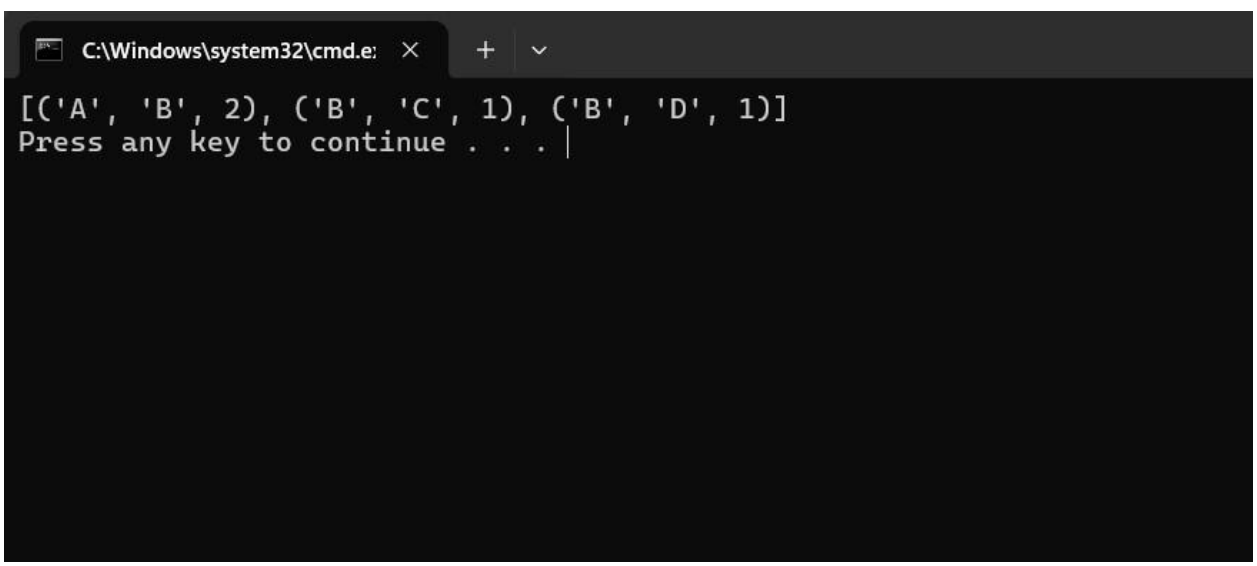
96) Prims Algorithm

CODE:

```
import heapq
 def prim(graph):
mst = []     visited =
set()
    start_node = list(graph.keys())[0]     visited.add(start_node)
    edges = [(cost, start_node, neighbor)
 for neighbor, cost in graph[start_node]]     heapq.heapify(edges)
     while edges:
         cost, n1, n2 = heapq.heappop(edges)
        if n2 not in visited:
            visited.add(n2)            mst.append((n1, n2, cost))
            for neighbor, c in graph[n2]:
                if neighbor not in visited:
heapq.heappush(edges, (c, n2, neighbor))

    return mst
 graph = {
    'A': [('B', 2), ('C', 3)],
    'B': [('A', 2), ('C', 1), ('D', 1)],
    'C': [('A', 3), ('B', 1), ('D', 1)],
    'D': [('B', 1), ('C', 1)]
}
minimum_spanning_tree = prim(graph)
print(minimum_spanning_tree)
```

OUTPUT:



TIME COMPLEXITY : O(nlogn)