Министерство образования Республики Беларусь

Учреждения образования

Брестский государственный технический университет

Кафедра ИИТ

Лабораторная работа №3

По дисциплине ТМОИС

За 2 семестр

Тема: остовные деревья

Выполнил :

Студент 1 курса

Группы ИИ-21

Корпач Д.Р.

Проверил

Глущенко

Брест 2022

Вариант 7

**Алгоритм Прима**

from math import inf

V=9

G=[[0,2,3,0,0,0,0,0,0],

   [2,0,4,0,0,0,0,0,0],

   [3,4,0,5,0,0,7,0,0],

   [0,0,5,0,3,5,6,0,0],

   [0,0,0,3,0,1,0,0,0],

   [0,0,0,5,1,0,0,0,0],

   [0,0,7,6,0,0,0,1,1],

   [0,0,0,0,0,0,1,0,1],

   [0,0,0,0,0,0,1,1,0]]

selected=[0 for i in range(V)]

no\_edge=0

selected[0]=True

print("edge  :  weight")

while no\_edge<V-1:

    min=inf

    x,y=0,0

    for i in range(V):

        if selected[i]:

            for j in range(V):

                if not selected[j] and G[i][j]:

                    if min>G[i][j]:

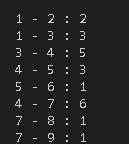
                        min=G[i][j]

                        x,y=i,j

    print(f"{x+1} - {y+1} : {G[x][y]}")

    selected[y]=True

    no\_edge+=1

****

**алгоритм Крускаля**

class Graph:

    def \_\_init\_\_(self, vertices):

        self.V = vertices

        self.graph = []

    def add\_edge(self, u, v, w):

        self.graph.append([u, v, w])

    def find(self, parent, i):

        if parent[i] == i:

            return i

        return self.find(parent, parent[i])

    def apply\_union(self, parent, rank, x, y):

        xroot = self.find(parent, x)

        yroot = self.find(parent, y)

        if rank[xroot] < rank[yroot]:

            parent[xroot] = yroot

        elif rank[xroot] > rank[yroot]:

            parent[yroot] = xroot

        else:

            parent[yroot] = xroot

            rank[xroot] += 1

    def kruskal\_algo(self):

        result = []

        i, e = 0, 0

        self.graph = sorted(self.graph, key=lambda item: item[2])

        parent = []

        rank = []

        for node in range(self.V):

            parent.append(node)

            rank.append(0)

        while e < self.V - 1:

            u, v, w = self.graph[i]

            i = i + 1

            x = self.find(parent, u)

            y = self.find(parent, v)

            if x != y:

                e = e + 1

                result.append([u, v, w])

                self.apply\_union(parent, rank, x, y)

        for u, v, weight in result:

            print("%d - %d: %d" % (u+1, v+1, weight))

g = Graph(9)

g.add\_edge(0, 1, 2)

g.add\_edge(0, 2, 3)

g.add\_edge(1, 2, 4)

g.add\_edge(2, 3, 5)

g.add\_edge(2, 6, 7)

g.add\_edge(3, 6, 6)

g.add\_edge(3, 4, 3)

g.add\_edge(3, 5, 5)

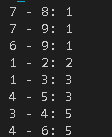
g.add\_edge(6, 7, 1)

g.add\_edge(6, 8, 1)

g.add\_edge(5, 8, 1)

g.add\_edge(7, 8, 2)

g.kruskal\_algo()

****