Problem 1

This problem is **NP-complete** problem.

1. The Problem is in NP

We construct a verifier checks if a given scheduling of jobs satisfies all the constraints and ensures that all jobs are completed by the deadline D.

Algorithm 1 Verifier for the Heterogeneous Multiprocessor Scheduling Problem

procedure VERIFIER($DAG, Scheduling, m, D$)
    Initialize $currentTime[1..m]$ to 0 ▷ Array to keep track of current time for each processor

    for each job v in $Scheduling$ (in order) **do**
        if there exists an edge (u, v) in DAG **then**
            if $currentTime[\pi(u)] + e(u) > currentTime[\pi(v)]$ **then return** False ▷ Job v starts before job u completes
            end if
        end if
        $currentTime[\pi(v)] \mathrel{+}= e(v)$ ▷ Update the completion time for the job on its processor
        if $currentTime[\pi(v)] > D$ **then return** False ▷ Job v does not complete before deadline D
        end if
    end for
    return True ▷ The scheduling is valid
end procedure

Here: - DAG represents the directed acyclic graph of jobs.
- $Scheduling$ is the proposed order in which jobs are scheduled.
- m is the number of processors.
- D is the deadline.
- $e(v)$ denotes the execution time of job v.
- $\pi(v)$ denotes the processor assignment for job v.

2. The Problem is NP-hard

We will use JOB-SHOP-SCHEDULING to prove this(Wiki-for JSSP problem)

Given an instance of JSSP with a set of jobs J and a set of machines M, each job j has a sequence of operations O_j which are processed on specific machines in a specific order. The aim is to find a schedule to minimize the makespan.

We transform this JSSP instance into an HMSP instance as follows:

1. For each operation o in J, create a vertex v in the DAG.

2. Set $e(v)$ equal to the processing time of o.

3. Set $\pi(v)$ equal to the machine processing o.

4. For operations o_1 and o_2 in job j such that o_1 precedes o_2, create a directed edge from the vertex representing o_1 to the vertex representing o_2.

The deadline D for HMSP will be the makespan of the JSSP solution. A solution to this HMSP instance corresponds to a solution for our JSSP instance.

Problem 2

This problem is **co-NP-Hard problem**

Definition of Related NP Problem: "Subset Sum Problem" Given a set of integers S and an integer t, does there exist a subset of S whose sum equals t?

Algorithm 2 Reduction from Subset Sum Problem to Subset No-Sum Problem

procedure SSP_TO_SNSP(S, t)
    $A \leftarrow S$
    $B \leftarrow \emptyset$
    for each s in S **do**
        $B \leftarrow B \cup \{2s\}$
    end for
    $T \leftarrow 2t$
    return A, B, T
end procedure

Correctness of the Reduction

If there exists a subset $A' \subset A$ such that the sum of its elements is t, then there must be a subset $B' \subset B$ whose elements sum to $T - t$. This is because B is constructed by doubling every element of S, ensuring such a subset exists in B for every possible subset of A.

Conversely, if there exists a subset $A' \subset A$ and a subset $B' \subset B$ such that $\sum_{a \in A'} a + \sum_{b \in B'} b = T$, then there exists a subset in S that sums to t.

Since the "Subset Sum Problem" is NP-complete, this polynomial-time reduction shows that its complement, the "Subset No-Sum Problem", is co-NP-hard.