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(1)

Кирилович А.А.

Проверил:

Глущенко Т. А.

Брест 2022

**Цель:** изучить алгоритмы для нахождения минимального остовного дерева связанного неориентированного графа.

**Ход работы**

Вариант 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***№*** | ***Кол.***  ***верш.*** | ***Кол.***  ***ребер*** | ***Список ребер*** | ***Веса ребер*** |
| 5. | 5 | 8 | (a,b),(a,c),(a,d),(b,c),  (b,e),(c,d),(c,e),(d,e) | 5,3,6,3,  4,2,5,1 |

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

Алгоритм Крускаля.

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

R = [(5, 1, 2), (3, 1, 3), (6, 1, 4), (3, 2, 3), (4, 2, 5),

     (2, 3, 4), (5, 3, 5), (1, 4, 5)]

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:

            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)



Алгоритм Прима

import math

from unicodedata import name

def get\_min(R, U):

    rm = (math.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 math.inf)

        if rm[0] > rr[0]:

            rm = rr

    return rm

if \_\_name\_\_ == '\_\_main\_\_':

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

    # первое значение возвращается, если нет минимальных ребер

    R = [(math.inf, -1, -1), (5, 1, 2), (3, 1, 3), (6, 1, 4), (3, 2, 3), (4, 2, 5),

         (2, 3, 4), (5, 3, 5), (1, 4, 5)]

    N = 6 # число вершин в графе

    U = {1} # множество соединенных вершин

    T = [] # список ребер остова

    while len(U) < N:

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

        if r[0] == math.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])

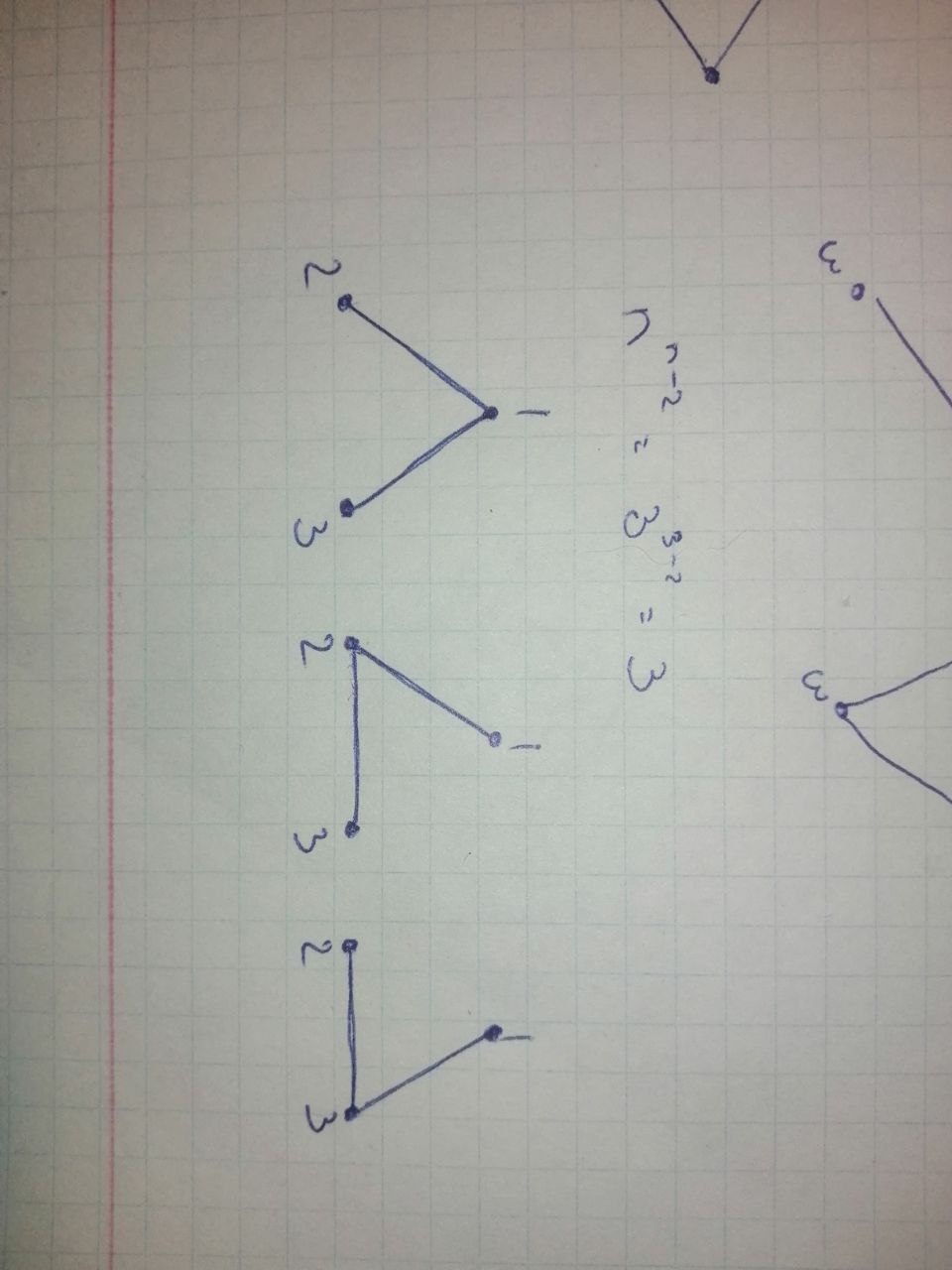
    print(T)



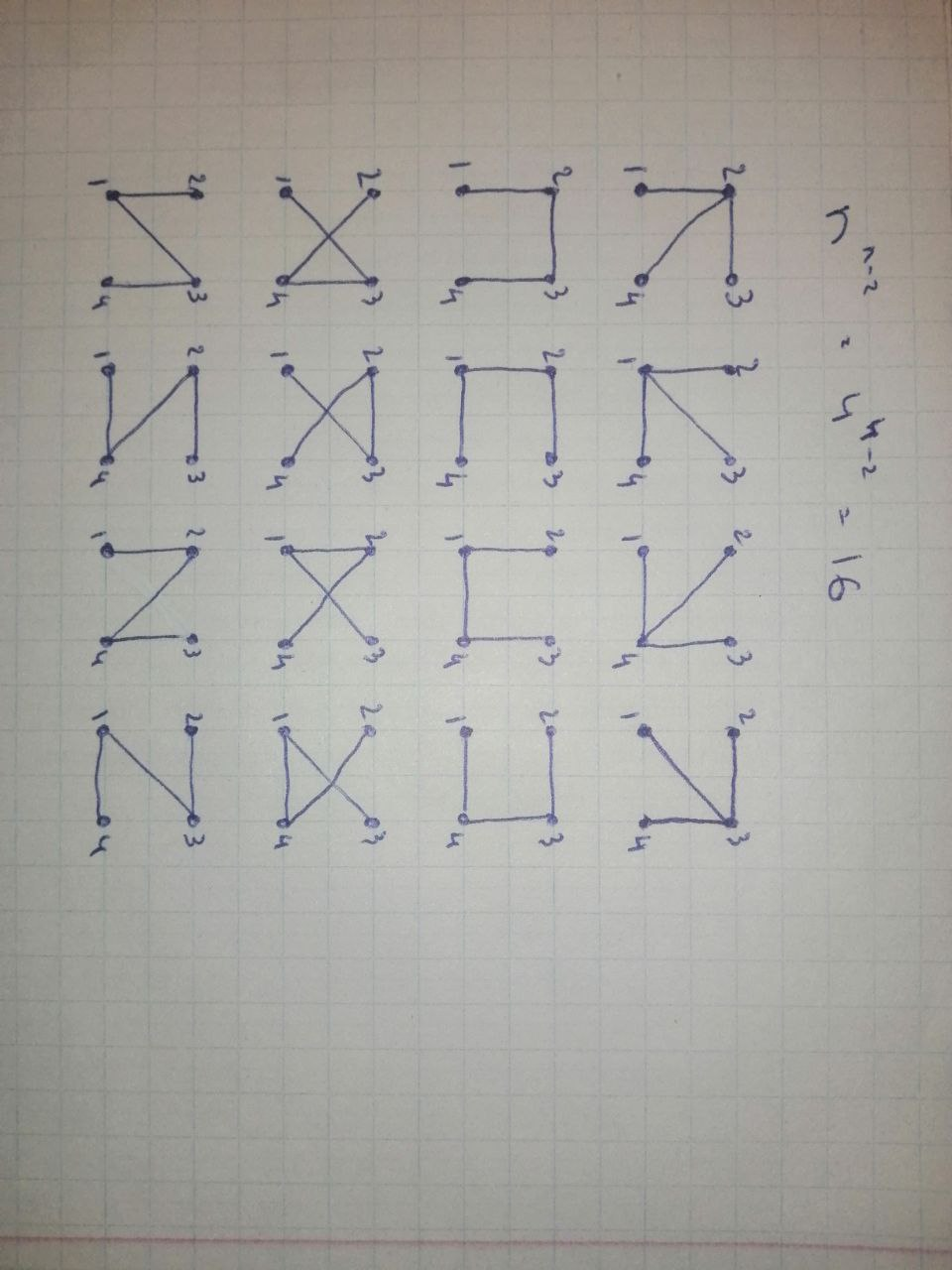
1. Ответить на поставленные вопросы.
2. Для какого графа определяет число остовных деревьев формула *Кэли*.

Ответ: для полного.

1. Подсчитать по формуле *Кэли* и нарисовать число остовных деревьев для *n = 3*.



1. \* пункт 2 выполнить для *n = 4.*



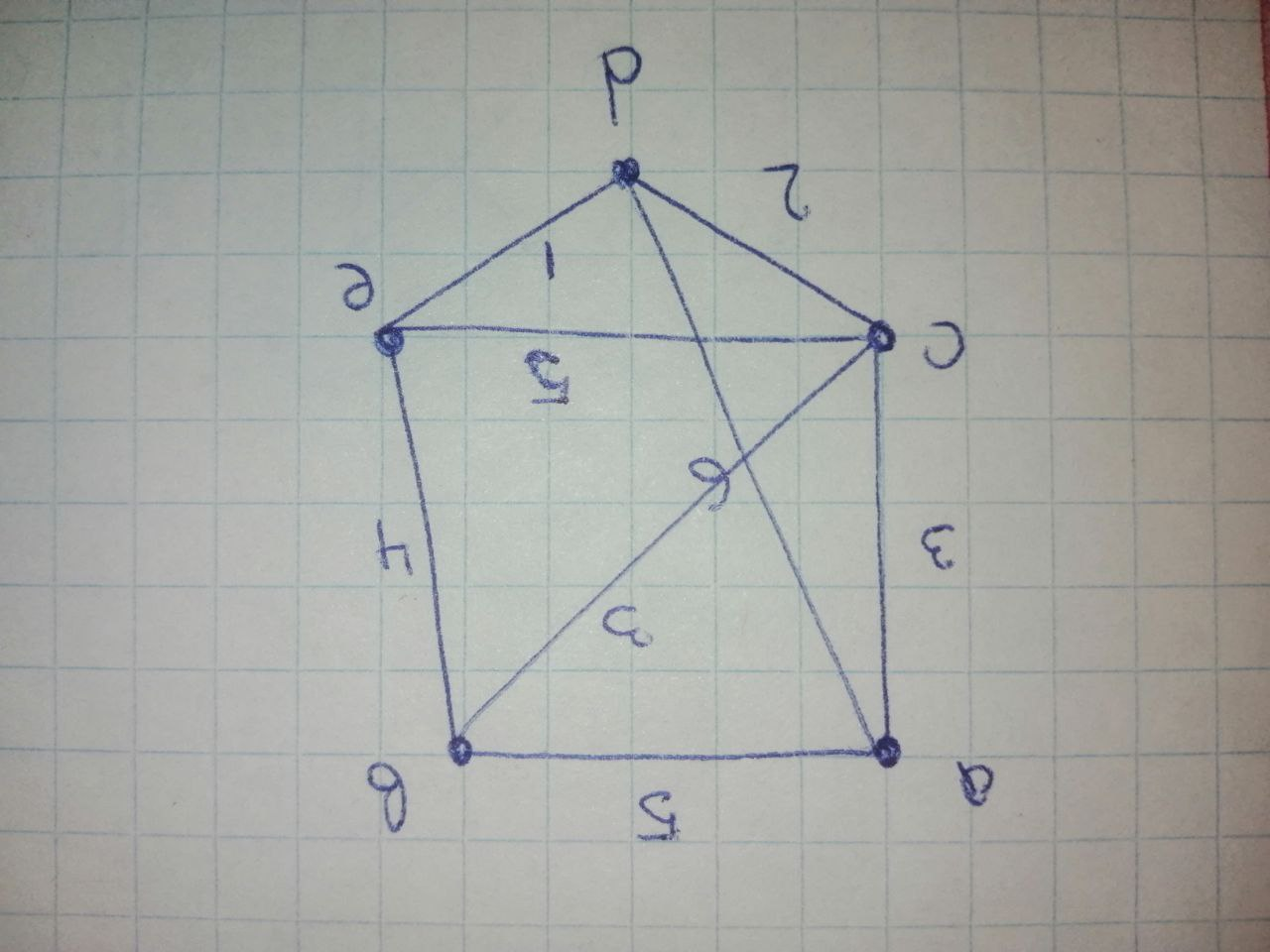
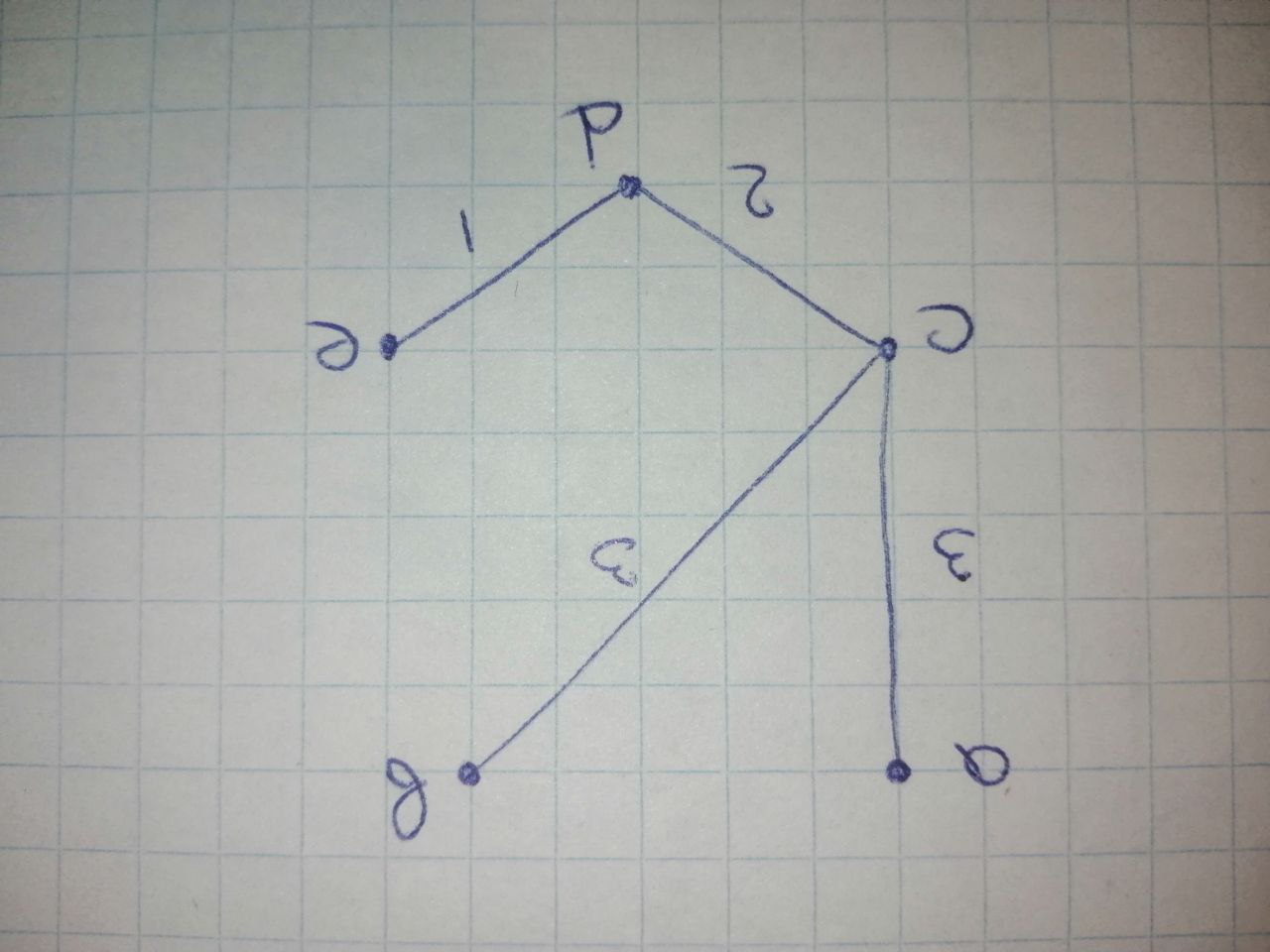
1. Какое остовное дерево находится алгоритмом *Дейкстры*?

Ответ: минимальное.

1. Может ли быть несколько *минимальных* остовных деревьев?

Ответ: да.

1. Графически изобразить граф и его минимальное остовное дерево.

**Вывод:** изучил алгоритмы для нахождения минимального остовного дерева связанного неориентированного графа.