Conrad Lewin
CS 325
5/22/2016

1) Both b and d can be inferred from that fact that TSP is NP-complete assuming $P \neq NP$.

b - There does not exist an algorithm that solves an arbitrary instance of the TSP problem in polynomial time.
d - The TSP is not in P.

Explanation:
If a problem is shown to be NP-Complete, then it exists in a set of problems that can be verified in polynomial time, but, unlike those problems in P, that cannot be solved in polynomial time. All problems that are NP-complete are also in the set NP, and P is a subset of NP. Thus, if $P \neq NP$, that is those elements in NP – P cannot collapse the barrier between P and NP to form a single set of problems solvable and verifiable in polynomial time, then those problems in P and those in NP-Complete must be mutually exclusive, insofar as P is subset of NP and NP-Complete is a subset of NP, but P is not a subset of NP-Complete nor is NP-Complete a subset of P. Hence, there can be no efficient algorithm for NP-Complete problems, including TSP which cannot be in P.

2) Considering two decisions problems X and Y, both d and g can be inferred from that fact that X reduces to Y.

d - If X is NP-Complete and Y is in NP then Y is NP-Complete.
g - If Y is in P, then X is in P.

Explanation:
Since X can be reduced to Y, then X is no harder than Y. This means that X can be solved in the worst case as efficiently as Y and may in fact allow for a faster solution. Thus, since X must be less than or equal to Y, if X is NP-Complete then Y must be at least NP-Complete. Likewise, if Y is in P, then X can be no more difficult than P.

3) Prove HAM-PATH is NP-Complete.

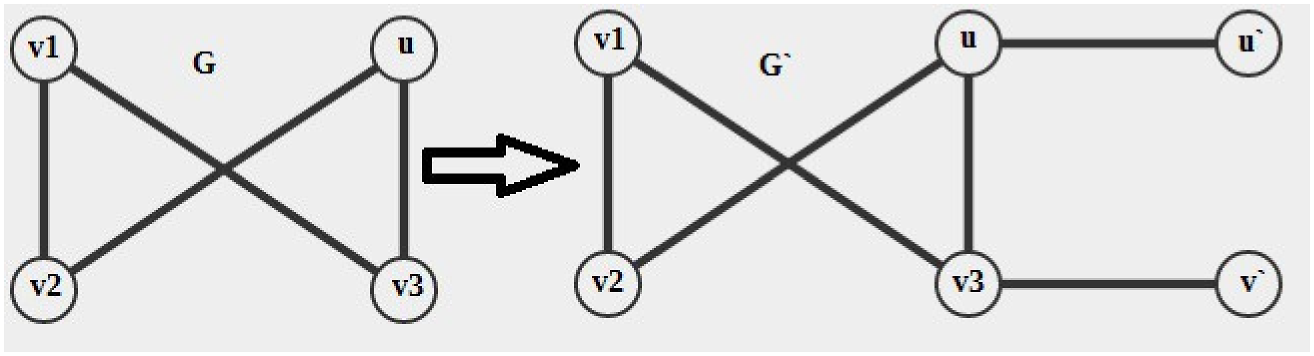
I. HAM-PATH $\in$ NP

Given a graph $G = (V, E)$ and a certificate $C = (v_1, v_2, v_3, \ldots, v_{n-1}, v_n)$ where n is the total number of vertices in V, we can verify that the certificate is in fact a Hamiltonian path in polynomial time. The algorithm employed for this verification might involve iterating through the graph's adjacency list starting with vertex v_1, ensuring that not only is there a vertex $v_1 \in V$, but that there is also an edge between v_1 and v_2, and beyond that between $v_i \in V$ and $v_j \in V$. A check must also be made to determine whether each $v_i \in V$ appears in the path at most a single time. This can be accomplished in $O(E)$ time by iterating through the edge list of each each vertex in the certificate searching for the next vertex in the given path, marking each found vertex as visited and quitting if a found vertex has already been traversed. Thus, HAM-PATH $\in$ NP.

II. HAM-CYCLE $\leq_p$ HAM-PATH

a. HAM-CYLCE reduction in polynomial time

The input for a HAM-CYLCE is defined as follows: {*G*, *u*} where *G* = (*V*, *E*) and *u* is both a source and target vertex such that a simple cycle beginning at *u* and traversing v_i through v_k will return to *u* after having visited all $v_j \in V$ not equal to *u* at most one time. The input for HAM-PATH is similar and is defined as such: {*G*`, *s*, *t*} where *G*` = (*V*`, *E*`) and *s* and *t* represent the source and destination vertices that demarcate a path from *s* to *t* in which all $v_j \in V$` are visited at most one time. We must transform *G* in such as way as to assert that *G* has a Hamiltonian path, if and only if it has a Hamiltonian cycle. This can be done in polynomial time by adding vertices *u*` and *v*` to *G* forming *G*` in which *u*` = *s* and is adjacent only to *u* (*u*`, *u*) and *v*` = *t* and is only adjacent to some vertex v_i adjacent to *u.* Below is an illustration of one particular reduction:



b. HAM-CYCLE solution using HAM-PATH

Since determining whether a graph with no more than 2 vertices has a Hamiltonian path is trivial and can be accomplished in polynomial time, insofar the algorithm need only confirm that the source vertex is connected to the target vertex, this proof will assume that *G* is graph with at least 3 vertices. We must show that if a graph with a Hamiltonian cycle is input into the HAM-PATH function, then there must necessarily be a Hamiltonian path, and furthermore, if a graph lacking a Hamiltonian cycle is input in the HAM-PATH function, then there is necessarily no Hamiltonian path. Considering the fact that *v*` is adjacent to some vertex $v_i$ adjacent to *u*, if a Hamiltonian cycle exists then there will be an edge (*u*, $v_i$) that when removed will break the cycle. Now, if we consider a path from *u*` to *v*` ignoring the edge (*u*, $v_i$) and in its stead traversing the edge (*v*`, v_i), there there will necessarily be a Hamiltonian path from *u* to *v*` since $v_i$ would be the last vertex visited before completing the cycle, meaning all vertices not equal to *v*` have been visited at most once, and since *v*` = *t* it will be the last vertex visited on the path, hence all vertices in *G*` will have been visited a single time. Trivially, since *u*` = *s*, *u*` is adjacent only to *u* and *u* is a vertex in a Hamiltonian path *u*` must also be a vertex on this path.

Furthermore, if *G* does not have a Hamiltonian cycle, then there is no path exists that will visit each vertex in *V* at most once such that the last vertex visited will be adjacent to *u*. In this way, there is no valid (previously untraveled) edge (*u*, v_i) that can be replaced with (*v*`, v_i) at the final moment of the path's generation. ■

We have shown that a solution to the HAM-PATH problem can be verified in polynomial time, that HAM-CYCLE can be reduced to HAM-PATH in polynomial time and that HAM-CYCLE $\leq_p$ HAM-PATH. Therefore, HAM-PATH is NP-Complete.

4) Prove LONG-PATH is NP-Complete

I. LONG-PATH ∈ NP

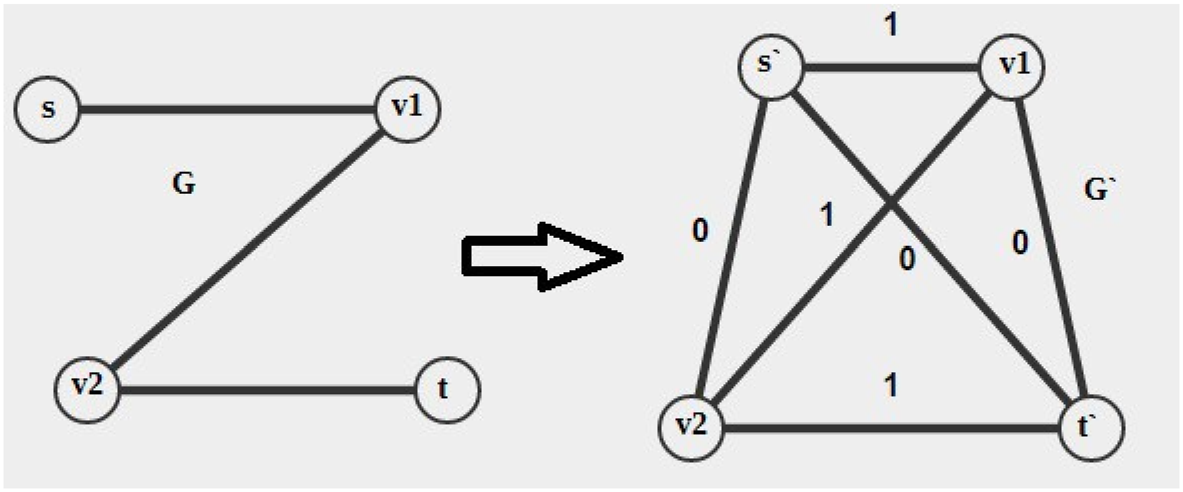
Given a graph $G = (V, E)$ and a certificate $C = (v_1, v_2, v_3, \ldots, v_{n-1}, v_n)$ where n is the total number of vertices in a longest simple path P, we can verify that the certificate is in fact a longest simple path of weight $w_p \geq k$, where k is a given weight value corresponding to the a longest simple path in G, in polynomial time. This can be verified in linear time by iterating through the edge weights defined in C, summing each value and determining whether or not that sum is greater than or equal to k.

II. HAM-PATH $\leq_p$ LONG-PATH

a. HAM-PATH reduction in polynomial time

The input for HAM-PATH is defined as follows: $\{G, s, t\}$ where $G = (V, E)$ and s and t represent the source and destination vertices that demarcate a path from s to t in which all $v_j \in V$ are visited at most one time. Similarly, the input for LONG-PATH is defined as such: $\{G`, s`, t`, k\}$ where $G` = (V`, E`)$, $s`$ and $t`$ represent the source and destination vertices that demarcate a weighted path from $s`$ to $t`$ in which all $v_i \in V`$ are visited at most one time and k is the sum cost of weights in a simple path from s to t where k is greater than or equal to some value q.

We can transform the input for HAM-PATH in polynomial time by making two minor modifications. First, an algorithm can be defined to iterate through the list of edges in E, adding a weight of 1 to each edge. Second, the graph must be made complete by iterating though the list of edges and connecting each vertex v_i to all other vertices in V not equal to v_i and assigning each of these new edges a weight of 0. Below is an illustration of one particular reduction:



b. HAM-PATH solution using LONG-PATH

Note that if G is already complete when passed to the LONG-PATH function, that is, there are no edge weights of 0, then G necessarily has a Hamiltonian cycle and, furthermore, it has a Hamiltonian path, insofar as this path can be formed by removing a single edge incident on s. This simple path will then have a weight of $n - 1$ where n is the total number of vertices in $V`$ which will be the longest possible path in $G`$. If G is not complete when passed to the LONG-PATH function, then after the transformation, $G`$ will be a complete graph with a set of additional edges having weights of 0 and whose original edges now have a weight of 1. In this way, the longest simple path from $s`$ to $t`$ should have a total weight $k \geq n - 1$. This holds true if there is a Hamiltonian path in $G$ since traversing this path will involve visiting all the vertices in $V`$ that are also in V. Since the weights of the edges that are in both E and $E`$ will incur a cost of 1 for each traversal, then the longest simple path in $G`$ will necessarily have been discovered insofar as the maximum number of non-negative weighted edges comprises the path. If, on the other hand, G does not contain a Hamiltonian path, then there will be no path from $s`$ to $t`$ that will include only edges whose weight is 1, and so the weight of this path will necessarily be $w_p < k$. ■

We have shown that a solution to the LONG-PATH problem can be verified in polynomial time, that HAM-PATH can be reduced to LONG-PATH in polynomial time and that HAM-PATH $\leq_p$ LONG-PATH. Therefore, LONG-PATH is NP-Complete.

5a) Given a graph $G = (V, E)$ we can determine whether or not there is a two coloring of G in polynomial time. Let these two colors be red and blue. We begin by starting at some arbitrary vertex $s \in V$ and coloring that vertex red. Next, the algorithm should commence a breadth first search on G starting at s. Let each vertex adjacent to s, and subsequently to v_i, form a set of vertices V_a. Each $v_j \in V_a$ is then examined to determine whether or not some vertex v_j is adjacent to some other vertex $v_k \in V_a$, where $v_j \neq v_k$. In other words, if there exists a complete subgraph involving $v_i \in V$ and both v_j, $v_k \in V_a$, then there is no valid two coloring of the G since adding either a red or blue vertex to this complete subgraph will inevitably mean coloring a vertex the same color as one adjacent to it. If there is no such subgraph, then a valid two coloring of G could be achieved by simply running the breadth first search and coloring each vertex adjacent to v_i that color which does not characterize v_i. Since the breadth first search can be accomplish in $O(V + E)$ and iterating through sections of the graph's adjacency list, albeit multiple times for each v_i, can be accomplished in polynomial time, the entire algorithm can be run in polynomial time.

5b) The graph-coloring problem can be cast as a decision problem k-color in the following manner: Is there a k-coloring of an undirected graph $G = (V, E)$, where k is the number of colors available for use in the coloring of each vertex $v_i \in V$ such that no vertex is adjacent to another vertex of the same color?

If a k-coloring for G exists, and this can be determined in polynomial time, then finding a particular coloring of G can be accomplished by assigning some color c in a set of available colors C to each vertex $v_i \in V$ using some version of the polynomial time algorithm stated in 5a. Thus, the k-color decision problem can be solved in polynomial time by examining the number of colors n in C, insofar as there is a valid k-coloring of G if and only if $n \geq k$.

5c) Given a certificate c demonstrating a particular k-coloring of some graph $G = (V, E)$, we can verify the given solution in polynomial time by iterating through the each vertex $v_i \in V$, ensuring that each vertex has a valid color and that there are no adjacent vertices that share the same color. Thus, K-COLOR $\in$ NP.

5d) Prove 4-COLOR is NP-Complete.

I. 4-COLOR $\in$ NP

Verifying a certificate for a 4-coloring of some arbitrary graph can be accomplish in the same manner as was expressed in the answer to question 5c, in which it was shown that any k-coloring of a given graph be accomplished in polynomial time. Since 4 is a valid value of k, 4-COLOR $\in$ NP.

II. 3-COLOR $\leq_p$ 4-COLOR

Given a 3-color graph $G = (V, E)$, we can transform G in polynomial time by simply adding a single vertex v\` to $G$ and connecting it to each vertex $v_i \in V$. This will create a new graph $G$\` = (V\`, $E$\`) in which the existence of a valid 3-coloring of G implies the existence of a valid 4-coloring of G\`. This conditions holds true because if $G$ has a valid 3-coloring, then the addition of $v$\`, dependent on the availability of some fourth color, will allow for v\` to be colored with this new hue, hence avoiding an outcome in which neighboring vertices share the same color and permitting $G$\` to express a valid 4-coloring. If, on the other hand, G does not have a valid 3-coloring, then the addition of v\` and some fourth color will not be enough to provide a valid 4-coloring for $G$\`, insofar as even if one of those adjacent vertices in V that share a common color is imbued with this fourth color, hence fixing the condition that invalidated the 3-coloring of G, that fact that v\` is connected to all other vertices in $V$ necessarily means that any color chose to mark $v$\` will be match one of the vertices in V, thus invalidating the 4-coloring of G\`. ■

We have shown that a solution to the 4-COLOR problem can be verified in polynomial time, that 3-COLOR can be reduced to LONG-PATH in polynomial time and that 3-COLOR $\leq_p$ 4-COLOR. Therefore, 4-COLOR is NP-Complete.