Exercises

1.8 **Satisfactory map colorings.**[1] [2] (Computer science, Computation, Mathematics) ③

Many problems in computer science involve finding a good answer among a large number of possibilities. One example is *3-colorability* (Fig. 1.8). Can the N nodes of a graph be colored in three colors (say red, green, and blue) so that no two nodes joined by an edge have the same color?[3] For an N-node graph one can of course explore the entire ensemble of 3^N colorings, but that takes a time exponential in N. Sadly, there are no known shortcuts that fundamentally change this; there is no known algorithm for determining whether a given N-node graph is three-colorable that guarantees an answer in a time that grows only as a power of N.[4]

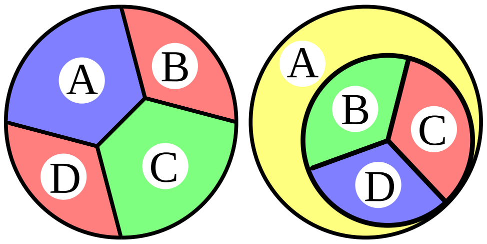

Fig. 1.8 Graph coloring. Two simple examples of graphs with $N = 4$ nodes that can and cannot be colored with three colors.

Another good example is *logical satisfiability* (**SAT**). Suppose one has a long logical expression involving N boolean variables. The logical expression can use the operations NOT ($\neg$), AND ($\wedge$), and OR ($\vee$). It is *satisfiable* if there is some assignment of *True* and *False* to its variables that makes the expression *True*. Can we solve a general satisfiability problem with N variables in a worst-case time that grows less quickly than exponentially in N? In this exercise, you will show that logical satisfiability is in a sense computationally at least as hard as 3-colorability. That is, you will show that a 3-colorability problem with N nodes can be mapped onto a logical satisfiability problem with $3N$ variables, so a polynomial-time (non-exponential) algorithm for the **SAT** would imply a (hitherto unknown) polynomial-time solution algorithm for 3-colorability.

If we use the notation A_R to denote a variable which is true when node A is colored red, then $\neg(A_R \wedge A_G)$ is the statement that node A is not colored both red and green, while $A_R \vee A_G \vee A_B$ is true if node A is colored one of the three colors.[5]

There are three types of expressions needed to write the colorability of a graph as a logical satisfiability problem: A has some color (above), A has only one color, and A and a neighbor B have different colors.

(a) *Write out the logical expression that states that A is colored with only a single color. Write out the logical expression that states that A and B are not colored with the same color.* Hint: Both should be a conjunction (AND, $\wedge$) of three clauses each involving two variables.

[1]From *Statistical Mechanics: Entropy, Order Parameters, and Complexity* by James P. Sethna, copyright Oxford University Press, 2007, page 12. A pdf of the text is available at pages.physics.cornell.edu/sethna/StatMech/ (select the picture of the text). Hyperlinks from this exercise into the text will work if the latter PDF is downloaded into the same directory/folder as this PDF.
[2]This exercise and the associated software were developed in collaboration with Christopher Myers, with help from Bart Selman and Carla Gomes.
[3]The famous four-color theorem, that any map of countries on the world can be colored in four colors, shows that all planar graphs are 4-colorable.
[4]Because 3-colorability is **NP**–complete (see Exercise 8.15), finding such a polynomial-time algorithm would allow one to solve traveling salesman problems and find spin-glass ground states in polynomial time too.
[5]The operations AND ($\wedge$) and NOT $\neg$ correspond to common English usage ($\wedge$ is true only if both are true, $\neg$ is true only if the expression following is false). However, OR ($\vee$) is an *inclusive or*—false only if both clauses are false. In common English usage 'or' is usually *exclusive*, false also if both are true. ('Choose door number one or door number two' normally does not imply that one may select both.)

Any logical expression can be rewritten into a standard format, the *conjunctive normal form.* A *literal* is either one of our boolean variables or its negation; a logical expression is in conjunctive normal form if it is a conjunction of a series of clauses, each of which is a disjunction (OR, $\vee$) of literals.

(b) *Show that, for two boolean variables X and Y, that $\neg(X \wedge Y)$ is equivalent to a disjunction of literals $(\neg X) \vee (\neg Y)$.* (Hint: Test each of the four cases). *Write your answers to part (a) in conjunctive normal form. What is the maximum number of literals in each clause you used? Is it the maximum needed for a general 3-colorability problem?*

In part (b), you showed that any 3-colorability problem can be mapped onto a logical satisfiability problem in conjunctive normal form with at most three literals in each clause, and with three times the number of boolean variables as there were nodes in the original graph. (Consider this a hint for part (b).) Logical satisfiability problems with at most k literals per clause in conjunctive normal form are called **kSAT** problems.

(c) *Argue that the time needed to translate the 3-colorability problem into a* **3SAT** *problem grows at most quadratically in the number of nodes M in the graph (less than αM^2 for some α for large M).* (Hint: the number of edges of a graph is at most M^2.) *Given an algorithm that guarantees a solution to any N-variable* **3SAT** *problem in a time $T(N)$, use it to give a bound on the time needed to solve an M-node 3-colorability problem. If $T(N)$ were a polynomial-time algorithm (running in time less than N^x for some integer x), show that 3-colorability would be solvable in a time bounded by a polynomial in M.*

We will return to logical satisfiability, **kSAT**, and **NP**–completeness in Exercise 8.15. There we will study a statistical ensemble of **kSAT** problems, and explore a phase transition in the fraction of satisfiable clauses, and the divergence of the typical computational difficulty near that transition.