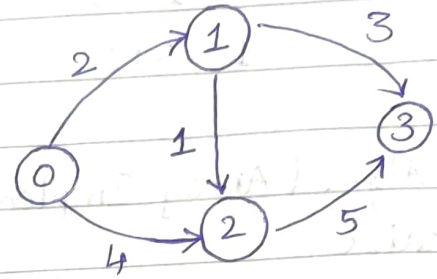
Exp. 4

Greedy Approach

eg. Graph -

Vertices: {0, 1, 2, 3}

Edges: 0-1 : 2
0-2 : 4
1-2 : 1
1-3 : 3
2-3 : 5



1) Kruskal's Algo.

(i) Sort all edges in non-decreasing order of wt.

1-2 : 1
0-1 : 2
1-3 : 3
0-2 : 4
2-3 : 5

(ii) Initialize DSUF

parent = [0, 1, 2, 3]
rank = [1, 1, 1, 1]

(iii) Add edges to MST

Edge 1-2: vertices 1 & 2 are in diff. sets ⟹ add edge to MST.
Update DSUF : parent = [0, 1, 1, 3]
rank = [1, 2, 1, 1]

Edge 0-1: vertices 0 & 1 are in diff. sets ⟹ add edge to MST.
parent = [1, 1, 1, 3]
rank = [1, 2, 1, 1]

Edge 1-3: diff. sets ⟹ add edge to MST
parent = [1, 1, 1, 1]
rank = [1, 2, 1, 1]

Edge 0-2: vertices have same parent ⟹ form a cycle ⟹ skip edge

Edge 2-3: same parent 1 ⟹ form a cycle ⟹ ~~ske~~ skip edge.

Final MST :

1 - 2 : 1
0 - 1 : 2
1 - 3 : 3

2) Prim's Algo. (Array Implementation)

(i) Initialize

key = [0, ∞, ∞, ∞] ..... key values for vertices.
parent = [-1, -1, -1, -1] ..... parent of each vertex.
inMST = [false, false, false, false] ..... track vertices in MST.

(ii) Start with vertex 0

inMST[0] = true.

Update keys of adj. vertices

key = [0, 2, 4, ∞]
parent = [~~-1~~, -1, 0, 0, -1]

Find vertex with min. key not in MST :

vertex 1 has min. key (key[1] = 2)

Add vertex 1 to MST

Add edge 0-1 to MST, update keys of vertices 2 & 3

key = [0, 2, 1, 3]
parent = [-1, 0, 1, 1]

(iii) Repeat by finding ~~vertex~~ vertices with min. keys.

vertex 2 has min. key (key[2] = 1)
add edge 1-2 to MST.
Update keys of vertex 3

key = [0, 2, 1, 3]
parent = [-1, 0, 1, 1]

vertex 3 has min. key (key[3] = 3)
add edge 1-3 to MST

Final MST :

0 - 1 : 2
1 - 2 : 1
1 - 3 : 3

3) Prim's Algo. (Min-Heap Implementation)

(i) Initialize

key = [0, ∞, ∞, ∞]
parent = [-1, -1, -1, -1]
inMST = [f, f, f, f]
minHeap = [(0, 0)] (key, vertex)

(ii) Extract vertex 0 from heap & add to MST (inMST[0] = t)
update adj. keys & push vertices to heap
key = [0, 2, 4, ∞]
parent = [-1, 0, 0, -1]
Heap = [(2, 1), (4, 2)]
extract vertex with min. key from heap (1)
add edge 0-1 to MST
update keys of vertices 2 & 3
push vertices 2 & 3 into heap
Heap = [(1, 2), (3, 3), (4, 2)]
key = [0, 2, 1, 3]
parent = [-1, 0, 1, 1]

(iii) Extract vertex 2 from heap & add edge 1-2 to MST
Extract vertex 3 from heap & add edge 1-3 to MST

Final MST :

0 - 1 : 2
1 - 2 : 1
1 - 3 : 3

DSUF : ~~suf~~ eff. union by rank & find by path compression.