Algorithms and Complexity: Homework #5

Due on March 24, 2019 at the start of class

Professor Bradford

Patrik Sokolowski

Problem 1

Given a digraph $G = (V, E)$ and the edge weigth function w where

$$w : E \longrightarrow \mathbb{Z}.$$

That is, each edge has an integer weight.

Definition 1 (Exact path length problem)**.** Consider an integer edge weighted digraph G. Given an integer κ, the **EPL** *(exact path length problem)* is to determine whether there is a path between a given pair vertices costing exactly κ.

Show the exact path problem is $\mathcal{NP}$-complete.

We can show that a problem is NP-complete by (1) Showing it's NP hard and (2) relating it to another NP complete problem. This problem here is 100 % NP-complete as we can solve this in polynomial time. If we take a set of k nodes, in polynomial time we can check each distance to each node and sum them for the exact path length.

(1)To show it's NP hard we can simply show it is P hard. P class algorithms are by default also NP hard. We can show this by proving a linear time algorithm exists to do this, here is my psuedo code. This code takes a begining node and runs off in the graph until it finds the node it was looking for. This is NP-hard as there is a polynomial time reduction from every piece as well. By varifying this solution, there exists a deterministic way to solve the problem, which means it is still NP-Hard.

Grab the first node
Verify the node's edges
Choose an edge to go off of.
If the edge hits a dead end, we go back and continue off the previous node.
With given edges (saved) continue through the graph.
while traversing, we will save the weights/paths.
if we hit a dead end, there is no path, meaning there is no cost k that will ever lead us to given node (possible that the node to go to doesn't exist)
if we hit the wanted node, sum the weights/paths. (a subset of sums in the graph)
if the path at any point costs more than k, we go back to previous node and try again.
if the path completes and the cost is lower than k, we go back.
if the path finds our wanted node and cost = k, return the path.
otherwise k isn't applicable to the graph.

This can possibly also tell us that the SubsetSum problem and the EPL problem are similar. This helps us in (2).

(2) Every problem that is NP-complete can be reduced to a 3SAT problem. With that, we can also compare the EPL problem to the SubSet problem as we are grabing a subset of the graph and summing once the conditions are met. SubSet Sum (SS) can also be reduced to 3SAT. Given an instance of the EPL problem and SS problem, we can compare their 'algorithm routines' mid-run and show that they equal.

When EPL algorith mis running we are constantly checking/saving sums/paths. SS does this as well when applied to a graph.

Problem 2

Page 505, # 2.

2. A store trying to analyze the behavior of its customers will ofte a two-dimensional array A, where the rows correspond to its and the columns correspond to the products it sells. The entry $A[i, j]$ specifies the quantity of product j that has been purchased by customer i.

Here's a tiny example of such an array A.

	liquid detergent	beer	diapers	cat litter
Raj	0	6	0	3
Alanis	2	3	0	0
Chelsea	0	0	0	7

One thing that a store might want to do with this data is the following. Let us say that a subset S of the customers is *diverse* if no two of the of the customers in S have ever bought the same product (i.e., for each product, at most one of the customers in S has ever bought it). A diverse set of customers can be useful, for example, as a target pool for market research.

We can now define the Diverse Subset Problem as follows: Given an $m \times n$ array A as defined above, and a number $k \leq m$, is there a subset of at least k of customers that is *diverse*?

Show that Diverse Subset is NP-complete.

This problem NP-complete. If we grab a set of k customers we can prove in polynomial time that no 2 bought the same item in a given instance. Given that we can get our diverse set of customers.

With a graph G and a customer per node, we can assign the product to an edge. We can make a list saying customer k bought product e if e is an edge to that customer node. We can ask the list if it has a diverse-subset of size k.

If there is a diverse subset in the graph, those nodes will have unique edges that aren't shared, with that we can get the independent set of size k and return it for said research. This can be directly related to if they are diverse or not as if each unique customer (k) holds a unique edge, they are by default having said unique product that no two customers have bought.

Psuedo code to prove this:

in Grapg G start at a random point (or un-random, just a point).
Run through and catch those unique k's.
Once the graph has been run-through, we should have all of our given k's (if graph properties hold true).
We can compare the list of actual k customers to the wanted number if we want a specific number of k customers that are diverse.
By default any k in the post-search list should be diverse.

This too can be reduced into 3SAT. If any instance of 3SAT is satisfiable, then our transformed instance of the Diverse Subset problem has k-indep nodes. Given the boolean satifiability problem (SAT, which is by nature a reduced 3SAT) we can show Diverse Subset problem is NP-complete. We showed our Independent set was Lower than or equal to (sub p) to the diverse set such as the SAT proves the given boolean's satisfiability true via lower than or equal to (sub p) the given condition. If we say B is the amount of the nodes (or cost) we can set this equal to k and prove it using the above explanations.

Problem 2 continued on next page...

Problem 3

Page 511, #13.

13. A *combinatorial auction* is a particular mechanism developed by economists for selling a collection of items to a collection of potential buyers. (The Federal Communications Commission has studied this type of auction for assigning stations on the radio spectrum to broadcasting companies.)

Here's a simple type of combinatorial auction. There are n items for sale, labeled $I_1, \ldots, I_n$. Each item is indivisible and can only be sold to one person. Now, m different people place *bids*: The i^{th} bid specifies a subset S_i of the items, and an *offering price* x_i that the bidder is willing to pay for the items in the set S_i, as a single unit. (We'll represent this bid as the pair (S_i, x_i).)

An auctioneer now looks at the set of all m bids; she chooses to *accept* some of these bids and to *reject* the others. Each person whose bid i is accepted gets to take all the items in the corresponding set S_i. Thus the rule is that no two accepted bids can specify sets that contain a common item, since this would involve giving the same item to two different people.

The auctioneer collects the sum of the offering prices of all accepted bids. (Note that this is a "one-shot" auction; there is no opportunity to place further bids.) The auctioneer's goal is to collect as much money as possible.

Thus, the problem of *Winner Determination for Combinatorial Auctions* asks: Given items $I_1, \ldots, I_n$, bids $(S_1, x_1), \ldots, (S_m, x_m)$, and a bound B, is there a collection of bids that the auctioneer can accept so as to collect an amount of money that is at least B?

Example. Suppose an auctioneer decides to use this method to sell some excess computer equipment. There are four items labeled "PC," "monitor," "printer", and "scanner"; and three people place bids. Define

$$S_1 = \{\texttt{PC}, \texttt{monitor}\}, S_2 = \{\texttt{PC}, \texttt{printer}\}, S_3 = \{\texttt{monitor}, \texttt{printer}, \texttt{scanner}\}$$

and

$$x_1 = x_2 = x_3 = 1.$$

The bids are (S_1, x_1), (S_2, x_2), (S_3, x_3), and the bound B is equal to 2.

Then the answer to this instance is no: The auctioneer can accept at most one of the bids (since any two bids have a desired item in common), and this results in a total monetary value of only 1.

Prove that the problem of Winner Determination in Combinatorial Auctions is NP-complete.

The Winner Determination in Combinatorial Auctions problem is NP-complete. If we take a set of biders, in polynomial time we can check that no 2 bought/bid on the same item (their total value being B). Just like problem 2, we can prove one subset is lower than or equal to another that we are looking for and show through a psuedo code/explanation that it is NP-complete.(this problem is wierdly similar to the Diverse Subset problem as in a perspective we can sort of look at this as a conditional Diverse Subset problem).

We can show a set in our graph is (through Karp reduction type denotation) is lower than or equal to (sub p) our 'Diverse set'. Given a graph G and a number of k's (such as the last problem) we can set a bidder to each node (just like our customers earlier). Just like problem 2 we can set each edge of G to an item for bidding, so that each bidder places a bid on each item e. Each e is unique as they can't 'overlap' and share very specific nodes. If we set each bid to 1 (or any equal arbitrary value) we can ask each bidder if they can go with such costs (set of bids) resultig is a total value of B = k.

This holds as the previous properties we discussed in problem 2's graphs hold here as well. We can also prove this is NP-complete still as it is a conditional version of the Diverse Subset problem which is proven to be NP complete. (two are similar by nature, just different names and different costs to check/return).