Session 6: Greedy Algorithms

Saravana Senthilkumar - 3122247001057

February 17, 2026

Contents

1 1. Union-Find Data Structure

1.1 1. Design and Implement UnionFind functions

This block implements Union-Find using standard array access, avoiding list methods.

```
def find(parent, i):
    while parent[i] != i:
        parent[i] = parent[parent[i]]
        i = parent[i]
    return i

def union(parent, i, j):
    root_i = find(parent, i)
    root_j = find(parent, j)
    if root_i != root_j:
        parent[root_i] = root_j
        return 1
```

```
    return 0

parent = [0, 1, 2, 3, 4]
union(parent, 0, 1)
print(find(parent, 0))
print(find(parent, 1))
```

2 2. Priority Queue

2.1 1. Design and Implement PriorityQ functions

This block implements a Priority Queue. instead of ‘sort‘ or ‘pop‘, it manually iterates to find the smallest weight and uses slicing to remove it.

```
def pq_add(pq, w, u, v):
    pq = pq + [[w, u, v]]
    return pq

def pq_remove_min(pq):
    if len(pq) == 0:
        return [], pq

    min_idx = 0
    min_val = pq[0][0]

    i = 1
    while i < len(pq):
        if pq[i][0] < min_val:
            min_val = pq[i][0]
            min_idx = i
        i = i + 1

    item = pq[min_idx]

    # Remove by slicing: elements before + elements after
    pq = pq[:min_idx] + pq[min_idx+1:]

    return item, pq

pq = []
```

```
pq = pq_add(pq, 10, 'A', 'B')
pq = pq_add(pq, 5, 'C', 'D')
item, pq = pq_remove_min(pq)
print(item)
```

3 3. Kruskal's Algorithm

3.1 1. Design KruskalMST(vertices, edges)

This block implements Kruskal's algorithm using the manual helper functions defined above.

```
def KruskalMST(vertices, edges):
    v_map = {}
    idx = 0
    for v in vertices:
        v_map[v] = idx
        idx = idx + 1

    n = len(vertices)
    parent = list(range(n))

    pq = []
    for edge in edges:
        pq = pq_add(pq, edge[2], edge[0], edge[1])

    mst = []
    cost = 0

    while len(pq) > 0:
        item, pq = pq_remove_min(pq)
        w = item[0]
        u = item[1]
        v = item[2]

        u_id = v_map[u]
        v_id = v_map[v]

        if find(parent, u_id) != find(parent, v_id):
            union(parent, u_id, v_id)
```

```
            mst = mst + [[u, v, w]]
            cost = cost + w

    return mst, cost

nodes = ['A', 'B', 'C', 'D', 'E', 'F', 'G']
graph_edges = [
    ('E', 'G', 8), ('G', 'F', 5), ('E', 'F', 10),
    ('E', 'B', 18), ('E', 'C', 2), ('F', 'C', 3),
    ('F', 'D', 16), ('B', 'C', 12), ('C', 'D', 30),
    ('B', 'A', 4), ('C', 'A', 14), ('D', 'A', 26)
]

mst, cost = KruskalMST(nodes, graph_edges)
print(cost)
print(mst)
```