





# Krushka's MST Algo

class DSU:

    def \_\_init\_\_(self, n) -> None:

        self.n = n

        self.par = [x for x in range(0, n+1)]

        self.size = [1 for x in range(0, n+1)]

    def get(self, x):

        if x == self.par[x]:

            return x

        self.par[x] = self.get(self.par[x])

        return self.par[x]

    def same\_set(self, x, y):

        return self.get(x) == self.get(y)

    def unify(self, x, y):

        x = self.get(x)

        y = self.get(y)

        if (x == y):

            return 0

        if (self.size[x] < self.size[y]):

            x, y = y, x

        if (self.size[x] == self.size[y]):

            self.size[x] += 1

        self.par[y] = x

        return 1

class MST:

    # edge list representation

    def \_\_init\_\_(self, n) -> None:

        self.n = n

        self.edges = []

    def get\_input(self):

        print("Enter u, v, wt")

        for i in range(1, self.n+1):

            u, v, wt = [int(x) for x in input().split()]

            if (u > v):

                u, v = v, u

            self.edges.append((u, v, wt))

        print("The Graph is: ")

        self.print\_graph(self.edges)

    def create\_mst(self):

        self.edges = sorted(

            self.edges, key=lambda tpl: (tpl[2], tpl[0], tpl[1]))

        ds = DSU(self.n)

        self.mst\_edges = []

        mst\_wt = 0

        for (u, v, wt) in self.edges:

            if not ds.same\_set(u, v):

                ds.unify(u, v)

                mst\_wt += wt

                self.mst\_edges.append((u, v, wt))

        print("Weight of MST is ", mst\_wt)

        print("MST edges are:")

        self.print\_graph(self.mst\_edges)

    def print\_graph(self, edges):

        for tpl in edges:

            print(tpl)

def main():

    n = int(input("Enter number of nodes: "))

    mst = MST(n)

    mst.get\_input()

    mst.create\_mst()

main()

**Testcase:**





