**5.11 Disjoint Set Union (DSU)**

**Aim:** The program uses a Disjoint Set Union (DSU) or Union-Find data structure to efficiently detect cycles.

**Algorithm:**

1. We are given an undirected graph G = (V, E) with vertices V = {1, 2, …, n} and edges E.
2. The aim is to detect whether the graph contains a cycle using the Disjoint Set Union (DSU) data structure.
3. Initialize DSU: Each vertex is its own parent: parent[v] = v.
4. Rank/size array may also be maintained for optimization.
5. Define two operations:

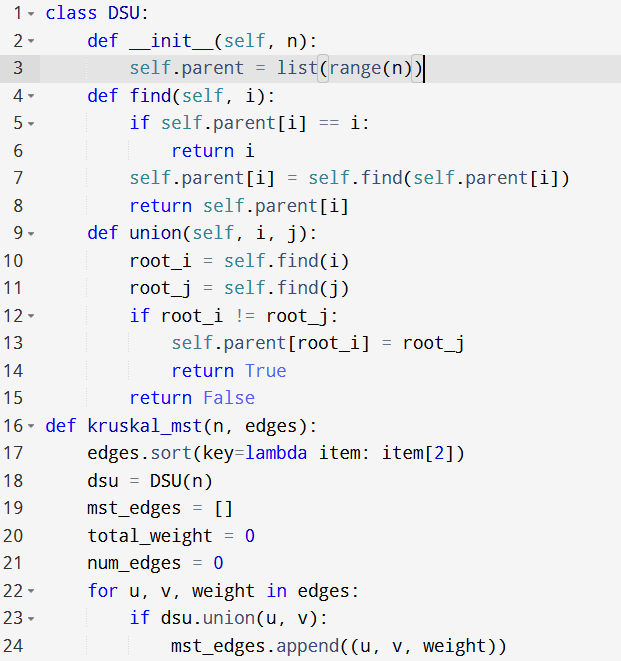
* Find(v): Returns the root representative of set containing v. Uses path compression for efficiency.
* Union (u, v): Merges the sets containing u and v. Uses union by rank/size to keep tree shallow.

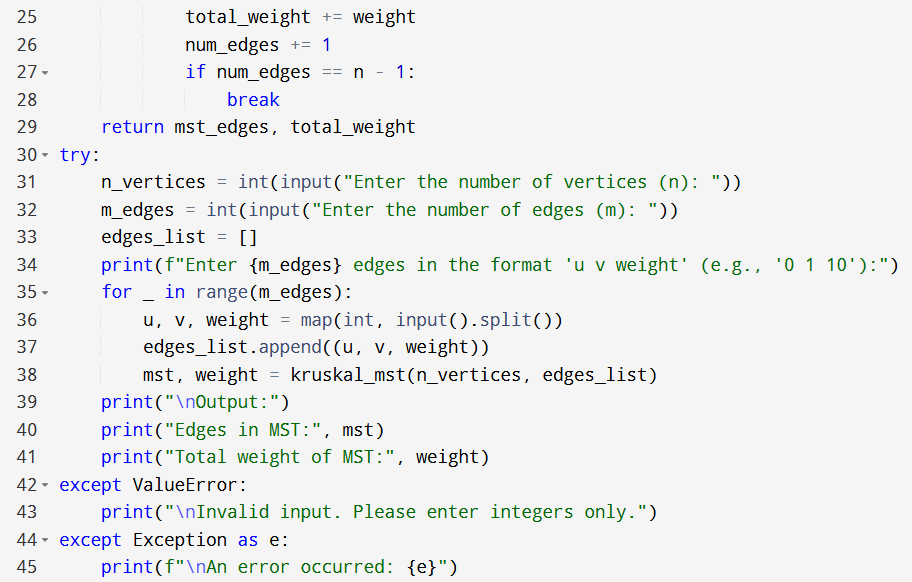
1. For each edge (u, v) in E:

* Find the roots: rootU = Find(u), rootV = Find(v).
* If rootU == rootV,  
   then u and v are already in the same set → Cycle detected.

1. Else, perform Union (rootU, rootV) to merge the sets.
2. If no such case is found after processing all edges,  
   then the graph has no cycles.

**Program:**

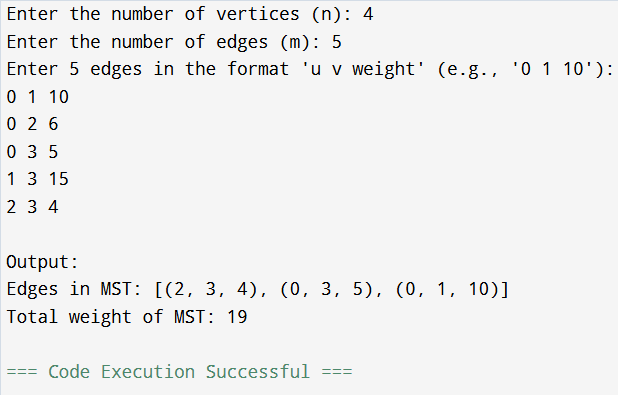
****



**Input:**

* Enter the number of vertices (n):
* Enter the number of edges (m):
* Enter 5 edges in the format 'u v weight' (e.g., '0 1 10'):

**Output:**

****

**Result:** Thus, the program is executed successfully and output is verified.

**Performance analysis:**

* Time Complexity: O(eloge) or O(elogv)
* Space Complexity: O(v+e).