Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №4**

По дисциплине «Теоретические и множественные основы интеллектуальных систем»

Тема: «Нахождение минимального остовного дерева связанного неориентированного графа»

**Выполнил:**

Студент 1 курса

Группы ИИ-21

Литвинюк Т. В.

**Проверил:**

Глущенко Т. А.

Брест 2022

**Цель работы:** познакомиться с алгоритмом Прима и Краскала.

**Ход работы:**

**Вариант 8**

**Задание 1**

Найти минимальное остовное дерево для заданного графа G алгоритмом Прима и Крускаля.

**Алгоритм Прима:**

from math import inf

def get\_min(R, U):

rm = (inf, -1, -1)

for v in U:

rr = min(R, key=lambda x: x[0] if (x[1] == v or x[2] == v) and (x[1] not in U or x[2] not in U) else inf)

if rm[0] > rr[0]:

rm = rr

return rm

# список ребер графа (длина, вершина 1, вершина 2)

# первое значение возвращается, если нет минимальных ребер

R = [(inf, -1, -1), (7, 'a', 'b'), (4, 'a', 'd'), (5, 'b', 'c'), (9, 'b', 'd'), (9, 'c', 'd'), (13, 'c', 'e'), (10, 'd', 'e'), (11, 'e', 'f')]

N = 6 # число вершин в графе

U = {'a'} # множество соединенных вершин

T = [] # список ребер остова

while len(U) < N:

r = get\_min(R, U) # ребро с минимальным весом

if r[0] == inf: # если ребер нет, то остов построен

break

T.append(r) # добавляем ребро в остов

U.add(r[1]) # добавляем вершины в множество U

U.add(r[2])

print(T)

****

**Алгоритм Краскала:**

R = [(7, 'a', 'b'), (4, 'a', 'd'), (5, 'b', 'c'), (9, 'b', 'd'), (9, 'c', 'd'),

(13, 'c', 'e'), (10, 'd', 'e'), (11, 'e', 'f')]

Rs = sorted(R, key=lambda x: x[0])

U = set() # список соединенных вершин

D = {} # словарь списка изолированных групп вершин

T = [] # список ребер остова

for r in Rs:

if r[1] not in U or r[2] not in U: # проверка для исключения циклов в остове

if r[1] not in U and r[2] not in U: # если обе вершины не соединены, то

D[r[1]] = [r[1], r[2]] # формируем в словаре ключ с номерами вершин

D[r[2]] = D[r[1]] # и связываем их с одним и тем же списком вершин

else: # иначе

if not D.get(r[1]): # если в словаре нет первой вершины, то

D[r[2]].append(r[1]) # добавляем в список первую вершину

D[r[1]] = D[r[2]] # и добавляем ключ с номером первой вершины

else:

D[r[1]].append(r[2]) # иначе, все то же самое делаем со второй вершиной

D[r[2]] = D[r[1]]

T.append(r) # добавляем ребро в остов

U.add(r[1]) # добавляем вершины в множество U

U.add(r[2])

for r in Rs: # проходим по ребрам второй раз и объединяем разрозненные группы вершин

if r[2] not in D[r[1]]: # если вершины принадлежат разным группам, то объединяем

T.append(r) # добавляем ребро в остов

gr1 = D[r[1]]

D[r[1]] += D[r[2]] # объединем списки двух групп вершин

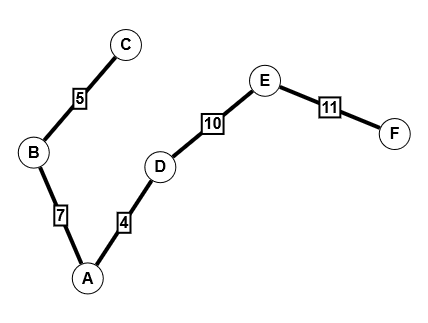
D[r[2]] += gr1

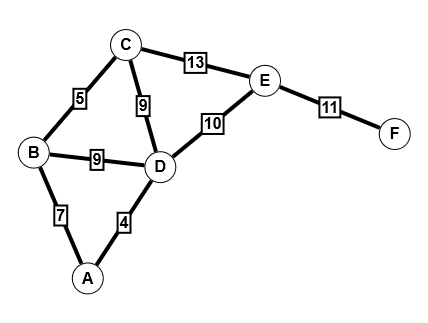
print(T)



**Задание 3**

Графически изобразить граф и его минимальное остовное дерево.





**Вывод:** в ходе лабораторной работы я узнал про деревья и про алгоритмы Прима и Краскала.