’s planar graph formula) (1879), Kempe’s method must be modernized and adjusted, using standardized formulas and current, mathematical terminologies, as to better explain and avoid confusion regarding Kempe’s “solution” to the Four-Colour problem. The following details the author’s customized and contemporary version of Kempe’s “proof” of the problem.

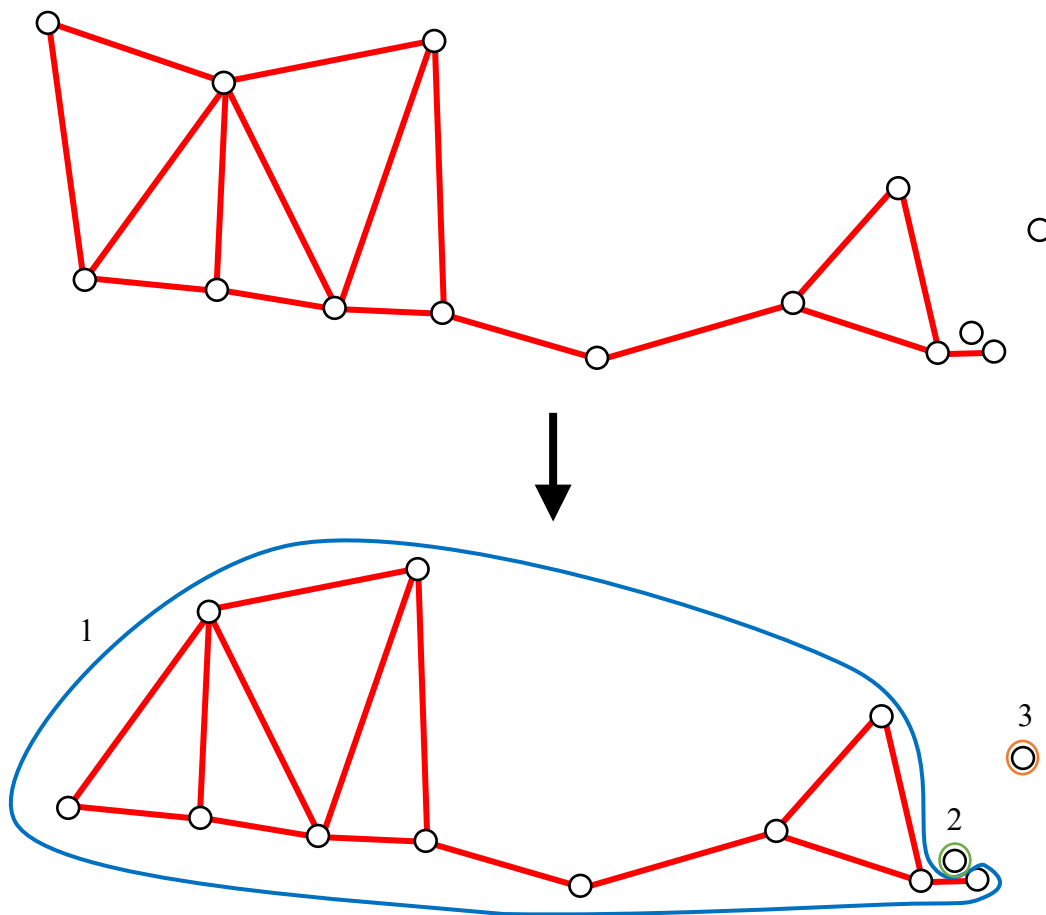
In the first step of the reworked Kempe's method (Kempe 1879), another graph must be constructed on top of the original convention map to simplify and represent the connection between the different territories. Each territory is to be represented by a vertex (a point in the planar graph) and the connection or sharing of borders between two or more vertices is to be represented by one edge (a line, either curved or straight, starting and ending on the same vertex or starting on one vertex and ending on another vertex) between the connecting vertices. This is the dual graph form of the conventional map upon which Kempe's method can be utilized and explained.

Figure 1.1: Dual graph of the conventional map of Canada (Kevin 2017).



Kempe's proof (1879) is practically a proof by induction even though Kempe did not mention this in his paper (Sipka 2002). As with any other induction proof, Kempe first said that any planar graphs with four or fewer vertices can be vertex four colourable (Sipka 2002). Then he assumed that an  $n$  vertices graph would also be four colourable and tried to prove that a  $n + 1$  vertices graph would be four colourable as well (Sipka 2002). To do this, Kempe uses the "patching" technique (Kempe 1879). Essentially, he would remove a vertex of degree five or less from the graph and continue to do so until there is no vertex left (Kempe 1879). And should upon removing one vertex, there be disconnected vertices then the disconnected portion shall be treated as a distinct graph.

Figure 1.2: Kempe's patching process where there are three distinct graphs at both steps.



However, why must vertices of degree five or less be removed? It is because it can be proven that in any planar graph, there would be at least one of such vertices available to be removed, and thus it is a safe operation. In a planar graph, there would be  $F$  faces (an area enclosed by one or more edges),  $E$  edges and  $V$  vertices. And as per Euler's planar graph formula,

$$V - E + F = 2.$$

Now,  $V$  is the number of vertices in the graph:

$$V = v_0 + v_1 + v_2 + v_3 + v_4 + \dots v_x$$

$v_x$  is the number of vertices with a degree of  $x$ .

Next,  $E$  is the number of edges in the graph and considering that every edge ends between two vertices:

$$2E = v_1 + 2v_2 + 3v_3 + 4v_4 + \dots xv_x$$

Another equation of  $E$  could be formed that takes into consideration that the degree of a face is the number of edges and that a degree of a face is only one side of an edge:

$$2E = f_1 + 2f_2 + 3f_3 + 4f_4 + \dots + xf_x$$

$f_x$  is the number of faces with a degree of  $x$ .

Then,  $F$  is the number of faces in the graph:

$$F = f_0 + f_1 + f_2 + f_3 + f_4 + f_5 + \dots + f_x$$

Finally,  $F$ ,  $E$  and  $V$  can be substituted into Euler's formula as follow:

$$V - E + F = 2$$





$$6V - 6E + 6F = 12$$

$$6V - 2E + 6F - 4E = 12$$

$$\begin{aligned}
& 6(v_0 + v_1 + v_2 + v_3 + v_4 + \dots + v_x) - (v_1 + 2v_2 + 3v_3 + 4v_4 + \dots + xv_x) \\
& + 6(f_0 + f_1 + f_2 + f_3 + \dots + f_x) - 2(f_1 + 2f_2 + 3f_3 + 4f_4 + \dots + xf_x) = 12 \\
& 6v_0 + 5v_1 + 4v_2 + 3v_3 + 2v_4 + v_5 - \dots + (6-x)v_x + 6f_0 + 4f_1 + 2f_2 - 2f_4 - 4f_5 \\
& - 6f_6 - \dots + (6-2x)f_x = 12
\end{aligned}$$

From the equation, because of the presence of  $+6f_0 + 4f_1 + 2f_2$ , it is not clear whether or not  $6v_0 + 5v_1 + 4v_2 + 3v_3 + 2v_4 + v_5$  would be the sole cause for the total of the equation to 12 ( $+(6-x)v_x$  and  $+(6-2x)f_x$  would always be negative for degree values larger than five and thus is a non-issue). In other words, it is not clear whether there must be at least one vertex of degree five or less in the graph. This issue, however, can easily be resolved if each case of  $f_0$ ,  $f_1$ , and  $f_2$  is examined.

Table 1.1: Cases of face and vertex, their possible graphs, and valid conditions.

Cases	Possible graphs	Conditions
$f_0 \geq 1$ $v_1 = 1$		Only one vertex is present.
$f_1 \geq 1$ $v_1 = 1$		Edge(s) start and end at the same vertex. Not possible as a territory (vertex) can not be the neighbour of itself.
$f_2 \geq 1$ $v_2 = 2$	 	Only two vertices are connected to each other with one edge. Two vertices connected to each other with two or more edges. Not possible as each connection is represented by one

		and only one edge.
--	--	--------------------

As seen in Table 1.1, all cases of  $f_0$ ,  $f_1$  and  $f_2$  are either only possible when the graph only has exactly one or two vertices or not possible at all. Thus, it is safe to exclude them for graphs with at least three vertices and a new equation could be made:

$$5v_1 + 4v_2 + 3v_3 + 2v_4 + v_5 - \dots + (6 - x)v_x - 2f_4 - 4f_5 - 6f_6 - \dots + (6 - 2x)f_x = 12$$

From this, it is evident that there must be at least one vertex of degree five or less for the equation to be positive. This means that in a graph with three or more vertices, there is always at least one vertex of a degree less than six. Combining with the fact there is always at least one vertex of degree five or less in a graph with one or two vertices (as shown in Table 1.1), it is possible to only remove vertices of degree five exactly one or two vertices or less until there are no vertices left.

After removing the vertices, they are to be added back in the order in which they were removed (Kempe 1879). Assuming that after removing a vertex, the  $n$  vertices graph is coloured using four or fewer colours (Kempe 1879). After adding back the removed vertex, there are four unavoidable cases (Kempe 1879):

1. The added vertex is connected to less than four other vertices. The added vertex can be coloured using the fourth colour.
2. The added vertex is connected to four or five other vertices that were coloured with at most three colours. The added vertex can be coloured using the fourth colour.
3. The added vertex is connected to four other vertices that were coloured with four colours.

To free up the fourth colour, one can interchange the colour of one of the surrounding disconnected vertices that are of different colour regions (not in the same chain of their



colours – from one vertex, it is possible to reach the other by tracing vertices that have either of the two colours of the vertices of interest) (Sipka 2002).

Figure 2.1: Added vertex, X, connected to vertices A, B, C and D with A and C in different colour regions (adapted from illustration by Sipka (2002)).

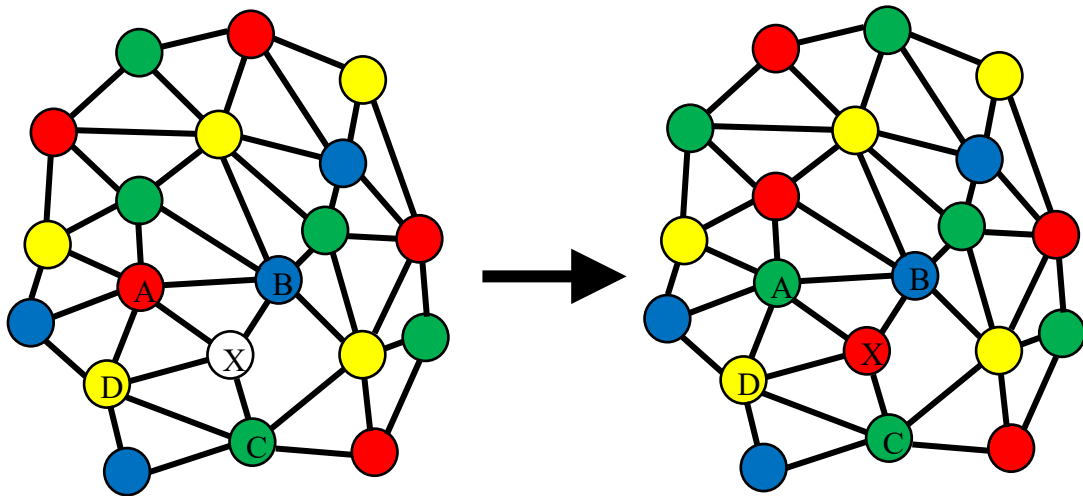
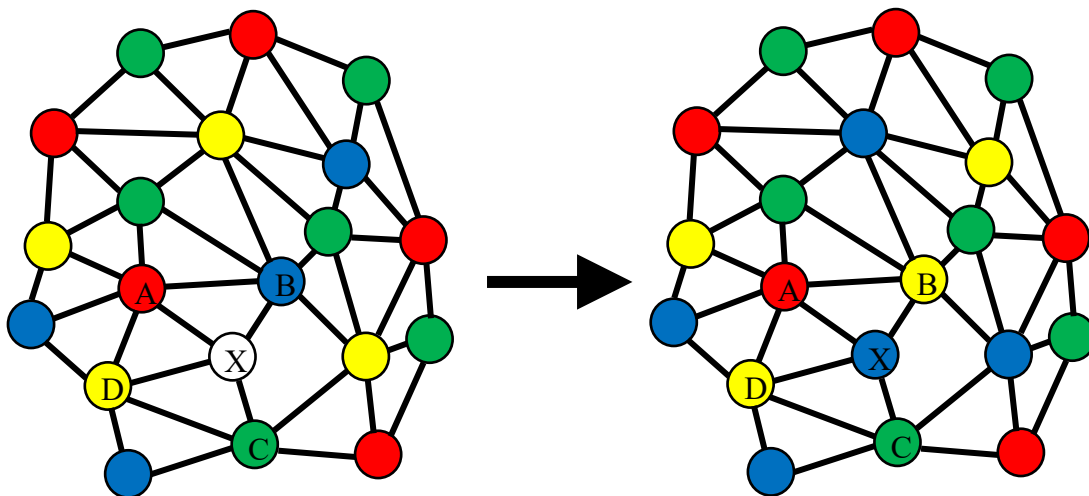


Figure 2.2: Added vertex, X, connected to vertices A, B, C and D with A and C in the same colour region (adapted from illustration by Sipka (2002)).



4. The added vertex is connected to five other vertices that are coloured using four colours.

If two disconnected vertices are of different colour regions, one can interchange the

colour of one of them (Sipka 2002). If two pairs of disconnected vertices are of the same colour regions, one can interchange the colour of the two vertices that are not in the two pairs above (Sipka 2002). With that, a fourth colour is available to colour the added vertex with.

Figure 3.1: Added vertex, X, connected to vertices A, B, C, D and E with A and C in different colour regions (adapted from illustration by Sipka (2002)).

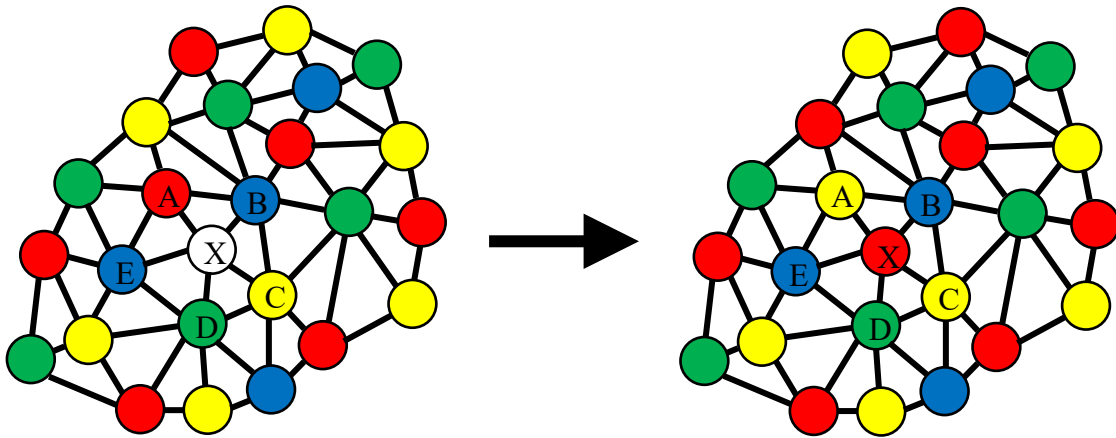
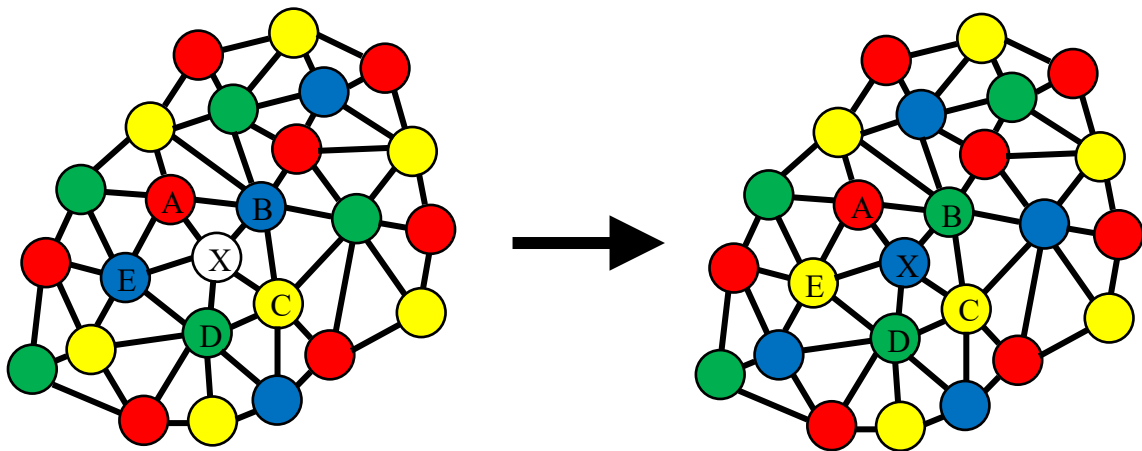


Figure 3.2: Added vertex, X, connected to vertices A, B, C, D and E with A and C in the same colour region and A and D in the same colour region (adapted from illustration by Sipka (2002)).



### Counterproof

However, Kempe's solution was flawed as demonstrated by Percy John Heawood in 1890 and the Four-Colour conjecture returned to being unproven (Sipka 2002). Heawood

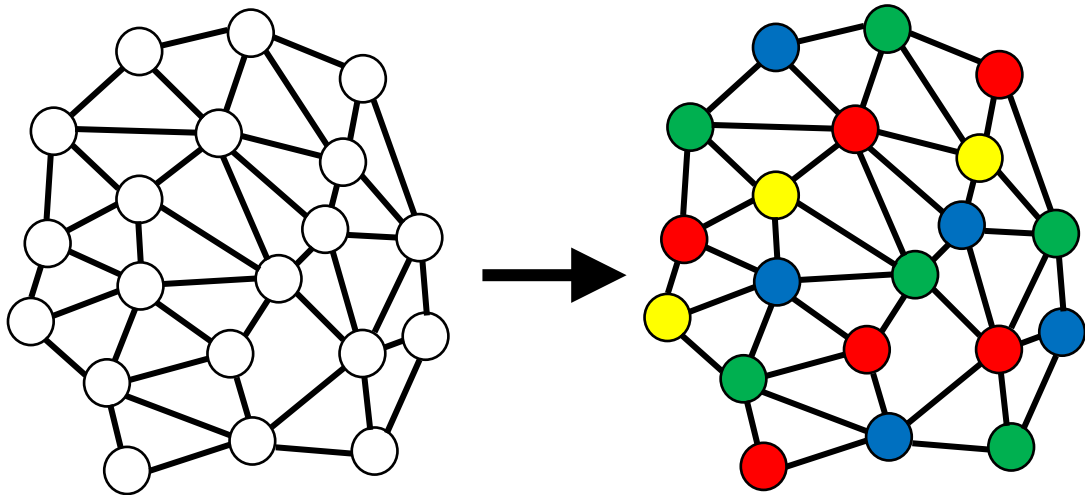
presented a situation in case four that when interchanging the colours of the blue-green and blue-yellow regions so that “[B] becomes green and [E] yellow, [A], [C] and [D] remaining unchanged” (Kempe 1879), Kempe had failed to consider the fact that the blue-green region that B is in might be adjacent to the blue-yellow region that contains E (Sipka 2002). In this situation, interchanging the colours of those regions like Kempe did would result in the vertex from the blue-green region that borders the adjacent vertex from the blue-green region to both be assigned the colour blue (Sipka 2002). This violates the assertion that no two vertices that share an edge shall receive the same colour, invalidating Kempe’s proof. Knowing of the flaw, Kempe attempted to rectify it but to no avail and the Four-Colour conjecture was once again unproven (Sipka 2002).

### **Computerized equivalence and its benefits**

Kempe’s method, while not the correct proof, is still valid for many graphs. One can still utilize Kempe’s method to colour with four colours any trivial graphs that at any point during the “patching and removing” process, the revealed vertex only has a degree of four as only Kempe’s five-degree case is flawed. However, as one would have guessed, Kempe’s “patching and removing” process was slow, painfully so for larger graphs. This is where the power of computers comes in. Kempe’s method can be programmed onto a computer using a programming language such as Python. Exhibit 1.1 (see Appendix) demonstrates a rough program written by the author using Python. It should be noted that the program did not include Kempe’s case where the revealed vertex is of degree five. This is because Kempe’s five-degree case has been proven to be flawed and in the interests of time, was not programmed for it brings little value to the program. Using this program, the process of colouring any graph can be significantly sped up.

Using an uncoloured version of Figure 2.1 from Kempe's case three, the speed advantage of the computer can be demonstrated.

Figure 4.1: Computer-coloured version of Figure 2.1:



To colour an uncoloured version of Figure 2.1, the program takes on average (of 10 runs) 0.0104 seconds to complete on an ASUS Vivobook X509DA with an AMD Ryzen 7 3700U processor (plugged in), 12 gigabytes of DDR4 RAM (double data rate random access memory generation four) at 2400 MHz, Microsoft Windows 11 Home version 22H2 (Operating System build 22621.1265) and Python 3.10 interpreter (version 3.10.2800.0). This is an incredible time that humans have no hope to match. No humans on earth can colour the same graph in less than one second. Thus, it is clear that computers have a major speed advantage compared to traditional, manual methods of solving. Moreover, due to the systematic and inhuman nature of the computer, the rate of human errors could be greatly reduced. Humans, as living organisms, are susceptible to fatigue, distraction, and other forms of organic responses. All of these could cause the human mind to not work at its peak capacity, resulting in performance degradation and errors. Furthermore, humans need regular rest whereas computers do not. Therefore, not only the computer is faster, but it can also work for longer and in a more reliable manner compared to humans. Therefore, with all these advantages, the computer could be used to run many tests, proofs, and validations, much more than a human can, to either demonstrate that the proof is

correct and reinforce the proof or to invalidate the proof by stumbling on an invalid configuration. Lastly, computers make mathematics more accessible to a wider range of people. Even a high school student, such as the author, with some extra mathematical and programming knowledge, can use computers to solve or attempt to solve complex mathematical problems like the Four-Colour problem effectively, quickly and without much trouble, as demonstrated above. Thus, considering everything, it can be seen that computers and mathematics could intertwine and complement each other.

### **The first correct proof**

#### **How it comes to be**

The first correct proof is, in some sense, similar to Kempe's proof. The proof by Wolfgang Haken and Kenneth Appel relies on the two concepts of "unavoidability" and "reducibility" which were already inadvertently introduced with Kempe's proof (Appel and Haken 1977). "Unavoidability" is when the configuration is not avoidable in a planar graph. This means that in a planar graph, there must be at least one instance of such configuration contained in the graph (Appel and Haken 1977). An example of this is Kempe's assertion that there must be at least one vertex of degree five or less in any planar graph (Appel and Haken 1977). On the other hand, "reducibility" refers to configurations that can be reduced when it can be proven that the configuration "cannot possibly appear in a minimal five-chromatic map minimal five-chromatic map" (Appel and Haken 1977). Kempe's cases one to three are examples of such reducible configurations (case four is false as proven by Heawood) (Appel and Haken 1977). These two concepts, "unavoidability" and "reducibility" together would form the basis of both Kempe's and the first correct proof, for both are essentially attempts at "constructing an unavoidable set of reducible configurations" (Appel and Haken 1977). The difference is that

Haken and Appel's proof utilized a set of “some 1,500 complex figures” (Appel and Haken 1977) in place of Kempe’s four simple configurations (Appel and Haken 1977). There is only one problem, how does one come up with so many complex configurations?

The journey to find the required set of unavoidable and reducible configurations can be separated into two parts, one for unavoidable configurations and one for reducible configurations. For unavoidable configurations, many great mathematicians such as Paul Wernicke, Philip Franklin, and Henri Lebesgue worked to find other unavoidable configurations to replace Kempe’s faulty configuration (Wilson 2016). The unavoidable set was slowly but surely expanded in 1904 by Wernicke, 1922 by Franklin and 1940 by Lebesgue (Wilson 2016). However, it was until the era of Heinrich Heesch that the process of finding unavoidable configurations is systematized with Heesch’s discharging method (Wilson 2016). The discharging method is a proof by contradiction. An example could be done with one of the configurations in Wernicke’s unavoidable set: that in a planar graph of minimal degree five, there is at least one vertex of degree five being adjacent to another vertex of degree five or a vertex of degree six (Walters 2004). First, an opposite hypothesis was made that a degree five vertex can only be adjacent to a vertex of degree seven or more (Wilson 2016). Then, Heesch would assign a “charge” to each vertex so that the total “charge” is positive (Wilson 2016). In this case, a charge of  $6 - d(v)$  (where  $d(v)$  is the degree of the vertex  $v$ ) would allow the total charge of  $\sum_{v \in V} 6 - d(v) = 4v_2 + 3v_3 + 2v_4 + v_5 - \dots + (6 - x)v_x$  (where  $v$  is a vertex belonging to  $V$  which represents every vertex in the graph and  $v_x$  is the number of vertices with degree  $x$ ) to be positive. This can easily be proven using Euler’s polyhedron formula:

$$V - E + F = 2$$

$$6(V - E + F) = 12$$

$$6V - 6E + 6F = 12$$

$$6V - 2E + 6F - 4E = 12$$

$$4v_2 + 3v_3 + 2v_4 + v_5 - \dots + (6 - x)v_x - 2f_4 - 4f_5 - 6f_6 - \dots + (6 - 2x)f_x = 12$$

$$\left( \sum_{v \in V} 6 - d(v) \right) - 2f_4 - 4f_5 - 6f_6 - \dots + (6 - 2x)f_x = 12$$

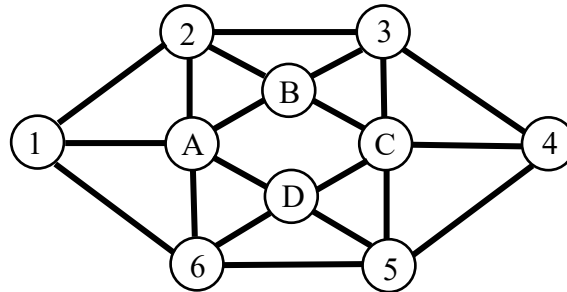
The charges can now be transferred in one-fifth from each five-degree vertex to each of its negatively charged neighbours, making it neutral, without creating or destroying any “charge” (Wilson 2016). And as per the hypothesis, vertices with degree five are not adjacent to any vertices with degree six and so all vertices of degree six remain neutral. On the other hand, a negatively charged vertex of degree seven would need at least six vertices of degree five to be positive. However, this configuration would cause at least two five-degree vertices to be adjacent to each other, which is not allowed as per the hypothesis (Wilson 2016). Therefore, vertices of degree seven can only remain negative. The same argument could be used for vertices with degrees more than seven (Wilson 2016). Thus, with the vertices with degrees five and six being neutral and the vertices with degrees seven and more being negative, the total charge could only be negative, which contradicts the assertion above that the total charge should be positive. This means that the hypothesis that a vertex of degree five can only be adjacent to vertices of degree seven or more is false and so there must be at least one vertex of degree five being adjacent to another vertex of degree five or six. This proves that this configuration of Wernicke is unavoidable (Wilson 2016). So, using this method of “discharging” and adapting it for the



specificities of each case, Heesch and eventually, Haken and Appel were able to find thousands of unavoidable configurations which are essential to the first correct proof of the Four-Colour problem (Wilson 2016).

At the same time, the process of finding reducible configurations was ongoing. In 1913, George David Birkhoff published a paper detailing an approach to finding reducible configurations (Wilson 2016). Using his infamous Birkhoff diamond as an example, the process is as follows.

Figure 5.1: Dual graph form of the Birkhoff diamond.



Birkhoff first assumes that the graph above is colourable with minimally five colours (Wilson 2016). If the vertices A, B, C and D are removed, the remaining vertices 1, 2, 3, 4, 5 and 6 can be coloured using four colours (Wilson 2016). Birkhoff then attempts to find colourings of the vertices 1 to 6 that could be extended to vertices A to D, the so-called “good colourings” (Wilson 2016). He found out that of the 31 possible colourings of vertices 1 to 6, 16 are “good colourings” while the remaining colourings, through Kempe’s colour interchanging method, could be converted to “good colourings” (Wilson 2016). This means the graph is four-colourable and so contradicts the above assumption. Thus, the Birkhoff diamond is reducible (Wilson 2016). However, much like with unavoidable sets, the process of finding reducible configurations was slow and ineffective. But once again, Heesch comes to the rescue with the D-reducibility” and

“C-reducibility” concepts (Wilson 2016). First, “D-reducibility” is a systematized version of Birkhoff’s approach and works by contradiction (Wilson 2016). First, the graph is assumed to be colourable with at least five colours, Next, all colourings of the outermost vertex ring are examined and if all colourings of the ring could be extended to the inner vertices, the configuration is four colourable which contradicts that the graph could be minimally five coloured and thus the configuration is reducible. On the hand, “C-reducibility” works by induction (Steinberger 2010). First, the graph is modified (Wilson 2016). For example, Birkhoff’s diamond can be modified in such a way that the new graph could be coloured with four colours. This new graph could then be used in the induction step which one can then prove and complete the proof by induction (Steinberger 2010). Therefore, by first finding “D-reducible” and if that does not work, finding “C-reducible”, Heesch would be able to check for reducibility of a large number of configurations (Wilson 2016).

### **The computer’s role and effects in the first correct proof**

Much like with Kempe’s method, a computer can be programmed to systematically find unavoidable and reducible sets (Walters 2004). For unavoidable sets, Heesch’s discharging method can be implemented in a computer program to automatically check all the configurations (Walters 2004). Similarly, the check for reducibility can be programmed into a computer. While the check for “C-reducibility” might not best be programmed due to its backup nature and tricky modification and induction process that requires detailed human examination (Steinberger 2010), the check for “D-reducibility” can certainly be brute force through the computerized iteration of all possible colourings of outer vertices. Then the program could attempt to loop through all possible extensions of colours for the inner vertices and interchanging colours if needed. Finally, the program could check if all possible colourings either directly or indirectly have a maximum

of four colours. This way, the computer could check for reducibility systematically and quickly. After all, for each increment (+1) in ring size, the combinations of colourings possible roughly triple that of the previous ring size. For example, with ring size of 14, there would be 199291 possible outer colourings (Walters 2004). That is a number of colourings that humans, by hands alone, could not ever possibly check one by one in a reasonable amount of time and effort.

Figure 2.1: Ring sizes and numbers of possible colourings (Walters 2004).

Ring size	6	7	8	9	10	11	12	13	14
Colourings	31	91	274	820	2461	7381	22144	64430	199291

With the computer, however, even such many colourings could be checked in a fraction of the time it takes for humans to do the same thing. Instead of days and months, the time to check could be reduced to just six hours for ring size 12 with a computer of the day (Wilson 2016). With modern computers, this process would perhaps take mere minutes if not seconds. Thus, with the apparent speed and reliable advantage mentioned before, computers are instrumental to finding reducible configurations which are essential for solving the Four-Colour problem. The program for checking reducible and unavoidable configurations could be written (as done by Appel and Haken later) and let ran for days and weeks at a rate much faster than a human. This would, first of all, save time for the mathematicians as they would not be limited by manually and slowly checking the configurations anymore and can get the results and complete the solution much faster. With the Four-Colour problem, this is very important as there are just so many configurations to check. Secondly, by offloading the checking onto the computer, the mathematicians could focus on a different aspect of the solution, working in parallel with the computer. This would then, once again, allow time to be saved and the solution to be completed quicker than without a computer. Therefore, combined with the advantages mentioned before,

for a complex problem like the Four-Colour problem, the presence of the computer would be incredibly helpful and important for the eventual first correct proof of the problem. And through this, it is clear that computers and mathematics are indeed intertwined with each other.

### **The finalé**

Despite the apparent advantages of computers, the problem is still too much for Heesch to solve by himself. The Four-Colour problem thus remains unproven until Wolfgang Haken and Kenneth Appel appeared on the scene (Walters 2004). They expanded upon Heesch's ideas, finding unavoidable and reducible sets. They even came up with a breakthrough that allows all configurations to have a ring size of no more than 15 (Walters 2004). This allows them to work even more quickly. Eventually, before anyone realized it, the work was done, and the solution was finished. After hundreds of pages of details, over 1,200 hours of computer time, and checking around 2000 cases with up to 500,000 logical operations, the Four-Colour theorem was proved again for the second and final time in 1976 (Walters 2004).

### **The after...math**

Reactions are mixed when Haken presents their proof (Wilson 2016). Due to the complex and massive nature of the proof, it is simply impossible to check by hand without a computer. It is for this reason that many mathematicians remain skeptical of the proof, not believing in the validity of a proof generated by a computer (Wilson 2016). This skepticism sometimes even leads to situations where Haken and Appel were shunted, dismissed, etc. for their proof being nothing but ugly and long (Wilson 2016). It is not beautiful nor elegant, not when the proof requires such a colossal number of cases and computing power that could not be verified without a computer (Wilson 2016). Regardless, Haken and Appel's proof was still accepted officially as

the first complete proof of the Four-Colour theorem (Wilson 2016). Their proof opened a whole new definition of what is a mathematical proof; “What is a Proof Today?” (Wilson 2016). Are computer-assisted proofs still proofs? This Theory of Knowledge question is still getting debated over then and now likewise. The Four-Colour problem had changed what the world thinks about a computer-assisted mathematical proof. No wonder the Four-Colour theorem is one of the most infamous mathematical problems of the 19<sup>th</sup>, 20<sup>th</sup>, and 21<sup>st</sup> centuries.

## **Conclusion**

### **Journey’s end**

Throughout its history as one of the most controversial mathematical theorems in the modern era, the Four-Colour theorem had proved itself to be not only interesting but also world-changing. From Kempe’s original but flawed method of induction to the advanced computation algorithms of Haken and Appel, the Four-Colour theorem had never failed to amaze. If anything, that just shows how beautiful the theorem is, even if the first correct proof is by no means elegant or beautiful. Yet, let not that be a complete downside. Its ugly but ultimately unique and exotic proof led to one of the most significant breakthroughs in modern mathematics. It singlehandedly changed how one considers what is a mathematical proof and demonstrated the role and effects of computers in mathematics. With such a big impact for a seemingly simple theorem, the Four-Colour theorem could very definitely bear the title of one of the most infamous theorems of the 19<sup>th</sup>, 20<sup>th</sup> and 21<sup>st</sup> centuries alike. If only a shorter, more elegant solution is found.

## Appendix

Exhibit 1.1: The author's Python 3.10 code that implements Kempe's method.

```
# Python 3.10

from copy import copy, deepcopy
from time import time

def getNumberOfConnections(vertex):
    return len(vertex["connections"])

def createChainHistory(current, history):
    current.sort(key=getNumberOfConnections)
    history.append(deepcopy(current))
    if len([vertex for vertex in history[-1] if vertex["color"] not in ["r", "g", "b", "y"]]) == 1:
        return history
    for vertexi in [vertex for vertex in current if vertex["id"] in current[0]["connections"]]:
        vertexi["connections"].remove(current[0]["id"])
    current.remove(current[0])
    return createChainHistory(current=deepcopy(current), history=history)

def sameColorRegion(colorPair, vertexB, currentVertex, previousVertex, allVertices):
    if currentVertex["id"] is vertexB["id"]: return True
    possibleNextVertices = [vertex for vertex in allVertices if vertex["id"] in currentVertex["connections"] and vertex["color"] in colorPair and vertex["id"] is not previousVertex["id"]]
    if len(possibleNextVertices) == 0: return False
    isSameColorRegion = False
    for vertexi in possibleNextVertices:
        if sameColorRegion(colorPair=colorPair, vertexB=vertexB, currentVertex=copy(vertexi),
previousVertex=copy(currentVertex), allVertices=allVertices):
            isSameColorRegion = True
            break
    return isSameColorRegion

def invertVertexChain(colorPair, currentVertex, previousVertex, allVertices, changedVertices):
    if currentVertex["color"] is colorPair[1]:
        for vertexi in allVertices:
            if vertexi["id"] is currentVertex["id"]:
                vertexi["color"] = copy(colorPair[0])
                changedVertices.append(copy(vertexi))
    else:
        for vertexi in allVertices:
            if vertexi["id"] is currentVertex["id"]:
                vertexi["color"] = copy(colorPair[1])
```

```

        changedVertices.append(copy(vertexi))
    possibleNextVertices = [vertex for vertex in allVertices if vertex["id"] in currentVertex["connections"] and vertex["color"] in
colorPair and vertex["id"] is not previousVertex["id"] and vertex["id"] not in [vertex["id"] for vertex in changedVertices]]
    if len(possibleNextVertices) == 0: return changedVertices
    for vertexi in possibleNextVertices:
        changedVertices.extend(invertVertexChain(colorPair=colorPair, currentVertex=copy(vertexi),
previousVertex=copy(currentVertex), allVertices=deepcopy(allVertices), changedVertices=copy(changedVertices)))
    changedVertices = list(set(changedVertices))
    for changedVertex in changedVertices:
        for vertexi in allVertices:
            if changedVertex["id"] is vertexi["id"]:
                vertexi["color"] = copy(changedVertex["color"])
    return allVertices

def caseDegreeFour(baseVertex, current):
    vertexA = ""
    vertexB = ""
    for vertexi in current:
        if vertexi["id"] in baseVertex["connections"]:
            vertexA = copy(vertexi)
            break
    for vertexi in current:
        if vertexi["id"] in baseVertex["connections"] and vertexi["id"] not in vertexA["connections"]:
            vertexB = copy(vertexi)

    if sameColorRegion(colorPair=[vertexA["color"], vertexB["color"]], vertexB=copy(vertexB), currentVertex=copy(vertexA),
previousVertex=copy(vertexA), allVertices=deepcopy(current)):
        for vertexi in current:
            if vertexi["id"] == baseVertex["connections"] and vertexi["id"] is not vertexA["id"] and vertexi["id"] is not vertexB["id"]:
                vertexA = copy(vertexi)
                break
        for vertexi in current:
            if vertexi["id"] in baseVertex["connections"] and vertexi["id"] not in vertexA["connections"]:
                vertexA = copy(vertexi)
                break
        current = invertVertexChain(colorPair=[vertexA["color"], vertexB["color"]], currentVertex=copy(vertexA),
previousVertex=copy(vertexA), allVertices=deepcopy(current), changedVertices=[])
        colorsUsed = [vertex["color"] for vertex in current if vertex["id"] in baseVertex["connections"]]
        [vertex for vertex in current if vertex["id"] is baseVertex["id"]][0]["color"] = [color for color in ["r", "g", "b", "y"] if color not in
colorsUsed][0]
        return current

def colorGraph(history):
    current = history.pop()

    while len(history) != 0:

```

```

for vertexHistory in history[-1]:
    for vertexCurrent in current:
        if vertexCurrent["id"] is vertexHistory["id"]:
            vertexHistory["color"] = copy(vertexCurrent["color"])
            break
current = history.pop()
for vertexi in current:
    if vertexi["color"] in ["r", "g", "b", "y"]: continue
    numberOfConnections = len(vertexi["connections"])
    if numberOfConnections < 4:
        colorsUsed = [vertex["color"] for vertex in current if vertex["id"] in vertexi["connections"]]
        vertexi["color"] = [color for color in ["r", "g", "b", "y"] if color not in colorsUsed][0]
        continue
    if numberOfConnections == 4:
        current = deepcopy(caseDegreeFour(baseVertex=copy(vertexi), current=deepcopy(current)))
        continue
    if numberOfConnections == 5:
        continue
return current

def checkAndPrintResult(output):
    ansGud = True
    for vertexi in output:
        colorsUsed = [vertex["color"] for vertex in output if vertex["id"] in vertexi["connections"]]
    if vertexi["color"] in colorsUsed:
        ansGud = False
    print(output)
    if ansGud:
        print("Gud - HELLL YEAHHH!")
    else:
        print("Bad - NOOOOOOOOOO!!!")

if __name__ == "__main__":
    startTime = time()

    input = #Input array here

    checkAndPrintResult(colorGraph(createChainHistory(current=deepcopy(input), history=[])))

    endTime = time()
    print("Execution time:", (endTime-startTime), "s")

```



## References

“Four Color Theorem and Kuratowski’s Theorem in Discrete Mathematics.” 2021.

GeeksforGeeks. July 10, 2021. <https://www.geeksforgeeks.org/four-color-theorem-and-kuratowskis-theorem-in-discrete-mathematics/>.

Appel, Kenneth, and Wolfgang Haken. “The Solution of the Four-Color-Map Problem.”

*Scientific American* 237, no. 4 (1977): 108–21.

<https://doi.org/10.1038/scientificamerican1077-108>.

Kempe, A. B. 1879. “On the Geographical Problem of the Four Colours.” *American Journal of Mathematics* 2 (3): 193–200. <https://doi.org/10.2307/2369235>.

Kevin. 2017. “Canada Map.” Guide of the World. June 28, 2017.

<https://www.guideoftheworld.com/canada-map.html>.

Sipka, Timothy. 2002. “Alfred Bray Kempe’s ‘Proof’ of the Four-Color Theorem.” *Math Horizons* 10 (2): 21–26. <https://doi.org/10.1080/10724117.2002.11974616>.

Steinberger, John P. 2010. “An Unavoidable Set of  $SD$ -Reducible Configurations.” *Transactions of the American Mathematical Society* 362 (12): 6633–33.

<https://doi.org/10.1090/s0002-9947-2010-05092-5>.

Walters, Mark. “It Appears That Four Colors Suffice: A Historical Overview of the Four-Color Theorem.” (2004).

Wilson, Robin. 2016. “Wolfgang Haken and the Four-Color Problem.” *Illinois Journal of Mathematics* 60 (1). <https://doi.org/10.1215/ijm/1498032028>.