Assignment 43

Alex Clemmer
Student number: u0458675

1

The key to this problem is to think of a path as a series of boolean operations. For example, if we know that $A \rightsquigarrow B \rightsquigarrow C$, we can also say that, in this sequence, A is the first element AND B is the second element AND C is the third element. Note that *any* path we pick will be expressible this way—no matter what sequence of vertices you pick, I can express them as a series of AND statements. If I have more than one option, I can create two complete paths and OR them together (for example: I went from Salt Lake to Austin to Eugene OR Salt Lake to Portland to Eugene).

So the first (and really only) obstacle here is to find a boolean sequence that encodes this directly. Let the notation (U, V) stand for the edge going from vertex U to vertex V. I chose this way because it is relatively easy to combinatorially generate every possible path. You start with some U, and find every vertex you can connect to, and then continue on, building every possible path you can. Note that you might think you'd run into problems if you have cycles, but you're only allowed to visit each node once. If you've tried every possibility, you should end up with some set of paths like these:

(a, t)
$(a, c) \rightsquigarrow (c, b) \rightsquigarrow (b, q)$
$(a, r) \rightsquigarrow (r, s)$

Transforming these paths into an expression is really easy. First, discard all paths that do not visit all vertices. Since we do not visit vertices more than once, we don't have to worry about that case. Second, to each of these paths, OR together all the edges that are not visited by that path and then NOT them—this is because if we want to go from Salt Lake to Austin to New York, we also want to NOT go from Salt Lake to New York OR New York to Austin. Third, OR all the valid paths together, as we did in the beginning paragraph.

So, for example, if we have a set of vertices $V = \{\text{Salt Lake, Austin, Eugene}\}$, our boolean expression might look like follows:

$$((\text{Salt Lake}, \text{Austin}) \wedge (\text{Austin}, \text{Eugene}) \wedge \neg((\text{Eugene}, \text{Salt Lake}))$$

We'd of course have more edges in the NOT block if this were a directed graph. Regardless, this isn't just equivalent to SAT, it's the *definition* of SAT. It must be true that we're going form SLC to Austin AND it must be true that we're going from Austin to Eugene, AND it must be true that we're NOT going from Eugene to Salt Lake. In this way, we can encode and solve any Redrata problem as SAT.