例 1.11 kruskal 和 Prime 算法实现
算法功能:求解最小生成树

修改参数：maps：图中边的连接关系和权值信息
        """

```
class Graph(object):
    def __init__(self, maps):
        self.maps = maps
        self.nodenum = self.get_nodenum()  #  节点数
        self.edgenum = self.get_edgenum()  #  边数

    def get_nodenum(self):
        return len(self.maps)

    def get_edgenum(self):
        count = 0
        for i in range(self.nodenum):
            for j in range(i):
                if self.maps[i][j] > 0 and self.maps[i][j] < 9999:
                    count += 1
        return count

    def kruskal(self):
        res = []
        if self.nodenum <= 0 or self.edgenum < self.nodenum - 1:
            return res
        edge_list = []
        for i in range(self.nodenum):
            for j in range(i, self.nodenum):
                if self.maps[i][j] < 9999:
                    edge_list.append([i, j, self.maps[i][j]])  # 按[begin, end,
weight]形式加入
        edge_list.sort(key=lambda a: a[2])  # 按边长度排序，得到已经排好序的边集
合

        group = [[i] for i in range(self.nodenum)]
        for edge in edge_list:
            for i in range(len(group)):
                if edge[0] in group[i]:
                    m = i
                if edge[1] in group[i]:
                    n = i
```

```
            if m != n:
                res.append(edge)
                group[m] = group[m] + group[n]
                group[n] = []
        return res

    def prim(self):
        res = []
        if self.nodenum <= 0 or self.edgenum < self.nodenum - 1:
            return res
        res = []
        seleted_node = [0]
        candidate_node = [i for i in range(1, self.nodenum)]

        while len(candidate_node) > 0:
            begin, end, minweight = 0, 0, 9999
            for i in seleted_node:
                for j in candidate_node:
                    if self.maps[i][j] < minweight:
                        minweight = self.maps[i][j]
                        begin = i
                        end = j
            res.append([begin, end, minweight])
            seleted_node.append(end)
            candidate_node.remove(end)
        return res

if __name__ == "__main__":
    Inf=float('inf')
    rowA = [0, 4, Inf, Inf, Inf, Inf, Inf, 8, Inf]
    rowB = [4, 0, 8, Inf, Inf, Inf, Inf, 11, Inf]
    rowC = [Inf, 8, 0, 7, Inf, 4, Inf, Inf, 2]
    rowD = [Inf, Inf, 6, 0, 9, 14, Inf, Inf, Inf]
    rowE = [Inf, Inf, Inf, 9, 0, 10, Inf, Inf, Inf]
    rowF = [Inf, Inf, 4, 14, 10, 0, 2, Inf, Inf]
    rowG = [Inf, Inf, Inf, Inf, Inf, 2, 0, 1, 6]
    rowH = [8, 11, Inf, Inf, Inf, Inf, Inf, 0, 7]
    rowI = [Inf, Inf, 2, Inf, Inf, Inf, 6, 7, 0]

    maps = [rowA, rowB, rowC, rowD, rowE, rowF, rowG, rowH, rowI]
    graph = Graph(maps)
    print('邻接矩阵为\n%s' % graph.maps)
    print('节点数据为%d，边数为%d\n' % (graph.nodenum, graph.edgenum))
    print('------最小生成树 kruskal 算法------')
```

```
print(graph.kruskal())
print('------最小生成树 prim 算法')
print(graph.prim())
```