CS41 Homework 9

This homework is due 11:59pm Friday November 24. Submit this homework as a *.tex file using github. For this homework, you will work with a partner. It’s ok to discuss approaches with others at a high level, but most of your discussions should just be with your partner. The only exception to this rule is work you’ve done with a lab partner while in lab. In this case, note who you’ve worked with and what parts were solved during lab. If there are questions about academic integrity, please visit the section on Academic Integrity on the course website.

0. Before final submission, make sure to fill out the README file.

1. Credit Networks. A credit network is a mathematical model that can capture financial relationships among banks or trust relationships between individuals. A credit network involves a set of agents \( A \) (the banks or people) and credit relationships between them. Each agent \( a \in A \) specifies an integer amount of credit they extend \( c_{ab} \geq 0 \) to each other agent \( b \in A \). This credit allows agents to engage in transactions: if agent \( a \) extends credit to agent \( b \), then \( b \) can spend that credit to obtain something of value from \( a \) (banks might purchase commodities; individuals might request favors). When this happens, some of the credit is expended: if the transaction has value \( v \leq c_{ab} \), then there will be \( c_{ab} - v \) units of credit remaining; at the same time, agent \( a \) gains \( v \) units of credit from agent \( b \) (equivalent to an I.O.U. indicating \( b \)'s obligation to later repay the debt). See Figure 1 for an example.

Agents in a credit network can also engage in remote transactions, obtaining something of value from an agent who hasn’t extended them credit directly. Here is how remote transactions work: suppose that agent \( a \) extends \( c_{ab} \) units of credit to agent \( b \), agent \( b \) extends \( c_{bd} \) units of credit to agent \( d \), and agent \( d \) extends \( c_{de} \) units of credit to agent \( e \). If agent \( e \) wants to make a purchase from agent \( a \), they can first transact with agent \( d \), using their credit from \( d \) to buy some of \( d \)'s credit from agent \( b \), then use that credit from agent \( b \) to buy credit from agent \( a \), and finally make a purchase from agent \( a \) using this credit. See Figure 2 for an example.

![Figure 1](image1.png)

(a) before transaction
(b) after transaction

Figure 1: \( b \) buys from \( a \), expending \( v \) units of credit.
(a) An important query about credit networks is whether a certain transaction is feasible. In the TRANSACTION FEASIBILITY problem we are given a credit network, two agents \( s \) and \( t \), and a transaction size \( v \). Our job is to determine whether it is possible for agent \( s \) to make a purchase from agent \( t \) of total value \( v \) in the credit network. Describe and analyze an efficient algorithm for the transaction feasibility problem.

(b) In some settings, agents might be unwilling to exchange large amounts of credit for a single transaction. This leads to a variant of the problem where we have the additional input of an exchange limit \( \ell_a \) for each agent \( a \in A \), and the additional constraint that for any agent \( a \) other than \( s \) and \( t \), the total amount of other agents’ credit they sell as a part of any given transaction cannot exceed \( \ell_a \). The EXCHANGE-LIMITED TRANSACTION FEASIBILITY problem asks whether a transaction of size \( v \) between agents \( s \) and \( t \) is feasible, while respecting the exchange limits. Describe and analyze an efficient algorithm for this problem.

2. **Preregistration.** At Eromhtraws College, a peculiar institution, students register for courses in the upcoming semester in an unusual way. In particular, students enroll in courses before the courses get assigned to times/classrooms.

   The usual procedure is as follows:

   Step 1: Announcement of courses \( C = \{c_1, \ldots, c_m\} \) offered for the next semester. Students \( S = \{s_1, \ldots, s_n\} \) consider and meet with their advisors.

   Step 2: Each student \( s_i \) submits a list \( \ell_i \subseteq C \) of courses they are eligible for and interested in taking, as well as an integer \( k_i \), the number of courses that they want to take.

   Step 3: The registrar is given the list of courses \( C \), the list of each student \( s_i \)'s preferences \((\ell_i, k_i)\), and a general limit on class size \( L \). The registrar attempts to produce a list of class enrollments satisfying two properties:

   - Every student \( s_i \) is enrolled in exactly \( k_i \) courses.
   - Every course has at most \( L \) students enrolled.
The problem of producing such a list or determining that it is impossible is called enrollment. (When the answer is impossible, the registrar usually barricades himself in his office and produces a schedule by hand with the goal of satisfying no one’s preferences.)

Step 4: The building managers are given a list of course enrollments, the total number \( r \) of classrooms on campus, and the number \( c \) of weekly timeslots during which classes can be scheduled. The building managers must assign each class a regular weekly time and room so that:

- Each course is assigned a classroom and a timeslot.
- Each classroom is used by at most one class during any of the \( c \) timeslots.
- No student is forced to bilocate. (That is, if there is a student enrolled in both class A and class B, then those classes must be scheduled at different times.)

The problem of deciding whether it is possible to schedule all classes is called classrooms. (The building managers at Eromhtraws College have a lot of power.)

This procedure is complicated, and frequently runs into snags. Sometimes the enrollment problem gets stuck because it is not possible for every student to be enrolled in the minimum necessary number of courses. Sometimes the classroom problem gets stuck because it is not possible for all the classes to be scheduled in the available spaces.

(a) Design and analyze a polynomial-time algorithm for enrollment which decides if a valid enrollment is possible, and if it is, gives the list of students enrolled in each course.

(b) Show that the decision problem classrooms is NP-Complete.

This suggests that Eromhtraws College’s preregistration process is backwards, indeed.

3. (K&T 8.6) Consider an instance of the Satisfiability problem, specified by clauses \( c_1, \ldots, c_k \) over a set of Boolean variables \( x_1, \ldots, x_n \). We say that the instance is monotone if each term in each clause consists of a nonnegated variable; that is, each term is equal to \( x_i \), for some \( i \), rather than \( \bar{x}_i \). Monotone instances of SAT are very easy to solve: they are always satisfiable, by setting each variable to “true”.

For example, suppose that we have the three clauses:

\[(x_1 \lor x_2), (x_2 \lor x_3), (x_1 \lor x_3)\]

This is monotone, and the all-true assignment satisfies all the clauses. But this is not the only satisfying assignment: we could have set \( x_1 \) and \( x_2 \) to true, and \( x_3 \) to false. Indeed, for any monotone instance, it is natural to ask how few variables we need to set “true” in order to satisfy all clauses.

Given a monotone instance of satisfiability, together with a number \( k \), the problem of monotone satisfiability with few true variables asks: is there a satisfying assignment for the instance in which at most \( k \) variables are set to “true”? Prove this problem is NP-Complete.
4. **Extra credit.** In a credit network, sometimes instead of setting an exchange limit, agents are allowed to charge an exchange rate. If agent $a \in A$ charges exchange rate $r_a$, then exchanging each unit of credit incurs a cost of $r_a$. For example, when agent $s$ purchases $v$ units of agent $b$’s credit from agent $a$, agent $s$ must pay a cost of $r_a \cdot v$.

(a) The **min-cost transaction** problem is to determine whether a transaction of size $v$ between agents $s$ and $t$ is feasible, and if so, to determine the set of credit exchanges that will complete the transaction at the lowest total cost (summed over all nodes where credit is exchanged). Describe and analyze an efficient algorithm for the **min-cost transaction** problem.

(b) In a credit network, it is natural to denominate costs not in a common currency, but in units of the credit being exchanged. In the **max-credit transaction** problem, exchange rates must be paid with additional credit, so purchasing $v$ units of credit from agent $a$ costs $(1 + r_a)v$ units of agent $a$’s credit. In this case, we must determine whether a transaction is feasible, and if so, maximize the total direct credit available to the purchasing agent $t$ after the transaction: $\max \sum_{a \in A} c_{at}$ Describe and analyze an efficient algorithm for the **max-credit transaction** problem.