# Assignment - 3

## Posel and Lattices

Submitted By:
Prashant Bhattarai
2315002589 (49)
Section: 2G2

## Question No.1

To be a poset, a set with a relation must satisfy the following properties:-

1. Reflexivity : Every element is related to itself. For any integer a, we have, $a \leq a$.
   This property holds.

2. Antisymmetry : If $a \leq b$ and $b \leq a$, then $a = b$.
   For integers, if $a \leq b$ and $b \leq a$, then a and b must be the same integer. Thus, this property holds.

3. Transitivity : If $a \leq b$ and $b \leq c$, then $a \leq c$. This is true for integers ~~between~~ because if a, is less than or equal to b, and b is less than or equal to c, then a must be less than or equal to c.
   So, This property holds.

Since the relation satisfies reflexibity, antisymmetric, and transitivity the set of integers under the "less than or equal to" relation forms a poset.

## Question No.2

"Task A can be started before task B."
Let's check the properties of a partial order in this project management context:

1. Reflexibility: For any task A, it is trivially true that A can be

started before or at the same time as itself. Thus $A \le A$ holds, the relation is reflexive.

2. Antisymmetry : If $A \le B$ (task A can be started before task B) and $B \le A$ (task B can be started before task A), then A and B must be the same task. In other words, if two tasks can start before each other, they must be identical. This property holds, so the relation is antisymmetric.

3. Transitivity : If task $A \le B$ (task A can be started before task B) and task $B \le C$ (task B can be started before task C), then task $A \le C$ (task A can be started before task C).

This proves in the context of task dependencies in project management, where if one task precedes a second, and the second precedes a third, the first must precede the third. The relation is transitive.

Since the relation " task A can be started before task B" statisfies reflexivity, antisymmetry, and transitivity, the set of tasks under this relation forms a poset.

## Question No 3

1. Meet (Greatest Lower Bound, Intersection)

- for any two subsets A and B, their meet is the largest subset that is a subset of both A and B. There is $A \cap B$.

- The intersection of two subsets is always a subset of both, and if any subset is a subset of both A and B, it must also be a subset of $A \cap B$. Thus, the intersection is the greatest lower bound.

2. Join (Least Upper Bound, Union):

- For any two subsets A and B, their join is the smallest subset that contains both A and B. This is simply the union of A and B, denoted A ∪ B.
- The union of two subsets contains both A and B, and if any subset contains both, it must contain A ∪ B. Thus, the union is the least upper bound.

∴ The set of all subsets of a given set S, with the subset relation ⊆, forms a lattice because both the meet and join exist for any pair of subsets.

---

Question No. 4

The hierarchy of job positions in a company forms a partially ordered set (poset), not a totally ordered set.

It is because:

- Partially Ordered Set (Poset): In a poset, some pairs of elements are comparable (i.e., one is "greater" than the other), but not every pair has to be comparable. This means that for some pairs of elements, there is no defined order between them.

- Totally Ordered Set: In a totally ordered set, every pair of elements is comparable, meaning for any two elements A and B, either $A \leq B$ or $B \leq A$ holds.

Since some job positions can be compared in terms of hierarchy, but others may not be directly comparable, the hierarchy of job positions in a company forms a partially ordered set (poset).
It is not totally ordered because not every pair of positions is comparable.

## Question No 5

Yes, the set $\{0,1\}$ with the operations AND (denoted as $\wedge$), OR (denoted as $\vee$), and NOT forms a lattice in the context of Boolean Algebra.

1. Meet (Greatest Lower Bound, AND):
   - For any two elements $a, b \in \{0,1\}$, the meet is defined by $a \wedge b$.
   - $0 \wedge 0 = 0$, $0 \wedge 1 = 0$, $1 \wedge 0 = 0$, and $1 \wedge 1 = 1$.
   - This is the greatest lower bound because $a \wedge b$ gives the largest element that is less than or equal to both a and b.

2. Join
   - For any two elements $a, b \in \{0,1\}$, the join is defined as $a \vee b$.
   - $0 \vee 0 = 0$, $0 \vee 1 = 1$, $1 \vee 0 = 1$ and $1 \vee 1 = 1$.
   - This is the least upper bound because $a \vee b$ gives the smallest element that is greater than or equal to both a and b.

The set $\{0,1\}$ under the operations AND and OR satisfies the lattice properties because both the meet and join exist for any pair of elements. Therefore, the Boolean algebra $\{0,1\}$ forms a lattice.

## Question No 6:

To show that the relation R (where $xRy$ means "person x is older than person y") is not a partial ordering we need to check wheather R satisfies the three properties required for a partial order:

1. Reflexivity
   - A relation R is reflexive if, for every element x in the set, $xRx$ holds. In this case, $xRx$ would mean that every person is older than themselves, which is clearly false. R is not reflexive.

## 2. Antisymmetric

- A relation R is antisymmetric if, for all x and y, if xRy (i.e. x is older than y) and yRx (i.e. y is older than x), then x = y.

- In this case, if xRy holds, meaning x is older than y, then yRx cannot hold, because one person cannot be both older and younge than the other.

  ∴ Antisymmetry is not an issue here because it's logically impossible for both xRy and yRx to hold unless x and y are distin people.

## 3. Transitivity

- A relation R is transitive if, for all x, y, and z, whenever xRy (i.e., x is older than y) and yRz (i.e., y is older than z), it must follow than xRz (i.e., x is older than z).

- This property does hold in the context of age. If x is older than y and y is older than z, then x must be older than z. So, transitivity is satisfied.

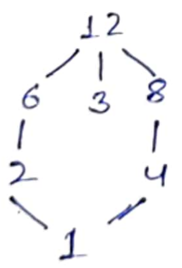
∴ The relation R fails to satisfy reflexivity.

∴ The lack of reflexivity means that R is not a partial ordering.

---

## Question No. 7.

Given set is {1, 2, 3, 4, 6, 8, 12}



- 1 divides all the numbers
- 2 divides 4, 6, and 12.
- 3 divides 6 and 12
- 4 divides 8.
- 6 divides 2.