COMP 314 Exam 2, Spring 2023 Solutions

Qu 1 To show UHC $\leq_p$ 3HC, we need a conversion algorithm C mapping instances of UHC to instance of 3HC such that (1) C runs in polynomial time; (2) C maps positive instances to positive instances; (3) C maps negative instances to negative instances.

Define C as follows. Given input G (an undirected graph), C(G) consists of 3 identical copies of G. Example:

G $\longmapsto$ C(G)

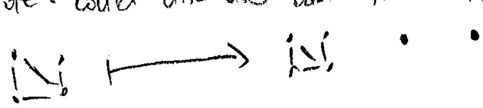
(1) Generating C(G) can be done in linear time, which is certainly polynomial.

(2) If G is positive, it has a Hamilton cycle. We can use the same cycle on each copy of G in C(G) to visit all nodes in 3 disjoint cycles, so C(G) is a positive instance of 3HC.

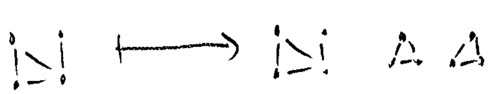
(3) We show the contrapositive: C(G) positive implies G positive. If C(G) is positive, we have 3 disjoint cycles, each of which is a Hamilton cycle on one of the copies of G. Picking any one of these cycles demonstrates that G was a positive instance of UHC.

This completes the proof.

Note: could also use conversions like this



← add 2 disconnected nodes



← add any 2 disconnected cycles.

$\boxed{\text{Qu 2}}$

First we show that 3HC $\in$ NP. This follows because we can check a positive instance of 3HC in polynomial time: given a hint claiming to consist of 3 disjoint cycles that covers an instance G, we can check that these really are cycles that are disjoint, and that all vertices of G are visited — this requires only efficient algorithms (roughly linear cost if we store G's edges and vertices in hash tables, or quadratic if we use naive scanning of lists of the edges and vertices). Hence, 3HC $\in$ NP.

Now recall that <u>any</u> NP problem polyreduces to <u>any</u> NP-complete problem. In particular, 3HC $\leq_P$ C, as desired.

$\boxed{\text{Qu 3}}$

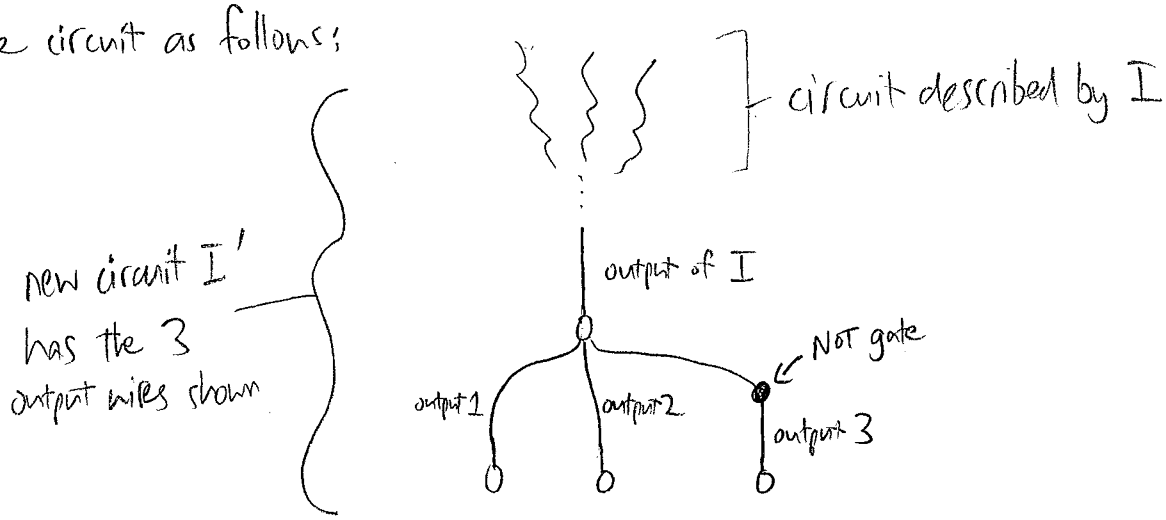
a) Yes, it is a verifier. Given a positive instance I, representing an integer M that is a perfect square, we can take $H = \sqrt{M}$. The variable 'total' will be computed as $\underbrace{H+H+\dots H}_{H \text{ times}} = H^2 = M$. But if I is a negative instance, there is no integer hint H that will return 'correct'.

b) No, it does not run in polynomial time. The loop at line 7 is executed H times, which requires at least $10^{|H|}$ operations, so the running time is exponential (and therefore not polynomial) as a function of the length of the inputs. [Detail: we <u>can</u> have len(H) = len(I), so the running time <u>can</u> require exponential time as a function of $|I|$].

Qn 4

Yes, Circuit6 is NP-complete. To prove this, we exhibit a polyreduction CircuitSat $\leq_p$ Circuit6.

Define a conversion C, mapping instances of CircuitSat to instances of Circuit6, as follows. Given a circuit I that is a valid instance of CircuitSat, extend the circuit as follows:



Note that if the output of I is 1, then the output of I' is 110, representing 6. If the output of I is 0, then the output of I' is 001, representing 1. Therefore, I' is a positive instance of Circuit6 if and only if I is a positive instance of CircuitSat.

Since the transformation C adds only a constant number of gates and wires, it can clearly be accomplished in polynomial time.

This completes the proof that CircuitSAT $\leq_p$ Circuit6.

Finally note Circuit6 $\in$ NP as we can easily check a satisfying assignment.

Since CircuitSAT is known to be NP-complete, we conclude Circuit6 is also NP-complete.

Qn 5

IsIncString belongs to all classes: Const, Lin, LogLin, Quad.

A Python program can solve this problem in constant time by examining at most the first 52 characters of the string. After that, one of the 52 alphabetic characters would be repeated (or another illegal character would be used), violating the requirement of strictly increasing ordering.