1. **Asymptotic complexity (20 points)**
Consider the multiplication of two \( n \times n \) matrices \( C = A \cdot B \). We may view each matrix as being composed of four \( \frac{n}{2} \times \frac{n}{2} \) submatrices, as shown below:

\[
\begin{pmatrix}
C_{11} & C_{12} \\
C_{21} & C_{22}
\end{pmatrix} = 
\begin{pmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{pmatrix} \cdot 
\begin{pmatrix}
B_{11} & B_{12} \\
B_{21} & B_{22}
\end{pmatrix}
\]

(a) (5 points) The matrix multiplication may be expressed recursively (i.e. in terms of submatrix products) as shown on the right.

Express the asymptotic complexity of this variant of matrix multiplication as a recurrence. Assume that the cost of adding two matrices of \( n \times n \) size is \( \Theta(n^2) \).

(b) (5 points)
Express the asymptotic complexity of this alternate variant of matrix multiplication as a recurrence.

(c) (10 points) Provide asymptotic tight bounds for the two recurrences using the Master Method.

2. **Graph Algorithms (25 points)**
Activity ordering: You are given a set of activities \( A = \{a_1, a_2, a_3, \ldots, a_n\} \) and a set of prerequisites activities \( P = \{(a_i, a_j) \mid 1 \leq i, j \leq n\} \) that define the dependence of activity \( a_j \) on activity \( a_i \). A feasible activity order is defined as some permutation of all activities in \( A \) such that if \( a_i \) occurs before \( a_j \) in that permutation, then \( (a_j, a_i) \notin P \).

Describe an algorithm (in plain English, NOT pseudocode) to (a) decide whether there exists any feasible activity order. (b) emit the feasible activity order if it is decided in part (a) that such a feasible order exists.

3. **Dynamic Programming/Greedy Algorithms (20 points)**
The fractional knapsack problem can be stated as follows: There is a knapsack of capacity (in terms of weight) \( W \) and there are \( n \) items where each item \( i \) \((0 < i <= n)\) has weight \( w_i \) and value \( v_i \). The goal is to select items (potentially including fractions of items) to place in the knapsack such that (a) the sum of weights in the knapsack does not exceed \( W \) and (b) the value of the knapsack’s contents is maximized.

Prove that the greedy choice of picking items with the highest value per unit weight is optimal.
4. NP completeness (35 points)
TILE-FIT (D,W,H) is a decision problem that has as inputs (a) the height (H) and width(W) of a
tray and a set of positive integers D = \{d_1, d_2, ... d_n\} specifying dimensions of n tiles as 1xd_1,
1xd_2, ... 1xd_n. The decision problem returns a YES if and only if ALL the tiles may be placed in
the tray with horizontal orientation such that (a) the placement must be entirely contained within
the tray and (b) there must be no overlapping of tiles. Note, the tiles need not fill the entire area
of the tray. One instance of the TILE-FIT problem is illustrated below. The illustration is NOT to
scale. Because the tiles specified by D (as shown on the left) can fit within the tray boundary (as
shown on the right), this problem instance would return a YES. The shaded area is the unfilled
area in the tray.

1. (10 points) Prove that TILE-FIT is in NP.
2. (25 points) Prove that TILE-FIT is NP-complete. (Hint: Partition(S) is a known NP-
Complete decision problem. It accepts a finite set S of positive integers and returns YES
iff the set can be partitioned into two non-intersecting sets A and S-A whose sums are
equal.)