CSC 373 - Algorithm Design, Analysis, and Complexity **Summer 2014**

Lalla Mouatadid

Introduction to Complexity Theory: 3-Colouring is NP-complete

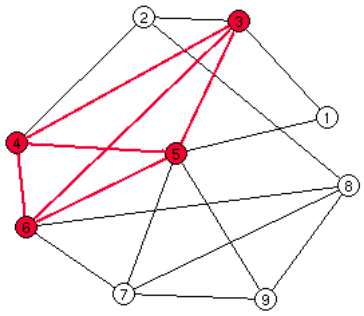
We next show that 3-colouring is NP-complete. What's the **colouring problem** on graphs?

Given a graph $G(V, E)$, the colouring problem asks for an assignment of k colours to the vertices $c : V \to \{1, 2, ..., k\}$. We say that a colouring is **proper** if adjacent vertices receives different colours: $\forall (u, v) \in E : c(u) \neq c(v)$.

The **minimum colouring problem** asks for the smallest k to properly colour G. The **k-colouring problem** asks whether G can be properly coloured using at most k colours.

We call the subset of vertices that receive the same colour a **colour class**. It is easy to see that every colour class is an independent set. The **chromatic number** of a graph is the smallest k, such that G admits a k-proper colouring.

One way to deal with NP-completeness is to restrict the problem to subsets of the input (in this assignment, we restricted "arbitrary" graphs to interval graphs). In this lecture we will show that the colouring problem on arbitrary graphs becomes NP-complete even for $k = 3$! Crazy, No? Think about it: One easy to rule out a 3-colouring is to check if G has a clique of size 4, like in the example below[1]:



How hard is it to check for a clique of size at least $k = 4$? Not hard actually, it takes polynomial time, but as we saw in class, there are graphs with arbitrary large chromatic number, and yet the graph doesn't even have a triangle as a sub-clique. Without further ado, let's prove our theorem:

Theorem 1. *3-COLOURING is NP-complete.*

Where:

3-COLOURING: Given a graph $G(V, E)$, return 1 if and only if there is a proper colouring of G using at most 3 colours.

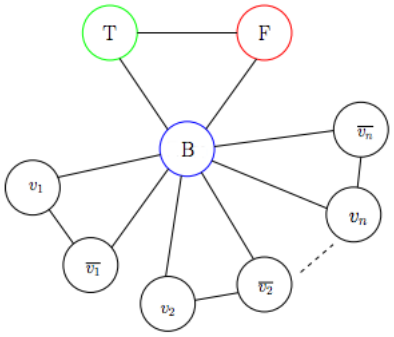
Proof. To show the problem is in NP, our verifier takes a graph $G(V, E)$ and a colouring c, and checks in $\mathcal{O}(n^2)$ time whether c is a proper colou ring by checking if the end points of every edge $e \in E$ have different colours.

[1] All the pictures are stolen from Google Images and UIUC's algo course.

To show that 3-COLOURING is NP-hard, we give a polytime reduction from 3-SAT to 3-COLOURING. That is, given an instance ϕ of 3-SAT, we will construct an instance of 3-COLOURING (i.e. a graph $G(V, E)$) where G is 3-colourable iff ϕ is satisfiable.

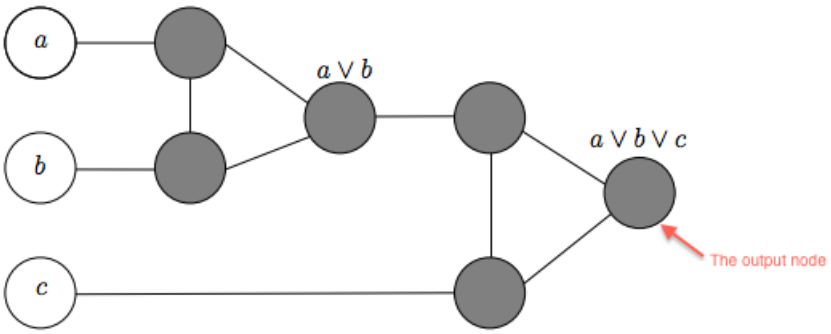
Let ϕ be a 3-SAT instance and $C_1, C_2, ..., C_m$ be the clauses of ϕ defined over the variables $\{x_1, x_2, ..., x_n\}$. The graph $G(V, E)$ that we will construct needs to capture two things: 1. somehow establish the truth assignment for $x_1, x_2, ..., x_n$ via the colors of the vertices of G; and 2. somehow capture the satisfiability of every clause C_i in ϕ.

To achieve these two goals, we will first create a triangle in G with three vertices $\{T, F, B\}$ where T stands for True, F for False and B for Base. Think of $\{T, F, B\}$ as the set of colours we will use to colour (label) the vertices of G. Since this triangle is part of G, we already need 3 colours to colour G. We next add two vertices $v_i, \overline{v}_i$ for every literal x_i and create a triangle $B, v_i, \overline{v}_i$ for every $(v_i, \overline{v}_i)$ pair, as shown below:



Notice that so far, this construction captures the truth assignment of the literals. Since if G is 3-colourable, then either v_i or $\overline{v}_i$ gets the colour T, and we just interpret this as the truth assignment to v_i. Now we just need to add constraints (edges? extra vertices?) to G to capture the satisfiability of the clauses of ϕ. To do so, we introduce the **Clause Satisfiability Gadget**, a.k.a the OR-gadget.

For a clause $C_i = (a \vee b \vee c)$, we need to express the OR of its literals using our colours $\{T, F, B\}$. We achieve this by creating a small *gadget* graph that we connect to the literals of the clause. The OR-gadget is constructed as follows:



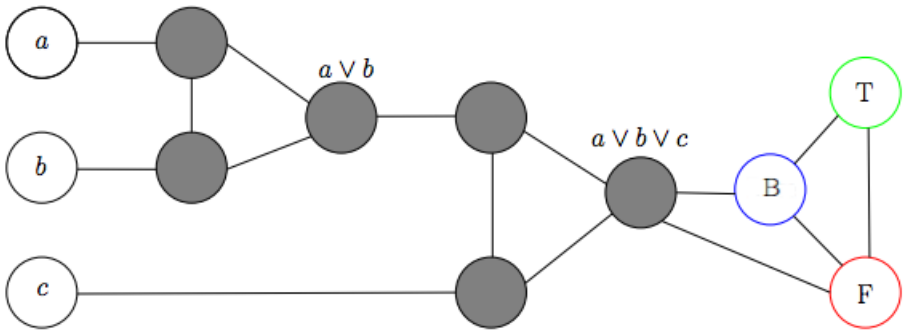
But what is it actually doing? You can think of this gadget graph as a circuit whose output is the node labeled $a \vee b \vee c$. We basically want this node to be coloured T if C_i is satisfied and F otherwise. Notice

that this is a two step construction: The node labelled $a \vee b$ captures the output of $(a \vee b)$ and we repeat the same operation for $((a \vee b) \vee c)$.

If you play around with some assignments to a, b, c, you will notice that the gadget satisfies the following properties:

1. If a, b, c are all coloured F in a 3-colouring, then the output node of the OR-gadget **has** to be coloured F. Thus capturing the unsatisfiability of the clause $C_i = (a \vee b \vee c)$.
2. If one of a, b, c is coloured T, then **there exists a** valid 3-colouring of the OR-gadget where the output node is coloured T. Thus again capturing the satisfiability of the clause.

We're almost done our construction. Once we add the OR-gadget of every C_i in ϕ, we connect the output node of every gadget to the Base vertex **and** to the False vertex of the initial triangle, as follows:



Done :) Now we prove that our initial 3-SAT instance ϕ is satisfiable if and only the graph G as constructed above is 3-colourable.

Suppose ϕ is satisfiable and let $(x_1^*, x_2^*, ..., x_n^*)$ be the satisfying assignment. If x_i^* is assigned True, we colour v_i with T and $\overline{v}_i$ with F (recall they're connected to the Base vertex, coloured B, so this is a valid colouring). Since ϕ is satisfiable, every clause $C_i = (a \vee b \vee c)$ must be satisfiable, i.e. at least of a, b, c is set to True. By the second property of the OR-gadget, we know that the gadget corresponding to C_i can be 3-coloured so that the output node is coloured T. And because the output node is adjacent to the False and Base vertices of the initial triangle only, this is a proper 3-colouring.

Conversely, suppose G is 3-colourable. We construct an assignment of the literals of ϕ by setting x_i to True if v_i is coloured T and vice versa. Now suppose this assignment is not a satisfying assignment to ϕ, then this means there exists at least one clause $C_i = (a \vee b \vee c)$ that was not satisfiable. That is, all of a, b, c were set to False. But if this is the case, then the output node of corresponding OR-gadget of C_i must be coloured F (by property (1)). But this output node is adjacent to the False vertex coloured F; thus contradicting the 3-colourability of G!

To conclude, weve shown that 3-COLOURING is in NP and that it is NP-hard by giving a reduction from 3-SAT. Therefore 3-COLOURING is NP-complete. □