CS 170 Homework 11

Due **4/17/2023, at 10:00 pm (grace period until 11:59pm)**

1 Study Group

List the names and SIDs of the members in your study group. If you have no collaborators, you must explicitly write "none".

2 Warm up: Basic Complexity Concepts

Complexity theory (reductions, **NP**-completeness, etc.) is the most conceptually heavy part of the course. To help guide you through lecture and chapter 8/9 reading, we recommend doing this problem first.

1. Suppose we reduce a problem A to another problem B. This means that if we have an algorithm that solves _____, we immediately have an algorithm that solves _____.

2. Again, suppose we show a (polynomial-time) reduction from A to B. This implies that up to polynomial factors, _____ cannot be any easier than _____.

3. Define the class **NP** in one sentence (search version of **NP**).

4. Define the class **NP**-hard in one sentence.

5. Define the class **NP**-complete in one sentence.

3 California Cycle

Prove that the following problem is **NP**-hard.

Input: A directed graph $G = (V, E)$ with each vertex colored blue or gold, i.e., $V = V_{\text{blue}} \cup V_{\text{gold}}$.

Goal: Find a *Californian cycle*, which is a directed cycle through all vertices in G that alternates between blue and gold vertices.

(Hint: Directed Rudrata Cycle)

4 Reduction, Reduction, and Reduction

In this problem, you will prove three problems are **NP**-Complete. Please be aware that to prove a problem is **NP**-Complete, you need to argue it is in **NP** and then briefly justify a reduction that shows it is **NP**-hard.

(a) Alice and Bob go out on a date at a nice restaurant. At the end of the meal, Alice has eaten c_A dollars worth of food, and has in her wallet a set of bills $A = \{a_1, a_2, \ldots, a_n\}$. Similarly, Bob owes the restaurant c_B dollars and has bills $B = \{b_1, b_2, \ldots, b_m\}$. Now, Alice and Bob are very calculating people, so they agree that each of them should pay their fair share (c_A and c_B, respectively).

One thing they don't mind doing, however, is fairly trading bills. That is, Alice can exchange a subset $A' \subseteq A$ of her bills for for a subset $B' \subseteq B$ of Bob's bills, so long as $\sum_{a \in A'} a = \sum_{b \in B'} b$. Under the above conditions, Alice and Bob wish to find, after trading as many times as desired, subsets of their bills A^*, B^* such that $\sum_{a \in A^*} a = c_A$ and $\sum_{b \in B^*} b = c_B$ so that they are able to pay fairly.

Show that Fair Date is **NP**-complete. (*Hint: You may assume* SUBSET SUM *is* ***NP****-complete.*)

(b) Consider the following problem Public Funds:

> You are looking to build a new fence for your mansion, to keep out pesky people protesting profligate purchases. You have m bank accounts at your disposal to use to pay for your fence; each account i has a balance of b_i. You must choose one of n options for your fence; each fence j costs c_j dollars. You would like to withdraw from at most k of the bank accounts to build the fence, and due to peculiar UC accounting rules, if you use a particular bank account, you must use the whole balance (all b_m dollars.)
> Determine whether it is possible to pay for a new fence by exactly paying for some fence j using k bank accounts. In other words, determine if there is a fence $j \in \{1, \ldots, n\}$ and bank accounts $\{\ell_i\}_{i=1}^{k} \in \{1, \ldots, m\}$ such that
>
> $$c_j = \sum_{i=1}^{k} b_{\ell_i},$$
>
> and if so return the corresponding choice of fence and set of bank accounts that you withdraw from.

Show that Public Funds is **NP**-Complete.

(c) Consider the search problem Max-Acyclic-Induced-Subgraph:

Input: A *directed* graph $G = (V, E)$, and a positive integer k.

Output: A subset of vertices $S \subseteq V$ of size k such that resulting induced graph $G_S = (S, \{(u, v) \in E : u, v \in S\})$ is a DAG.

Show that Max-Acyclic-Induced-Subgraph is **NP**-complete.

5 More Reductions

Given an array $A = [a_1, a_2, \ldots, a_n]$ of nonnegative integers, consider the following problems:

1 **Partition**: Determine whether there is a subset $P \subseteq [n]$ ($[n] := \{1, 2, \cdots, n\}$) such that $\sum_{i \in P} a_i = \sum_{j \in [n] \setminus P} a_j$.

2 **Subset Sum**: Given some integer k, determine whether there is a subset $P \subseteq [n]$ such that $\sum_{i \in P} a_i = k$.

3 **Knapsack**: Given some set of items each with weight w_i and value v_i, and fixed numbers W and V, determine whether there is some subset $P \subseteq [n]$ such that $\sum_{i \in P} w_i \leq W$ and $\sum_{i \in P} v_i \geq V$.

For each of the following clearly describe your reduction, justify runtime and correctness.

(a) Find a linear time reduction from Subset Sum to Partition.

(b) Find a linear time reduction from Subset Sum to Knapsack.

6 Coding Question

For this week's homework, you'll implement a reduction of Set Cover to ILP in the python jupyter notebook called `reductions.ipynb`. There are two ways you can access the notebook and complete the problems:

1. Click here and navigate to the `HW11` folder if you prefer to complete this question on Berkeley DataHub.

2. Run

   ```
   git clone https://github.com/Berkeley-CS170/cs170-coding-notebooks-sp23
   ```

   in your computer's terminal (and navigate to the `HW11` folder) if you prefer to complete it locally. If you run into any issues with python import or autograder errors, please refer to the local setup instructions here.

Notes:

- *Submission Instructions:* Please download your completed `reductions.ipynb` file and submit it to the gradescope assignment titled "Homework 11 Coding Portion".

- *OH/HWP Instructions:* There will be designated office hours for you to come to ask for conceptual and debugging help. Please visit the coding OH post for more information.

- *Academic Honesty Guideline:* We realize that code for some of the algorithms we ask you to implement may be readily available online, but we strongly encourage you to not directly copy code from these sources. Instead, try to refer to the resources mentioned in the notebook and come up with code yourself. That being said, we **do acknowledge** that there may not be many different ways to code up particular algorithms and that your solution may be similar to other solutions available online.