Kruskal's Algorithm

Problem : Finding a minimum spanning tree.

↳ A tree with all n vertices of the graph $G(V, E)$ and exactly $(n-1)$ edges.

Given : A weighted undirected connected graph.

Idea :

1. Sort the edges of the graph in increasing order of their weights.
2. Start with a graph with vertices only and each vertex "i" corresponds to a component "i" of the graph.
3. Iteratively add edges in the graph to combine components.

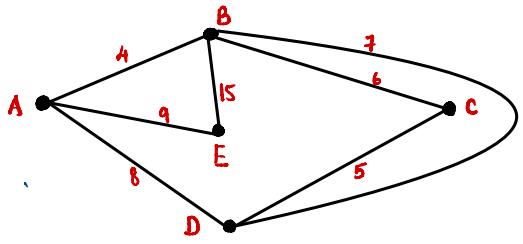
```
kruskalAlgorithm ( )
    Given weighted undirected graph G(V, E)
    sortedE = [ e1, e2, e3, ..., em ] be a sorted (increasing) list of edges.
                                      (order)
    Here, Weight on edge (ei) ≤ Weight on edge ej ; if i < j
    STE ← ∅ (Spanning Tree Edges)
    i ← 1
    while size of (STE) < (n-1)        ← No. of edges in a Spanning tree with n vertices.
        if sortedE[i] U STE does not form a cycle.
            STE ← STE U sortedE[i]
            i ← i+1                    ← ith edge in sorted list of edges.
        end if
    End While
End kruskalAlgorithm
```

Correctness proof of the Algorithm establishes kruskal's idea Achieves Global Optimum.

Observations :- Unlike Prim's Algorithm, STE (tree which we are growing iteratively) may not necessarily be connected.

Greedy Algorithm : In each iteration, we are trying to add an edge with minimum weight.

Example.



sortedEdges = [AB , CD , BC , BD , AD , AE , BE]

STE = ∅ , i = 1

STE = { AB } , i = 2

STE = { AB , CD } , i = 3

STE = { AB , CD , BC } , i = 4

STE = { AB , CD , BC , BD } , i = 5 — forms a cycle.

STE = { AB , CD , BC , AD } , i = 6 — forms a cycle.

STE = { AB , CD , BC , AE } , i = 7 — Set with (n-1) edges

---

kruskalAlgorithm( )

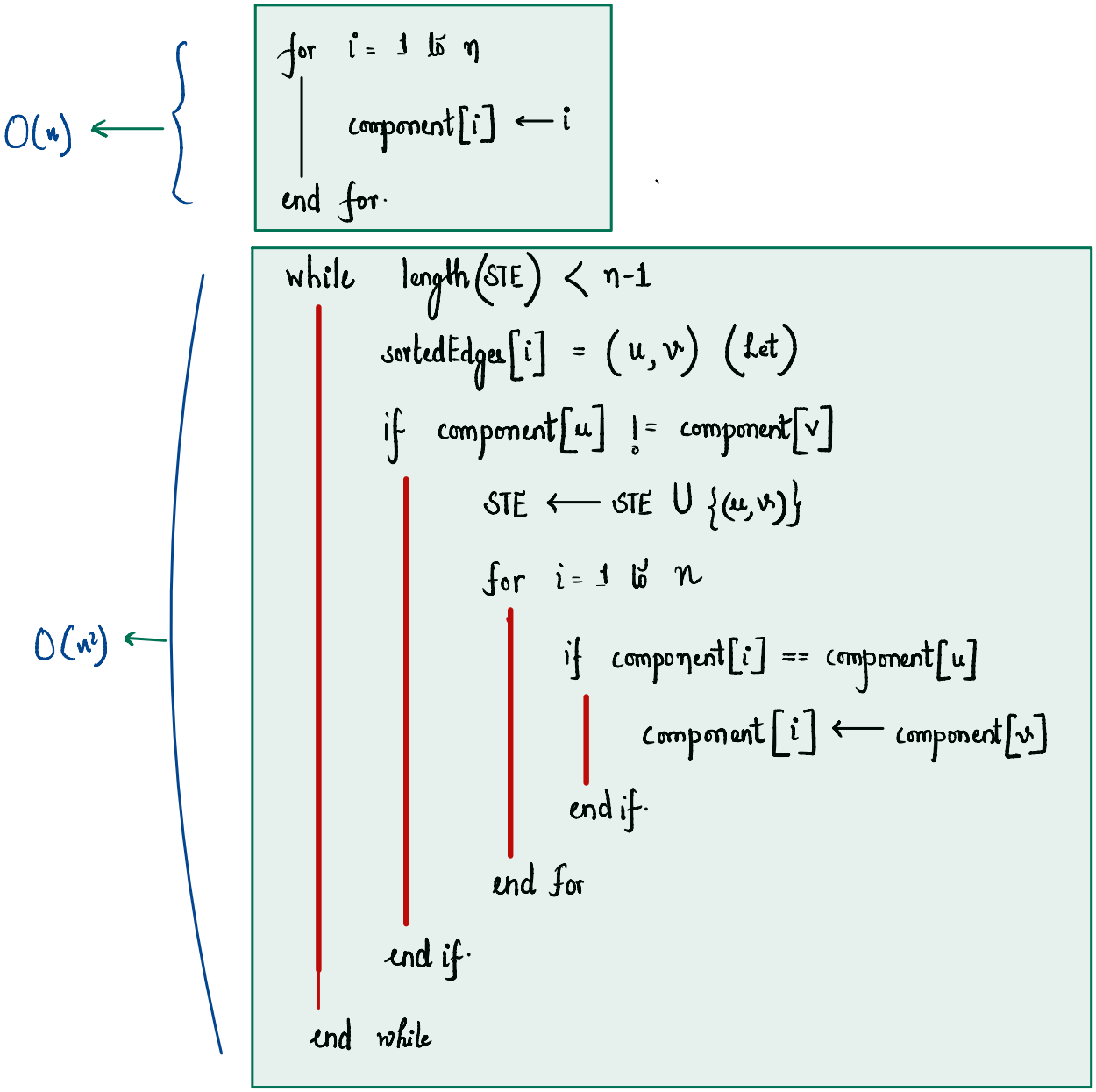
Given weighted undirected graph $G(V, E)$

sortedE = $[e_1, e_2, e_3, \cdots, e_m]$ be a sorted (increasing) list of edges. — $O(m \log m)$

Here, Weight on edge $(e_i) \leq$ Weight on edge e_j ; if $i < j$

STE ← ∅ (Spanning Tree Edges) — $O(1)$

$i \leftarrow 1$



BottleNeck merging the components

→ Use Find-Union Data Structure to bring down the complexity O(nlogn)