Question 1 [12 marks] Assuming that $T(n) \in \Theta(1)$ for all $n \leq n_0$ for a suitable constant n_0, solve each of the following recurrences to obtain an asymptotic bound on their complexity as a closed form. Make your bounds as tight as possible.

a. [4 marks] $T(n) = 6T(n/4) + n^2$

b. [4 marks] $T(n) = 10T(n/3) + n^2$

c. [4 marks] $T(n) = T(n/2) + \log_2 n$

Question 2 [25 marks] A rainwater drainage system is described by a weighted, directed acyclic graph $G = (V, E, w)$, where

- the directed edges, $G.E$, represent one-directional pipes in the drainage system,
- the vertices, $G.V$, are junctions in the system where pipes connect, and
- for each directed edge in the graph, $(u, v) \in G.E$, the weight function $G.w(u, v)$ describes the fraction of water flowing into junction u that then flows down pipe (u, v) into v.

Vertices with no incoming edges are referred to as inlets, and vertices with no outgoing edges are referred to as outlets. The rainwater drainage system can have multiple inlets and outlets.

Since weights in the graph represent fractions, they must be non-negative. For each vertex u that is not an outlet we also require the fractions on its outgoing edges to sum to one.

If the rainwater drainage system is initially empty and there is a deluge of rain that causes, for each inlet v_i, $x(v_i)$ liters of water to flow into v_i, we can use the drainage system graph to calculate, for any outlet v_o, the total amount of deluge water that will flow into v_o.

For example, for graph G with vertices $G.V = \{v_1, v_2, v_3, v_4\}$ and edges $G.E = \{(v_1, v_2), (v_1, v_3), (v_4, v_2)\}$ and weight function $G.w = \{(v_1, v_2) \mapsto 0.4, (v_1, v_3) \mapsto 0.6, (v_4, v_2) \mapsto 1\}$, if $x(v_1) = 10$ liters of water flows into inlet v_1, and $x(v_4) = 5$ liters of water flows into inlet v_4, then in total 9 liters of deluge water will flow into the outlet v_2, and 6 liters of deluge water will flow into the outlet v_3.

Let G be a rainwater drainage system graph, and x be a mapping from the inlets of G to the number of liters of water that flows into each of those inlets from the deluge.

a. [5 marks] What is a topological sort of a directed acyclic graph, and how can you efficiently compute one?

b. [5 marks] Define a recurrence $M(G, x, v)$ that describes, for any arbitrary vertex v of G, the total number of liters of deluge water that will flow into vertex v. Be sure to define the base cases of the recurrence as well as the more general cases.

c. [10 marks] Given any outlet $v_o \in G$, write an efficient algorithm MAXIMUM-FLOW(G, x, v_o) in pseudocode, that calculates $M(G, x, v_o)$. You may assume the existence of an efficient algorithm that can perform a topological sort of your graph.

d. [5 marks] What is the time complexity of your algorithm? State any assumptions that you make and briefly justify your answer.

Question 3 [20 marks]

This question involves performing an amortised analysis of a data structure.

Consider a data structure that keeps track of the inventory in a warehouse that has two operations, BUY(s) that adds one item of stock s with price $s.key$ to the inventory, and SELL(m) that removes the cheapest m items of stock in the inventory and returns the cost of purchasing those m cheapest items. The concrete implementation below uses a min-priority queue, called $inventory$, that stores one element for each item of stock in the inventory. The key associated with each element of stock, s, in the inventory is its price $s.key$. The min-priority queue is implemented using a binary heap.

```
BUY(s)
  inventory.INSERT(s)
```

```
SELL(m)
  cost = 0
  while m > 0 AND NOT inventory.IS-EMPTY()
        s = inventory.EXTRACT-MIN()
        cost = cost + s.key
        m = m - 1
  return cost
```

It is not straightforward using *aggregate analysis* to show that any sequence of m operations, starting from an empty inventory, has amortised cost $O(m \log m)$. Instead show how one could apply the *accounting method* and *potential method*, by answering the following questions:

a. [4 marks] What is the *actual cost* of the BUY and SELL operations? Answer in terms of the size, N, of the inventory, letting the actual cost of the priority-queue INSERT operation be $\lfloor \log_2(N+1) \rfloor + 1$, EXTRACT-MIN be $\lfloor \log_2 N \rfloor + 1$, and IS-EMPTY be 1.

b. [8 marks] Give the *amortised cost* for BUY and SELL for use in the *accounting method*. Argue that the total amortised costs minus the total actual costs is never negative for any sequence of m operations.

c. [8 marks] Give a *potential function* Φ for the warehouse data structure implementation, as would be used in analysing complexity with the *potential method*, and show that the value of the potential function after any sequence of m operations, starting from an empty inventory, is bound below by the initial value of the potential function. Calculate the amortised cost of operations BUY and SELL from Φ (show your working).

Question 4 [25 marks total]

A sequence Z of k characters $\langle z_1, z_2, \cdots, z_k \rangle$ is a palindrome if $z_i = z_{k-i+1}$ for all i in $1, 2, \cdots \lfloor k/2 \rfloor$.

Given a sequence $X = \langle x_1, x_2, \cdots, x_m \rangle$, we have that $Z = \langle z_1, z_2, \cdots, z_k \rangle$ is a subsequence of X if there exists a strictly increasing sequence $\langle i_1, i_2, \cdots, i_k \rangle$ of indices of X such that for all $j = 1, 2, \cdots, k$, we have that $x_{i_j} = z_j$.

Given a sequence $X = \langle x_1, x_2, \cdots, x_m \rangle$, the problem is to design an efficient algorithm for finding a longest subsequence of X that is also a palindrome. For example, the longest subsequence of the sequence $\langle p, e, n, e, l, o, p, e \rangle$ that is also a palindrome is $\langle p, e, e, p \rangle$.

a. [15 marks] This problem can be solved by dynamic programming. Let $A(i, j)$ be the length of the longest subsequence of $\langle x_i, x_{i+1}, \cdots, x_j \rangle$ that is also a palindrome. The solution we seek is $A(1, m)$. Give a recurrence defining $A(i, j)$ for $1 \leq i \leq m$ and $0 \leq j \leq m$.
You do NOT have to give a dynamic programming solution, just the recurrence.
Be sure to define the base cases of the recurrence as well as the more general cases.

b. [10 marks] For the dynamic programming solution indicate in what order the elements of the matrix A corresponding to the recurrence should be calculated. As part of answering this question you could either give pseudocode indicating the evaluation order or draw a table and indicate the dependencies of typical elements and which elements have no dependencies.

Question 5 [18 marks total]

Below we describe two computer science problems.

The subset-sum problem Given a set A of n distinct positive integers and an integer C, the subset-sum decision problem is to decide if there is a subset B of A such that $\sum_{b \in B} b = C$. This problem is known to be NP-complete (assuming a standard encoding of inputs).

The multiple-partition problem Given a set A of n distinct positive integers and m integers $C_1, C_2 \cdots C_m$, the multiple-partition decision problem is to decide if A can be partitioned into m disjoint subsets $B_1, B_2, \cdots B_m$ of A such that for each i in $1, 2, \cdots, m$, $\sum_{b:B_i} b = C_i$.

a. [8 marks] Prove that the multiple-partition problem is NP-hard.

b. [6 marks] Show that the multiple-partition problem is NP-complete and clearly state any assumptions that you make.

c. [4 marks] Explain the significance of the existence of NP-complete problems.