Algorithms HW2

Greyson Brothers

February 21, 2023

Statement of Integrity: I, Greyson Brothers, attempted to answer each question honestly and to the best of my abilities. I cited any and all help that I received in completing this assignment.

1 Problem

Show that the $\leq_P$ relation is a transitive relation on languages. That is, show that if $L1 \leq_P L2$ and $L2 \leq_P L3$, then $L1 \leq_P L3$.

The following definition arises from the textbook's formal-language framework for decision problems (CLRS Section 34.3): A language L1 is polynomial-time reducible to L2 (i.e. $L1 \leq_P L2$) if there exists a polynomial-time computable function:

$$f : \{0,1\}^* \to \{0,1\}^*$$

such that for all $x \in \{0,1\}^*$,

$$x \in L1 \iff f(x) \in L2$$

So, applying the definition to our problem statement we have:

$$L1 \leq_P L2 \Rightarrow \exists f(L1) \to L2$$

$$L2 \leq_P L3 \Rightarrow \exists g(L2) \to L3$$

Let h be the composition $f \circ g$. Since polynomials are closed under composition, h must also be a polynomial-time computable function. It follows that

$$h(L1) = g(f(L1)) = g(L2) = L3$$

Per the CLRS definition, the existence of a function $h(L1) \to L3$ implies that $L1$ is polynomial-time reducible to $L3$, i.e. $L1 \leq_P L3$. Thus the $\leq_P$ relation is transitive ■

2 Problem

Recall the definition of a complete graph K_n is a graph with n vertices such that every vertex is connected to every other vertex. Recall also that a clique is a complete subset of some graph. The graph coloring problem consists of assigning a color to each of the vertices of a graph such that adjacent vertices have different colors and the total number of colors used is minimized. We define the chromatic number of a graph G to be this minimum number of colors required to color graph G. Prove that the chromatic number of a graph G is no less than the size of the maximal clique of G.

Suppose the maximal clique C of G contains $|C| = m$ nodes. For our graph coloring problem, since each node is connected to each other node, no node in the clique can have the same color. This means in order to color the clique, m different colors are needed. In the simplest case, the clique is the graph $(C = G)$ and the chromatic number $n = m$. Adding additional nodes to the graph outside of the maximal clique will either require reusing colors from the clique or adding a new color depending on the connected-ness of the additional nodes, so the chromatic number of G is at least the size of the maximal clique of G ■

3 Problem

Suppose you're helping to organize a summer sports camp, and the following problem comes up. The camp is supposed to have at least one counselor who's skilled at each of the n sports covered by the camp (baseball, volleyball, and so on). They have received job applications from m potential counselors. For each of the n sports, there is some subset of the m applicants qualified in that sport. The question is "For a given number $k \leq m$, is is possible to hire at most k of the counselors and have at least one counselor qualified in each of the n-sports?" We'll call this the Efficient Recruiting Problem. Prove that Efficient Recruiting is NP-complete.

First we want to show that the Efficient Recruiting Problem $(ERP) \in NP$ by finding a polynomial-time algorithm to verify solutions to the problem. To do this, we must first define what a solution to the problem looks like. One acceptable form would be an k x n matrix, where the rows represent k candidates and the columns represent n sports. A 1 at $matrix[i, j]$ means the i^{th} candidate is skilled at the j^{th} sport, and a 0 means they are not.

To verify a solution, we simply want to check that the sum of each column is at least 1. This involves looping through each column, and on each iteration

summing the k values of the column. If the sum is zero, the verification algorithm returns false. If the sum for each column is ≥ 1, then the algorithm returns true. The time complexity for this verification is $O(kn)$ which is firmly in polynomial-time, so $ERP \in NP$

Now, we must show that $ERP \in NP - Hard$, which we will do by showing that an NP-Complete can be reduced to ERP. In the group discussion, Corbin Jones recommended using the Set Cover Problem as a target for reduction, which I will explore here. An instance of the Set Cover Problem (SCP) is defined in CLRS 35.3 as a finite set X and a family F of subsets of X such that every element of X belongs to at least one subset in F. As Corbin pointed out, this lends itself very naturally to the ERP.

An instance of the ERP would be an m x n matrix with m potential counselors and n sports, almost identical to the solution form presented above. To convert between an SCP instance and an ERP instance, let $|F| = m$ and $|X| = n$. Then create an m x n matrix of zeros, which takes $O(nm)$. Here each element of the set X gets its own column and is treated as a sport. Then for each subset i in F find the corresponding row i in the matrix and set 1 for any sports columns it contains. These family subsets act as our potential counselors, with the elements in the subset being the sports that the counselor is skilled at. This second operation also takes $O(nm)$, and finishes the reduction of a instance of SCP to an instance of ERP. The total reduction time complexity is $O(2nm) = O(nm)$ which is linear, so $SCP \leq_P ERP$

From CLRS Lemme 34.8, $ERP \in NP$ and $ERP \in NP - Hard \Rightarrow ERP \in NPC$■

4 Problem

We start by defining the Independent Set Problem (IS). Given a graph $G = (V, E)$,we say a set of nodes SV is independent if no two nodes in S are joined by an edge. The Independent Set Problem, which we denote IS, is the following. Given G, find an independent set that is as large as possible. Stated as a decision problem, IS answers the question: "Does there exist a set $S \subseteq V$ such that $|S| \geq k$?" Then set k as large as possible. For this problem, you may take as given that IS is NP-complete.

A store trying to analyze the behavior of its customers will often maintain a table where the rows of the table correspond to the customers and the columns (or fields) correspond to products the store sells. The entry $A[i, j]$ specifies the quantity of product j that has been purchased by customer i. For example, Table 1 shows one such table.

One thing that a store might want to do with this data is the following. Let's say that a subset S of the customers is diverse if no two 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 (DS) 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 customers that is diverse? Prove that DS is NP-complete.

First we show that $DS \in$ NP. To do this, we can show that there exists a polynomial time verification algorithm for DS. A solution to this problem is a subset of customers, or $k \leq m$ rows of the matrix in our example. To check if the solution is valid, we need to assert that no pairs in the subset have purchased the same product.

This can be done by checking each pairwise combination of customers, an $O(k^2)$ operation. For each customer combination we need to check that at most one of the two customers has purchased each of the n products, an $O(n)$ operation. If any pairwise combination of customers have purchased the same product, the algorithm returns false. If not, the algorithm loops through all customer pairs and products and returns in $O(nk^2)$ for an input of size $O(nk)$, well within polynomial time. Thus $DS \in$ NP.

Now, we must show that $DS \in$ NP-Hard. This can be done by showing that a problem in NPC reduces to DS, and here we will show $IS \leq_P DS$, where $IS \in NPC$ is given by the problem statement. To do so, we must find some polynomial-time computable function which can transform an instance of IS to an instance of DS. We must define these two instances before transforming between them.

An instance of IS is a graph $G = (V, E)$ of m vertices and n edges. This can be represented as an m x m adjacency matrix, where a 1 at cell [i,j] means that nodes i and j have an edge connecting them and a 0 means the nodes are not connected, for $0 \leq i, j \leq m$.

An instance of DS is an m x n matrix of customers (m) and products (n). We want an algorithm to convert this matrix into an adjacency matrix across the customers, where an edge between two customer nodes indicates that they purchased at least one product in common. This can be done in a very similar fashion to our verification algorithm in the first part of the proof.

First we allocate an m x m array of zeros, taking $O(m^2)$. Once this operation is complete, we loop through each pairwise combination of customers i and j ($O(m^2)$, and with each pair we check all n products ($O(n)$). If customers i and j both purchased one of the products, we set array[i,j] = 1 and break. Otherwise we continue until we have checked all customer pairs and their products. In total, this algorithm takes $O(nm^2)$ to convert any DS customer/product table into an adjacency matrix equivalent to a graph $G = (V, E)$ instance of the IS problem. Thus we can say $IS \leq_P DS$ and $DS \in$ NP-Hard.

From CLRS Lemme 34.8, $DS \in NP$ and $DS \in NP-Hard \Rightarrow DS \in NPC$■