Zhao Zhang <ZZHANG19@ubishops.ca>,
Shiji Jiang <SJIANG18@ubishops.ca>,
Qichuan Shi <QSHI192@ubishops.ca>

Graph 3-colouring: Given a graph G = (V, E), is there a function f : V → {red, green, blue}, such that f(x) ≠ f(y) whenever (x, y) ∈ E.
Suggested reduction From 3-SAT

Graph 3-colouring is NP-Complete

3-Coloring is NP-Complete if it is NP and NP-hard, here are the proofs:

Prove it's a NP problem:

It could be verified in polynomial time.
Define verifier *VF* for 3-color problem:
VF : On input <*G*, *c*>: (G is the graph, c is the list of colors, in the same order with vertices)

1. Check *c* has only 3 colors. [O(V)]
2. Color each vertices with c. [O(V)]
3. For each vertex, check it's color is different than neighbours.
    a. Using adjacency list O(V+E)
    b. Using adjacency matrix/edge list O(V^2)
4. Accept if (3) is valid for all vertices, otherwise reject.

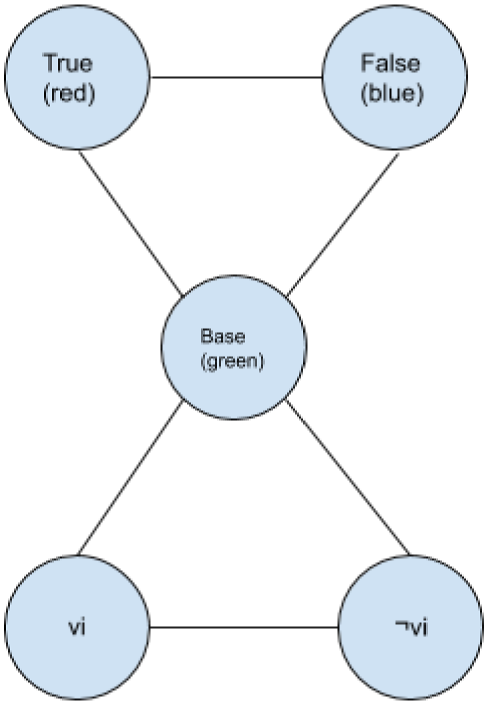
The overall complexity is polynomial time, so 3-coloring is a NP problem.

Prove it's NP hard:

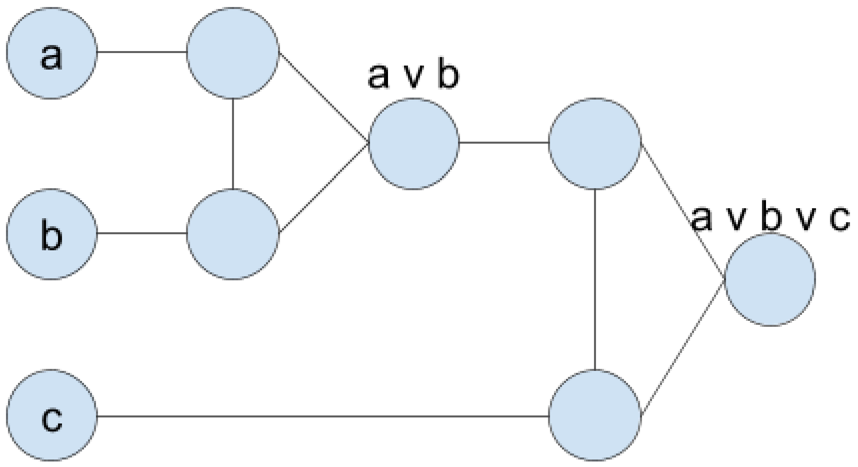
By giving a polynomial time reduction from 3SAT to 3-Coloring:

In order to do a reduction from 3-SAT to 3-Colouring. Given φ is an instance of 3-SAT, which has n variables x_1, x_2, ... x_n and m clauses C_1, C_2 … C_m. Graph G is 3-colorable, iff φ is satisfiable. First, we will build the truth assignment for variables x_1, x_2, ... x_n via the colour of vertexes. Then, we need to find some to capture the satisfiability of clause C_m in φ.
To achieve these two missions, we build a triangle in G with three vertexes {True, False, Base}, which match the three colours {red, green, blue}. For every variable x_i, we add two nodes v_i or

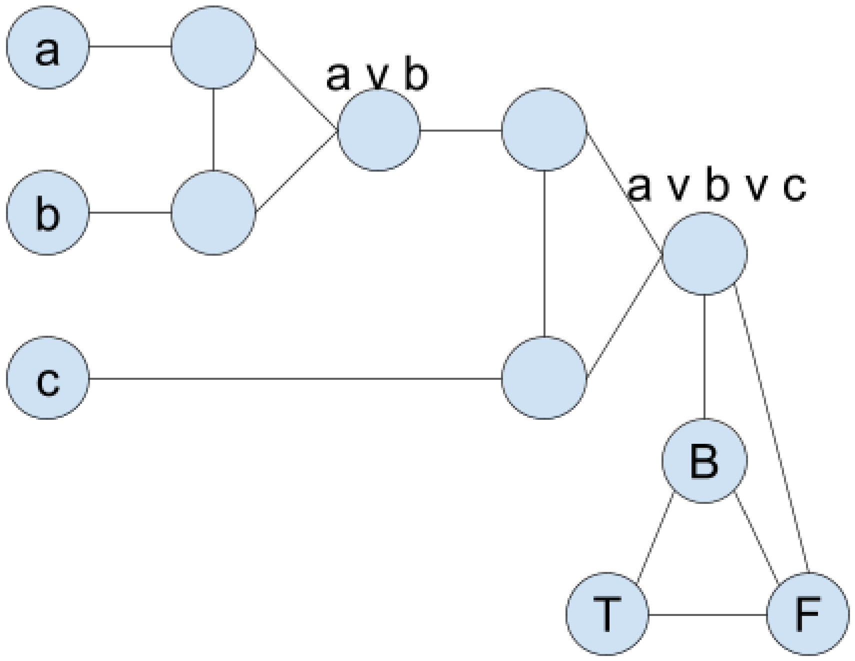
$\overline{v_i}$ and connect to the vertex coloured Base, which shows in the figure following. In this case, if G is 3-colourable, we can get the truth assignment to v_i no matter v_i or $\overline{v_i}$ is coloured True.



The next step is to get the satisfiability of φ. Take any clause C_i = (a ⋁ b ⋁ c) in G, to express the function OR, we will use a graph to build a gadget whose function likes an OR gate, which is shown in the following figure.¬



In this gadget, we want the output node is coloured red, which means that output is True if C_i is satisfied, or the output node will be coloured green which represents False. In this graph, we can simply verify that if variables a, b, c are coloured False, then the output node (a ⋁ b ⋁ c) will be coloured False. When no matter which of the input node is coloured True, the output will be True. The final step is to connect the output to the Base node and False node, shown in the following figure.



Optimization problems:

For the 3 coloring problem, the optimization problem would be:

Given a graph G, Find the smallest *k* such that G is *k*-colorable.

One of the way to solve that would be performing a search to find the optimal k.
Starting from k = 0, 1, 2, 3, ……. n.
We are making at most n calls. When an upper bound k is given, a binary search could be used as well, making log(n) calls to the decision problem.

Back to the decision problem itself, 1-Color and 2 Color are in P, because the neighbors' color is deterministic. NP-complete starts from k >= 3, because of the non deterministic color of neighbours.

References

- http://cs.bme.hu/thalg/3sat-to-3col.pdf
- https://cgi.csc.liv.ac.uk/~igor/COMP309/3CP.pdf
- https://en.wikipedia.org/wiki/Gadget_(computer_science)
- https://www.youtube.com/watch?v=B5_m8lKULlI&t=315s
- https://courses.csail.mit.edu/6.006/fall10/lectures/lecture24.pdf
- https://www.cs.cmu.edu/~anupamg/adv-approx/lecture15.pdf