Welcome to CR12 !
Little history of Four-Color Theorem proofs

Michaël Rao

CNRS - ENS Lyon
LIP - Laboratoire de l'Informatique du Parallélisme
équipe MC2

by Pascal Ochem (LIRMM, Montpellier) and Michael Rao (me, LIP)

Recall :

- How a computer can help in mathematics or computer science ?
- Presentation of the history/proof of some computer aided theorems (TODAY : 4 color theorem)
- Presentation of classic tools for combinatorial exploration : reduction to SAT or LP, backtracking, transfer matrix methods...
- Concrete examples of different approaches to accelerate explorations : reduction of the search space, heuristics, speed up code...
- Lot of practise !
- Evaluation by homework and projects.
- Note : no "Formal proof assistants" like Coq here !

Today : One see the history of the Four color theorem (and ideas of the proofs)

Today : One see the history of the Four color theorem (and ideas of the proofs)

Why :

- One of the first "big theorem" proved using the help of the computer, and explains well several parts of the advantages, disadvantages, scepticism of this kind of approaches
- lot of scepticism at the beginning
- some other proofs later always using computer
- up to now : no "simple" proofs, no proof by hand...
- also, we will (quickly) show the "discharging method", a tool often used in planar graph theory

Today : One see the history of the Four color theorem (and ideas of the proofs)
Why :

- One of the first “big theorem” proved using the help of the computer, and explains well several parts of the advantages, disadvantages, scepticism of this kind of approaches
- lot of scepticism at the beginning
- some other proofs later always using computer
- up to now : no “simple” proofs, no proof by hand...
- also, we will (quickly) show the “discharging method”, a tool often used in planar graph theory

Tomorrow : practise with transfer matrices and power iteration.

Today : One see the history of the Four color theorem (and ideas of the proofs)

Why :

- One of the first "big theorem" proved using the help of the computer, and explains well several parts of the advantages, disadvantages, scepticism of this kind of approaches
- lot of scepticism at the beginning
- some other proofs later always using computer
- up to now : no "simple" proofs, no proof by hand...
- also, we will (quickly) show the "discharging method", a tool often used in planar graph theory

Tomorrow : practise with transfer matrices and power iteration.

Next week : Pascal Ochem will talk about backtracking (and others things)

How a computer can help a mathematician ?

- Computer algebra system (Maple, Mathematica, Sage...)

How a computer can help a mathematician ?

- Computer algebra system (Maple, Mathematica, Sage...)

  A computer can do exact calculus with rationals, algebraic number... and usually without error.

- Test an idea or a conjecture on numerous examples

How a computer can help a mathematician ?

- Computer algebra system (Maple, Mathematica, Sage...)

  A computer can do exact calculus with rationals, algebraic number... and usually without error.

- Test an idea or a conjecture on numerous examples
- Find an "object" with some properties

How a computer can help a mathematician ?

- Computer algebra system (Maple, Mathematica, Sage...)

  A computer can do exact calculus with rationals, algebraic number... and usually without error.
- Test an idea or a conjecture on numerous examples
- Find an "object" with some properties
  - Either to prove the existence of something

How a computer can help a mathematician ?

- Computer algebra system (Maple, Mathematica, Sage...)

  A computer can do exact calculus with rationals, algebraic number... and usually without error.
- Test an idea or a conjecture on numerous examples
- Find an "object" with some properties
  - Either to prove the existence of something
  - Or to "break" a conjecture

How a computer can help a mathematician ?

- Computer algebra system (Maple, Mathematica, Sage...)

  A computer can do exact calculus with rationals, algebraic number... and usually without error.
- Test an idea or a conjecture on numerous examples
- Find an "object" with some properties
  - Either to prove the existence of something
  - Or to "break" a conjecture
- Case study : prove something on numerous but "similar" cases

How a computer can help a mathematician ?

- Computer algebra system (Maple, Mathematica, Sage...)

  A computer can do exact calculus with rationals, algebraic number... and usually without error.

- Test an idea or a conjecture on numerous examples
- Find an "object" with some properties
  - Either to prove the existence of something
  - Or to "break" a conjecture
- Case study : prove something on numerous but "similar" cases
- Formal proof assistants (Coq...)

How a computer can help a mathematician ?

- Computer algebra system (Maple, Mathematica, Sage...)

  A computer can do exact calculus with rationals, algebraic number... and usually without error.
- Test an idea or a conjecture on numerous examples
- Find an "object" with some properties
  - Either to prove the existence of something
  - Or to "break" a conjecture
- Case study : prove something on numerous but "similar" cases
- Formal proof assistants (Coq...)

Advantages of the computer : rigorous, and are quick

How a computer can help a mathematician ?

- Computer algebra system (Maple, Mathematica, Sage...)

  A computer can do exact calculus with rationals, algebraic number... and usually without error.
- Test an idea or a conjecture on numerous examples
- Find an "object" with some properties
  - Either to prove the existence of something
  - Or to "break" a conjecture
- Case study : prove something on numerous but "similar" cases
- Formal proof assistants (Coq...)

Advantages of the computer : rigorous, and are quick

Disadvantages : know how to program, sometimes it's more difficult to find "errors" the the proofs, and the "power" of the computer is not infinite...

Some problems solved with the help of a computer

- Four-Color Theorem (Appel & Haken, 1976, and others...)

Some problems solved with the help of a computer

- Four-Color Theorem (Appel & Haken, 1976, and others...)
- Proof of Kepler Conjecture(Hales & Ferguson, 1998)
- Non-existence of a finite projective plane of order 10(Lam, 1991)
- Games : 4 in a row, Awalé, Checkers 8×8...
- Rubik's cube, Sudoku...
- The smallest aperiodic Wang tileset is 11 (Jeandel & Rao, 2015)
- ...

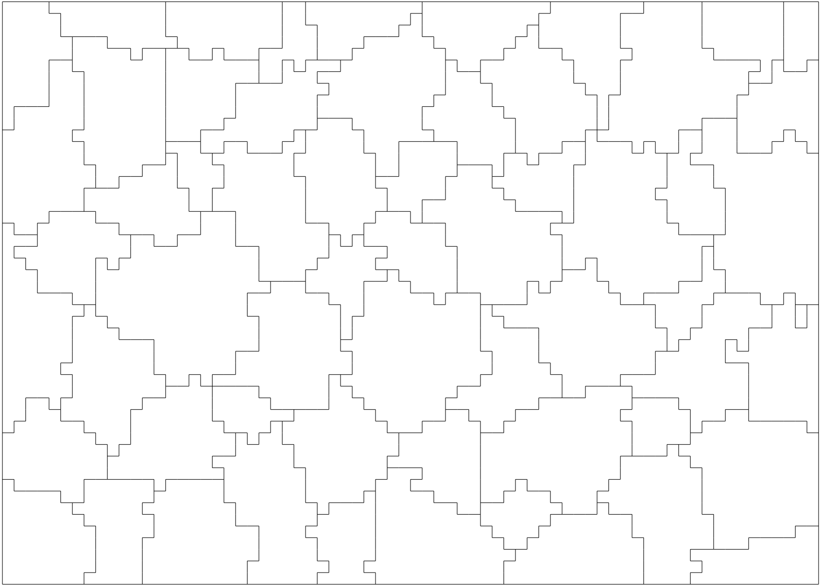
Some problems solved with the help of a computer

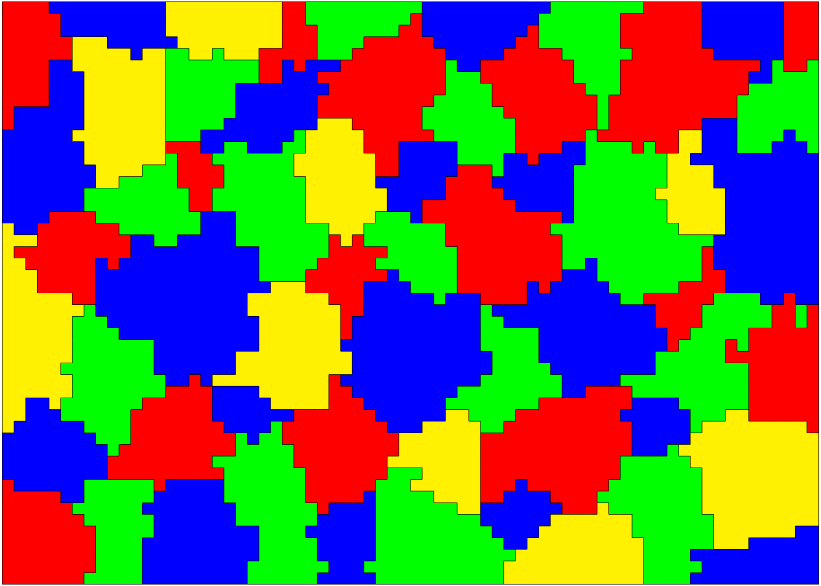
- Four-Color Theorem (Appel & Haken, 1976, and others...)
- Proof of Kepler Conjecture(Hales & Ferguson, 1998)
- Non-existence of a finite projective plane of order 10(Lam, 1991)
- Games : 4 in a row, Awalé, Checkers 8×8...
- Rubik's cube, Sudoku...
- The smallest aperiodic Wang tileset is 11 (Jeandel & Rao, 2015)
- ...

The Four-Color Theorem

Every map can be coloured with (at most) 4 colors, in such a way that no two adjacent regions have the same color.

(note : regions are connected)





The Four-Color Theorem

Every map can be coloured with (at most) 4 colors, in such a way that no two adjacent regions have the same color.

(note : regions are connected)

The Four-Color Theorem

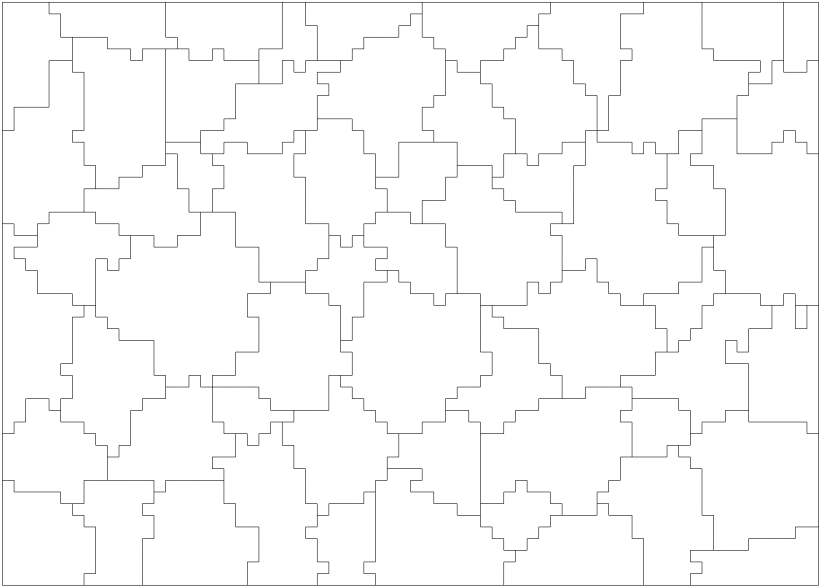
Every map can be coloured with (at most) 4 colors, in such a way that no two adjacent regions have the same color.

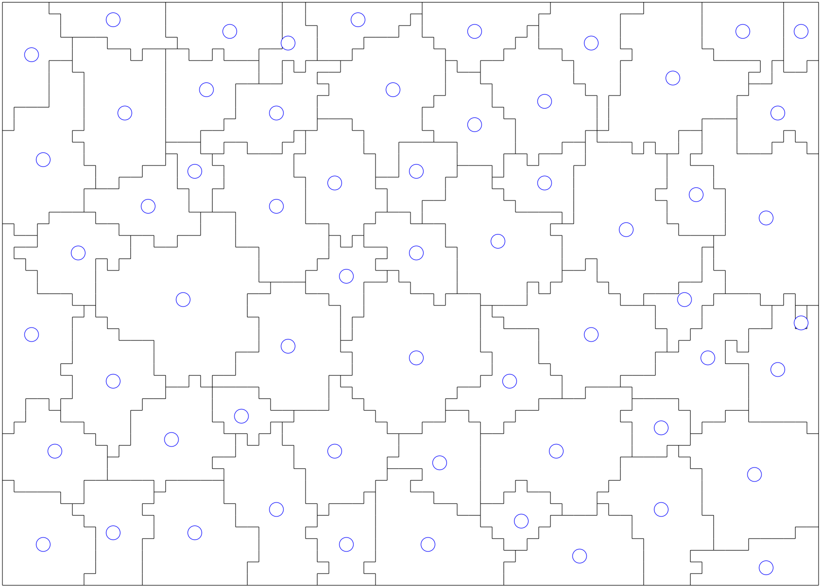
(note : regions are connected)

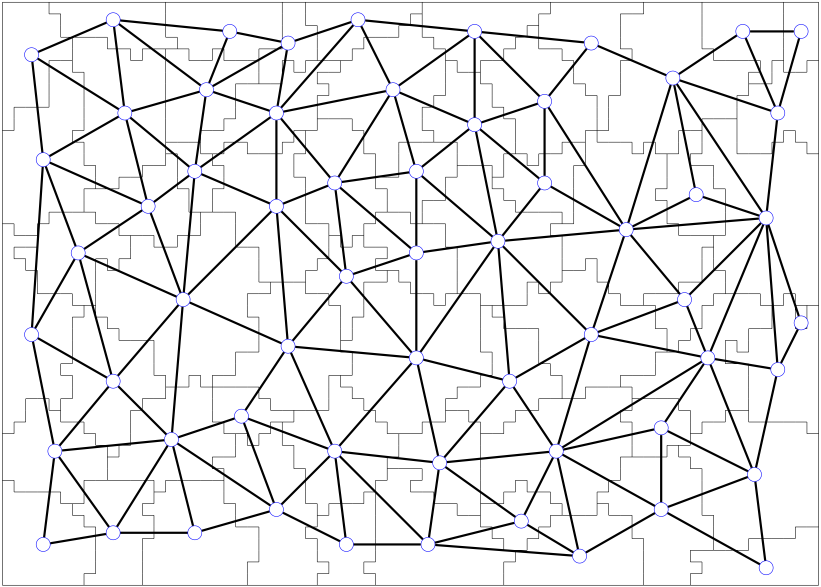
In "graph theory" :

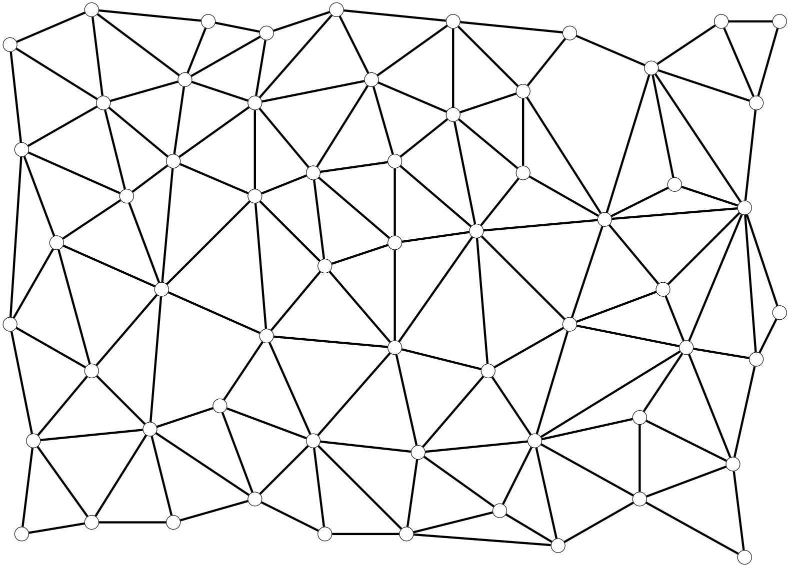
The Four-Color Theorem

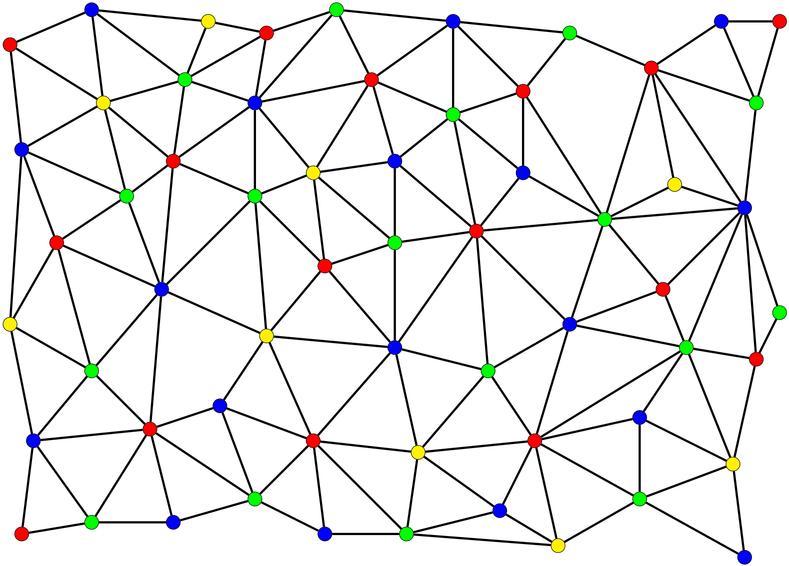
Every planar graph is 4-colorable











Four-Color Theorem : history

- Conjecture stated by Guthrie in 1852

Four-Color Theorem : history

- Conjecture stated by Guthrie in 1852
- Proof by Kempe in 1879

Lemma

In every planar graph, there is a vertex with degree at most 5.

Lemma

In every planar graph, there is a vertex with degree at most 5.

proof. Using the Euler formula :

$$\#\text{vertices} - \#\text{edges} + \#\text{faces} = 2$$

Lemma

In every planar graph, there is a vertex with degree at most 5.

proof. Using the Euler formula :

$$\#\text{vertices} - \#\text{edges} + \#\text{faces} = 2$$

A face has at least 3 edges :

Lemma

In every planar graph, there is a vertex with degree at most 5.

proof. Using the Euler formula :

$$\#\text{vertices} - \#\text{edges} + \#\text{faces} = 2$$

A face has at least 3 edges : $\#\text{faces} \leq \frac{2}{3} \times \#\text{edges}$

Euler formula and 6 colors

Lemma

In every planar graph, there is a vertex with degree at most 5.

proof. Using the Euler formula :

$$\#\text{vertices} - \#\text{edges} + \#\text{faces} = 2$$

A face has at least 3 edges : $\#\text{faces} \leq \frac{2}{3} \times \#\text{edges}$
If every vertices has a degree ≥ 6 :

Euler formula and 6 colors

Lemma

In every planar graph, there is a vertex with degree at most 5.

proof. Using the Euler formula :

$$\#\text{vertices} - \#\text{edges} + \#\text{faces} = 2$$

A face has at least 3 edges : $\#\text{faces} \leq \frac{2}{3} \times \#\text{edges}$
If every vertices has a degree ≥ 6 : $\#\text{vertices} \leq \frac{2}{6} \times \#\text{edges}$

Euler formula and 6 colors

Lemma

In every planar graph, there is a vertex with degree at most 5.

proof. Using the Euler formula :

$$\#\text{vertices} - \#\text{edges} + \#\text{faces} = 2$$

A face has at least 3 edges : $\#\text{faces} \leq \frac{2}{3} \times \#\text{edges}$
If every vertices has a degree ≥ 6 : $\#\text{vertices} \leq \frac{2}{6} \times \#\text{edges}$

$$\#\text{faces} + \#\text{vertices} \leq \left(\frac{2}{6} + \frac{2}{3}\right) \times \#\text{edges}$$

Euler formula and 6 colors

Lemma

In every planar graph, there is a vertex with degree at most 5.

proof. Using the Euler formula :

$$\#\text{vertices} - \#\text{edges} + \#\text{faces} = 2$$

A face has at least 3 edges : $\#\text{faces} \leq \frac{2}{3} \times \#\text{edges}$
If every vertices has a degree ≥ 6 : $\#\text{vertices} \leq \frac{2}{6} \times \#\text{edges}$

$$\#\text{faces} + \#\text{vertices} \leq \left(\frac{2}{6} + \frac{2}{3}\right) \times \#\text{edges}$$

Contradiction !

Euler formula and 6 colors

Every planar graph is 6-colorable :

Recursive algorithm to color with 6 colors

- Let v be a vertex of G with degree at most 5
- We color $G - v$ with 6 colors ("recursive call")
- At least, one of the 6 colors is not used by the neighbors of v. We color v with a free color.

Euler formula and 6 colors

Every planar graph is 6-colorable :

Recursive algorithm to color with 6 colors

- Let v be a vertex of G with degree at most 5
- We color $G - v$ with 6 colors ("recursive call")
- At least, one of the 6 colors is not used by the neighbors of v. We color v with a free color.

One can call it the "6-Color Theorem".

6-Color Theorem

Recursive algorithm to color a planar graph G with 6 colors.

6-Color Theorem

Recursive algorithm to color a planar graph G with 6 colors.

- Let v be a vertex of G with degree at most 5

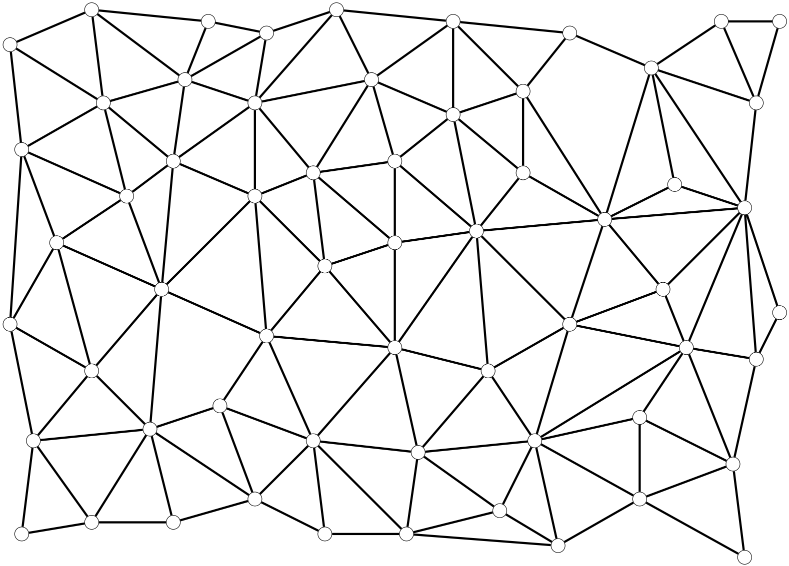
Recursive algorithm to color a planar graph G with 6 colors.

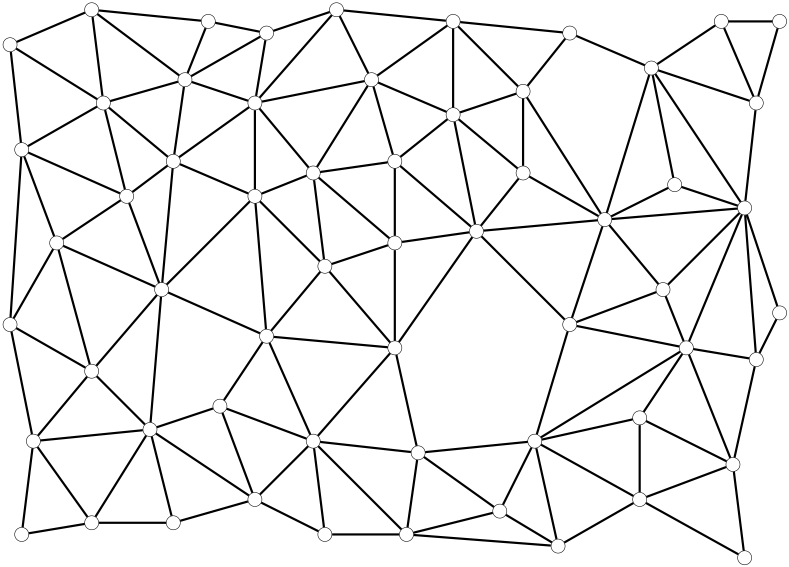
- Let v be a vertex of G with degree at most 5
- We color $G - v$ with 6 colors ("recursive call")

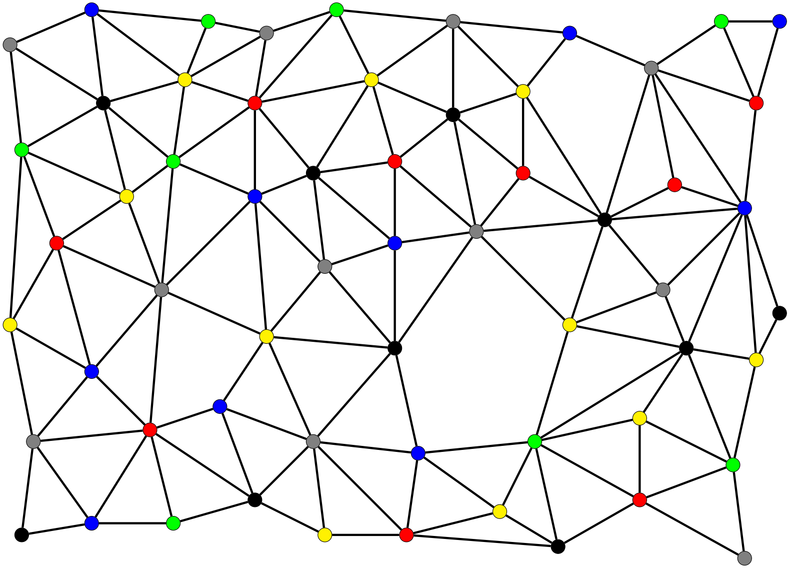
6-Color Theorem

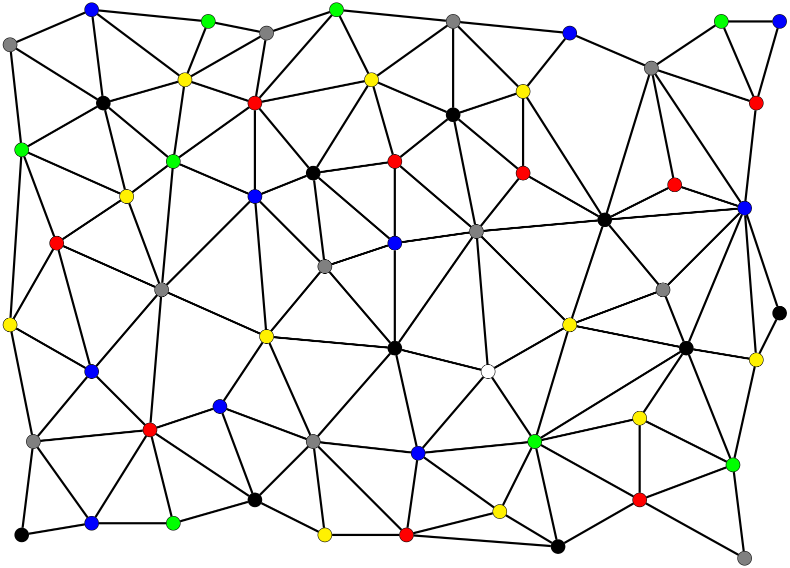
Recursive algorithm to color a planar graph G with 6 colors.

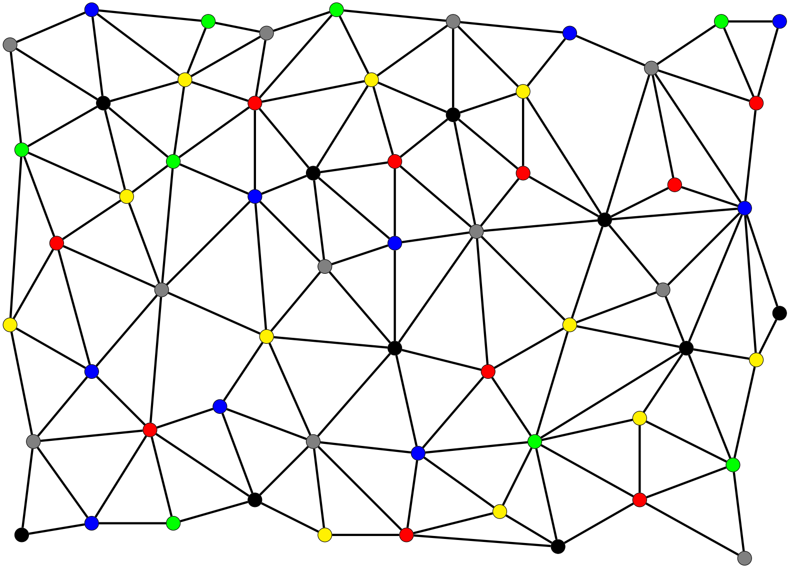
- Let v be a vertex of G with degree at most 5
- We color $G - v$ with 6 colors ("recursive call")
- At least, one of the 6 colors is not used by the neighbors of v. We color v with a free color.











6-Color Theorem

Recursive algorithm to color a planar graph G with 6 colors.

- Let v be a vertex of G with degree at most 5
- We color $G - v$ with 6 colors ("recursive call")
- At least, one of the 6 colors is not used by the neighbors of v. We color v with a free color.

Four-Color Theorem (Kempe)

Recursive algorithm to color a planar graph G with 4 colors :

Four-Color Theorem (Kempe)

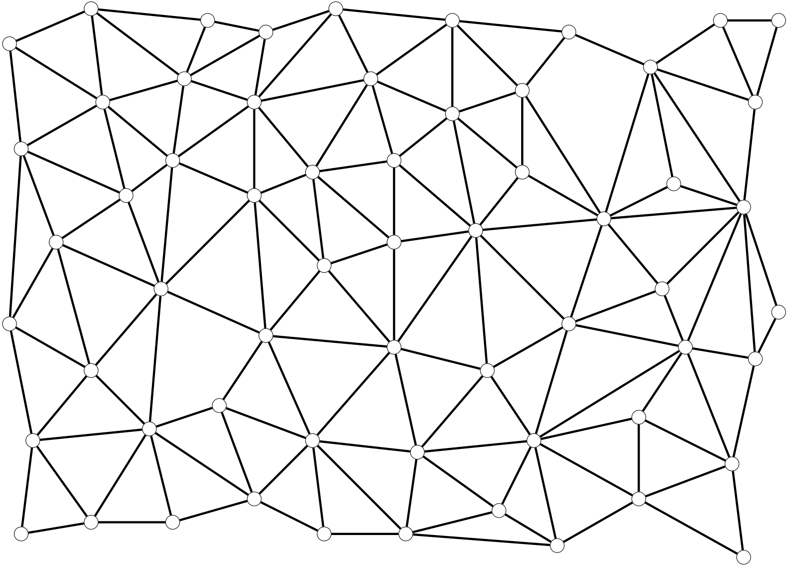
Recursive algorithm to color a planar graph G with 4 colors :

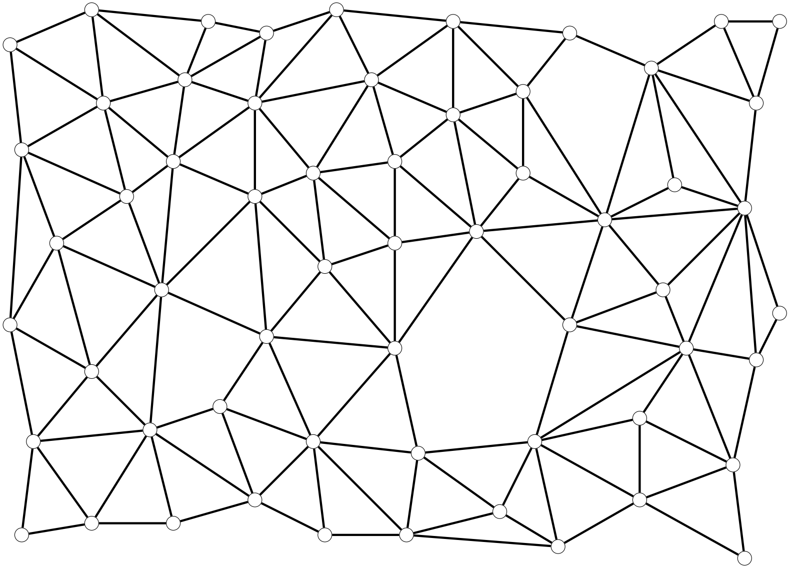
- Let v be a vertex of G with degree at most 5

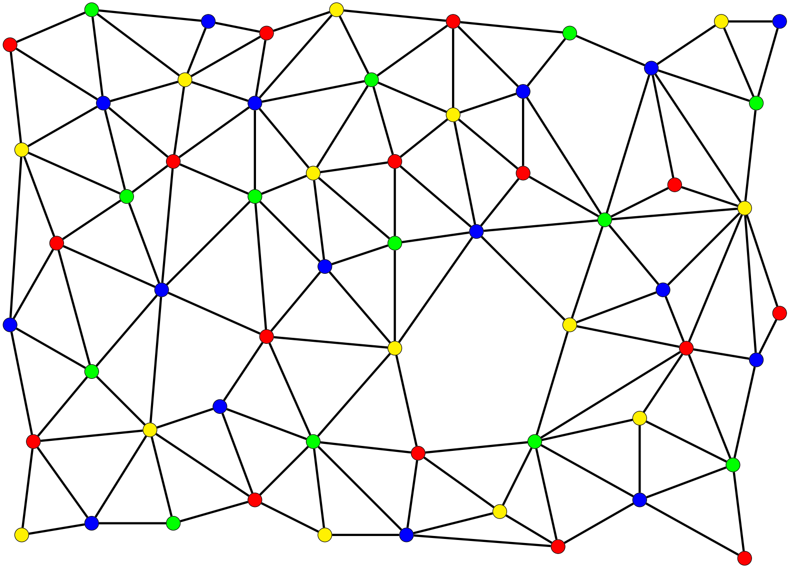
Four-Color Theorem (Kempe) ?

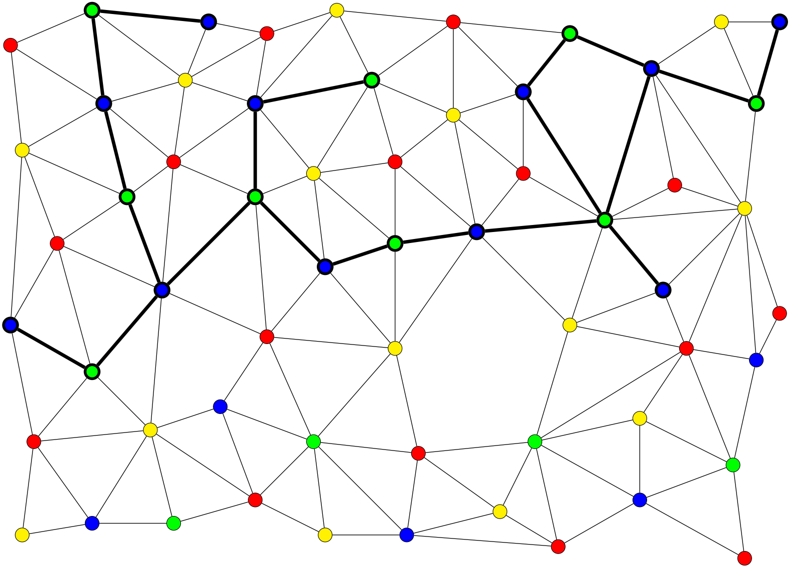
Recursive algorithm to color a planar graph G with 4 colors :

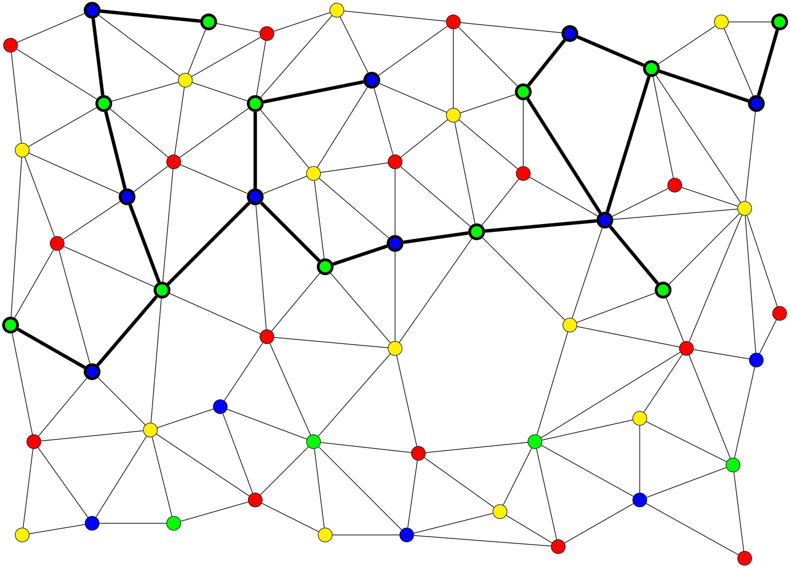
- Let v be a vertex of G with degree at most 5
- We color $G - v$ with 4 colors ("recursive call")
- If one of the 4 colors are not used among the neighbors of v in G, use it to color v
- Otherwise, do local changes (via "Kempe chains") to « win » one color.

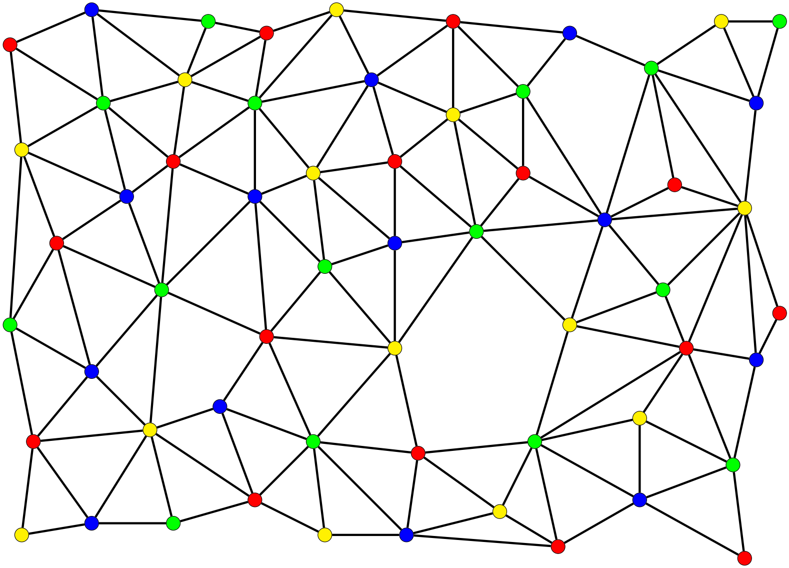


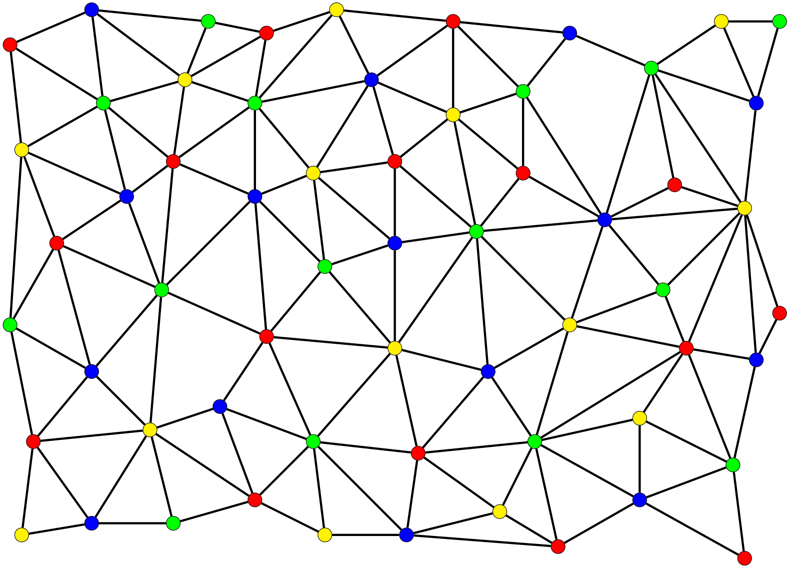


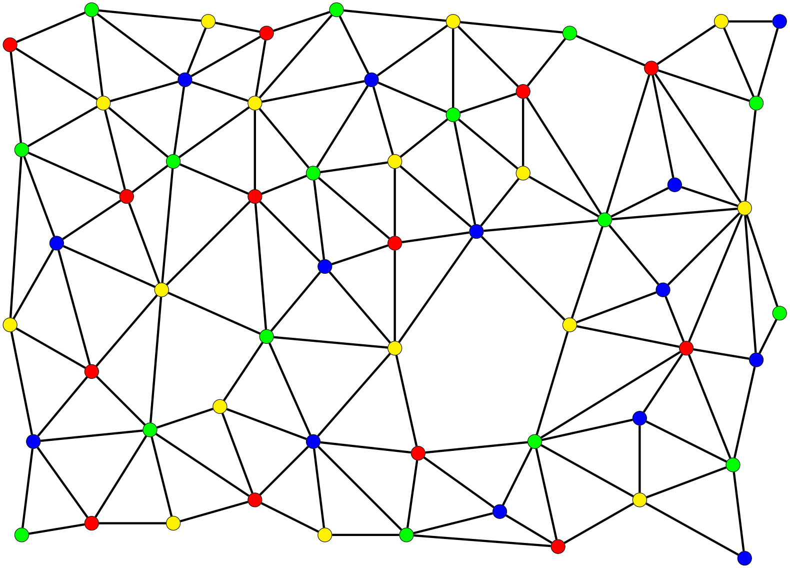


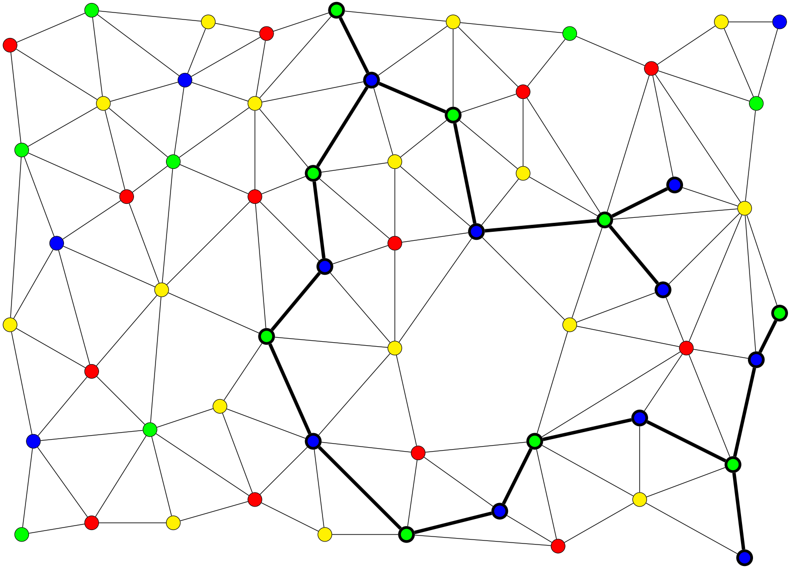


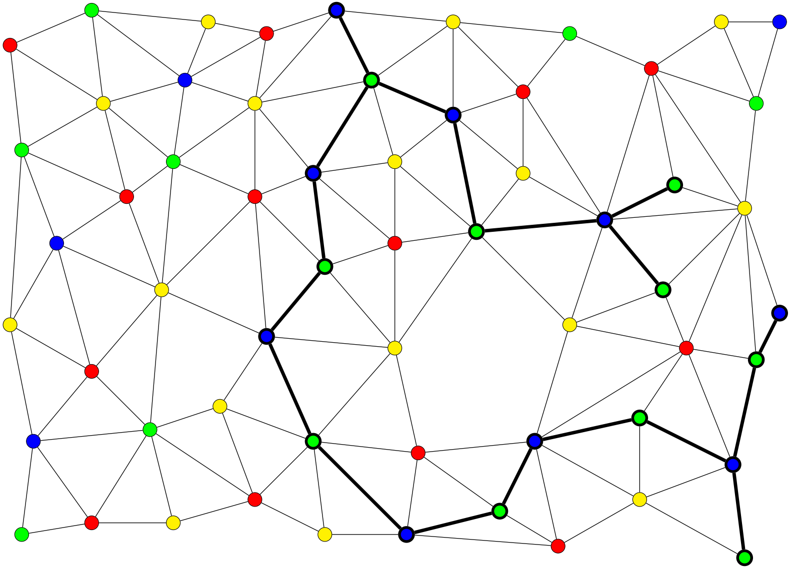


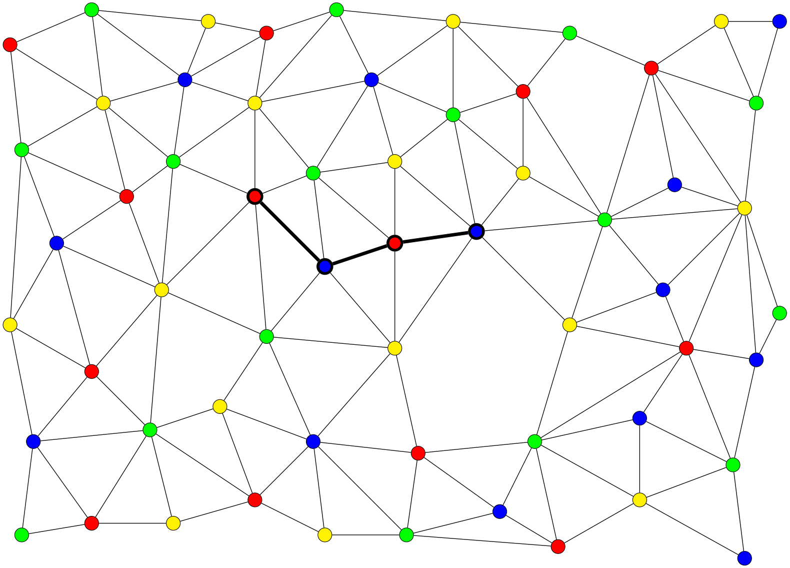


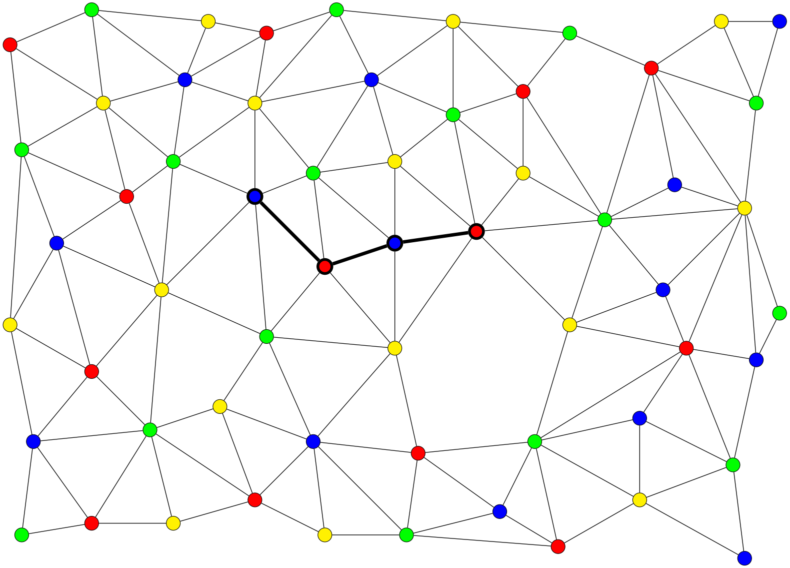


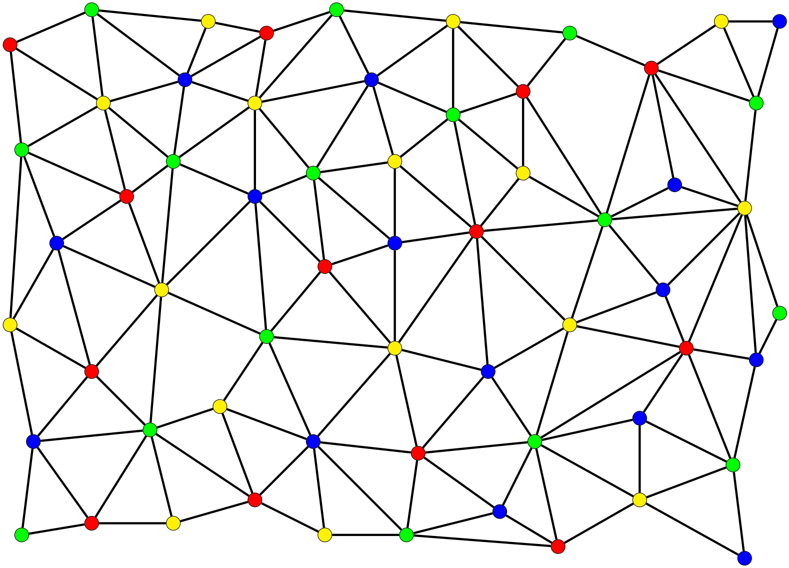












Four-Color Theorem (Kempe) ?

Recursive algorithm to color a planar graph G with 4 colors :

- Let v be a vertex of G with degree at most 5
- We color $G - v$ with 4 colors ("recursive call")
- If one of the 4 colors are not used among the neighbors of v in G, use it to color v
- Otherwise, do local changes (via "Kempe chains") to « win » one color.

Four-Color Theorem (Kempe)

Recursive algorithm to color a planar graph G with 4 colors :

- Let v be a vertex of G with degree at most 5
- We color $G - v$ with 4 colors ("recursive call")
- If one of the 4 colors are not used among the neighbors of v in G, use it to color v
- Otherwise, do local changes (via "Kempe chains") to « win » one color.

Every planar graph is 4-colorable

The four color conjecture is a theorem

Four-Color Theorem (Kempe)

Recursive algorithm to color a planar graph G with 4 colors :

- Let v be a vertex of G with degree at most 5
- We color $G - v$ with 4 colors ("recursive call")
- If one of the 4 colors are not used among the neighbors of v in G, use it to color v
- Otherwise, do local changes (via "Kempe chains") to « win » one color.

Every planar graph is 4-colorable

The four color conjecture is a theorem (?)

Four-Color Theorem : history

- Conjecture stated by Guthrie in 1852
- Proof by Kempe in 1879

Four-Color Theorem : history

- Conjecture stated by Guthrie in 1852
- ~~Proof~~ by Kempe in 1879
- Heawood finds a flaw in the proof of Kempe in 1890 !

Four-Color Theorem : history

- Conjecture stated by Guthrie in 1852
- ~~Proof~~ by Kempe in 1879
- Alternative proof by Tait in 1880
- Heawood finds a flaw in the proof of Kempe in 1890 !

Four-Color Theorem : history

- Conjecture stated by Guthrie in 1852
- ~~Proof~~ by Kempe in 1879
- ~~Alternative proof~~ by Tait in 1880
- Heawood finds a flaw in the proof of Kempe in 1890 !
- Petersen finds a flaw in the proof of Tait in 1891 !

Four-Color Theorem : history

- Conjecture stated by Guthrie in 1852
- ~~Proof~~ by Kempe in 1879
- ~~Alternative proof~~ by Tait in 1880
- Heawood finds a flaw in the proof of Kempe in 1890 !
- Petersen finds a flaw in the proof of Tait in 1891 !
- Heesch in 1955 : first idea of a "computer" proof scheme

Four-Color Theorem : history

- Conjecture stated by Guthrie in 1852
- ~~Proof~~ by Kempe in 1879
- ~~Alternative proof~~ by Tait in 1880
- Heawood finds a flaw in the proof of Kempe in 1890 !
- Petersen finds a flaw in the proof of Tait in 1891 !
- Heesch in 1955 : first idea of a "computer" proof scheme but the number of configurations to test is too big for that moment

Four-Color Theorem : history

- Conjecture stated by Guthrie in 1852
- ~~Proof~~ by Kempe in 1879
- ~~Alternative proof~~ by Tait in 1880
- Heawood finds a flaw in the proof of Kempe in 1890 !
- Petersen finds a flaw in the proof of Tait in 1891 !
- Heesch in 1955 : first idea of a "computer" proof scheme but the number of configurations to test is too big for that moment
- Proof by Appel & Haken in 1977 (with the help of the computer for some parts)

Four-Color Theorem : history

- Conjecture stated by Guthrie in 1852
- ~~Proof~~ by Kempe in 1879
- ~~Alternative proof~~ by Tait in 1880
- Heawood finds a flaw in the proof of Kempe in 1890 !
- Petersen finds a flaw in the proof of Tait in 1891 !
- Heesch in 1955 : first idea of a "computer" proof scheme but the number of configurations to test is too big for that moment
- Proof by Appel & Haken in 1977 (with the help of the computer for some parts) but a log of criticism !

Four-Color Theorem : history

- Conjecture stated by Guthrie in 1852
- ~~Proof~~ by Kempe in 1879
- ~~Alternative proof~~ by Tait in 1880
- Heawood finds a flaw in the proof of Kempe in 1890 !
- Petersen finds a flaw in the proof of Tait in 1891 !
- Heesch in 1955 : first idea of a "computer" proof scheme but the number of configurations to test is too big for that moment
- Proof by Appel & Haken in 1977 (with the help of the computer for some parts) but a log of criticism !
- Proof by Robertson, Sanders, Seymour & Thomas in 1997 (with the help of the computer for every parts)

Four-Color Theorem : history

- Conjecture stated by Guthrie in 1852
- ~~Proof~~ by Kempe in 1879
- ~~Alternative proof~~ by Tait in 1880
- Heawood finds a flaw in the proof of Kempe in 1890 !
- Petersen finds a flaw in the proof of Tait in 1891 !
- Heesch in 1955 : first idea of a "computer" proof scheme but the number of configurations to test is too big for that moment
- Proof by Appel & Haken in 1977 (with the help of the computer for some parts) but a log of criticism !
- Proof by Robertson, Sanders, Seymour & Thomas in 1997 (with the help of the computer for every parts) much less criticism ! but...

Four-Color Theorem : history

- Conjecture stated by Guthrie in 1852
- ~~Proof~~ by Kempe in 1879
- ~~Alternative proof~~ by Tait in 1880
- Heawood finds a flaw in the proof of Kempe in 1890 !
- Petersen finds a flaw in the proof of Tait in 1891 !
- Heesch in 1955 : first idea of a "computer" proof scheme but the number of configurations to test is too big for that moment
- Proof by Appel & Haken in 1977 (with the help of the computer for some parts) but a log of criticism !
- Proof by Robertson, Sanders, Seymour & Thomas in 1997 (with the help of the computer for every parts) much less criticism ! but...
- Verification of the proof by Gonthier in 2005, using a "Formal proof assistant" (Coq)

Go back to Kempe "proof" : an error, but...

The proof of Keepe has an error : two "Kempe chains" can cross...

Go back to Kempe "proof" : an error, but...

The proof of Keepe has an error : two "Kempe chains" can cross...

But, there are already a lot of ideas inside !

Go back to Kempe "proof" : an error, but...

The proof of Keepe has an error : two "Kempe chains" can cross...

But, there are already a lot of ideas inside !

The "recursive approach" and the general outline was the "good" one, and used in the (correct) proofs

Go back to Kempe "proof" : an error, but...

The proof of Keepe has an error : two "Kempe chains" can cross...

But, there are already a lot of ideas inside !

The "recursive approach" and the general outline was the "good" one, and used in the (correct) proofs

The ideas of "Kempe chains" was also useful in (correct) proofs

Go back to Kempe “proof” : an error, but...

The proof of Keepe has an error : two “Kempe chains” can cross...

But, there are already a lot of ideas inside !

The “recursive approach” and the general outline was the “good” one, and used in the (correct) proofs

The ideas of “Kempe chains” was also useful in (correct) proofs

But there are (much) more cases to manage...

Idea for a proof (which is also an algorithm)

- If the theorem is false, there is a minimal counter-example (in number of vertices) : a planar graph which is not 4-colorable, but if we remove a vertex, it becomes 4-colorable

Idea for a proof (which is also an algorithm)

- If the theorem is false, there is a minimal counter-example (in number of vertices) : a planar graph which is not 4-colorable, but if we remove a vertex, it becomes 4-colorable
- Now, try to find "reducibles" configurations

Idea for a proof (which is also an algorithm)

- If the theorem is false, there is a minimal counter-example (in number of vertices) : a planar graph which is not 4-colorable, but if we remove a vertex, it becomes 4-colorable
- Now, try to find "reducibles" configurations

What is a "reducible" configuration ?

Idea for a proof (which is also an algorithm)

- If the theorem is false, there is a minimal counter-example (in number of vertices) : a planar graph which is not 4-colorable, but if we remove a vertex, it becomes 4-colorable
- Now, try to find "reducibles" configurations

What is a "reducible" configuration ?

A **configuration** is a planar graph with an "external" face.

Idea for a proof (which is also an algorithm)

- If the theorem is false, there is a minimal counter-example (in number of vertices) : a planar graph which is not 4-colorable, but if we remove a vertex, it becomes 4-colorable
- Now, try to find "reducibles" configurations

What is a "reducible" configuration ?

A **configuration** is a planar graph with an "external" face.

A configuration C appears in G if there is a injection between edges/vertices/inner faces of C and some edges/vertices/faces of G which respects the adjacencies.

Idea for a proof (which is also an algorithm)

- If the theorem is false, there is a minimal counter-example (in number of vertices) : a planar graph which is not 4-colorable, but if we remove a vertex, it becomes 4-colorable
- Now, try to find “reducibles” configurations

What is a “reducible” configuration ?

A **configuration** is a planar graph with an “external” face.

A configuration C appears in G if there is a injection between edges/vertices/inner faces of C and some edges/vertices/faces of G which respects the adjacencies.

Now (roughly) a configuration is **reducible** if G can be 4-colorated if we know a coloration of G-C ;

Idea for a proof (which is also an algorithm)

- If the theorem is false, there is a minimal counter-example (in number of vertices) : a planar graph which is not 4-colorable, but if we remove a vertex, it becomes 4-colorable
- Now, try to find "reducibles" configurations

What is a "reducible" configuration ?

A **configuration** is a planar graph with an "external" face.

A configuration C appears in G if there is a injection between edges/vertices/inner faces of C and some edges/vertices/faces of G which respects the adjacencies.

Now (roughly) a configuration is **reducible** if G can be 4-colorated if we know a coloration of G-C ;

Thus : a reducible configuration cannot appear in a minimal counter example !

reducible configurations

There are several type of “reducible” configuration... We already see some

There are several type of “reducible” configuration... We already see some

Some easy “reducibles” configurations :

reducible configurations

There are several type of "reducible" configuration... We already see some

Some easy "reducibles" configurations :

- a vertex of degree at most 3

reducible configurations

There are several type of “reducible” configuration... We already see some

Some easy “reducibles” configurations :

- a vertex of degree at most 3
- a vertex of degree 4

reducible configurations

There are several type of "reducible" configuration... We already see some

Some easy "reducibles" configurations :

- a vertex of degree at most 3
- a vertex of degree 4 (Kempe method works for them)

reducible configurations

There are several type of "reducible" configuration... We already see some

Some easy "reducibles" configurations :

- a vertex of degree at most 3
- a vertex of degree 4 (Kempe method works for them)

But there are others reducible configurations. E.g :

- if for every valid coloring of the border of C, one can extend it to a coloring if the rest of C, C is reducible.

There are several type of "reducible" configuration... We already see some

Some easy "reducibles" configurations :

- a vertex of degree at most 3
- a vertex of degree 4 (Kempe method works for them)

But there are others reducible configurations. E.g :

- if for every valid coloring of the border of C, one can extend it to a coloring if the rest of C, C is reducible.
- if there exists two configurations C and C' such that $|C| > |C'|$ and the set of the colorings of the borders of C' is a subset of the colorings of the borders of C (i.e., C' "mimic" C), then C is reducible.

reducible configurations

There are several type of "reducible" configuration... We already see some

Some easy "reducibles" configurations :

- a vertex of degree at most 3
- a vertex of degree 4 (Kempe method works for them)

But there are others reducible configurations. E.g :

- if for every valid coloring of the border of C, one can extend it to a coloring if the rest of C, C is reducible.
- if there exists two configurations C and C' such that $|C| > |C'|$ and the set of the colorings of the borders of C' is a subset of the colorings of the borders of C (i.e., C' "mimic" C), then C is reducible.

This can be checked by computer : just test every set of colorings.

reducible configurations

There are several type of "reducible" configuration... We already see some

Some easy "reducibles" configurations :

- a vertex of degree at most 3
- a vertex of degree 4 (Kempe method works for them)

But there are others reducible configurations. E.g :

- if for every valid coloring of the border of C, one can extend it to a coloring if the rest of C, C is reducible.
- if there exists two configurations C and C' such that $|C| > |C'|$ and the set of the colorings of the borders of C' is a subset of the colorings of the borders of C (i.e., C' "mimic" C), then C is reducible.

This can be checked by computer : just test every set of colorings. It's stupid and very repetitive, so it's perfect for a computer.

Now, how to prove that a set of configuration is unavoidable on a planar graph ?

Now, how to prove that a set of configuration is unavoidable on a planar graph ? Using the **discharging method** !
(this tool is often used in planar graph theory)

Start : we suppose that one have a planar graph G, avoiding every configuration in C

1. We put weight on the vertices (and faces), in such a way that the sum is negative. For this, we use Euler's formula

Now, how to prove that a set of configuration is unavoidable on a planar graph ? Using the **discharging method** !
(this tool is often used in planar graph theory)

Start : we suppose that one have a planar graph G, avoiding every configuration in C

1. We put weight on the vertices (and faces), in such a way that the sum is negative. For this, we use Euler's formula
2. We "move" weights according local rules.
   The overall sum does not change

Discharging : proving that a set is unavoidable

Now, how to prove that a set of configuration is unavoidable on a planar graph ? Using the **discharging method** !
(this tool is often used in planar graph theory)

Start : we suppose that one have a planar graph G, avoiding every configuration in C

1. We put weight on the vertices (and faces), in such a way that the sum is negative. For this, we use Euler's formula
2. We "move" weights according local rules.
   The overall sum does not change
3. We show that after all the moves, every vertex in the graph has non-negative weight

Discharging : proving that a set is unavoidable

Now, how to prove that a set of configuration is unavoidable on a planar graph ? Using the **discharging method** !
(this tool is often used in planar graph theory)

Start : we suppose that one have a planar graph G, avoiding every configuration in C

1. We put weight on the vertices (and faces), in such a way that the sum is negative. For this, we use Euler's formula
2. We "move" weights according local rules.
   The overall sum does not change
3. We show that after all the moves, every vertex in the graph has non-negative weight
4. We get a contradiction !
   Such a graph G cannot exists !

Theorem

A planar graph with minimum degree 5 has either an edge 5-5 or an edge 5-6

Give the weight $d(v) - 6$ to each vertex v and and the weight $2d(f) - 6$ to each face f.

Theorem

A planar graph with minimum degree 5 has either an edge 5-5 or an edge 5-6

Give the weight $d(v) - 6$ to each vertex v and and the weight $2d(f) - 6$ to each face f.

The overall sum is -12.

Now move $1/5$ from every neighboor of a 5 degree vertex v to v.

If we suppose that there is no $5 - 5$ nor $5 - 6$, every weight becomes positive.
Contratiction !

Proving the 4CT

Two "big" computational parts in this approach :

1. Find a set $\mathcal{C}$ of reducible configurations
2. Show that $\mathcal{C}$ is unavoidable in a planar graph

Proving the 4CT

Two "big" computational parts in this approach :

1. Find a set $\mathcal{C}$ of reducible configurations
2. Show that $\mathcal{C}$ is unavoidable in a planar graph

The two parts are slightly different.

Approach to show 4CT :

1. Find a big enough set $\mathcal{C}$, with all "reducibility tools" we have
2. Try to show, using the discharging method, that $\mathcal{C}$ is unavoidable
   if we fail, go back in (1)

Estimation of Heesch in 1955

Heesch do some simulation, and estimate that one should have $|\mathcal{C}| \sim 8900$ to succeed.

Estimation of Heesch in 1955

Heesch do some simulation, and estimate that one should have $|\mathcal{C}| \sim 8900$ to succeed.

$\Rightarrow$ the usage of a computer seems unavoidable

Computers are not powerful enough. A "race" begins...

Appel and Haken announce that they completed the proof in 1977.

First proof in 1977

Appel and Haken announce that they completed the proof in 1977.
There a two papers (one for each part) :

- Appel, Kenneth ; Haken, Wolfgang (1977), "Every Planar Map is Four Colorable. I. Discharging", Illinois Journal of Mathematics, 21 (3) : 429–490
- Appel, Kenneth ; Haken, Wolfgang ; Koch, John (1977), "Every Planar Map is Four Colorable. II. Reducibility", Illinois Journal of Mathematics, 21 (3) : 491–567

First proof in 1977

Appel and Haken announce that they completed the proof in 1977.
There a two papers (one for each part) :

- Appel, Kenneth ; Haken, Wolfgang (1977), "Every Planar Map is Four Colorable. I. Discharging", Illinois Journal of Mathematics, 21 (3) : 429–490
- Appel, Kenneth ; Haken, Wolfgang ; Koch, John (1977), "Every Planar Map is Four Colorable. II. Reducibility", Illinois Journal of Mathematics, 21 (3) : 491–567

But, now well accepted...

Criticism on proof from Appel and Haken

The proof of Appel and Haken has not been accepted by everyone...

Criticism on proof from Appel and Haken

The proof of Appel and Haken has not been accepted by everyone...

There is a scepticism of long proofs :

- The first part is done "by hand", and is long
- Some (small) errors was found

Criticism on proof from Appel and Haken

The proof of Appel and Haken has not been accepted by everyone...

There is a scepticism of long proofs :

- The first part is done "by hand", and is long
- Some (small) errors was found

But some new problems...

Criticism on proof from Appel and Haken

The proof of Appel and Haken has not been accepted by everyone...

There is a scepticism of long proofs :

- The first part is done "by hand", and is long
- Some (small) errors was found

But some new problems...

- The second part is done by a computer
- The program is programmed in assembly, on a IBM 370/168
- How can be sure that there is no bug in the computer program ?
- Few people know really what is a computer, how it works, have access to it...
- Almost nobody known how to program a computer
- How can be sure that the computer has no hardware bug ? or no "computation" error ? (computers are not very reliable at this time)

```
HELLO    CSECT               The name of this program is 'HELLO'
*                            Register 15 points here on entry from OPSYS or caller.
         STM   14,12,12(13)  Save registers 14,15, and 0 thru 12 in caller's Save area
         LR    12,15         Set up base register with program's entry point address
         USING HELLO,12      Tell assembler which register we are using for pgm. base
         LA    15,SAVE       Now Point at our own save area
         ST    15,8(13)      Set forward chain
         ST    13,4(15)      Set back chain
         LR    13,15         Set R13 to address of new save area
*                            -end of housekeeping (similar for most programs) -
         WTO   'Hello World' Write To Operator  (Operating System macro)
*
         L     13,4(13)      restore address to caller-provided save area
         XC    8(4,13),8(13) Clear forward chain
         LM    14,12,12(13)  Restore registers as on entry
         DROP  12            The opposite of 'USING'
         SR    15,15         Set register 15 to 0 so that the return code (R15) is Zero
         BR    14            Return to caller
*
SAVE     DS    18F           Define 18 fullwords to save calling program registers
         END   HELLO         This is the end of the program
```

New proof of Robertson et al in 1997

New proof of Robertson et al in 1997

The approach is still the same : reductions and discharging.

New proof of Robertson et al in 1997

The approach is still the same : reductions and discharging.

What change :

- they restarted everything from scratch

New proof of Robertson et al in 1997

The approach is still the same : reductions and discharging.

What change :

- they restarted everything from scratch
- everything in only one paper

New proof of Robertson et al in 1997

The approach is still the same : reductions and discharging.
What change :

- they restarted everything from scratch
- everything in only one paper
- all checks are done by computer (not only the reduction part)
- nevertheless, the proof not fully "automatic", since the discharging rules was found "by hand"

New proof of Robertson et al in 1997

The approach is still the same : reductions and discharging.
What change :

- they restarted everything from scratch
- everything in only one paper
- all checks are done by computer (not only the reduction part)
- nevertheless, the proof not fully "automatic", since the discharging rules was found "by hand"
- the computer programs are in C (not perfect, but way better than "assembly code")
- programs are available for everyone, and can be checked/launched by everyone. Also, in 1997, everyone has access to a computer
- there is also a companion technical paper to explain the computer program.

New proof of Robertson et al in 1997

The approach is still the same : reductions and discharging.
What change :

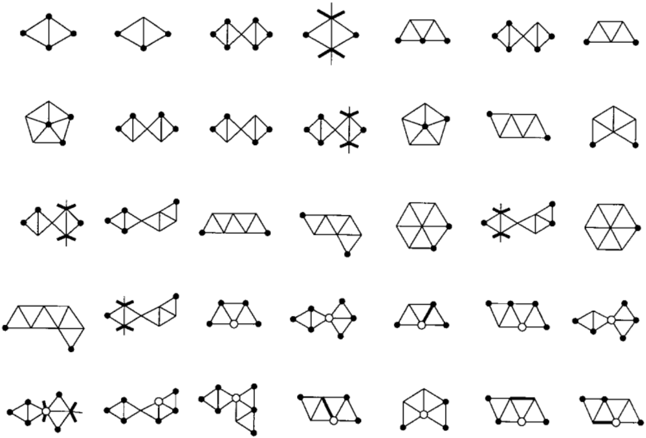
- they restarted everything from scratch
- everything in only one paper
- all checks are done by computer (not only the reduction part)
- nevertheless, the proof not fully "automatic", since the discharging rules was found "by hand"
- the computer programs are in C (not perfect, but way better than "assembly code")
- programs are available for everyone, and can be checked/launched by everyone. Also, in 1997, everyone has access to a computer
- there is also a companion technical paper to explain the computer program.
- water has flowed under the bridge....

Everything is done to avoid doubts, and indeed, the proof is better accepted

In the proof of Robertson et al. 1997

- 633 reducibles configurations (each one checked by computer)
- 32 discharging rules (found by hand)
- The discharging "check" is done by computer

APPENDIX: THE UNAVOIDABLE SET OF REDUCIBLE CONFIGURATIONS



(and so on...)

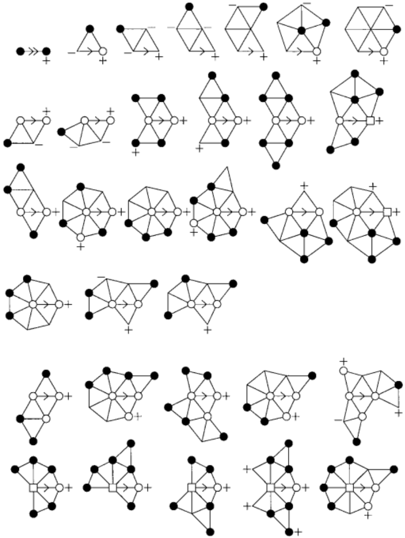


Fig. 4. The rules.

last step : Verification by Gonthier in 2005

Using a "Formal proof assistant" (Coq) Georges Gonthier finished the computer verification of the proof in 2005

last step : Verification by Gonthier in 2005

Using a “Formal proof assistant” (Coq) Georges Gonthier finished the computer verification of the proof in 2005

The verification is a translation, in Coq, of the proof of RSST’97.

Thus, the “proof scheme” is not a new one.

last step : Verification by Gonthier in 2005

Using a "Formal proof assistant" (Coq) Georges Gonthier finished the computer verification of the proof in 2005
The verification is a translation, in Coq, of the proof of RSST'97.
Thus, the "proof scheme" is not a new one.

Hard work... : one have to formalize all the aspects into the theory used by the proof assistant

last step : Verification by Gonthier in 2005

Using a “Formal proof assistant” (Coq) Georges Gonthier finished the computer verification of the proof in 2005
The verification is a translation, in Coq, of the proof of RSST’97.
Thus, the “proof scheme” is not a new one.

Hard work... : one have to formalize all the aspects into the theory used by the proof assistant

Some others computer aided proofs have been also formally showed.
For example, the proof of Kepler conjecture (Hales & al., 2014)

Using a “Formal proof assistant” (Coq) Georges Gonthier finished the computer verification of the proof in 2005
The verification is a translation, in Coq, of the proof of RSST’97.
Thus, the “proof scheme” is not a new one.

Hard work... : one have to formalize all the aspects into the theory used by the proof assistant

Some others computer aided proofs have been also formally showed.
For example, the proof of Kepler conjecture (Hales & al., 2014)

It’s not only for “computer” aided proofs :
For example, the Feit-Thompson theorem (odd order theorem) has been formally verified by Gonthier et al. (2012)

last step : Verification by Gonthier in 2005

Using a "Formal proof assistant" (Coq) Georges Gonthier finished the computer verification of the proof in 2005
The verification is a translation, in Coq, of the proof of RSST'97.
Thus, the "proof scheme" is not a new one.

Hard work... : one have to formalize all the aspects into the theory used by the proof assistant

Some others computer aided proofs have been also formally showed.
For example, the proof of Kepler conjecture (Hales & al., 2014)

It's not only for "computer" aided proofs :
For example, the Feit-Thompson theorem (odd order theorem) has been formally verified by Gonthier et al. (2012)

(But, it's not the subject of this course...)