#Q11 - Union by Rank and Union by Size

#Union by Rank

class DisjointSet:

def \_\_init\_\_(self, size):

self.parent = [i for i in range(size)]

self.rank = [0] \* size

def find(self, i):

if self.parent[i] != i:

self.parent[i] = self.find(self.parent[i])

return self.parent[i]

def union\_by\_rank(self, i, j):

irep = self.find(i)

jrep = self.find(j)

if irep == jrep:

return

irank = self.rank[irep]

jrank = self.rank[jrep]

if irank < jrank:

self.parent[irep] = jrep

elif jrank < irank:

self.parent[jrep] = irep

else:

self.parent[irep] = jrep

self.rank[jrep] += 1

def main():

size = 5

ds = DisjointSet(size)

ds.union\_by\_rank(0, 1)

ds.union\_by\_rank(2, 3)

ds.union\_by\_rank(1, 3)

for i in range(size):

print(f"Element {i} belongs to the set with representative {ds.find(i)}")

if \_\_name\_\_ == "\_\_main\_\_":

main()

#Union by Size

class UnionFind:

def \_\_init\_\_(self, n):

self.Parent = list(range(n))

self.Size = [1] \* n

def find(self, i):

if self.Parent[i] != i:

self.Parent[i] = self.find(self.Parent[i])

return self.Parent[i]

def unionBySize(self, i, j):

irep = self.find(i)

jrep = self.find(j)

if irep == jrep:

return

isize = self.Size[irep]

jsize = self.Size[jrep]

if isize < jsize:

self.Parent[irep] = jrep

self.Size[jrep] += self.Size[irep]

else:

self.Parent[jrep] = irep

self.Size[irep] += self.Size[jrep]

n = 5

unionFind = UnionFind(n)

unionFind.unionBySize(0, 1)

unionFind.unionBySize(2, 3)

unionFind.unionBySize(0, 4)

for i in range(n):

print("Element {}: Representative = {}".format(i, unionFind.find(i)))