For this example, you can use your notes and the notes on the course page, but you should not.

**Problem 1.** Prove that $\frac{1}{2}n^3 + 2(n - 1)^2 + 42 \in O(n^4)$.

For the next three problems, consider the following.

For a matrix $M$ of dimensions $2n$ by $2m$, we can divide it into four equally sized block matrices of size $n$ by $m$. For example, when $n = m = 2$, we could have

$$M = \begin{bmatrix}
  1 & 2 & 3 & 4 \\
  5 & 6 & 7 & 8 \\
  9 & 10 & 11 & 12 \\
  13 & 14 & 15 & 16
\end{bmatrix} = \begin{bmatrix}
  M_{11} & M_{12} \\
  M_{21} & M_{22}
\end{bmatrix}.$$

To multiply two matrices $AB$ of size $2^n$ by $2^n$ together ($n > 0$), the naive approach would result in

$$AB = \begin{bmatrix}
  A_{11} & A_{12} \\
  A_{21} & A_{22}
\end{bmatrix} \begin{bmatrix}
  B_{11} & B_{12} \\
  B_{21} & B_{22}
\end{bmatrix} = \begin{bmatrix}
  A_{11}B_{11} + A_{12}B_{21} & A_{11}B_{12} + A_{12}B_{22} \\
  A_{21}B_{11} + A_{22}B_{21} & A_{21}B_{12} + A_{22}B_{22}
\end{bmatrix}.$$
Algorithm 1: Strassen($A,B$)

Input: Two matrices $A$ and $B$ of size $2^n$ by $2^n$
Output: The product $C = AB$ of size $2^n$ by $2^n$

if $n = 0$ then
  return $AB$

$M_1 = \text{Strassen}(A_{11} + A_{22}, B_{11} + B_{22})$
$M_2 = \text{Strassen}(A_{21} + A_{22}, B_{11})$
$M_3 = \text{Strassen}(A_{11}, B_{12} - B_{22})$
$M_4 = \text{Strassen}(A_{22}, B_{21} - B_{11})$
$M_5 = \text{Strassen}(A_{11} + A_{12}, B_{22})$
$M_6 = \text{Strassen}(A_{21} - A_{11}, B_{11} + B_{12})$
$M_7 = \text{Strassen}(A_{12} - A_{22}, B_{21} + B_{22})$

return $\begin{bmatrix}
  M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\
  M_2 + M_4 & M_1 - M_2 + M_3 + M_6
\end{bmatrix}$

Problem 2. Prove that

$$AB = \begin{bmatrix}
  M_1 + M_4 - M_5 + M_7 & M_3 + M_5 \\
  M_2 + M_4 & M_1 - M_2 + M_3 + M_6
\end{bmatrix},$$

where each $M_i$ is as described in the Strassen algorithm (assuming the recursive calls correctly return the product of their inputs).

Problem 3. Prove that the Strassen algorithm is correct.

Problem 4. Give a recurrence relation $T(m)$ for Strassen (where $m$ is the width/height of the matrix) and determine the big-oh runtime of the algorithm (your bound should be as good as possible). Show your work for the latter.

Recall that the ceiling of a number $r$, written $\lceil r \rceil$, is the unique integer $z$ such that $r \leq z < r + 1$. For the next two problems, consider the following algorithm.

Algorithm 2: CLog($n$)

Input: $n \in \mathbb{Z}^+$
Output: $\log_2 n$
Let $\log = 0$
Let $i = 1$
while $i < n$ do
  $\log = \log + 1$
  $i = 2 \cdot i$
return $\log$

Problem 5. Give a (useful) loop invariant for CLog and prove that it is a loop invariant. Do not prove that CLog is correct.

Problem 6. Prove that CLog is correct.
For the next three problems, consider the following.

*Arbitrage* is the process of buy and selling goods to take advantage of market inefficiencies to make money. For example, if you can buy $1 for ¥148 and then buy ¥149 for $1, you can make ¥1 by buying a dollar and then selling it for yen.

Let $C$ be a collection of $n \geq 2$ currencies, and let $w : C \times C \to \mathbb{Q}^+$ represent the exchange rates between them. In the example above,

$$w(\$, ¥) = \frac{148}{149} \quad w(¥, \$) = \frac{1}{148}.$$

**Problem 7.** In terms of the variables given to you above, describe when arbitrage is possible.

**Problem 8.** In general, multiplication is a harder problem than addition. Transform your arbitrage condition from the previous problem from being about a product into being about a sum.

**Problem 9.** Give a polynomial time (pseudocode) algorithm to detect when arbitrage is possible (your algorithm is given $C$ and $w$). You do not need to come up with the exchange cycle, only detect its existence.

---

**Problem 10.** Let $G = (V, E)$ be a connected graph, and let $T$ be a minimum spanning tree of $G$. Prove that the shortest path between any pair of vertices of $V$ does not necessarily use only the edges in $T$.

---

For the next three problems, consider the following.

Let $G = (V, E)$ be a connected graph, and let $w : E \to \mathbb{Q}$ be the weights of its edges. The *bottleneck value* of a spanning tree $T = (V_T, E_T)$ of $G$, written $b(T)$, is the maximum edge weight of $E_T$, i.e.

$$b(T) = \max_{e \in E_T} w(e).$$

A *minimum bottleneck spanning tree* of $G$ is a spanning tree $T^B$ that minimizes $b(T^B)$.

In other words, let $\mathcal{T}$ be the set of all spanning trees of $G$. Then $T^B$ is a minimum bottleneck spanning tree of $G$ if for every $T \in \mathcal{T}$,

$$b(T^B) \leq b(T).$$
**Problem 11.** Let $G = (V, E)$ be a connected graph, $w : E \to \mathbb{Q}$ be the weights of $G$’s edges, and $b \in \mathbb{Q}$. Give a polynomial time (pseudocode) algorithm that, given $G$, $w$, and $b$, determines if there is a bottleneck spanning tree $T$ of $G$ with $b(T) \leq b$.

**Problem 12.** Let $G = (V, E)$ be a connected graph, and let $w : E \to \mathbb{Q}$ be the weights of $G$’s edges. Give a polynomial time (pseudocode) algorithm that, given $G$ and $w$, determines the bottleneck value of a minimum bottleneck spanning tree of $G$. 