```
def kruskal_algorithm(self):
    """
    Returns the minimum spanning tree of the graph
    :return:
    """
    # creating an empty graph
    minimum_spanning_tree = Graph()

    # creating the list of edges sorted by cost
    sorted_edges = []
    for edge in self.edges_iterator():
        sorted_edges.append((self.get_cost(edge[0], edge[1]), edge[0],
edge[1]))
    sorted_edges.sort()

    # creating the list of sets
    components = []
    for vertex in self.vertices_iterator():
        components.append({vertex}) # each vertex is in a component by
itself

    # creating the minimum spanning tree
    for edge in sorted_edges:
        component1 = None
        component2 = None
        for component in components:
            if edge[1] in component:
                component1 = component
            if edge[2] in component:
                component2 = component
        if component1 != component2:
            # adding the nodes if they do not already exist
            if edge[1] not in minimum_spanning_tree.vertices_iterator():
                minimum_spanning_tree.add_vertex(edge[1])
            if edge[2] not in minimum_spanning_tree.vertices_iterator():
                minimum_spanning_tree.add_vertex(edge[2])

            # adding the edge to the minimum spanning tree
            minimum_spanning_tree.add_edge(edge[1], edge[2], edge[0])

            # updating the components
            component1.update(component2)
            components.remove(component2)

    return minimum_spanning_tree
```