from collections import defaultdict

# Selection Sort Function

def selectionSort(arr):

for i in range(len(arr)):

min\_idx = i

for j in range(i + 1, len(arr)):

if arr[j] < arr[min\_idx]:

min\_idx = j

arr[i], arr[min\_idx] = arr[min\_idx], arr[i]

return arr

# Disjoint Set for Kruskal's Algorithm

class DisjointSet: #avoid cycles

def \_\_init\_\_(self, n): # Initializes each node to be its own parent (disconnected components).

self.parent = list(range(n))

self.rank = [0] \* n

def find(self, x): # Returns the root parent of a node (with path compression).

if self.parent[x] != x:

self.parent[x] = self.find(self.parent[x]) # Path compression

return self.parent[x]

def union(self, x, y): # Merges two subsets if they are not already connected.

rootX = self.find(x)

rootY = self.find(y)

if rootX != rootY:

if self.rank[rootX] > self.rank[rootY]: #**Rank** is an estimate of the **height** of the tree.

self.parent[rootY] = rootX

elif self.rank[rootX] < self.rank[rootY]:

self.parent[rootX] = rootY

else:

self.parent[rootY] = rootX

self.rank[rootX] += 1

# Kruskal's Algorithm Function

def kruskal(n, edges):

ds = DisjointSet(n)

mst = []

total\_weight = 0

edges.sort(key=lambda x: x[2]) #func extracts the wt from each edge for sorting

for u, v, weight in edges:

if ds.find(u) != ds.find(v):

ds.union(u, v)

mst.append((u, v, weight))

total\_weight += weight

return mst, total\_weight

# -------- Main Program Loop --------

print("Name: Prachi Karande")

print("Roll no.: TACO22134")

while True:

print("\nChoose the algorithm to run:")

print("1. Selection Sort")

print("2. Kruskal's Algorithm (MST)")

print("0. Exit")

choice = input("Enter your choice (0, 1 or 2): ")

if choice == '1':

arr = list(map(int, input("Enter the array elements separated by space: ").split()))

sorted\_arr = selectionSort(arr)

print("Sorted Array:", sorted\_arr)

print("\nHeuristic Function Used:")

print("In Selection Sort, the heuristic function is based on comparison.")

print("At each iteration, it selects the minimum from the unsorted part and swaps it.")

print("f(node) = min(arr[i], arr[j]) for all j in range(i+1, n)")

elif choice == '2':

n = int(input("Enter the number of vertices: "))

e = int(input("Enter the number of edges: "))

edges = []

print("Enter edges in the format: u v weight")

for \_ in range(e):

u, v, weight = map(int, input().split())

edges.append((u, v, weight))

mst, total\_weight = kruskal(n, edges)

print("Minimum Spanning Tree:", mst)

print(f"Total cost of the Minimum Spanning Tree: {total\_weight}")

print("\nExpression for Edge Selection (Heuristic Function used):")

print("f(u, v) = w(u, v) if find(u) != find(v)")

elif choice == '0':

print("Exiting program. Goodbye!")

break

else:

print("Invalid choice. Please enter 0, 1, or 2.")