96. Boruvka's Algorithm

Program:

```
def boruvka(graph):
   mst = []
   subsets = [[i] for i in range(len(graph))]

   while len(mst) < len(graph) - 1:
      cheapest = [None] * len(graph)

      for i in range(len(graph)):
         for j in range(len(graph[i])):
            if i != j:
               root_i = find(subsets, i)
               root_j = find(subsets, j)

               if root_i != root_j:
                  if cheapest[root_i] is None or graph[i][j] < graph[cheapest[root_i]][root_i]:
                     cheapest[root_i] = j
                  if cheapest[root_j] is None or graph[i][j] < graph[cheapest[root_j]][root_j]:
                     cheapest[root_j] = i

      for i in range(len(graph)):
         if cheapest[i] is not None:
            root_i = find(subsets, i)
            root_j = find(subsets, cheapest[i])

            if root_i != root_j:
               mst.append([i, cheapest[i], graph[i][cheapest[i]]])
               union(subsets, root_i, root_j)

   return mst

def find(subsets, node):
   for i in range(len(subsets)):
      if node in subsets[i]:
         return i

def union(subsets, a, b):
   subsets[a] += subsets[b]
   subsets.pop(b)

# Example Usage
graph = [
   [0, 2, 0, 6, 0],
   [2, 0, 3, 8, 5],
   [0, 3, 0, 0, 7],
   [6, 8, 0, 0, 9],
   [0, 5, 7, 9, 0]
]

minimum_spanning_tree = boruvka(graph)
print(minimum_spanning_tree)
```

Output:

```
[[0, 2, 0], [1, 0, 2], [3, 2, 0], [4, 0, 0]]

=== Code Execution Successful ===
```

Time complexity:

O(Elog V)