3-Coloring Is NP-Complete

In this lecture, we will explain NP-completeness of yet another problem: 3-coloring.

1 What Is 3-Coloring

Where coloring problems come from. Let us first explain where this problem came from. This historical explanation will not be on the test, but it helps understand the problem.

So here is where this problem came from. Coloring problems come from cartography – the art of making maps. In the political map of the world, different countries are colored. To make it clearer where the borders are, neighboring countries are colored in different colors.

Before computer-based printers, for each color, we needed a different printing step. Thus, the more colors the map uses, the more expensive the map. So, cartographers tried to minimize the number of colors. How can we do it? It is important to make sure that neighboring countries are colored in different colors, but there is no confusion if countries with no common orders are colored in the same color. For example, US and Mexico should be colored in different colors, but it is OK if Mexico and Canada get the same color on the map.

It is known that every map can be colored in 4 colors. Some – but not all – maps can be colored in 3 colors, some even in two colors. It would be nice to know, given a map, whether this map can be colored in 3 colors, and if, yes, how exactly. This is exactly the 3-coloring problem.

How to describe this problem in precise terms. Usually, we make a graph, in which:

- vertices are countries, and
- two vertices are connected by an edge if and only if the corresponding countries have a common border.

For example, for North America, we will have three vertices corresponding to Mexico (M), US (U), and Canada (C). Here:

- M is connected with U,
- U is connected with C, but

- M is not connected with C.

Formulation of the 3-coloring problem in precise terms.

- *given:* a graph,
- *find:* a 3-coloring of this graph, i.e., a way to assign, to each vertex, one of the three colors so that any two directly connected vertices will get different colors (or a message that for this graph, 3-coloring is not possible).

2 Let Us Prove That 3-Coloring Is NP-Complete

How we will prove it. We already know that 3-SAT is NP-complete. So, similar to what we argued when we proved that fact, it is sufficient to prove that 3-SAT can be reduced to 3-coloring.

This reduction will be described in this lecture.

Reducing 3-SAT to 3-coloring. Let us show how, based on a 3-CNF formula, we can build a graph which is 3-colorable if and only if the original formula is satisfiable. This reduction will consist of several steps.

In this reduction, the three colors will be called "true" (T), "false" (F), and "unknown" (U). These weird names for colors will be explained later – they indeed will correspond to truth values.

First step: building a palette. First, we build what the book called a *palette*: three vertices connected to each other (as a triangle). Since each two of these three vertices are connected to each other by an edge, if this triangle is colored, all three edges must be of different colors: one of them gets color T, another gets color F, and the third one get color U.

In view of this, let us denote one of these vertices by T, another by F, and the third one by U.

Second step: adding vertices corresponding to literals. Then, for each Boolean variable v:

- we add two vertices – we will denote then by v and $\neg v$, and
- we connect them to each other and to the vertex U.

Since both vertices v and $\neg v$ are connected to U – the vertex which is colored in U color – the vertices v and $\neg v$ cannot be colored in the U color. So, each of these vertices must be colored either by the color T or by the color F.

Since the vertices v and $\neg v$ are connected to each other, they cannot be of the same color. So, one of them must be of color T, another of color F.

This is how we will read the truth values of the corresponding variable from the 3-coloring of the graph (this is the algorithm U_2):

- if the vertex v is colored in T color, we take $v =$"true",
- if the vertex v is colored in F color, we take $v =$"false".

Third step: dealing with clauses. So far, we did not take the formula – and its clauses – into account at all. Let us show how to do it.

Since we have a 3-CNF formula, each clause has either 2 or 3 literals. Let us first show how to deal with a clause consisting of 2 literals, i.e., with a clause of the type $a \vee b$.

Clause with 2 literals. To deal with each clause C_j of the type $a \vee b$, we use a construction that the book calls an *or-gadget*: namely, for each such clause, we:

- add two auxiliary vertices a_j and b_j;
- connect both auxiliary vertices a_j and b_j with the T vertex and with each other;
- connect a_j with a; and
- connect b_j with b.

Let us show that this configuration can be colored in three colors if and only if $a \vee b$ is true. Indeed, if $a \vee b$ is of true, then we have three possible cases for a and b: true-true, true-false, and false-true. Let us show that in all three cases, the or-gadget can be colored in 3 colors:

- if both a and b are true, then we can color the vertex a_j in color F and b_j in color U – and, as we can easily see (draw it, you will see it) every two connected vertices indeed have different colors;
- if a is true and b is false, then we can color a_j in color F and b_j in color U;
- if a is false and b is true, then we can color a_j in color U and b_j in color F.

On the other hand, if the statement $a \vee b$ is false, i.e., if both a and b are false, then:

- a_j cannot be of color T – since a_j is connected to the vertex T which is of color T, and
- a_j cannot be of color F – since a_j is connected with the vertex a which is false (i.e., of color F).

Thus, the vertex a_j has to be of the only remaining color U. But now, the vertex b_j is connected:

- to the vertex T of color T,
- to the vertex b of color F, and

- to the vertex a_j of color U,

so the vertex b_j cannot be of any of these three colors. This shows that when both a and b are false, the or-gadget cannot be colored in 3 colors.

Clause with 3 literals. We have shown how to describe a clause with two literals. Suppose now that we have a clause C_j with 3 literals, i.e., of the type $a \vee b \vee c$. If we need to compute the value of this expression for given a, b, and c, how do we do it?

- First, we compute the expression $a \vee b$;
- Then, we compute the value $(a \vee b) \vee c$.

To describe these two or's, we use a combination of two or-gadgets. Namely:

- we add a new vertex $a \vee b$ and connect it to U (to make sure that its value is either T or F);
- then, we build an or-gadget corresponding to $(a \vee b) \vee c$, i.e., the or-gadget corresponding to the new "literal" $a \vee b$ and to the old literal c;
- finally, we build a new or-gadget-like construction: we add two new vertices a_j and b_j, connect both to the vertex $a \vee b$ and to each other, and connect a_j to a and b_j to b.

Let us show that this construction is 3-colorable if and only if the clause $a \vee b \vee c$ is true. Indeed, as we have learned from the description of the or-gadget, if we have a 3-coloring, then either $a \vee b$ is true or c is true.

- if c is true, then clearly $a \vee b \vee c$ is true;
- if $a \vee b$ is true, then the or-gadget-like construction becomes a real or-gadget, so $a \vee b$ is true, which means that $a \vee b \vee c$ is true.

Let us now repeat the same construction as an algorithm, skipping motivations. As usual, *the algorithm is, of course, the most important part of this lecture.*

3 Reducing 3-CNF to 3-Coloring: Algorithm

Algorithm U_1: designing a graph based on the 3-CNF formula. First, we build a palette: three vertices T, and F, and U all connected to each other.

Then, for each Boolean variable v, we:

- add two vertices v and $\neg v$, and
- connect both vertices v and $\neg v$ to U and to each other.

For each 2-literal clause C_j of the type $a \vee b$, we add an or-gadget: namely,

- we add two new vertices a_j and b_j,

- we connect both vertices a_j and b_j to T and to each other, and
- we connect a to a_j and we connect b to b_j.

For each 3-literal clause C_j of the type $a \vee b \vee c$, we:

- add a new vertex $a \vee b$,
- we connect this vertex to U,
- we add an or-gadget for $(a \vee b) \vee c$, i.e.:
  - we add new vertices $(a \vee b)_j$ and c_j,
  - we connect both vertices $(a \vee b)_j$ and c_j to T and to each other, and
  - we connect the vertex $(a \vee b)_j$ to the vertex $a \vee b$, and we connect the vertex c_j to the vertex c;
- we add vertices a_j and b_j and connect them to the vertex $a \vee b$ and to each other, and
- we connect a_j to a and we connect b_j to b.

What is U_2 here. How do you transform a 3-coloring y' of the graph $x' = U_1(x)$ into a satisfying vector $y = U_2(y')$ of the original formula? This is straightforward: in the 3-coloring, each original Boolean variable is either colored T, or colored F (since the vertex v is connected to U). Now:

- if v is colored T, we take $v =$"true";
- if v is colored F, we take $v =$"false".

What is U_3 here. How can we build a 3-coloring based on the satisfying vector?

- For elements of the palette, colors are pre-determined.
- For vertices corresponding to Boolean variables, we take the color of the corresponding truth value.
- For auxiliary vertices corresponding to or-gadgets, we described the truth assignment in the previous section.

4 Example

Algorithm U_1. Let us apply the algorithm U_1 to the formula

$$(\neg p \vee q)\,\&\,(p \vee q \vee \neg r).$$

This formula consists of two clauses: $C_1 = \neg p \vee q$ and $C_2 = p \vee q \vee \neg r$.

- First, we build a palette: we add three vertices T, F, and U, and connect them to each other.
- Then, we add vertices p and $\neg p$ and connect them to each other and to the vertex U.
- Then, we add vertices q and $\neg q$ and connect them to each other and to the vertex U.
- Then, we add vertices r and $\neg r$ and connect them to each other and to the vertex U.
- After that, we build an or-gadget corresponding to the clause $C_1 = \neg p \vee q$: we add vertices $\neg p_1$ and q_1, we connect them to T and to each other, and we connect $\neg p_1$ to $\neg p$ and q_1 to q.
- Finally, we build a complex or-gadget corresponding to the clause

$$C_2 = p \vee q \vee \neg r :$$

  - first, we add a new vertex $p \vee q$ and connect it to U;
  - then, we build an or-gadget for the formula $(p \vee q) \vee \neg r$: we add two vertices $(p \vee q)_2$ and $\neg r_2$, we connect them to each other and to T, and we connect $(p \vee q)_2$ to $p \vee q$ and $\neg r_2$ to $\neg r$;
  - then, we add an or-gadget-like construction corresponding to $p \vee q$: we add two vertices p_2 and q_2, we connect them to $p \vee q$ and to each other, and we connect p_2 to p and q_2 to q.

Algorithm U_3. One of the satisfying vectors for the original formula is $p =$"false", $q =$"true", and $r =$"true". According to the algorithm, we color the above-described graph $x' = U_1(x)$ as follows:

- we color the vertex T in color T, the vertex F in color F, and the vertex U in color U;
- then, we color p in color F, $\neg p$ in color T, q in color T, $\neg q$ in color F, r in color T, and $\neg r$ in color F;
- for the auxiliary vertices from the or-gadget corresponding to the clause C_1, since both $\neg p$ and q are true, we color $\neg p_1$ in color F and q_1 in color U;
- for the or-gadget corresponding to $(p \vee q) \vee \neg r$, since $p \vee q$ is true and $\neg r$ is false, we color $(p \vee q)_2$ in color F and $\neg r_2$ in color U;
- finally, for the or-gadget-like construction corresponding to the true statement $p \vee q$, since p is false and q is true, we color p_2 in color U and q_2 in color F;

One can check that no two neighboring vertices have the same color (check!).

Algorithm U_2. If we have the above coloring:

- we take p to be false, since it was colored in color F;
- we take q to be true, since it was colored in color T; and
- we take r to be true, since it was colored in color T.

Try it yourself. Try this reduction on the example of the following 3-CNF formula: $(p \vee \neg q) \,\&\, (\neg p \vee \neg q \vee \neg r)$.

Homework. Now, you are hopefully ready to do the corresponding homework.