An `NP-COMPLETE` Problem, §8.4

We give a somewhat different formulation than §8.4 for an `NP-COMPLETE` Problem. We'll call our problem class `RECTANGLECOLOR`. The problem class will deal with finding a coloring of the one by one squares of an M by $T+1$ rectangle. The squares will be coordinatized (m,t) where $1 \leq m \leq M$ and $0 \leq t \leq T$. (We think of the first coordinate as the Turing Machine tape and the second as the time.)

INPUT

1. A finite set of colors C. One particular color is called `YES`. A specified subset of the colors are called *nonheader* colors.

2. A specification of *some* of the colors of the squares in the zero-th row.

3. A set of *rules* to generate colors in a row from the colors in the previous row. The rules are of the form:

   If on row $t-1$ positions $m-1, m, m+1$ have colors c_1, c_2, c_3 respectively then on row t position m has color c_4.

   For each triple c_1, c_2, c_3 we have a c_4. (Formally, the set of rules is a function $RULE : C \times C \times C \rightarrow C$.) At the edges (this is not so important but is needed for completeness) we have special rules so that the colors of $(t-1,1)$ and $(t-1,2)$ determine the color of $(t,1)$ and the color of $(t-1,M-1)$ and $(t-1,M)$ determine the color of (t,M).

Because the rules cover every case, a coloring of the zero-th row will determine the color of the first row and thence the second and so on, completely determining the coloring of the rectangle. But, critically, in the input not all the colors in the zero-th row are given.

`RECTANGLECOLOR:` Given the INPUT is there some way to color the remainder of the zero-th row with nonheader colors so that the coloring generated by the rules will have position $(T,1)$ colored `YES`.

We first claim that `RECTANGLECOLOR` is in `NP`. Given the input the certificate will be the coloring of the remainder of the zero-th row. Now in polynomial time we follow the rules and generate the coloring of the entire rectangle and see if position $(T,1)$ is colored `YES`.

The remarkable thing is the `RECTANGLECOLOR` in `NP-COMPLETE`.

`Turing Machine Primer:` A Turing Machine consists of an infinite tape with positions $1, 2, 3, \ldots$, a finite alphabet of symbols, a finite set of states, and a header. There is a special symbol `BLANK`. Each position has a symbol, all but finitely many have `BLANK`. The header points to one of the positions, initially to 1. There are two special symbols `YES` and `NO` and a special state `SLEEP`. (Other formulations have `HALT` which works pretty much the same way.) There are a finite set of rules: When the header is at a position with symbol a and the machine is in state s it will

1. Maybe replace a by some a'
2. Maybe change to state s'
3. Possibly move the header one to the left or right.

Crucially, what it does depends only on a, s. If the state is `SLEEP` then nothing is changed, including the state. When the above changes are made (even if nothing is changed) we think of that as one time unit, time starting at zero. A Turing Machine is a *decision procedure* for a problem class X if

1. If a string $\alpha \in X$ then starting the Turing Machine with the tape as α it will get to `SLEEP` with `YES` in position one.
2. If a string $\alpha \notin X$ then starting the Turing Machine with the tape as α it will get to `SLEEP` with `NO` in position one.

Sounds simple! But Turing Machines encapsulate all possible programming languages!

Formally, a problem class X is in P if there is a Turing Machine which is a decision procedure for X and which runs (you stop counting time when you reach state `SLEEP`) in time $O(n^c)$ where n is the length of the input string α.

For X to be in `NP` we think of the Turing Machine with input string α followed by `BLANK` and then followed by the certificate string β. The Turing Machine with this input runs for polynomial time. If $\alpha \notin X$ there will be no certificate β so that the tape will have the special symbol `YES`. If $\alpha \in X$ there will be a certificate β so that at time T there will be symbol `YES` at the first position of the tape. (Note that in `NP` we need not, and generally do not, have a symbol for `NO`.)

Lets say the Turing Machine alphabet has r letters, counting the special symbols, and s states. We can encode these and the header together by thinking of $r(1+s)$ colors. The color encodes the letter and whether or not the header is at that spot and, if it is, what the state is. (Note that we only encode the state if the header is there.) We have space time coordinates,

(m,t) being position m (on the tape) at time t. So the history of the running of a Turing Machine is a coloration of the rectangle. (The nonheader colors are only a technical artifice, to assure that there is only one header.) The key is that the color at $(m,t+1)$ is completely determined by the color at $(m-1,t),(m,t),(m+1,t)$. Indeed if the header at time t is at none of $m-1,m,m+1$ then the color remains the same. When the header at time t was at $m\pm 1$ it might have moved to m thus changing the color of $(m,t+1)$. And, critically, when the header at time t was at m the new letter at time $t+1$ and position m and the new state depend in a deterministic way on the rules of the Turing Machine. So we can encode all of this into the rules for `RECTANGLECOLOR`. Now, given a problem class X that is in `NP` there is a Turing Machine with the above properties and therefore we would encode an instance of `RECTANGLECOLOR`. The answer to the `RECTANGLECOLOR` problem will be Yes precisely when $\alpha \in X$. This shows that $X \leq_P$ `RECTANGLECOLOR`, as if we had a Black Box for `RECTANGLECOLOR` and X was in `NP` then we could, given an instance α, call the Black Box to decide if $\alpha \in X$.

`Further Reductions:` We can reduce `RECTANGLECOLOR` to `SAT` as follows. Say there are $u = r(1+s)$ colors called $1,2,\ldots,u$. Let color 1 mean that the header is there and the state is `SLEEP`.

For every $1 \leq m \leq M$, $0 \leq t \leq T$ and each $1 \leq c \leq r(1+s)$ we make a Boolean variable $F(m,t,c)$ which has the interpretation that we have color c at spacetime coordinates (m,t). Now we make the following clauses:

1. (Specified Start) For each m for which the initial position on the tape is specified as some specific color c: $F(m,0,c)$

2. (Rest of Start) For each m for which the initial position of the tape is not specified: the disjunction of $F(m,0,c)$ over all nonheader colors c.

3. (Good Ending) At time T the header is at position 1 which has the special symbol `YES`: $F(1,T,1)$

4. (Everything is Colored) At every spacetime coordinate (m,t) there is a symbol: For each (m,t) the disjuntion of $F(m,t,c)$ over all colors c.

5. (Nothing Colored Twice) At every spacetime coordinate (m,t) there are not two symbols: For each (m,t) and every two distinct colors c,c' the clause $\neg F(m,t,c) \vee \neg F(m,t,c')$.

6. (The Big Enchalada) The rules are followed. For each $1 < m < M$ (the edge effects require only an easy modification) and each $0 \leq t < T$

and each c_1, c_2, c_3, c_4 so that $c_4 \neq RULE(c_1, c_2, c_3)$ the disjunction

$$\neg F(m-1, t, c_1) \vee \neg F(m, t, c_2) \vee \neg F(m+1, t, c_3) \vee \neg F(m, t+1, c_4)$$

(This rather roundabout approach is because we want a conjunction of disjunctions and so we say which 4-tuples are not allowed.)

Let C be the conjunction of all of the above clauses. We have arranged things so that the determination of the first row (the Boolean $F(m, 0, c)$) determines all the rest and C being satisfiable is exactly that we can fill out the first row so that we get $F(1, T, 1)$.

We have argued that `RECTANGLECOLOR` in NP-Complete. But this is reducible to `SAT` which is reducible to `3-SAT`. That is `RECTANGLECOLOR`$\leq_P$`3-SAT`. But as `3-SAT` is in NP we have the other direction `3-SAT`$\leq_P$`RECTANGLECOLOR`. As they are each polynomial time reducible to each other we therefore conclude:

3-SAT IS NP-COMPLETE