

## Hands\_on\_14

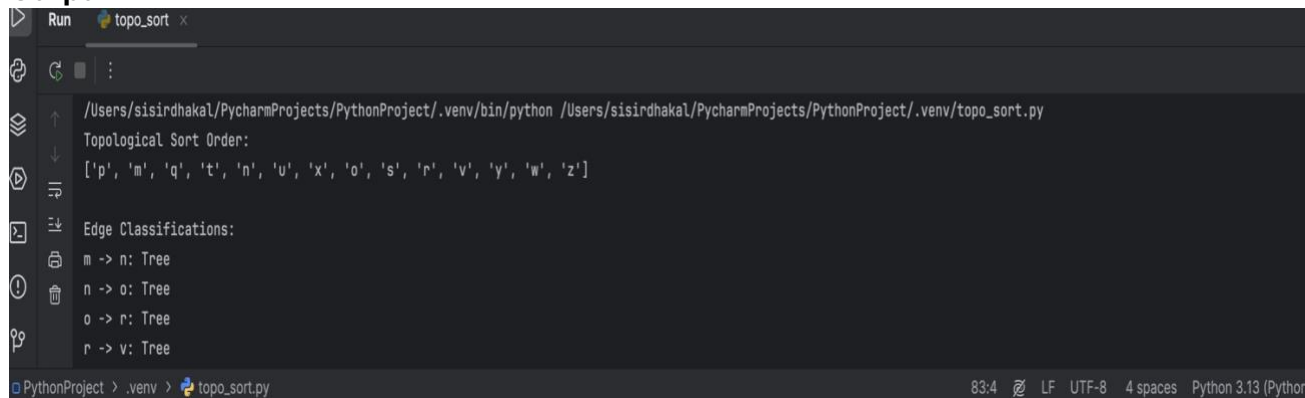
- ⇒ Implement Topological sort and test them on the examples from the book and upload your code and tests to Github.  
Show the ordering of vertices produced by TOPOLOGICAL-SORT when it is run on the dag of Figure 22.8 below, assume that the for loop of lines 5–7 of the DFS procedure considers the vertices in alphabetical order, and assume that each adjacency list is ordered alphabetically. Show the discovery and finishing times for each vertex and show the classification of each edge. implement code at python

⇒ Solution:

Here,

**Topological Order:** a valid linear ordering of the DAG.

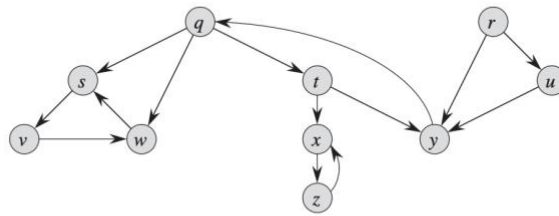
**Output:**



```
Run topo_sort x
/Users/sisirdhakal/PycharmProjects/PythonProject/.venv/bin/python /Users/sisirdhakal/PycharmProjects/PythonProject/.venv/topo_sort.py
Topological Sort Order:
['p', 'm', 'q', 't', 'n', 'u', 'x', 'o', 's', 'r', 'v', 'y', 'w', 'z']
Edge Classifications:
m -> n: Tree
n -> o: Tree
o -> r: Tree
r -> v: Tree
```

- 
- ⇒ Implement Depth-search and test them on the examples from the book.  
Show how depth-first search works on the graph of Figure 22.6. Assume that the for loop of lines 5–7 of the DFS procedure considers the vertices in alphabetical order and assume that each adjacency list is ordered alphabetically. Show the discovery and finishing times for each vertex and show the classification of each edge.

### 22.3 Depth-first search



**Figure 22.6** A directed graph for use in Exercises 22.3-2 and 22.5-2.

Solution:

Output

```
Run depth_search
/Users/sisirdhaka/PycharmProjects/PythonProject/.venv/bin/python /Users/sisirdhaka/PycharmProjects/PythonProject/.venv/depth_search.py
DFS Finish Order (Topological Postorder):
['z', 'x', 'y', 'q', 't', 's', 'v', 'w']

Edge Classifications:
q -> s: Tree
s -> v: Tree
v -> w: Tree
w -> s: Back
```

- ⇒ Implement Kruskal algorithm and test them on the examples from the book .  
Kruskal's algorithm can return different spanning trees for the same input graph  $G$ , depending on how it breaks ties when the edges are sorted into order. Show that for each minimum spanning tree  $T$  of  $G$ , there is a way to sort the edges of  $G$  in Kruskal's algorithm so that the algorithm returns  $T$  .  
⇒ Solution:

```
Run  kruskal x
/Users/sisirdhakal/PycharmProjects/PythonProject/.venv/bin/python /Users/sisirdhakal/PycharmProjects/PythonProject/.venv/kruskal.py
MST returned by Kruskal's Algorithm:
a - b (weight 1)
b - c (weight 1)
b - d (weight 2)
Process finished with exit code 0
```

PythonProject > .venv > kruskal.py 57:1 LF UTF-8 4 spaces Python 3.13 (PythonProject)