Problem 1  (20 points)

A circuit of a matroid is a dependent set of elements such that any proper subset is independent. For a given matroid \( M = (S, \mathcal{I}) \), we denote the set of circuits of the matroid by \( \mathcal{C} \). In other words, \( C \in \mathcal{C} \) if \( C \notin \mathcal{I} \) and \( C' \in \mathcal{I} \) for all \( C' \subseteq C \) (i.e., for all proper subsets \( C' \)). It is easy to see that the following properties hold:

Circuit property 1: \( \emptyset \notin \mathcal{C} \).

Circuit property 2: If \( C_1, C_2 \in \mathcal{C} \) and \( C_1 \subseteq C_2 \), then \( C_1 = C_2 \).

a.  (5 points)
Prove the following:

Circuit property 3: If \( C_1 \) and \( C_2 \) are distinct members of \( \mathcal{C} \) and \( x \in C_1 \cap C_2 \), then there exists a circuit \( C_3 \in \mathcal{C} \) such that \( C_3 \subseteq (C_1 \cup C_2) - x \).

b.  (5 points)
Let \( S \) be a set of elements, and let \( \mathcal{C} \) be a family of subsets of \( S \) that satisfy circuit properties 1, 2, and 3. Let \( \mathcal{I} \) be the family of subsets of \( S \) containing no member of \( \mathcal{C} \) (i.e., \( \mathcal{I} = \{ A \subseteq S : A' \notin \mathcal{C} \text{ for all } A' \subseteq A \} \)). Prove that \( (S, \mathcal{I}) \) is a matroid having \( \mathcal{C} \) as its family of circuits.

Observe that parts (a) and (b) imply the following (which you need not prove):

Let \( \mathcal{C} \) be a family of subsets of set \( S \). Then \( \mathcal{C} \) is the family of circuits of a matroid on \( S \) if and only if \( \mathcal{C} \) satisfies circuit properties 1, 2, and 3.
c. (5 points)
Let \( G = (V, E) \) be an undirected graph, and let \( C \) be the set of simple cycles of \( G \). (In a simple cycle, no vertex is repeated except for the first and last vertices are the same.) Prove that \( C \) is the set of circuits of some matroid \((E, \mathcal{I})\).

The \textbf{longest-simple-cycle} problem is the problem of determining a simple cycle of maximum length in an undirected graph. This problem is NP-complete.

d. (5 points)
Professor Merckx believes that he has proven the following “theorem”:

Let \( G = (V, E) \) be an undirected graph, and let \( C \) be the set of simple cycles of \( G \). Then \((E, C)\) is a matroid.

Discuss why Professor Merckx’s discovery, if correct, would be front-page news in the \textit{New York Times}.

---

**Problem 2** (20 points)

In the \textbf{parallel-machine-scheduling problem}, we are given \( n \) jobs, \( J_1, J_2, \ldots, J_n \), where each job \( J_k \) has an associated nonnegative processing time of \( p_k \). We are also given \( m \) identical machines, \( M_1, M_2, \ldots, M_m \). A \textbf{schedule} specifies, for each job \( J_k \), the machine on which it runs and the time period during which it runs. Each job \( J_k \) must run on some machine \( M_i \) for \( p_k \) consecutive time units, and during that time period no other job may run on \( M_i \). Let \( C_k \) denote the \textbf{completion time} of job \( J_k \), that is, the time at which job \( J_k \) completes processing. Given a schedule, we define \( C_{\text{max}} = \max_{1 \leq j \leq n} C_j \) to be the \textbf{makespan} of the schedule. The goal is to find a schedule whose makespan is minimum.

For example, suppose that we have two machines \( M_1 \) and \( M_2 \) and that we have four jobs \( J_1, J_2, J_3, J_4 \), with \( p_1 = 2, p_2 = 12, p_3 = 4, \) and \( p_4 = 5 \). Then one possible schedule runs, on machine \( M_1 \), job \( J_1 \) followed by job \( J_2 \), and on machine \( M_2 \), it runs job \( J_4 \) followed by job \( J_3 \). For this schedule, \( C_1 = 2, C_2 = 14, C_3 = 9, C_4 = 5, \) and \( C_{\text{max}} = 14 \). An optimal schedule runs \( J_2 \) on machine \( M_1 \), and it runs jobs \( J_1, J_3, \) and \( J_4 \) on machine \( M_2 \). For this schedule, \( C_1 = 2, C_2 = 12, C_3 = 6, C_4 = 11, \) and \( C_{\text{max}} = 12 \).

Given a parallel-machine-scheduling problem, we let \( C^*_\text{max} \) denote the makespan of an optimal schedule.

a. (5 points)
Show that the optimal makespan is at least as large as the greatest processing time, that is,

\[
C^*_\text{max} \geq \max_{1 \leq k \leq n} p_k .
\]

b. (5 points)
Show that the optimal makespan is at least as large as the average machine load, that is,

\[
C^*_\text{max} \geq \frac{1}{m} \sum_{1 \leq k \leq n} p_k .
\]
c. (5 points)
Suppose that we use the following greedy algorithm for parallel machine scheduling: whenever a
machine is idle, schedule any job that has not yet been scheduled. For the schedule returned by the
greedy algorithm, show that
\[ C_{\text{max}} \leq \frac{1}{m} \sum_{1 \leq k \leq n} p_k + \max_{1 \leq k \leq n} p_k. \]

d. (5 points)
Prove that the algorithm in part (c) is a polynomial-time 2-approximation algorithm for the parallel-
machine-scheduling problem.

Problem 3 (15 points)
Professor Dante has been studying the analysis of disjoint-set forests with union by rank and path com-
pression. He reasons that because node ranks increase strictly along a path to the root, node levels must
monotonically increase along the path. In other words, if \( \text{rank}(x) > 0 \) and \( \rho[x] \) is not a root, then
\( \text{level}(x) \leq \text{level}(\rho[x]) \). Is the professor correct? Justify your answer.

Problem 4 (20 points)
Show how to make the push-relabel method run in \( O(V^3) \) time if you always discharge a highest overflowing
vertex, and show that the running time is indeed \( O(V^3) \). If you require any properties in CLRS, please cite
them by the appropriate theorem, lemma, corollary, or equation number; there is no need to prove them in
your writeup.

Make sure that your description includes how to find a highest overflowing vertex so that the entire
algorithm runs in \( O(V^3) \) time. You may modify the Push, Relabel, and Discharge procedures, but
any changes you make to these procedures must be in addition to what they already do. If you add any new
data structures, describe them in sufficient (but not overbearing) detail.

Problem 5 (25 points)
Let \( \mathcal{H} \) be a class of hash functions in which each hash function \( h \in \mathcal{H} \) maps the universe \( U \) of keys
to \( \{0, 1, \ldots, m - 1\} \). We say that \( \mathcal{H} \) is \textit{k-universal} if, for every fixed sequence of \( k \) distinct keys
\( \{x^{(1)}, x^{(2)}, \ldots, x^{(k)}\} \) and for any \( h \) chosen at random from \( \mathcal{H} \), the sequence \( \{h(x^{(1)}), h(x^{(2)}), \ldots, h(x^{(k)})\} \)
is equally likely to be any of the \( m^k \) sequences of length \( k \) with elements drawn from \( \{0, 1, \ldots, m - 1\} \).

a. (5 points)
Show that if the family \( \mathcal{H} \) of hash functions is 2-universal, then it is universal.
b. (5 points)
Suppose that the universe $U$ is the set of $n$-tuples of values drawn from $\mathbb{Z}_p = \{0, 1, \ldots, p - 1\}$, where $p$ is prime. Consider an element $x = \langle x_0, x_1, \ldots, x_{n-1} \rangle \in U$. For any $n$-tuple $a = \langle a_0, a_1, \ldots, a_{n-1} \rangle \in U$, define the hash function $h_a$ by

$$h_a(x) = \left( \sum_{j=0}^{n-1} a_j x_j \right) \mod p .$$

Let $\mathcal{H} = \{h_a\}$. Show that $\mathcal{H}$ is not 2-universal. (*Hint: Find a key for which all hash functions in $\mathcal{H}$ produce the same value.*)

c. (10 points)
Suppose that we modify $\mathcal{H}$ slightly from part (b): for any $a \in U$ and for any $b \in \mathbb{Z}_p$, define

$$h_{a,b}^i(x) = \left( \sum_{j=0}^{n-1} a_j x_j + b \right) \mod p$$

and $\mathcal{H}' = \{h_{a,b}^i\}$. Argue that $\mathcal{H}'$ is 2-universal. (*Hint: Consider fixed $x \in U$ and $y \in U$, with $x_i \neq y_i$ for some $i$. What happens to $h_{a,b}^i(x)$ and $h_{a,b}^i(y)$ as $a_i$ and $b$ range over $\mathbb{Z}_p$?)

d. (5 points)
Suppose that Alice and Bob secretly agree on a hash function $h$ from a 2-universal family $\mathcal{H}$ of hash functions. Each $h \in \mathcal{H}$ maps from a universe of keys $U$ to $\mathbb{Z}_p$, where $p$ is prime. Later, Alice sends a message $m$ to Bob over the Internet, where $m \in U$. She authenticates this message to Bob by also sending an authentication tag $t = h(m)$, and Bob checks that the pair $(m, t)$ he receives indeed satisfies $t = h(m)$. Suppose that an adversary intercepts $(m, t)$ en route and tries to fool Bob by replacing the pair $(m, t)$ with a different pair $(m', t')$. Argue that the probability that the adversary succeeds in fooling Bob into accepting $(m', t')$ is at most $1/p$, no matter how much computing power the adversary has, and even if the adversary knows the family $\mathcal{H}$ of hash functions used.