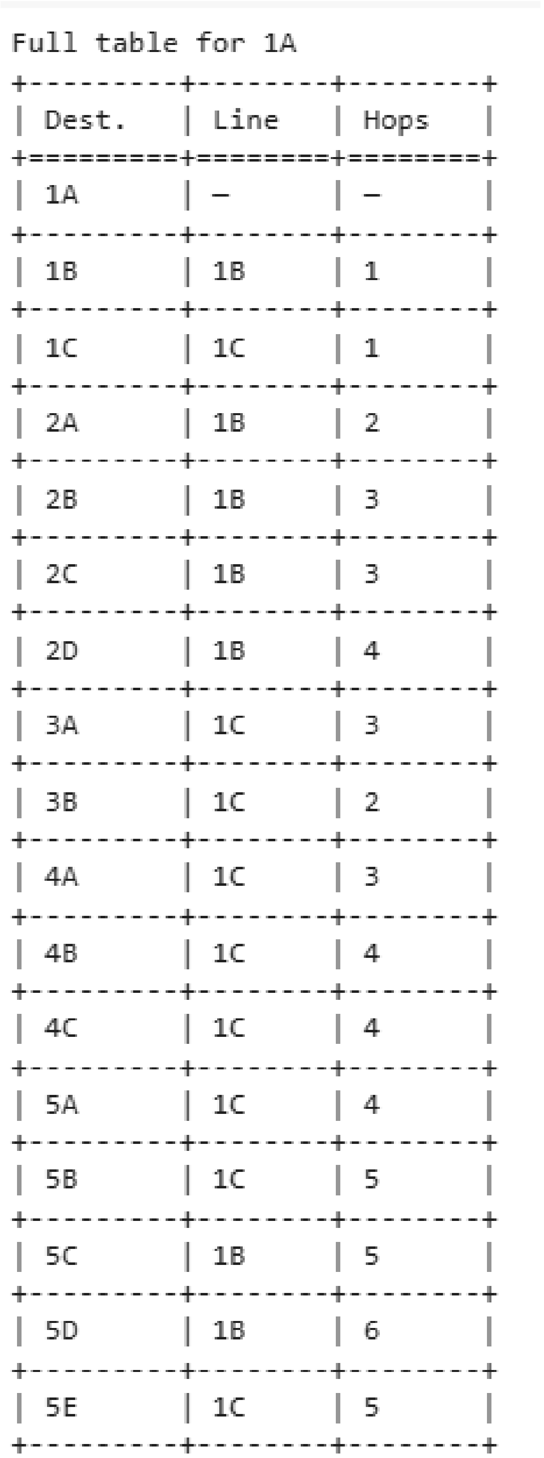
b) Hierarchical Routing Algorithm

Python Code:

```
from collections import defaultdict, deque

# Define network structure
all_nodes = ['1A', '1B', '1C', '2A', '2B', '2C', '2D', '3A', '3B', '4A',
'4B', '4C', '5A', '5B', '5C', '5D', '5E']

regions = {
    '1': ['1A', '1B', '1C'],
    '2': ['2A', '2B', '2C', '2D'],
    '3': ['3A', '3B'],
    '4': ['4A', '4B', '4C'],
    '5': ['5A', '5B', '5C', '5D', '5E']
}

intra_region_edges = {
    '1': [('1A', '1B'), ('1B', '1C'), ('1C', '1A')],
    '2': [('2A', '2B'), ('2B', '2D'), ('2D', '2C'), ('2C', '2A')],
    '3': [('3A', '3B')],
    '4': [('4A', '4B'), ('4B', '4C'), ('4C', '4A')],
    '5': [('5A', '5B'), ('5B', '5C'), ('5C', '5D'), ('5D', '5E'), ('5E',
'5A')]
}

inter_region_edges = [('1B', '2A'), ('1C', '3B'), ('3B', '4A'), ('4A',
'5A'), ('2D', '5C')]

# Build adjacency list
graph = defaultdict(list)
for region in intra_region_edges:
    for src, dst in intra_region_edges[region]:
        graph[src].append(dst)
        graph[dst].append(src)
for src, dst in inter_region_edges:
    graph[src].append(dst)
    graph[dst].append(src)

# Find shortest path and return path with hop count
def find_shortest_path(graph, start, end):
    if start == end:
        return [start], 0

    visited = set()
    queue = deque([(start, [start])])

    while queue:
        node, path = queue.popleft()
        if node not in visited:
            visited.add(node)
            for neighbor in graph[node]:
                if neighbor not in visited:
                    new_path = path + [neighbor]
                    if neighbor == end:
                        return new_path, len(new_path) - 1  # Hops = edges
traversed
                    queue.append((neighbor, new_path))
    return None, 0

# Generate full routing table with hop counts
def generate_full_table(source):
    table = {}
    for dest in all_nodes:
        path, hops = find_shortest_path(graph, source, dest)
        if path:
            if dest == source:
                table[dest] = ("--", "--")  # As per image format
            else:
                table[dest] = (path[1], hops)  # Next hop and hop count
    return table

# Generate hierarchical table with hop counts
def generate_hierarchical_table(source):
    table = {}
    source_region = '1'

    # Add source node entry
    table[source] = ("--", "--")

    # Intra-region routing (within Region 1)
    for dest in regions['1']:
        if dest != source:
            path, hops = find_shortest_path(graph, source, dest)
            if path:
                table[dest] = (path[1], hops)

    # Inter-region routing (group by region)
    gateway_nodes = {
        '2': '1B',  # 1B->2A
        '3': '1C',  # 1C->3B
        '4': '1C',  # 1C->3B->4A
        '5': '1C'   # 1C->3B->4A->5A
    }

    # For each region, find the hop count to the entry node of that region
    region_entry_nodes = {
        '2': '2A',  # Entry to Region 2 via 2A
        '3': '3B',  # Entry to Region 3 via 3B
        '4': '4A',  # Entry to Region 4 via 4A
        '5': '5A'   # Entry to Region 5 via 5A
    }

    for region in regions:
        if region != source_region:
            gateway = gateway_nodes.get(region)
            entry_node = region_entry_nodes.get(region)
            if gateway and entry_node:
                _, hops = find_shortest_path(graph, source, entry_node)
                table[region] = (gateway, hops)

    return table

# Generate tables
full_table = generate_full_table('1A')
hierarchical_table = generate_hierarchical_table('1A')

# Print tables in the exact format as the image
print("Full table for 1A")
print("Dest.  Line  Hops")
for dest, (next_hop, hops) in sorted(full_table.items(), key=lambda x:
x[0]):
    print(f"{dest:<6} {next_hop:<5} {hops}")

print("\nHierarchical table for 1A")
print("Dest.  Line  Hops")
for dest, (next_hop, hops) in sorted(hierarchical_table.items(), key=lambda
x: x[0]):
    print(f"{dest:<6} {next_hop:<5} {hops}")
```

Output:



```
Full table for 1A
+---------+--------+--------+
| Dest.   | Line   | Hops   |
+=========+========+========+
| 1A      | –      | –      |
+---------+--------+--------+
| 1B      | 1B     | 1      |
+---------+--------+--------+
| 1C      | 1C     | 1      |
+---------+--------+--------+
| 2A      | 1B     | 2      |
+---------+--------+--------+
| 2B      | 1B     | 3      |
+---------+--------+--------+
| 2C      | 1B     | 3      |
+---------+--------+--------+
| 2D      | 1B     | 4      |
+---------+--------+--------+
| 3A      | 1C     | 3      |
+---------+--------+--------+
| 3B      | 1C     | 2      |
+---------+--------+--------+
| 4A      | 1C     | 3      |
+---------+--------+--------+
| 4B      | 1C     | 4      |
+---------+--------+--------+
| 4C      | 1C     | 4      |
+---------+--------+--------+
| 5A      | 1C     | 4      |
+---------+--------+--------+
| 5B      | 1C     | 5      |
+---------+--------+--------+
| 5C      | 1B     | 5      |
+---------+--------+--------+
| 5D      | 1B     | 6      |
+---------+--------+--------+
| 5E      | 1C     | 5      |
+---------+--------+--------+
```

Hierarchical table for 1A

Dest.	Line	Hops
1A	–	–
1B	1B	1
1C	1C	1
2	1B	2
3	1C	2
4	1C	3
5	1C	4