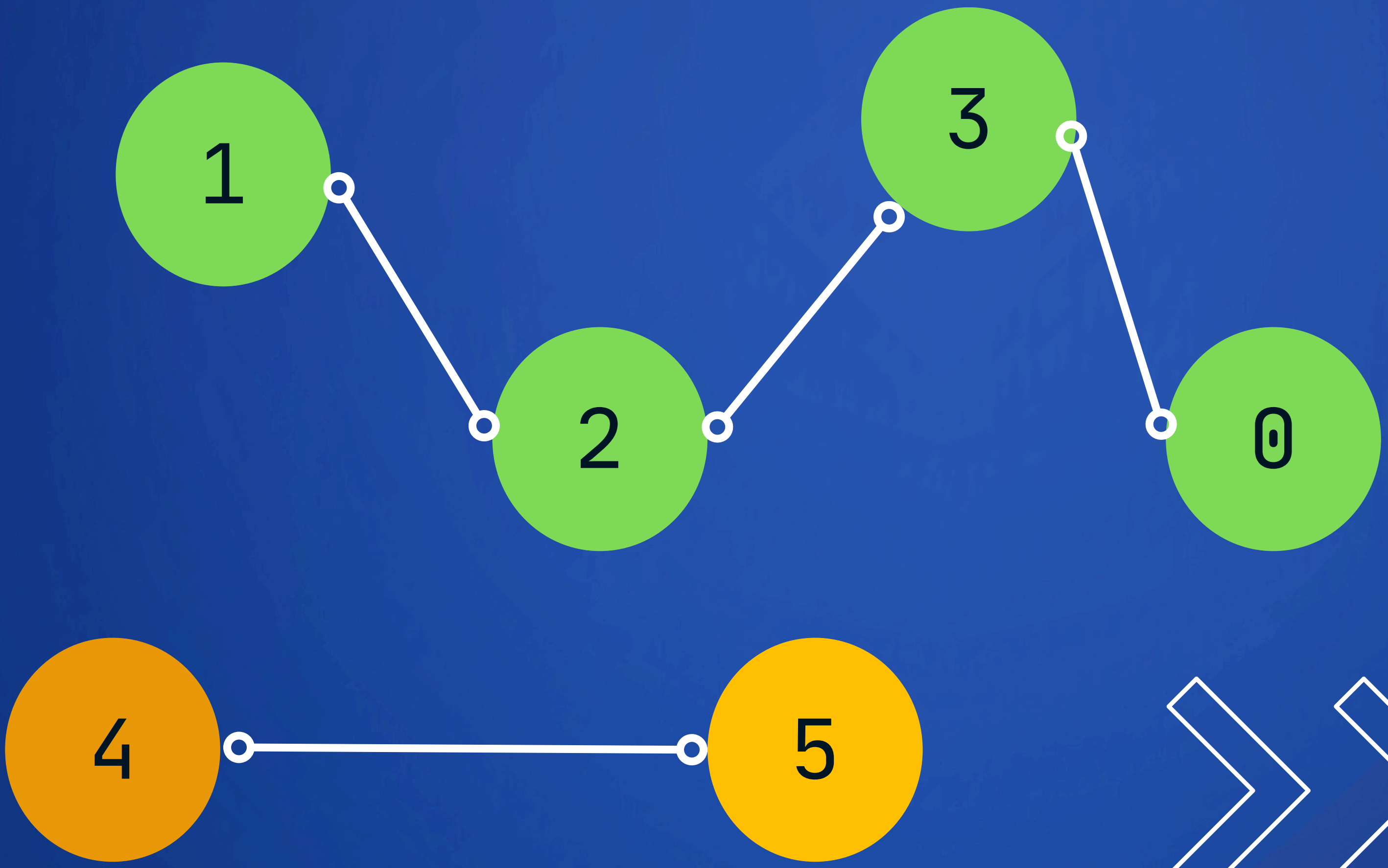
UNION FIND (DIS-JOINT SETS)

a popular algorithm for coding interviews

Problem statement - Find the number of independent group of connected nodes in a graph

edges = [[1,2], [2,3], [3,0], [4,5]]

following are two set / group of connected nodes



- Create a list parents that will hold the parent of each node in the graph
- in starting parent of every node is the node itself i.e parent_list = [0,1,2,3,4,5]

index represents the node and the value at the index is the parent

parent_list[2] = 2 (means parent of node 2 is 2)

- if two nodes are connected, make one node as child of the other node by changing the parent of child node to the other node

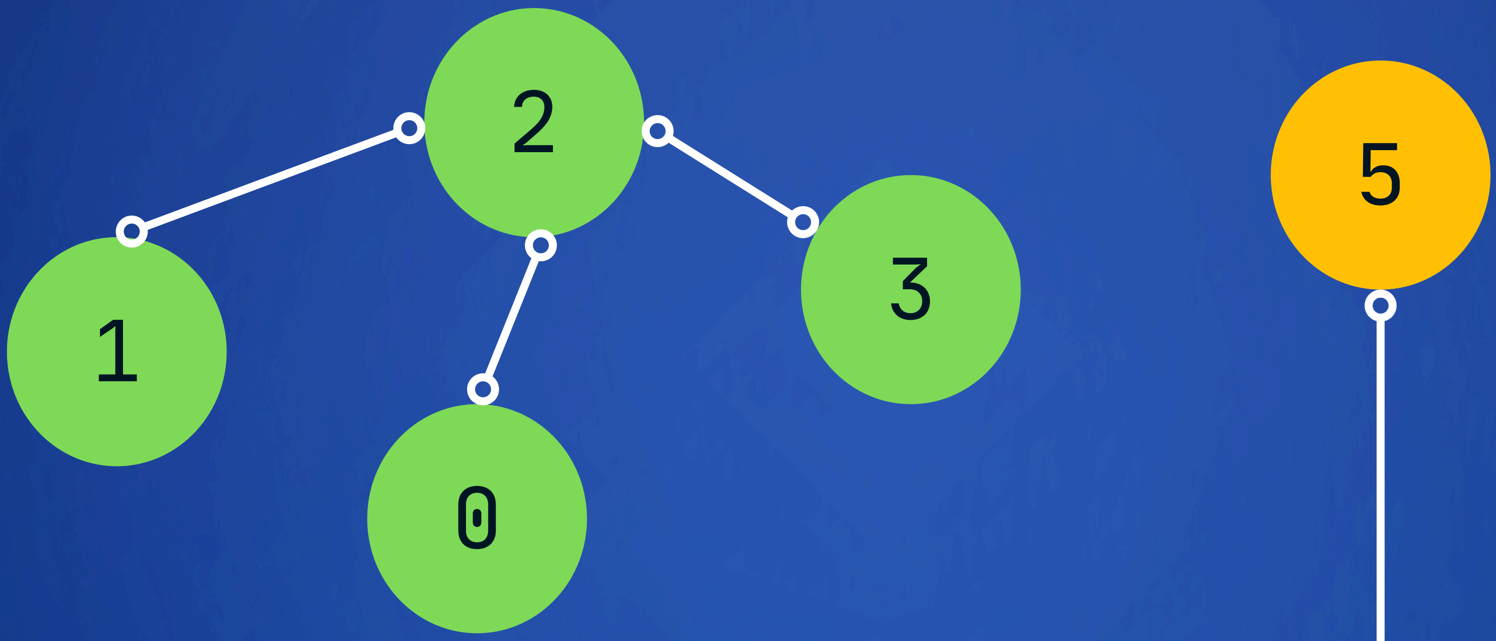
for e.g (3,0) 3 and 0 are connected, therefore the parent of 0 will be updated as 3 , however if 3 is not the root node for this set then parent of 3 will be updated as parent of 0, in this example it will be 2, because 2 will be the root of this set

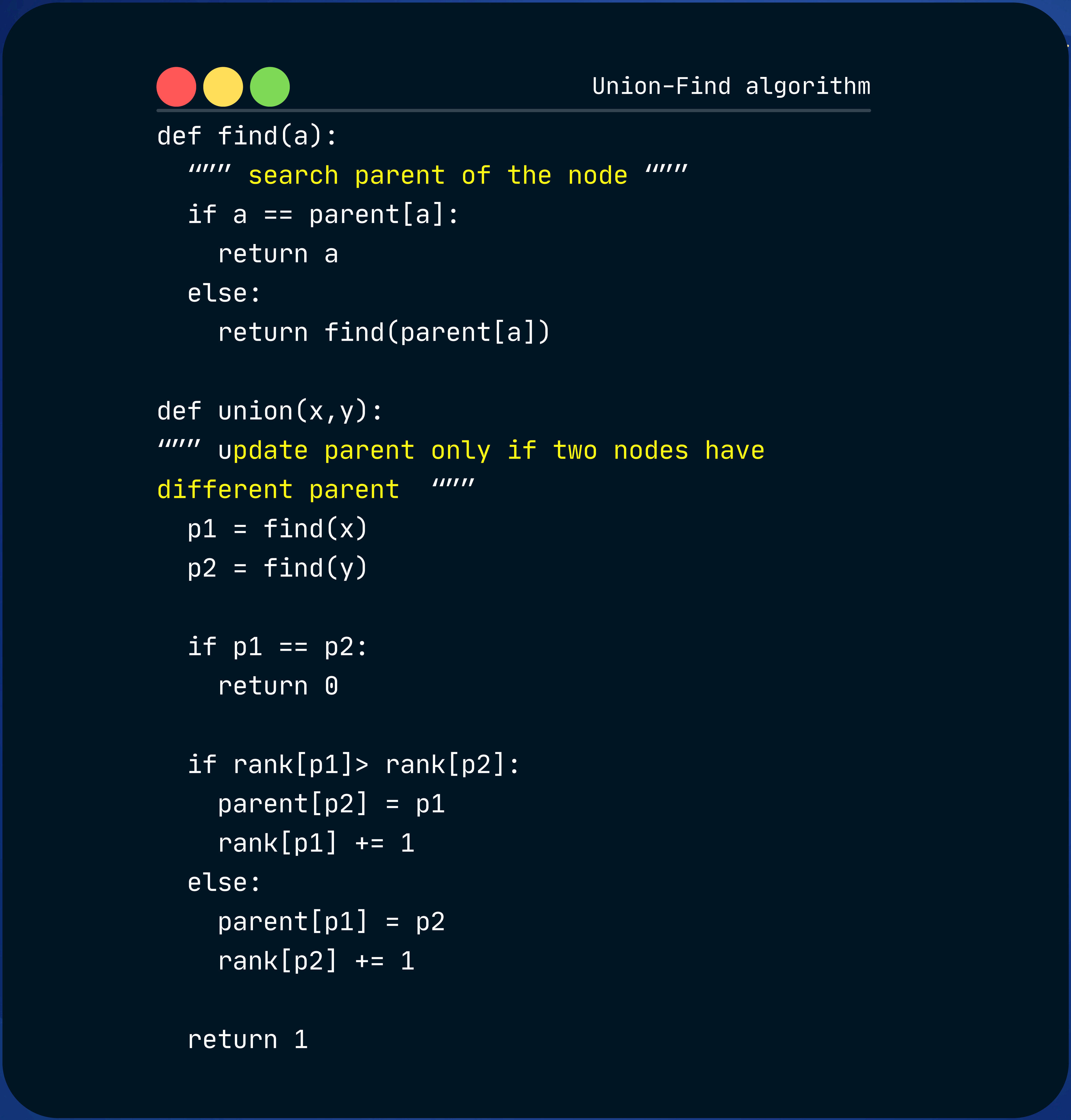
after updation , parent_list = [2,1,2,3,4,5]

parent_list[0] = 2 (means parent of node 0 is 2), by doing this we have mode 0 as child of 2 and hence they belong to same set

After updating the parent of all nodes the list will be

parent_list = [2,2,2,2,5,5]





Union-Find algorithm

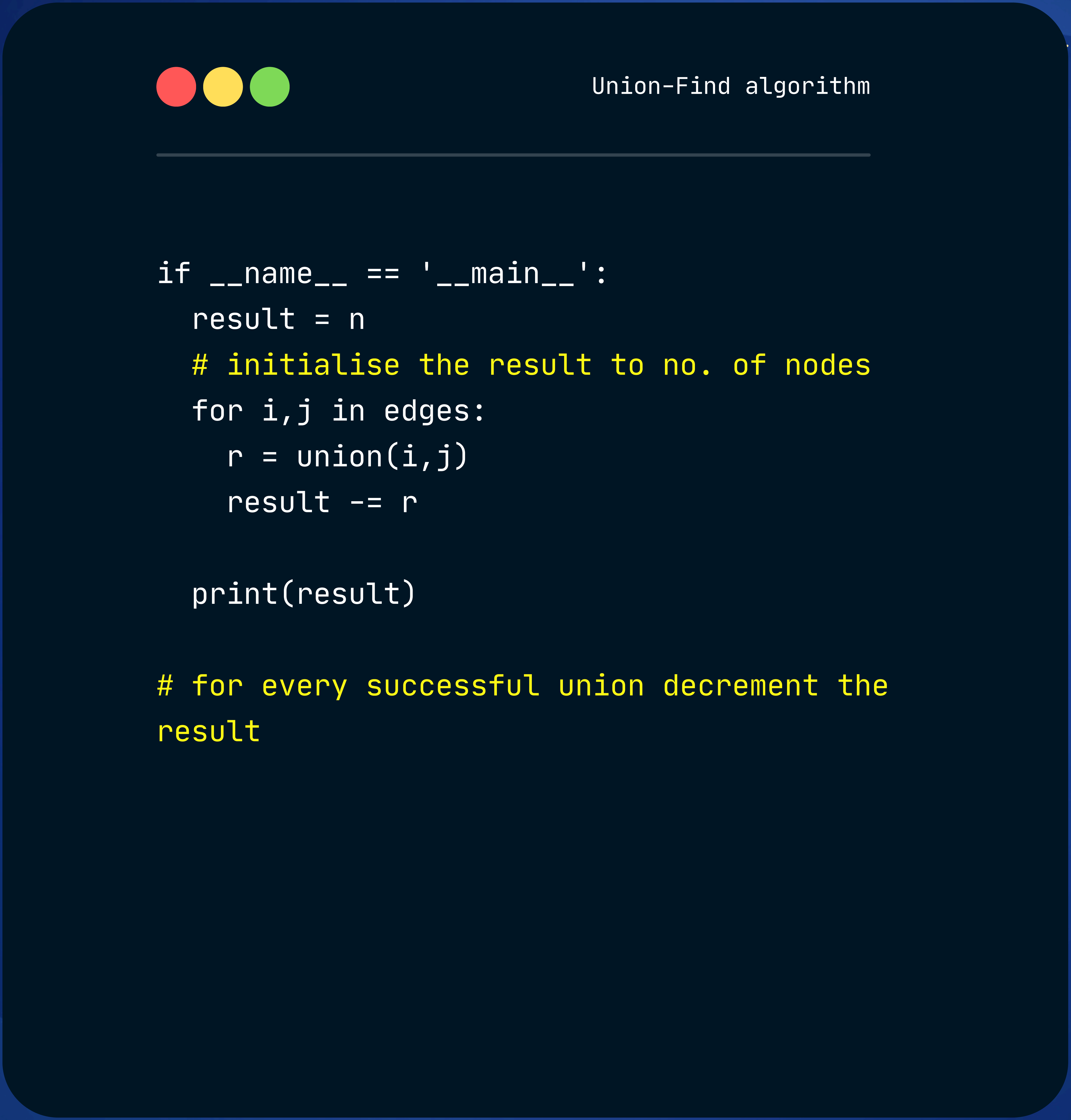
```
def find(a):
  “”” search parent of the node “””
  if a == parent[a]:
    return a
  else:
    return find(parent[a])

def union(x,y):
“”” update parent only if two nodes have
different parent  “””
  p1 = find(x)
  p2 = find(y)

  if p1 == p2:
    return 0

  if rank[p1]> rank[p2]:
    parent[p2] = p1
    rank[p1] += 1
  else:
    parent[p1] = p2
    rank[p2] += 1

  return 1
```



Union-Find algorithm

```
if __name__ == '__main__':
  result = n
  # initialise the result to no. of nodes
  for i,j in edges:
    r = union(i,j)
    result -= r

  print(result)

# for every successful union decrement the result
```