```
class UnionFind:
    def __init__(self, n):
        self.parent = [i for i in range(n)]

    def find(self, a):
        if self.parent[a] != a:
            self.parent[a] = self.find(self.parent[a])
        return self.parent[a]

    def union(self, a, b):
        root_a = self.find(a)
        root_b = self.find(b)
        self.parent[root_a] = root_b


def kruskal(edges, n):
    edges.sort()
    uf = UnionFind(n)
    min_cost = 0
    for cost, a, b in edges:
        if uf.find(a) != uf.find(b):
            min_cost += cost
            uf.union(a, b)
    return min_cost


if __name__ == "__main__":
    nodes = 4
    edges = 5
    graph = [
        (10, 0, 1),
        (18, 1, 2),
        (13, 2, 3),
        (21, 0, 2),
        (22, 1, 3)
    ]
    min_cost = kruskal(graph, nodes)
    print("Minimum cost is:", min_cost)
```