**Ministerul Educaţiei și Cercetării al Republicii Moldova**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică și Microelectronică**

**RAPORT**

Lucrarea de laborator nr.3

*Algoritmi Greedy.*

A efectuat:

st. gr. SI-211 Vozian Vladimir

A verificat:

asist. univ. Sergiu Scrob

Chişinău 2022

**Scopul lucrării:**

1. Studierea tehnicii greedy.

2. Analiza şi implementarea algoritmilor greedy.

**Codul programului:**

import random

def createGraf(varf, dens):

    G = [[0] \* varf] \* varf

    for i in range(varf):

        for j in range(varf):

            if (j > i):

                r = random.randint(0,100)

                if(r <= dens):

                    r = random.randint(1,10000)

                    G[i][j] = r

                else:

                    G[i][j] = G[j][i]

    for i in range(varf):

        for j in range(varf):

            if(G[i][j] == 0 and i != j):

                G[i][j] = 99999

    return G

def prim(V, G):

    iter = 0

    no\_edge = 0

    selected = [0]\*V

    selected[0] = True

    while (no\_edge < V - 1):

        iter += 1

        minimum = 99999

        x = 0

        y = 0

        for i in range(V):

            iter += 1

            if selected[i]:

                iter += 1

                for j in range(V):

                    iter += 1

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

                        iter += 1

                        if minimum > G[i][j]:

                            iter += 1

                            minimum = G[i][j]

                            x = i

                            y = j

        selected[y] = True

        no\_edge += 1

    print("iteratii prim:",iter)

iterKruskal = 0

parent = []

def find(i):

    global iterKruskal

    global parent

    while(parent[i]):

        i = parent[i]

        iterKruskal += 1

    return i

def union(x,y):

    global iterKruskal

    iterKruskal += 1

    if(x != y):

        parent[y] = x

def kruskal(V,G):

    global iterKruskal

    n = 0

    while(n < V - 1):

        iterKruskal += 1

        min = 99999

        for i in range(V):

            iterKruskal += 1

            for j in range(V):

                iterKruskal += 1

                if(G[i][j] < min and i != j):

                    iterKruskal += 1

                    min = G[i][j]

                    x = i

                    y = j

        u = find(x)

        v = find(y)

        union(u,v)

        G[x][y] = G[y][x] = 99999

        n += 1

    print("iterratii kruskal :",iterKruskal)

varf = [10,15,20]

dens = [20,50,100]

G = createGraf(5,100)

for i in dens:

    for j in varf:

        parent = [0] \* j

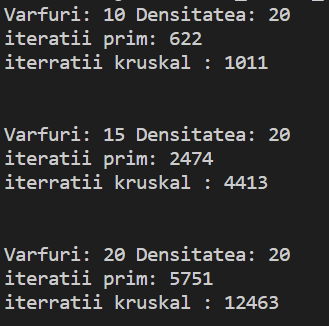
        print("Varfuri:",j,"Densitatea:",i)

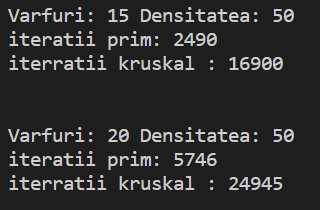
        G = createGraf(j,i)

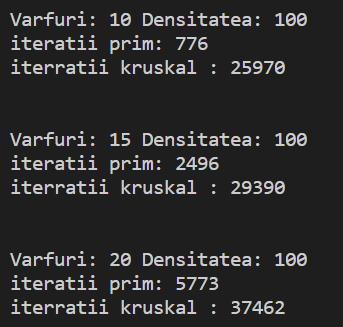
        prim(j,G)

        kruskal(j,G)

**Rezultatele:**







**Concluzii**

Pentru realizarea acestei lucrări de laborator, am studiat tehnica greedy prin elaborarea a doua algoritmi bazate pe aceasta tehnica. Din rezultate vedem diferenta numarului de iteratii exprimat prin numarul de varfuri si densitatea grafului, in toate cazurile algoritmul prim fiind unul mai bun avand un numar cu mult mai mic de iteratii indeosebi la un numar mai mare de varfuri si densitate mai mare.