# Union-Find

- Application in Kruskal's Algorithm
- Optimizing Union and Find Methods

# Minimum Spanning Trees

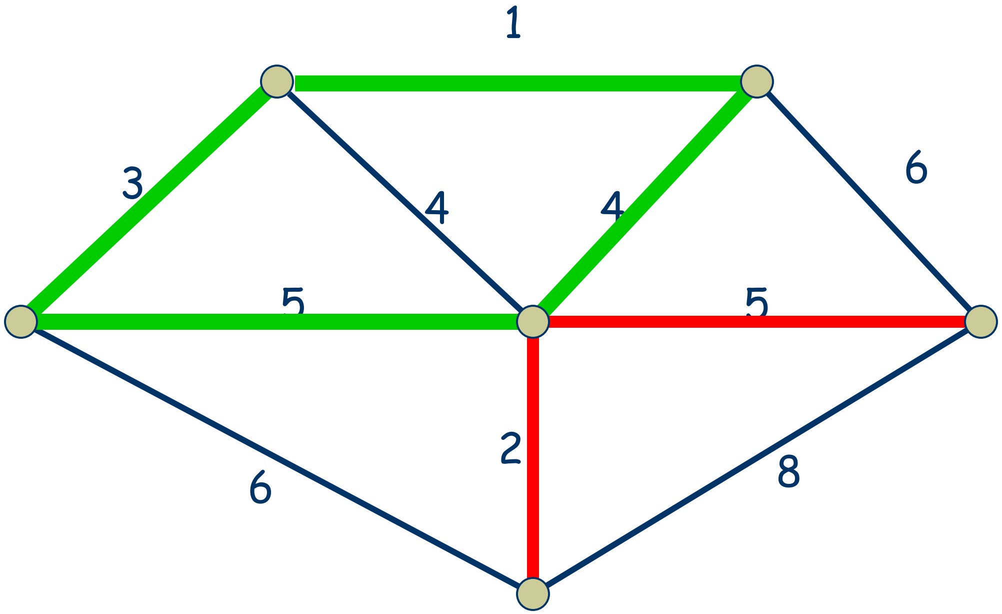
- Tree that connects all vertices of a graph
- Sum of the edge weights is a minimum

# Kruskal's Algorithm

- Sort edges in order of weights

- Start adding edges to sub-graph:
    - Start from lowest weight
    - Skip edge if it makes the sub-graph cyclic

# Kruskal's Algorithm



# Union-Find & Kruskal's Algorithm

- Vertices grouped in sets
- Can only add edges linking vertices not in same set

# Non-Optimal Solution

- Array of labels
- Change labels for a union
- O (n) for each union
- O (n^2)

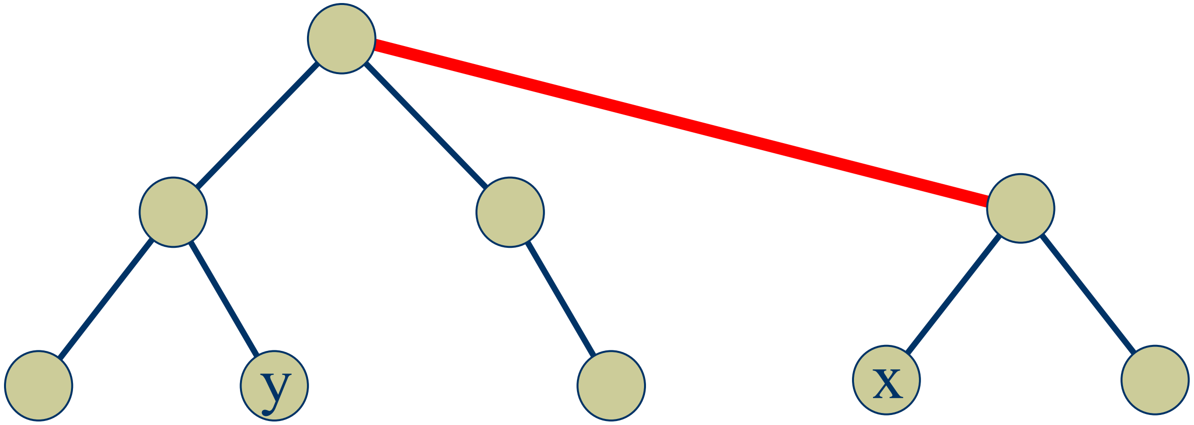
# Union-Find Methods

- makeSet ( x )
- union ( x , y )
- find ( x )

# Optimizing Union(x,y)

- Sets of vertices stored in trees
- Root of tree is label of set
- union(x,y) by joining two trees
- Root of smaller tree points to root of larger tree

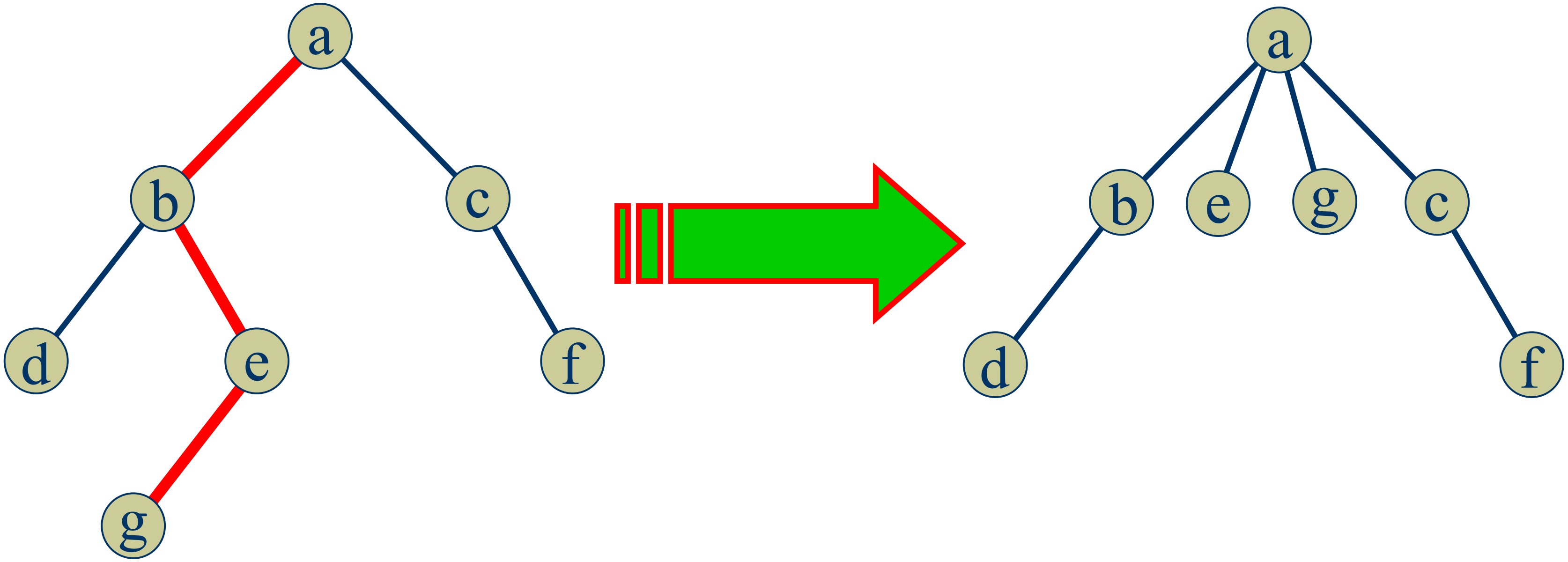
# Union(x,y) Illustration



# Path Compression

- Nodes from ‘x’ to root have same label
- Change these parent-pointers to the root

# Path Compression Illustration



# Time Efficiency

- Sorting is O(e log e)
- Find maximum is O(log n)
- Path compression makes future finds O(1)
- Calling find many times gives O(1) average
- Union is 2 finds and a pointer change: O(1)
- Kruskal becomes O(e log e)