DESIGN ANALYSIS AND ALGORITHMS

HANDSON-13

RASHMITHA RAMASANI

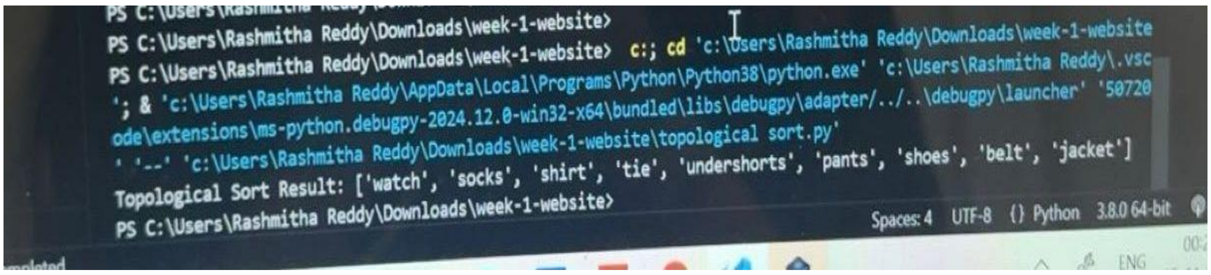
UTA ID:1002233393

1.TOPOLOGICAL SORT: Topological sort is an ordering of vertices in a directed acyclic graph (DAG) such that for every directed edge U→V Vertex U appears before V in the order. It is used for scheduling tasks with dependencies, ensuring prerequisite tasks are completed before dependent ones.

Key Characteristics:

1. **Directed Acyclic Graph (DAG)**: Topological sort only works on directed graphs without cycles. If a cycle is present, topological sorting is not possible.
2. **Order of Vertices**: The output is not unique; there can be multiple valid topological orderings for a given graph

Output:



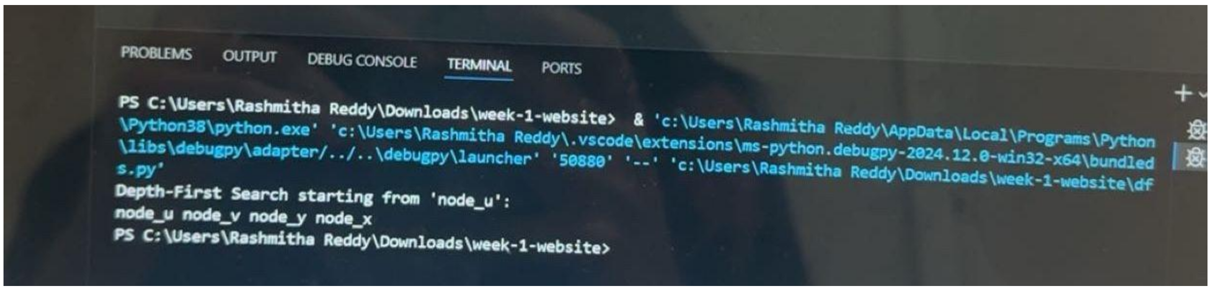
```
PS C:\Users\Rashmitha Reddy\Downloads\week-1-website>
PS C:\Users\Rashmitha Reddy\Downloads\week-1-website>  c:; cd 'c:\Users\Rashmitha Reddy\Downloads\week-1-website
PS C:\Users\Rashmitha Reddy\Downloads\week-1-website\ & 'c:\Users\Rashmitha Reddy\AppData\Local\Programs\Python\Python38\python.exe' 'c:\Users\Rashmitha Reddy\.vsc
'; & 'c:\Users\Rashmitha Reddy\AppData\Local\Programs\Python\Python38\python.exe' 'c:\Users\Rashmitha Reddy\.vsc
ode\extensions\ms-python.debugpy-2024.12.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '50720
' '--' 'c:\Users\Rashmitha Reddy\Downloads\week-1-website\topological sort.py'
Topological Sort Result: ['watch', 'socks', 'shirt', 'tie', 'undershorts', 'pants', 'shoes', 'belt', 'jacket']
PS C:\Users\Rashmitha Reddy\Downloads\week-1-website>
```

Spaces: 4 UTF-8 {} Python 3.8.0 64-bit

2.DEPTH FIRST SEARCH:

Depth First Search (DFS) is an algorithm used for traversing or searching tree or graph data structures. It explores as far as possible along each branch before backtracking. DFS uses a **stack** data structure, either explicitly (with a stack) or implicitly (with recursion), to keep track of the vertices to be explored.

Key Points:

- **Traversal Approach**: DFS starts at the root node (or an arbitrary node) and explores each branch as deep as possible before moving to the next branch.
- **Visited Tracking**: To avoid processing nodes more than once, DFS marks each node as visited.
- **Recursive or Iterative**: DFS can be implemented using recursion (natural for depth-first exploration) or iteratively using a stack.
- **OUTPUT**:
- 

```
PS C:\Users\Rashmitha Reddy\Downloads\week-1-website>  & 'c:\Users\Rashmitha Reddy\AppData\Local\Programs\Python
\Python38\python.exe' 'c:\Users\Rashmitha Reddy\.vscode\extensions\ms-python.debugpy-2024.12.0-win32-x64\bundled
\libs\debugpy\adapter/../..\debugpy\launcher' '50880' '--' 'c:\Users\Rashmitha Reddy\Downloads\week-1-website\df
s.py'
Depth-First Search starting from 'node_u':
node_u node_v node_y node_x
PS C:\Users\Rashmitha Reddy\Downloads\week-1-website>
```

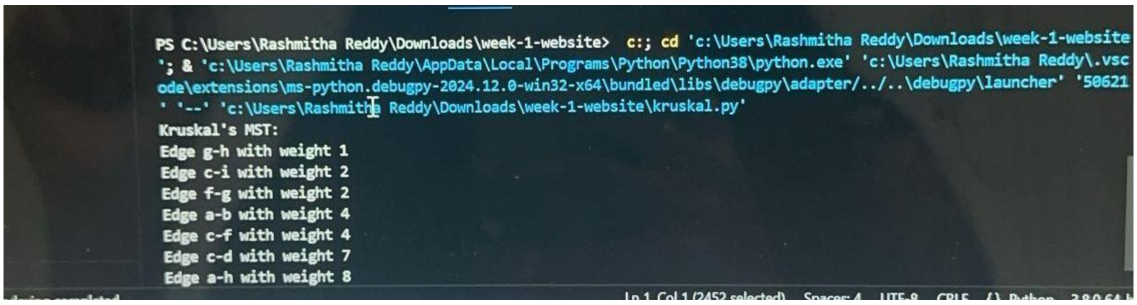
3.KRUSKAL ALGORITHM

Kruskal's Algorithm is a greedy algorithm used to find the Minimum Spanning Tree (MST) of a weighted, undirected graph. The MST is a subset of the edges that connects all vertices together, without any cycles and with the minimum possible total edge weight.

Key Concepts:

- **Greedy Approach:** Kruskal’s algorithm always selects the edge with the lowest weight that doesn’t form a cycle.
- **Spanning Tree:** A tree that includes all the vertices of the graph with the minimum number of edges.
- **Cycle Detection:** The algorithm ensures no cycles are formed while building the MST.

OUTPUT:



```
PS C:\Users\Rashmitha Reddy\Downloads\week-1-website>  c:; cd 'c:\Users\Rashmitha Reddy\Downloads\week-1-website
'; & 'c:\Users\Rashmitha Reddy\AppData\Local\Programs\Python\Python38\python.exe' 'c:\Users\Rashmitha Reddy\.vsc
ode\extensions\ms-python.debugpy-2024.12.0-win32-x64\bundled\libs\debugpy\adapter/../..\debugpy\launcher' '50621
' '--' 'c:\Users\Rashmitha Reddy\Downloads\week-1-website\kruskal.py'
Kruskal's MST:
Edge g-h with weight 1
Edge c-i with weight 2
Edge f-g with weight 2
Edge a-b with weight 4
Edge c-f with weight 4
Edge c-d with weight 7
Edge a-h with weight 8
```