James Hippler (HipplerJ)
Oregon State University
CS 325-400 (Spring 2018)
Analysis of Algorithms

Homework 7

Due Date: Sunday, May 27, 2018

1. *(7 pts)* Let X and Y be two decision problems. Suppose we know that X reduces to Y in polynomial time. Which of the following can we infer? Explain

    a. If Y is NP-complete then so is X.
    b. If X is NP-complete then so is Y.
    c. If Y is NP-complete and X is in NP then X is NP-complete.
    d. If X is NP-complete and Y is in NP then Y is NP-complete.
    e. X and Y can't both be NP-complete.
    f. If X is in P, then Y is in P.
    g. If Y is in P, then X is in P.

ANSWERS:

a. FALSE (CANNOT INFER): Y is at least as hard as X due to reduction. Since Y is NP-complete, X could be NP-complete.
b. FALSE (CANNOT INFER): Y is potentially harder than X. Therefore, Y could also be NP-complete
c. FALSE (CANNOT INFER): X can exist in NP but not NP-complete. NP-complete does not by default incorporate all aspects of NP.
d. TRUE (CAN INFER): X is not as hard as Y. Therefore, if Y is NP, it is not NP hard and must be NP complete.
e. FALSE (CANNOT INFER): Y and X can both exist in NP-complete
f. FALSE (CANNOT INFER): Y could exist outside of P problems.
g. **TRUE (CAN INFER): X reduces to Y, meaning that X is no harder than Y.**

2. *(4 pts)* Consider the problem COMPOSITE: given an integer y, does y have any factors other than one and itself? For this exercise, you may assume that COMPOSITE is in NP, and you will be comparing it to the well-known NP-complete problem SUBSET-SUM: given a set S of n integers and an integer target t, is there a subset of S whose sum is exactly t? Clearly explain whether or not each of the following statements follows from that fact that COMPOSITE is in NP and SUBSET-SUM is NP-complete:

    a. SUBSET-SUM ≤p COMPOSITE.
    b. If there is an $O(n^3)$ algorithm for SUBSET-SUM, then there is a polynomial time algorithm for COMPOSITE.
    c. If there is a polynomial algorithm for COMPOSITE, then P = NP.
    d. If P ≠ NP, then ***no*** problem in NP can be solved in polynomial time.

ANSWERS:

a. <u>FALSE</u>: SUBSET-SUM is NP-Complete and COMPOSITE is NP. NP-Complete problems do not reduce to an NP problem unless also NP-Complete.

b. <u>TRUE:</u> There is an $O(n^3)$ algorithm for SUBSET_SUM (NP-Complete) then there is a polynomial time algorithm for COMPOSITE (NP). NP problems can be solved using NP-Complete.

c. <u>FALSE:</u> COMPOSITE is an NP problem but is not definitively NP-Complete. Therefore, it cannot be inferred that P = NP.

d. <u>TRUE:</u> If P ≠ NP, then no algorithm exists to solve all NP problems in polynomial time. The solution for an NP Problem can still be verified in polynomial time, however, this does not mean that the actual solution can also be determined in polynomial time. If this was the case, then these would be considered P problems. This would be contradictory to the P ≠ NP statement.

3. *(3 pts)* Two well-known NP-complete problems are 3-SAT and TSP, the traveling salesman problem. The 2-SAT problem is a SAT variant in which each clause contains at most two literals. 2-SAT is known to have a polynomial-time algorithm. Is each of the following statements true or false? Justify your answer.

    a. 3-SAT $\leq_p$ TSP.
    b. If P $\neq$ NP, then 3-SAT $\leq_p$ 2-SAT.
    c. If P $\neq$ NP, then no NP-complete problem can be solved in polynomial time.

ANSWERS:

a. <u>TRUE:</u> Both 3-SAT and TSP are NP-Complete problems. 3-STAT can be reduced to DIR-HAM-CYCLE which can be reduced to HAM-CYCLE and can finally then be reduced to TSP.

b. <u>FALSE:</u> With P ≠ NP, there exists no polynomial algorithm for solving 3-SAT. In addition, 2-SAT is a P problem and 3-SAT is an NP-Complete problem. Reducing 3-SAT to a P Problem would imply that there is a polynomial algorithm for solving 3-SAT which would be a contradiction to the original statement.

c. <u>TRUE:</u> If it is possible to solve an NP-Complete problem in polynomial time, it must be true that all NP-Complete problems to be solved in polynomial time.

4. *(6 pts)* A Hamiltonian path in a graph is a simple path that visits every vertex exactly once. Show that HAM-PATH = { (G, u, v): there is a Hamiltonian path from u to v in G} is NP-complete. You may use the fact that HAM-CYCLE is NP-complete

ANSWER:

To confirm that the Hamiltonian Path (HAM-PATH) is also NP-Complete it is necessary to confirm the solution can be verified in polynomial time. The function will need a Graph, starting/finishing vertices, and a sequence of vertices from 1 to the total size (n). The function will need to traverse each vertex and confirm that none are revisited from beginning to end (No backtracking). As seen before, we know that traversing the graph is done in polynomial time which can be applied here to demonstrate that HAM-PATH is an NP problem.

To show that HAM-PATH is NP-Complete we next need to demonstrate a problem where we can reduce to HAM-PATH. From this week's course material (and because the question gave a hint), we know that HAM-CYCLE can successfully be reduced to HAM-PATH. A HAM-CYCLE must begin and end with the same vertex in the graph. Unlike a HAM-PATH that begins with one vertex and ends with another. They're similar in the sense that they traverse through each vertex without repeating, however, a HAM-CYCLE ends where it began and a HAM-PATH ends on a specified vertex. From this we can infer that if a HAM-CYCLE is present, there must also be a HAM-PATH present as all nodes are connected and a path exists where each vertex is visited only once. Since we can prove that HAM-CYCLE can be reduced to HAM-PATH, we can therefore also prove that HAM-PATH is an NP-Complete problem.

5. *(5 pts)* LONG-PATH is the problem of, given (G, u, v, k) where G is a graph, u and v vertices and k an integer, determining if there is a simple path in G from u to v of length at least k. Prove that LONG- PATH is NP-complete.

ANSWER:

Similar to problem 4, the initial step toward proving LONG-PATH is NP-Complete is to confirm the solution to LONG-PATH can be confirmed in polynomial time. We'd need to evaluate the set of edges provided for G and the solution for the path length. All edges given must be in G and the length of the past needs to be at least k. This operation can be performed in O(E) time complexity where E is the number of edges and also indicates that the solution is polynomial. From this, we can infer that LONG-PATH is an NP problem.

Next, we need to derive a problem that can be reduced into LONG-PATH to confirm the validity of NP-Complete. From the previous problem in the assignment we are able to determine that HAM-PATH is an NP-Complete Problem which is also capable of reducing to LONG-PATH. HAM-PATH is attempting to find the weight of the longest path without repeating vertices. The LONG PATH problem tries a path with a weight that is at least equivalent to the value of k (Can be greater). The difference is that LONG-PATH has a value for comparison, whereas HAM PATH is making decisions exclusively on its evaluation of the graph. Assuming that the graph is connected, with not outlying vertices, we are able to assign each edges a weight value of 1 and establish the value of k as the total number of vertices. LONG-PATH can now be performed on the problem to find a solution where a path is found of at least k. If a solution is discovered, this would prove that HAM PATH is reducible to LONG-PATH and therefore prove that LONG PATH is NP-Complete.

James Hippler (HipplerJ)
Oregon State University
CS 325-400 (Spring 2018)
Analysis of Algorithms

EXTRA CREDIT *(5 pts)*

The **Traveling Purchaser Problem (TPP)**:

The travelling purchaser travels from marketplace to marketplace searching for goods on his list. The travelling purchaser always returns to his "home" marketplace with all the goods on his list and having spent as little money as possible.

Given a list of marketplaces, the cost of travelling between different marketplaces, and a list of available goods together with the price of each such good at each marketplace, the task is to find for a given "shopping list" of goods the purchasing route with the minimum combined cost of purchases and traveling. The purchasing route solution will indicate both the order the marketplaces are visited and the goods that are purchased at each marketplace. The purchaser must start and finish at the same marketplace and visit each other marketplace at most once (it is not required that all marketplaces be visited).

The decision version of **TPP-D** would ask if there exists a route for a set of marketplaces, travel costs, list of goods, and costs of goods such that the total cost of purchases and travelling is at most k.

Formally **TPP-D** = { (G, C, L, A, P, k) : where G=(V,E) where the vertices are marketplaces, E is the set of edges which represent the road between the marketplaces, C(u,v) is the cost of travelling from marketplace u to v, L is the shopping list of goods that the purchaser wants to buy, A(v) is the list of goods available at marketplace v and P(x,v) is the price of the good x at marketplace v. G has a TPP route with total cost at most k }.

Prove that **TPP-D** is NP-Complete

ANSWER:

The goal for the Travelling Purchaser Problem (TPP) is to find the existence of a path for a set of market places (vertices), purchaser Travel Costs (edge weights), the list of goods, and the cost of goods such that all items are purchased, the purchaser returns to the original vertex, and that they expenses are at most k. To prove that the TPP-D problem is NP-Complete, it is necessary to prove that the problem is both NP and also reducible to a similar NP-Complete Problem.

James Hippler (HipplerJ)
Oregon State University

While the structure of this problem is similar to the Hamiltonian Cycle (HAM-CYCLE) presented in problem 4, we'll instead use the Travelling Salesperson Problem (TSP) as the basis for our model on proving that TPP-D is NP complete.

We know from experience with both the TSP and HAM-CYCLE that this problem can be solved in polynomial time as it involves traveling to vertices exactly once throughout the graph before returning to the original, starting vertex. Since the problem can be solved in O(E) polynomial time, we know that it must be an NP problem.

To perform our reduction, we'll use the Travelling Salesperson Problem (TSP) by proving that if there is an existence of an applicable path for the TSP then by definition there must also be a path in the TPP problem that is at most a cost of k. This is because there will be an edge (route) between all marketplaces. While the TSP adds edges with potential customers, the TPP adds edges with potential list items. Essentially TSP and TPP are the same problems with the only differences being minimizing cost over maximizing sales. Since we can prove that TSP can be reduced to TPP, we can therefore also prove that TPP is an NP-Complete problem.