Homework Answers
Ordered Sets in Data Analysis

Task1:

Since for each y in U there is x in U in other word: y = f(x).

let us define g: U → U, so g(y) = x (so it is surjective)

By condition y = f(x) if we put g(y) instead of x ➔ f(g(x)) = y then f is injective.

Some addition

Also, we add some thing by number of value f and g, g is surjective (by prove) and f is also surjective so the number of different value Ux = Uy = U (U is finite set) that means f is injective function.

Task2:

asymmetric and transitive ===> A relation R is a strict partial order.
Partial order is transitive and anti-symmetric and may or may not be reflexive or irreflexive.
It is not easy to calculate the number of relations, I found papers contain the number of relations (Partial order), the link is
https://dml.cz/bitstream/handle/10338.dmlcz/126183/MathBohem_122-1997-1_7.pdf
So approximately the number of strict partial order is ≤ number of partial order ≤ 4321

Task3:

Code in java:

- adjacency matrix

```
public int[] topological(int adjacency_matrix[][], int source)
        throws NullPointerException
   {
      int number_of_nodes = adjacency_matrix[source].length - 1;
      int[] topological_sort = new int[number_of_nodes + 1];
      int pos = 1;
      int j;
      int visited[] = new int[number_of_nodes + 1];
      int element = source;
      int i = source;
      visited[source] = 1;
      stack.push(source);
      while (!stack.isEmpty())
      {
         element = stack.peek();
         while (i <= number_of_nodes)
         {
            if (adjacency_matrix[element][i] == 1 && visited[i] == 1)
            {
               if (stack.contains(i))
               {
                  System.out.println("TOPOLOGICAL SORT NOT
POSSIBLE");
                  return null;
               }
            }
            if (adjacency_matrix[element][i] == 1 && visited[i] == 0)
            {
               stack.push(i);
               visited[i] = 1;
               element = i;
               i = 1;
               continue;
            }
            i++;
         }
         j = stack.pop();
```

```
            topological_sort[pos++] = j;
            i = ++j;
        }
            return topological_sort;
    }
```

- list of edges

```
        public void topologicalSort(){
        Stack stack = new Stack();
        // iterate through all the nodes and their neighbours if not already
visited.
        for (Character c : nodes){
            if(!nodeVisited.contains(c)){
                sort(c, stack);
            }
        }
        // print all the elements in the stack in reverse order
        while(!stack.empty()){
            System.out.print(stack.pop()+ " ");
        }
    }

    // this recursive method iterates through all the nodes and neighbours.
    // Pushes the visited items to stack
    public void sort(Character ch, Stack stack){
        // add the visited node to list, so we don't repeat this node again
        nodeVisited.add(ch);
        // the leaf nodes wouldn't have neighbors. A check added to avoid null
pointer
        if(edges.get(ch)!=null){
            // get all the neighbor nodes , by referring its edges
            Iterator iter = edges.get(ch).iterator();
            Character neighborNode;
            // if an edge exists for the node, then visit that neighbor node
            while(iter.hasNext()){
```

```
                neighborNode = iter.next();
                if(!nodeVisited.contains(neighborNode)){
                    sort(neighborNode,stack);
                }
            }
        }
        // push the latest node on to the stack
        stack.push(new Character(ch));
    }
```

Complexity for an adjacency matrix is O(|V|^2).

Complexity for a list of edges is O(|V| + |E|) it is the same (DFS algorithm),

Task4:

1. **reflexivity**

$$\begin{cases} x \leq x \\ y \leq y \end{cases} \Rightarrow (x, y)\ R\ (x, y)$$

2. **antisymmetricity**

$$\begin{cases} (x1, y1\ )\ R\ (x2, y2\ ) \\ (x2, y2\ )\ R\ (x1, y1\ ) \end{cases} \Rightarrow$$

$$\begin{cases} x1 \leq x2 \leq x1 \\ y1 \leq y2 \leq y1 \end{cases} \Rightarrow (x1, y1\ ) = (x2, y2\ )$$

3. **transitivity**

$$\begin{cases} (x1, y1\ )\ R\ (x2, y2\ ) \\ (x2, y2\ )\ R\ (x3, y3\ ) \end{cases} \Rightarrow$$

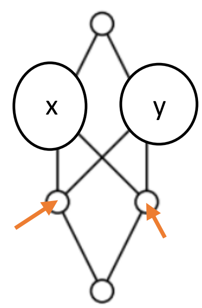
$$\begin{cases} x1 \leq x2 \\ x2 \leq x3 \end{cases}$$

$y1 \le y2$

$y2 \le y3 \quad \Rightarrow \quad \begin{cases} x1 \le x3 \\ y1 \le y3 \end{cases} \quad \Rightarrow \quad (x1, y1\ )\ R\ (x3, y3\ )$

For $A1$ the only maximal element is (3,4), As for the minimal element, there is no minimal element in this case.
For $A2 = \{(x, y) \in \mathbb{Z}^2 \mid x^2 + y^2 \le 4\} == \{(-2,0), (-1,-1), (-1,0), (-1,1), (0,-2), (0,-1), (0,0), (0,1), (0,2), (1,-1), (1,0), (1,1), (2,0)\}$ seen that maximal elements are $\{(0,2)\ , (1,1)\ , (2,0)\}$
and minimal elements are $\{(-2,0), (-1,-1), (0,-2)\}$.

Task5:

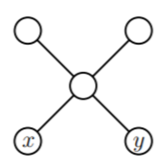


1. Does not represent a lattice, because, for instance, there exists no infimum for the elements {x, y}:

2. Is a lattice because each pair of elements has a unique supremum and infimum.

3. Is a lattice because each pair of elements has a unique supremum and infimum

4. Is a lattice because each pair of elements has a unique supremum and infimum



5. Does not represent a lattice, because, for instance, there exists no in mum for the elements {x, y}.

Since, 2,3,4 are finite lattice then they are complete lattices.

Task6:

1. By the definition, $x \vee x = \sup\{x\}$ ➔ $x \leq x \vee x$
   $(\forall d)(d \in L)(x \leq d) \Rightarrow (x \vee x \leq d)$➔ instead of d we can put x ➔ $x \vee x \leq x$.

2. Let us write $d = x \vee (x \wedge y)$. then $x \leq d$ and $x \wedge y \leq d$, but also $x \leq x$ and $x \wedge y \leq x$, so by the definition of the least upper bound $d \leq x$ ➔ $d = x$.

3. By definition, $x \vee y = d \in L$ such that $x \leq d$, $y \leq d$ and $\forall l \in L$ $x \leq l$, $y \leq l$ implies that $d \leq l$ which mean $y \vee x$, hence, two expressions are equal.

😊 😊 😊