CS-3250

Section 01

-

Homework #6

Patrick Pei

Professor Jeremy Spinrad

Due 04-10-2017

Exercises

1 Show that 3 of the following 4 problems are in NP.

Three of the following four problems will be shown to be in NP by the defintion of the complexity class NP, where NP is the class of languages that can be verified by a polynomial time algorithm. According to the CLRS textbook, a language L belongs to NP if and only if there exist a two-input poylnomial-time algorithm A and a constant c such that

$$L = \{x \in \{0,1\}^* :$$

There exists a certificate y with $|y| = O(|x|^c)$ such that $A(x, y) = 1$.

1a) Given a graph G and a number k, does G have a spanning tree of cost less than or equal to k? Clearly, given the vertices and edges of a spanning tree of cost less than or equal to k, it can be verified in polynomial time that G does have a spanning tree of cost less than or equal to k by simply adding the cost of all edges in the spanning tree to make sure that this total cost is less than or equal to k, as well as making sure that every vertex of G is contained within the given vertices of the spanning tree.

1b) Given a graph G and a number k, does G have no spanning tree of cost less than or equal to k? Using the minimum spanning tree (MST) algorithms learned in lecture: Prim's and Kruskal's algorithms, it is clear that given a graph G, the cost of the minimum spanning tree can be found in polynomial time since these are both polynomial time greedy algorithms. Thus, if the minimum cost of a spanning tree of G is found, it is easily discernable whether G has a spanning tree of cost less than or equal to k since this is only true if $k \geq$ minimum cost and evidently that G has no spanning tree of cost less than or equal to k if k is less than minimum cost.

1c) Given a CNF expression E and a number k, is there an assignment which makes at least k clauses true?

Given the assignment of every variable, it can be verified in polynomial time that the assignment makes at least k clauses true by substituting the values of every variable assignment and evaluating each individual clause of the CNF expression E. This evaluation can clearly be done in polynomial time (linear time with respect to the number of literals) since it is just ANDs and ORs in boolean logic. Finally, the count of the number of clauses that are made true can be kept and easily compared to k.

2 The simple k-path problem takes a graph G and a number k, and asks whether G has a path of at least k edges such that no vertex appears on the path more than once. Show that the simple k-path problem is NP-complete.

The simple k-path problem can be shown to be NP-complete by first demonstrating that the simple k-path problem is in NP and finally providing a clear reduction from a known NP-complete problem as well as proving that this reduction is correct. If given the list of vertices to be visited in order, we can follow this path from the start vertex to the end vertex and verify that this path does not repeat any vertices (any type of set or array would suffice). Next, the polynomial reduction will be performed from a known NP-complete problem and will be proven correct. The reduction for the simple k-path problem can be performed from the undirected hamilton cycle problem, which was proven to be NP-complete in lecture. Given a hamilton cycle problem with graph G, it will have a hamilton cycle of length l where l is equal to $|N|$. Since if a solution exists to the hamilton cycle problem, then there exists a path that visits every vertex only once and returns to the starting vertex (for this problem, the first vertex will not be revisited in order to satisfy the restriction that no vertex appears on the path more than once). Every subsection of this path is clearly a simple k-path where k is the number of vertices in the path. For the reverse, it is clear that if there is a simple k-path for some graph G, then clearly the path is a hamilton cycle since G only has k nodes.

3 Assume that it is NP-complete to determine whether a graph can be colored with 3 colors. Show that it is NP-complete to determine whether a graph can be colored with 4 colors.

The problem to determine whether a graph can be colored with 4 colors can be shown to be NP-complete by first demonstrating that the problem is in NP and finally providing a clear reduction from a known NP-complete problem as well as proving that this reduction is correct. In the first step of demonstrating that the problem is in NP, given a list of vertices and their corresponding color, it can be checked in polynomial time that the list contains less than or equal to 4 colors as well as that for every vertex, every adjacent vertex is not of the same color. Next, the polynomial reduction will be performed from a known NP-complete problem and will be proven correct. Since it can be assumed that it is NP-complete to determine whether a graph can be colored with 3 colors, the polynomial time reduction is from the 3 color graph coloring problem. Given a graph G, G' can be constructed by simply adding one new vertex to the original graph G and coloring that one vertex a color distinct from the colors in G. Then, an edge from the new vertex can be created so that it is adjacent to every other vertex in G'. Furthermore, it is clear that if G can be colored with 3 colors, G' can be colored with 4 colors since the new vertex is connected to every other vertex and every vertex and without the edge to the new vertex does not contain an adjacent vertex of the new color. In the same way, if G' can be colored with 4 colors, then G can be colored with 3 colors since the new vertex added is the only one of its color and is connected to every other vertex in G'. Evidently, if without this vertex of a distinct color connected to every other vertex, G must be able to be colored with 3 colors.

4 Show that it is NP-complete to determine whether a graph has a clique consisting of exactly half of the vertices in the graph.

The problem to determine whether a graph has a clique consisting of exactly half of the vertices in the graph can be shown to be NP-complete by first demonstrating that the problem is in NP and finally providing a clear reduction from a known NP-complete problem as well as proving that this reduction is correct. In the first step of demonstrating that the problem is in NP, it is clear that given a graph G and the vertices of the clique consisting of exactly half of the vertices in the graph, then the solution can be verified in $O(N^2)$ time by checking every pair of vertices. Next, the polynomial reduction will be performed from a known NP-complete problem and will be proven correct. Intuitively, the problem to determine whether a graph has a clique consisting of exactly half of the vertices in the graph is extremely similar to the clique problem covered in lecture. Thus, the reduction will be performed from the clique problem derived in class from the CNF-Satisfiability problem that was proven by Cook's theorem. The reduction from a clique problem is as follows: if k is $\frac{|N|}{2}$, clearly nothing needs to be changed in G' and k'. Next, if k $> \frac{|N|}{2}$, then vertices can be added to G' until N is $2k$. None of the added vertices will need any edges since this will make a difference to the cliques found in the graph. Lastly if $k < \frac{|N|}{2}$, then $|N| - 2k$ vertices will be added. Since the intended effect this time is not to "dilute" k to $\frac{1}{2}$, the new added vertices must connect to every other vertex in G'. The total number of vertices after this step will be $2|N| - 2k$. Finally, the correctness of this reduction will be proven. Since in the case where k is $\frac{|N|}{2}$, then if there is a clique in G of size k, then for each case as explained, there will be a clique of k' where k' is $\frac{|N'|}{2}$. Now in the cases of $k > \frac{|N|}{2}$ if G' has a clique consisting of exactly half of the vertices in the graph, then clearly it is a clique of size k in G since each of the new vertices added do not have edges anywhere thus not being included in any cliques. Lastly, in the case of $k < \frac{|N|}{2}$, the new graph G' has $2|N| - 2k$ vertices meaning that a clique consisting of exactly half of the vertices in the graph would be $|N| - k$. Since only $|N| - 2k$ vertices were added and connected to every other vertex, the clique in G must be of size

$$CLIQUESIZE(G) = |N| - k - added$$

$$CLIQUESIZE(G) = |N| - k - (|N| - 2k)$$

$$CLIQUESIZE(G) = |N| - k - |N| + 2k$$

$$CLIQUESIZE(G) = k$$