KRUSHKAL

```
def find(parent, v):
    if parent[v] != v:
        parent[v] = find(parent, parent[v])
    return parent[v]


def kruskal(graph):
    edges = sorted([(w, u, v) for u in graph for v, w in graph[u]])
    parent = {v: v for v in graph}
    mst = []


    for w, u, v in edges:
        if find(parent, u) != find(parent, v):
            parent[find(parent, u)] = find(parent, v)
            mst.append((u, v, w))


    return mst


# Graph definition
graph = {
    'A': [('B', 1), ('C', 4)],
    'B': [('A', 1), ('C', 2), ('D', 5)],
    'C': [('A', 4), ('B', 2), ('D', 1)],
    'D': [('B', 5), ('C', 1)]
}
# Edges Are:
#  [(1, 'A', 'B'), (1, 'B', 'A'), (1, 'C', 'D'), (1, 'D', 'C'), (2, 'B', 'C'), (2, 'C', 'B'), (4, 'A', 'C'), (4, 'C', 'A'), (5, 'B', 'D'), (5, 'D', 'B')]


# Main execution
```

```
tot_cost = 0
span_tree = kruskal(graph)

for source, target, cost in span_tree:
    tot_cost += cost

print("MST:", span_tree)
print("The cost of MST is:", tot_cost)
```