Homework #10 (Sections 7.4 – 7.5) Solutions

1) Let MAX-CLIQUE = { <G, k> | G is a graph and its largest clique is of size k}. If CLIQUE is in P, prove that MAX-CLIQUE is ALSO in P. (Namely, given a black box that solves the CLIQUE decision problem in polynomial time, design a solution to MAX-CLIQUE in polynomial time.

Solution

Let TM C solve CLIQUE in polynomial time.
Here is a TM D that solves MAX-CLIQUE in polynomial time:

```
D(Graph G, int k) {
   return C(G, k) && !C(G, k+1)
}
```

Basically, if G does have clique of size k but doesn't have a clique of size k+1, its maximum sized clique is size k. Thus, we can call C twice, and if these answers match the way we want, then G has a maximum sized clique of size k. If they don't match, the maximum sized clique is a different size. (Note that if C runs in polynomial time, two calls to C with the same input graph will also run in polynomial time.)

2) Why are 2 x 3 windows necessary in the proof of the Cook-Levin theorem?

Solution

If we only had 1 row, we could never verify any transitions, thus, 2 rows are necessary. 3 rows are not necessary since two different boxes with 2 rows each can verify two successive transitions.

A 2 x 2 window suffices to check Right moves, but it does NOT suffice to check Left moves. A configuration that occupies symbols k and k+1 prior to a left move occupies symbols k – 1 and k+1 subsequent to the move. Thus, we must be able to view column k – 1 and column k + 1 at the same time subsequent to any left move. The smallest window that fits around both squares has 3 columns in it. Any 2 x 2 window would be missing one of these two squares and would be un

3) Show a polynomial time reduction from 4-SAT to 3-SAT, where 4-SAT represents satisfiability for boolean formulas with four literals in each clause instead of 3, and the formula is still in conjunctive normal form. (Namely, show how to transform a boolean formula in 4-SAT form into an equivalent formula in 3-SAT form such that the input formula is satisfiable if and only if the output formula is.)

Solution

For each clause, $(a \vee b \vee c \vee d)$ in a 4-SAT instance, introduce a new variable z for your corresponding 3-SAT instance and place the two following clauses in the 3-SAT instance you are creating:

$$(a \vee b \vee z) \wedge (c \vee d \vee \bar{z})$$

We need to show that if and only if $(a \vee b \vee c \vee d)$ is satisfiable, $(a \vee b \vee z) \wedge (c \vee d \vee \bar{z})$ is satisfiable.

Consider the one truth setting that does not make the clause true, where each of a, b, c, and d are set to false. In this corresponding setting, we find that our output expression simplifies to $z \wedge \bar{z}$. Using basic logic, we see this simplifies further to false, as desired. Thus, in the case that the input clause is NOT satisfiable, neither is the output class.

Now, for the other 15 possible truth settings that satisfy the input clause, note that we can also satisfy the output clause. In particular, at least one of a, b, c or d must be true to satisfy the original clause. Automatically, this will satisfy one of the two clauses we added. We just need to show the other clause is satisfiable. It is, because we have the freedom to choose z appropriately. If a or b is true, we can set z = false to make the second clause true. Alternatively, if c or d is true, we can set z = true to make the first clause true.

Using this transformation on each clause of the input expression will create an output 3-SAT expression that is satisfiable if and only if the original 4-SAT expression was satisfiable. It runs in polynomial time because we do a constant amount of work for each clause, so the run time is on the order of the number of clauses in the expression, which is proportional to the input size.

4) The independent set problem is as follows: Given a graph G and an integer k, determine whether or not there are k vertices in G such that no two vertices out of the k share the same edge. Prove that the set INDEPENDET-SET is NP-Complete via a reduction from a known NP-Complete problem.

Solution

We will reduce CLIQUE to INDEPENDENT-SET in polynomial time to prove that INDEPENDENT-SET is NP-COMPLETE. Let the input to CLIQUE be a graph G and a positive integer k. The corresponding output for the reduction is $\overline{G}$ and k. Namely, if and only if a graph G has a clique of size k does its complement graph have an independent set of size k.

To prove this, consider a graph G with a CLIQUE of size k. These k vertices are ALL connected. Thus, in $\overline{G}$, NONE of these vertices are connected. Thus, none of these vertices share a common edge, since NONE of these edges are present in the complement graph.

Similarly, consider a graph G with NO CLIQUE of size k. This means that no matter which group of k vertices you choose in the graph G, at least one connection will be missing. Now, consider the complement graph, $\overline{G}$. For any selection of k vertices, it MUST have at least one connection (edge) because at least one is missing in all of these sets of vertices in G. Thus, no choice of k vertices leads to an independent set.

5) n people live in a house and wish to share their expenses equally. Their respective expenses before settling are $x_1, x_2, \ldots, x_n$. Assume that all of these are greater than 0. They agree to write each other checks so as to make each person's expenses equal the average cost. Naturally, they want to minimize the number of checks written. Formalize this as a decision problem and prove that it is NP-Complete.

Solution

A formalized version of the problem is as follows:

ROOMMATE = { $\langle S, k \rangle$ | $S = \{x_1, x_2, \ldots, x_n\}$, $x_i \in Z^+$ for $1 \leq i \leq n$, and k adjustments of the form $x_i = x_i + c_m$, $x_j = x_j - c_m$, for distinct integers i and j with $1 \leq i, j \leq n$ and for $c_m \in Z^+$ for $1 \leq i \leq k$ will result in $x_i = x_j$ for all integers i and j, $1 \leq i, j \leq n$ }

We can prove this problem is NP-Complete by reducing from SUBSET-SUM.

Consider an arbitrary instance of SUBSET-SUM with the set $\{s_1, s_2, \ldots, s_n\}$ and a target T.

We will create a ROOMMATE instance with n+2 values and set k = n. Let $S = x_1 + x_2 + \ldots + x_n$. Our roommates will have paid the following values:

$S - s_1, S - s_2, \ldots, S - s_n, S + T, 2S - T$

Quickly note that the sum of these n+2 values is (n+2)S, since the two T's cancel and the individual s_i's cancel with the extra S in the last term. Thus, each roommate was supposed to pay S dollars, and none of them have paid exactly that. (We are assuming that the subset sum input values are all positive.) The first n roommates have underpaid with the last two have overpaid.

Consider a situation where there exists a subset of values from the original set that adds up to T. We have some subset of values $\{s_1, s_2, \ldots, s_n\}$ that exactly adds up to T. In this corresponding instance of the roommate problem, we have those particular roommates who have collectively underpaid by T dollars. All of these roommates can pay roommate n+1, who has overpaid by T dollars. The rest of the roommates (all the ones NOT in the original subset) have underpaid by S – T dollars and can pay their debts to roommate n+2, who has overpaid by this amount exactly.

Alternatively, if the ROOMMATE problem has a solution, then we know that there exists a way to use exactly n checks to even every balance. Since n roommates have underpaid, if there is a solution to the ROOMMATE problem, we know that each of these n roommates must have written exactly one check. If they've written exactly one check each, then only one person can receive each of their checks. It must be either person n+1 or person n+2, who are both owed money. Thus, we can break up the first n roommates into two groups: those who pay person n+1 and those who pay person n+2. It must be the case that if this works, the corresponding answer in the SUBSET-SUM instance corresponds to the set of roommates who paid roommate n+1, since those set of checks will add up to T exactly, the amount overpaid by roommate n+1.