As always, sit with a partner and work through these together.

Activity 3: Kruskal Simulation

for each edge in shortest order
add edge to MST if it doesn't make a cycle

Activity 4: Runtime of Naïve Kruskal’s

Fill in the left-hand run times below for Kruskal’s Algorithm based on the naïve union-find implementation of merging clouds.

function kruskal(G):
//Input: undirected, weighted graph G
//Output: list of edges in MST
for vertices v in G: ------------------------ 1. O(______) 1. O(______)
    makeCloud(v)
MST = []
Sort edges by weight -------------------------- 2. O(______) 2. O(______)
for all edges (u,v): -------------------------- 3. O(______) 3. O(______)
    if u and v are not in same cloud:----- 4. O(______) 4. O(______)
        add (u,v) to MST
        merge clouds containing u and v ---- 5. O(______) 5. O(______)
return MST

Runtime of Naïve Kruskal’s: ________________

Activity 5: Runtime of Path-Compression Kruskal’s

Fill in the right-hand run times below of Kruskal’s Algorithm based on the path-compression union-find implementation of merging clouds.

function kruskal(G):
//Input: undirected, weighted graph G
//Output: list of edges in MST
for vertices v in G: ------------------------ 1. O(______) 1. O(______)
    makeCloud(v)
MST = []
Sort edges by weight -------------------------- 2. O(______) 2. O(______)
for all edges (u,v): -------------------------- 3. O(______) 3. O(______)
    if u and v are not in same cloud:----- 4. O(______) 4. O(______)
        add (u,v) to MST
        merge clouds containing u and v ---- 5. O(______) 5. O(______)
return MST

Runtime of Path-Compression Kruskal’s: ________________