

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ
Кафедра «КОМПЬЮТЕРНЫЕ ТЕХНОЛОГИИ»

Горюнов Ю.Ю., Леонова Т.Ю.

ДИСКРЕТНАЯ МАТЕМАТИКА

ЛАБОРАТОРНЫЙ ПРАКТИКУМ

Пенза, 2018

Оглавление	
Лабораторная работа № 1. Программирование на языке Python в среде Spyder	4
Лабораторная работа № 2. Операции над множествами	5
Задание к лабораторной работе №2.....	5
Рекомендации по программированию	Ошибка! Закладка не определена.
Лабораторная работа № 3. Отношения между множествами. Функции.....	7
Рекомендации по программированию	Ошибка! Закладка не определена.
Лабораторная работа № 4. Элементы комбинаторики	8
Лабораторная работа № 4. Графы и их представление в ЭВМ.....	12
Лабораторная работа № 5. Маршруты, цепи, циклы. Эйлеровы и гамильтоновы графы. Деревья и их свойства	20
Лабораторная работа № 6. Алгоритмы на графах. Алгоритмы Дейкстры, Краскала и Прима	20
Лабораторная работа № 7. Прикладные задачи на графах. Планарность. Раскраски графа.....	20
Приложение 1. Требования к оформлению отчетов по лабораторным работам	22
Содержание отчета	24
Приложение 2. Интегрированная среда Spyder.....	33
Приложение 3. Язык программирования Python.....	37
ПЗ.1 Программа на языке Python	38
ПЗ.2 Встроенные числовые типы (int, float)	39
ПЗ.3 Встроенные типы-последовательности (кортежи, строки, списки, диапазоны)	39
ПЗ.3.1 Кортежи.....	39
ПЗ.3.2 Строки	40
ПЗ.3.3 Списки	40
ПЗ.3.4 Диапазоны.....	41
ПЗ.4 Операторы.....	41
ПЗ.5 Инструкции.....	41
ПЗ.6 Объявление именованных функции пользователя (def).....	42
ПЗ.7 Встроенные модули	Ошибка! Закладка не определена.
ПЗ.7.1 Модуль math (функции вещественного аргумента.).....	Ошибка! Закладка не определена.
ПЗ.7.2 Модуль random (случайные числа)	Ошибка! Закладка не определена.
ПЗ.8 Графические методы.....	42
ПЗ.8.1 Построение графика $y = f(x)$	43
ПЗ.8.2 Построение графика функции $z = f(x, y)$ (поверхности).....	Ошибка! Закладка не определена.
ПЗ.9 Модули пользователя	Ошибка! Закладка не определена.
Литература	57

Лабораторная работа № 1. Программирование на языке Python в системе Spyder

Цель работы

Знакомство с программированием на языке Python в системе Spyder

Задание

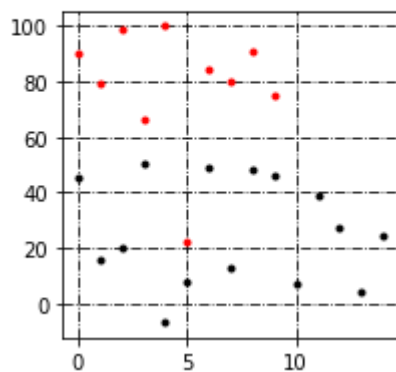
1. Ознакомьтесь с интерфейсом интегрированной среды Spyder (с. 33).
2. Ознакомьтесь и наберите в редакторе Spyder программу, в которой создаются два множества (списка) A и B , элементы которых выводятся на экран и в виде точек на плоскости.

```
import random
random.seed()
def rndSet(k, a, b): # k случайных целых чисел из отрезка [a; b],
    L = []
    for i in range(0, k):
        x = random.randint(a, b)
        while x in L:
            x = random.randint(a, b)
        L += [x]
    return L
IA = 10; A=rndSet(IA, 10, 100) # множество A, |A| = 10,
IB = 15; B=rndSet(IB, -10, 50) # множество B, |B| = 15,
print('A =', A) # вывод элементов множеств на экран,
print('B =', B)
# построение множеств точек:
xA = range(0, IA); xB = range(0, IB) # абсциссы точек для графиков,
import pylab # подключили необходимый модуль,
pylab.figure(figsize = (3, 3)) # размеры графического окна,
pylab.grid(color = 'k', linestyle='-.') # параметры решетки,
pylab.plot(xA, A, 'r.') # строим красным цветом точки множества A,
pylab.plot(xB, B, 'k.') # строим черным цветом точки множества B,
```

3. Получите скриншот с результатом работы программы ()

A = [90, 79, 99, 66, 100, 22, 84, 80, 91, 75]

B = [45, 16, 20, 50, -7, 8, 49, 13, 48, 46, 7, 39, 27, 4, 24]



В отчет включить:

- 1) текст программы,
- 2) скриншот с результатами работы программы,
- 3) выводы.

I Множества, отношения, функции

Лабораторная работа № 2. Операции над множествами

Задание

1. Доказать свойство операций над множествами.
2. Создать программу на языке Python для проверки первого свойства в Вашем варианте.
3. Требования к программе
 - множества интерпретировать списками;
 - мощности множеств вводятся интерактивно;
 - элементы множеств случайным образом выбираются из универсума [0; 100];
 - элементы полученных множеств вывести в консоль и изобразить графически разным цветом;
 - операции над множествами объявить именованными функциями пользователя;
4. В отчет включить:
 - 1) задание,
 - 2) доказательства свойств (доказательства следует набрать в Microsoft Equation.),
 - 3) текста программы с необходимыми комментариями
 - 4) скриншот результата выполнения программы.

Варианты заданий

Вариант 1

- 1) Коммутативность пересечения: $A \cap B = B \cap A$.
- 2) Свойство двойного дополнения: $\overline{\overline{A}} = A$.

Вариант 2

- 1) Ассоциативность объединения: $A \cup (B \cap C) = (A \cup B) \cap C$.
- 2) $A \cup \overline{A} = U$

Вариант 3

- 1) Ассоциативность пересечения: $A \cap (B \cap C) = (A \cap B) \cap C$.
- 2) $A \cup \emptyset = A$.

Вариант 4

- 1) Дистрибутивность объединения относительно пересечения:

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C).$$
- 2) $A \cap \overline{A} = \emptyset$.

Вариант 5

- 1) Дистрибутивность пересечения относительно объединения:

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C).$$
- 2) Свойство поглощения: $A \cap (A \cup B) = A$.

Вариант 6

- 1) Свойство: $A \cup U = U$.
- 2) Свойство Порецкого: $A \cup (\overline{A} \cap B) = A \cup B$.

Вариант 7

- 1) $A \cap U = A$
- 2) Свойство де Моргана: $\overline{A \cap B} = \overline{A} \cup \overline{B}$.

Вариант 8

- 1) Идемпотентность объединения: $A \cup A = A$.
- 2) Свойство склеивания: $(A \cup B) \cap (A \cup \overline{B}) = A$.

Вариант 9

- 1) Свойство де Моргана: $\overline{A \cup B} = \overline{A} \cap \overline{B}$.
- 2) Идемпотентность пересечения $A \cap A = A$.

Вариант 10

- 1) Свойство поглощения: $A \cup (A \cap B) = A$.
- 2) Свойство Порецкого: $A \cap (\overline{A} \cup B) = A \cap B$.

Вариант 11

- 1) Свойство склеивания: $(A \cap B) \cup (A \cap \overline{B}) = A$.
- 2) Свойство: $A \cap \overline{A} = \emptyset$

Лабораторная работа № 3. Задачи перебора. Код Грея

Задание

1. Создать программу на языке Python для решения задачи Вашего варианта.
2. Требования к программе
 - множество A интерпретировать списком;
 - $|A| = 5$;
 - множество A инициализировать случайными целыми числами из универсума $[-10; 10]$;
 - для получения всех подмножеств использовать функции `nextGr`, `kSubsets`, либо объявить свои функции.
3. В отчет включить:
 - 1) задание,
 - 2) текста программы с необходимыми комментариями
 - 3) скриншот результата выполнения программы.

Варианты заданий

1. Во множестве A найти три элемента, сумма которых наибольшая.
2. Во множестве A найти четыре элемента, модуль произведения которых наименьший.
3. У каждого элемента множества A есть вес (натуральное число). Найти во множестве A три элемента с наименьшим весом.
4. У каждого элемента множества A есть вес (натуральное число) и цена (натуральное число). Найти во множестве A три элемента с наименьшим весом, цена которых наибольшая.
5. Найти все разбиения множества A на два подмножества B и C , $|B| = 3$.
6. Найти все разбиения множества A на два подмножества B и C , $|B| = 4$.
7. Найти все разбиения множества A на три подмножества B , C и D , $|B| = 2$, $|C| = 3$.
8. Найти все разбиения множества A на три подмножества B , C и D , $|B| = 3$, $|C| = 4$.
9. Найти все разбиения множества A на два подмножества B и C такие, что сумма элементов множества B больше суммы элементов множества C , либо сообщить о их отсутствии.
10. Найти все разбиения множества A на два подмножества B и C такие, что сумма элементов множества B больше произведения элементов множества C , либо сообщить о их отсутствии.

Лабораторная работа № 4. Отношения между множествами

Задание

1. Сформулировать определение отношения между элементами множеств A_1, \dots, A_k .

2. Сформулировать определение бинарного отношения между элементами множеств A и B .
3. Сформулировать определения операций над бинарными отношениями: $\delta \cup \rho$, $\delta \cap \rho$, δ / ρ , $\delta \Delta \rho$, δ^{-1} , $\delta \circ \rho$.
4. Сформулировать определения рефлексивного, симметричного, антисимметричного и транзитивного бинарных отношений.
5. Сформулировать определения отношений строго, нестрого порядков, отношения эквивалентности.
6. Сформулировать определение фактор-множества множества A относительно отношения эквивалентности.
7. Задать множества: $A \subset \mathbb{Z}$, $|A| = 4$, $B \subset \{ 'a', 'b', \dots, 'z' \}$, $|B| = 3$, $C \subset \{ 'I', 'II', 'III', \dots, 'X' \}$, $|C| = 4$.
8. Найти $A \times B$.
9. Выбрать три бинарных отношения: $\delta \subset A \times B$, $|\delta| = 4$; $\rho \subset A \times B$, $|\rho| = 4$, $\sigma \subset B \times C$, $|\sigma| = 3$.
10. Изобразить графически бинарные отношения δ , ρ , σ .
11. Вычислить и изобразить графически: $\delta \cup \rho$, $\delta \cap \rho$, δ / ρ , $\delta \Delta \rho$, δ^{-1} , $\delta \circ \sigma$.
12. Проверить выполнимость свойств бинарных отношений:
 - a) $(\delta^{-1})^{-1} = \delta$,
 - b) $(\delta \circ \rho)^{-1} = \rho^{-1} \circ \delta^{-1}$.
13. Выбрать из $A \times A$ четыре бинарных отношения: только рефлексивное, только симметричное, только транзитивное и отношение эквивалентности τ .
14. Найти A/τ .

Графические изображения следует выполнять в MS Visio.

Лабораторная работа № 5. Функции

Задание

1. Сформулировать определения: функции, области определения и области значений функции.
2. Сформулировать определения: инъективной, сюръективной и биективной функций.
3. Задать множества: $A \subset \mathbb{Z}$, $|A| = 4$, $B \subset \{ a, b, c, \dots, z \}$, $|B| = 3$, $C \subset \{ I, II, III, \dots, X \}$, $|C| = 4$.
4. Выбрать и графически изобразить три функции:
 - инъективную, но не сюръективную функцию $f_1 : A \rightarrow B$, $|f_1| \geq 7$,
 - сюръективную, но не инъективную функцию $f_2 : A \rightarrow B$, $|f_2| \geq 7$,

- биективную функцию $g : B \rightarrow C$.

5. Найти и графически изобразить композиции функций: $f_1 \circ g$ и $f_2 \circ g$.

Графические изображения следует выполнять в MS Visio.

Лабораторная работа № 6. Матрицы бинарных отношений

Задание

1. Сформулировать определение матрицы бинарного отношения.
2. Создать программу для представления бинарных отношений матрицами.

Варианты заданий (требование к программе)

1. В тексте программы в виде списков задать множества: $A \subset \mathbb{Z}$, $|A| = 5$, $B \subset \{'a', 'b', \dots, 'z'\}$, $|B| = 5$.
2. В тексте программы в виде списка списков задать бинарные отношения: $\delta \subset A \times B$, $|\delta| = 5$, $\rho \subset A \times B$, $|\rho| = 5$, графически их изобразить (в MS Visio).
3. Вывести на экран матрицы:

Варианты 1-2. M_δ , M_ρ , $M_{\delta \cup \rho}$.

Варианты 3-4. M_δ , M_ρ , $M_{\delta \cap \rho}$.

Варианты 5-6. M_δ , M_ρ , $M_{\delta/\rho}$.

Варианты 7-8. M_δ , M_ρ , $M_{\delta/\rho}$.

4. Варианты 1-10: программно проверить выполнение равенства $M_{\delta \cap \delta^{-1}} = M_\delta \cdot M_\delta^T$.

В отчет включить:

- 1) определение матрицы бинарного отношения,
- 2) текст программы,
- 3) скриншот с результатами выполнения программы.

II Комбинаторика

Лабораторная работа № 7. Основные комбинаторные конфигурации

Задание

1. Решить задачи Вашего варианта, не вычисляя результатов.
2. Для вычисления результата создать программу на языке Python, объявив самостоятельно необходимые функции, либо воспользоваться объявлениями на с. 49; если входные данные не числовые, то самостоятельно присвоить им достаточно большие значения.

В отчет включить:

- 4) определения использованных комбинаторных конфигураций,
- 5) решения задач,

- б) текст программы,
- 7) скриншот с результатами выполнения программы.

Варианты заданий

1. Сколькими способами можно распределить три билета среди 20 студентов, если:
 - а) распределяются билеты в разные театры, а каждый студент может получить не более одного билета;
 - б) распределяются билеты в разные театры и на разные дни, а каждый студент может получить любое (не превышающее трех) число билетов?
2. Выяснить, сколькими способами можно выстроить девять человек:
 - а) в колонну по одному;
 - б) колонну по три, если в каждой шеренге люди выстраиваются по росту и нет людей одинакового роста?
3. Найти число векторов $\vec{a} = (a_1, \dots, a_n)$, координаты которых удовлетворяют условиям:
 - а) $a_i \in \{0, 1, \dots, k-1\}$ ($i = 1, 2, \dots, n$);
 - б) $a_i \in \{0, 1\}$ ($i = 1, 2, \dots, n$) и $a_1 + \dots + a_n = r$.
4. Каково число матриц из m строк и k столбцов с элементами из множества $\{0, 1\}$, если строки матриц:
 - а) не обязательно попарно различны;
 - б) попарно различны?
5.
 - а) Дано m предметов одного сорта и k другого. Найти число выборок, составленных из r предметов одного сорта и s предметов другого сорта.
 - б) Из n букв, среди которых a встречается α раз, буква b встречается β раз, а остальные буквы попарно различны, составляются слова. Сколько среди них будет различных r -буквенных слов, содержащих h раз букву a и k раз букву b ?
6. Имеется колода из $4n$ ($n \geq 5$) карт, которая содержит карты четырех мастей по n карт каждой масти, пронумерованных числами $1, 2, \dots, n$. Подсчитать, сколькими способами можно выбрать пять карт так, что среди них окажутся:
 - а) пять последовательных карт одной масти;
 - б) три карты с одним номером и две карты с другим?
7. Имеется колода из $4n$ ($n \geq 5$) карт, которая содержит карты четырех мастей по n карт каждой масти, пронумерованных числами $1, 2, \dots, n$. Подсчитать, сколькими способами можно выбрать пять карт так, что среди них окажутся:
 - а) пять карт какой-нибудь одной масти;

- b) пять последовательно занумерованных карт?
8. Имеется колода из $4n$ ($n \geq 5$) карт, которая содержит карты четырех мастей по n карт каждой масти, занумерованных числами $1, 2, \dots, n$. Подсчитать, сколькими способами можно выбрать пять карт так, что среди них окажутся:
- в точности три карты из пяти с одним и тем же номером;
 - не более двух карт каждой масти.
- 9.
- Сколькими способами из 28 костей домино можно выбрать две кости так, чтобы их можно было приложить друг к другу (т.е. чтобы некоторое одинаковое число очков встретилось на обеих костях)?
 - Бросают три игральные кости. Сколькими способами они могут упасть так, что все оказавшиеся сверху грани либо одинаковы, либо попарно различны?
- 10.
- Сколькими способами можно число n представить в виде суммы k натуральных слагаемых? (Представления, различающиеся лишь порядком слагаемых, считаются разными.)
 - Сколькими способами число 7^n можно представить в виде трех сомножителей? (Представления, различающиеся лишь порядком сомножителей, считаются разными.)
- 11.
- Сколькими способами можно расставить n нулей и k единиц так, чтобы между любыми двумя единицами находилось не менее m нулей?
 - Сколько существует неотрицательных целых чисел, не превышающих 10^n , цифры которых расположены в неубывающем порядке?
12. Пусть $n = p_1^{a_1} \dots p_k^{a_k}$ - разложение числа n в произведение простых попарно различных чисел. Найти:
- число всех делителей, не делящихся на квадрат никакого целого числа, отличного от 1;
 - сумму делителей числа n .

Лабораторная работа № 8. Числа Стирлинга второго рода



Джеймс Стирлинг (1692 - 1770)
шотландский математик.

Задание

1. Сформулировать определение чисел Стирлинга второго рода $S_2 \binom{k}{n}$.

2. Доказать теорему

$$S_2 \binom{k}{n} = k S_2 \binom{k-1}{n-1} + S_2 \binom{k-1}{n}.$$

3. Используя явную формулу:

$$S_2 \binom{k}{n} = \frac{1}{k!} \sum_{j=0}^k (-1)^{k+j} C_k^j j^n,$$

создать программу на языке Python для получения следующего треугольника чисел

Стирлинга второго рода $S_2 \binom{k}{n}$

$n \backslash k$	0	1	2	3	4	5	6
0	1	0	0	0	0	0	0
1	0	1	0	0	0	0	0
2	0	1	1	0	0	0	0
3	0	1	3	1	0	0	0
4	0	1	7	6	1	0	0
5	0	1	15	25	10	1	0
6	0	1	31	90	65	15	1

В отчет включить:

- 1) определение чисел Стирлинга второго рода,
- 2) доказательство теоремы,
- 3) текст программы,
- 4) скриншот с результатами выполнения программы.

Лабораторная работа № 9. Числа Стирлинга первого рода (без знака)

Задание

1. Сформулировать определение чисел Стирлинга второго рода $S_1 \binom{k}{n}$.

2. Доказать теорему

$$S_1 \binom{k}{n} = (n-1) S_1 \binom{k}{n-1} + S_1 \binom{k-1}{n-1}.$$

3. Доказать теорему

$$S_1 \binom{k}{n} = n! S_2 \binom{k}{n}.$$

4. Используя доказанную в п. 3 теорему и явную формулу для чисел Стирлинга второго рода, создать программу на языке Python для получения следующего треугольника чисел Стирлинга второго рода $S_1 \binom{k}{n}$:

$n \backslash k$	0	1	2	3	4	5	6	7	8	9
0	1									
1	0	1								
2	0	1	1							
3	0	2	3	1						
4	0	6	11	6	1					
5	0	24	50	35	10	1				
6	0	120	274	225	85	15	1			
7	0	720	1764	1624	735	175	21	1		
8	0	5040	13068	13132	6769	1960	322	28	1	
9	0	40320	109584	118124	67284	22449	4536	546	36	1

В отчет включить:

- 1) определение чисел Стирлинга первого рода,
- 2) доказательства теорем,
- 3) текст программы,
- 4) скриншот с результатами выполнения программы.

Лабораторная работа № 10. Числа Каталана



Каталан, Эжен Шарль (1814 - 1894)
бельгийский математик.

Задание

1. Сформулировать определение чисел Каталана C_n .
2. Доказать теорему

$$C_n = \sum_{k=0}^{n-1} C_k C_{n-1-k}, \quad C_0 \stackrel{\text{def}}{=} 1.$$

3. Используя явную формулу для чисел Каталана

$$C_n = \frac{1}{n+1} C_{2n}^n = \frac{1}{2n+1} C_{2n+1}^n = C_{2n}^n - C_{2n}^{n-1},$$

создать программу на языке Python для табулирования функции C_n и построения ее точечного графика на отрезке $[1; 20]$ с шагом 1.

В отчет включить:

- 1) определение чисел Каталана,

- 2) доказательство теоремы,
- 3) текст программы,
- 4) скриншот с результатами выполнения программы.

Лабораторная работа № 11. Числа Бернулли



Бернулли Якоб (1654 - 1705)

швейцарский математик, один из основателей теории вероятностей и математического анализа.

Задание

1. Сформулировать определение чисел Бернулли B_n .
2. Используя рекуррентное соотношение

$$B_n = \frac{-1}{n+1} \sum_{k=1}^n C_{n+1}^{k+1} B_{n-k}, B_0 = 1,$$

создать программу на языке Python для табулирования функции B_n и построения ее точечного графика на отрезке $[1.0; 15.0]$ с шагом 1.0.

Примечание. Первые 15 чисел Бернулли для тестирования

$B_0 = 1$	$B_4 = -\frac{1}{30}$	$B_8 = -\frac{1}{30}$	$B_{12} = -\frac{691}{2730}$
$B_1 = -\frac{1}{2}$	$B_5 = 0$	$B_9 = 0$	$B_{13} = 0$
$B_2 = \frac{1}{6}$	$B_6 = \frac{1}{42}$	$B_{10} = \frac{5}{66}$	$B_{14} = 1\frac{1}{6}$
$B_3 = 0$	$B_7 = 0$	$B_{11} = 0$	$B_{15} = 0$

В отчет включить:

- 1) определение чисел Бернулли,
- 2) текст программы,
- 3) скриншот с результатами выполнения программы.

Лабораторная работа № 12. Числа Фибоначчи



Пизанский Леонардо (1170 – 1250)

первый крупный математик средневековой Европы, наиболее известен под прозвищем Фибоначчи (сын Боначчи).

Задание

1. Сформулировать определение чисел Фибоначчи f_n .
2. Создать программу на языке Python в которой объявить две функции:

- 1) fbR(n), возвращающей n -е число Фибоначчи на основе рекурсивного определения f_n ;
- 2) fbBM(n), возвращающей n -е число Фибоначчи на основе формулы де Муавра-Бине

$$f_n = \frac{1}{\sqrt{5}} \left(\left(\frac{1+\sqrt{5}}{2} \right)^n - \left(\frac{1-\sqrt{5}}{2} \right)^n \right),$$

для табулирования функций fbR(n) и fbBM(n), и построения их точечных графиков в одной системе координат на отрезке [1; 30] с шагом 1.

В отчет включить:

- 1) определение чисел Фибоначчи,
- 2) текст программы,
- 3) скриншот с результатами выполнения программы.

Лабораторная работа № 13. Числа Эйлера первого рода



Эйлер Леонард (1707 - 1783)

швейцарский, немецкий и российский математик и механик, внесший фундаментальный вклад в развитие этих наук.

Задание

1. Сформулировать определение чисел Эйлера первого рода $E_1 \binom{k}{n}$.

2. Доказать теорему

$$E_1 \binom{k}{n} = (k+1) E_1 \binom{k}{n-1} + (n-k) E_1 \binom{k-1}{n-1}, \quad E_1 \binom{0}{0} \stackrel{def}{=} 1, \quad E_1 \binom{k}{0} \stackrel{def}{=} 0 \text{ при } k > 0.$$

3. Используя явную формулу для чисел Эйлера первого рода

$$E_1 \binom{k}{n} = \sum_{m=0}^k (-1)^m (k+1-m)^n C_{n+1}^m,$$

создать программу на языке Python для получения следующего треугольника чисел Эйлера первого рода

$n \backslash k$	0	1	2	3	4	5	6	7	8	9
0	1									
1	1	0								
2	1	1	0							
3	1	4	1	0						
4	1	11	11	1	0					
5	1	26	66	26	1	0				
6	1	57	302	302	57	1	0			
7	1	120	1191	2416	1191	120	1	0		
8	1	247	4293	15619	15619	4293	247	1	0	

В отчет включить:

- 1) определение чисел Эйлера первого рода,
- 2) доказательство теоремы,
- 3) текст программы,
- 4) скриншот с результатами выполнения программы.

II.1 Методы комбинаторики**Лабораторная работа № 14. Свойства биномиальных коэффициентов****Задание**

1. Используя формулы для количеств комбинаторных конфигураций, либо метод математической индукции, доказать свойство биномиальных коэффициентов Вашего варианта.

2. Создать программу на языке Python для проверки доказанного свойства:

✓ варианты 1-7 вычислить левые и правые части равенств для:

1. $n = 10, k = 1, 2, 3, \dots, 9;$

2. $n = 10, p = 7, m = 1, 2, 3, \dots, 6;$

3. $n = 10, k = 8, r = 1, 2, 3, \dots, 7;$

4, 5. $n = 10, k = 8;$

6. $n = 9, k = 1, 2, \dots, 9;$

7. $n = 10, k = 4;$

✓ вариант 8: построить точечный график для $n = 10, k = 1, 2, \dots, 10.$

✓ вариант 9: построить точечный график для $n = 10, k = 7, r = 1, 2, \dots, 6.$

✓ вариант 10:

a) для $p = 11$ (простое) и $k = 1, 2, \dots, p - 1$ вывести на экран пары чисел $C_p^k, C_p^k \% p,$

b) для $p = 12$ (составное) для $k = 1, 2, \dots, p - 1$ вывести на экран пары чисел $C_p^k, C_p^k \% p,$

где $a \% b$ – остаток от деления a на $b.$

В отчет включить:

- 1) доказательство свойства,
- 2) текст программы,
- 3) скриншот с результатами выполнения программы.

Варианты заданий

- | | |
|---|---|
| 1. $C_n^k = C_n^{n-k}$.
2. $C_n^p C_p^m = C_n^m C_{n-m}^{p-m}$.
3. $\frac{C_n^{k-r}}{C_n^k} = \frac{A_k^r}{A_{n-k+r}^r}$.
4. $C_n^k = \sum_{r=0}^n C_{n-r-1}^{k-r}$.
5. $\frac{C_{n-r}^{k-r}}{C_n^k} = \frac{A_k^r}{A_n^r}$.
6. $\frac{C_{n+1}^k}{C_n^k} = \frac{n+1}{n-k+1}$. | 7. $\sum_{r=k}^n C_r^k = C_{n+1}^{k+1}$.
8. C_n^k возрастает по n при фиксированном k .
9. C_{n-r}^{k-r} убывает по r при фиксированных n и k .
10. $\max_{0 \leq k \leq n} C_n^k = C_n^{\lceil n/2 \rceil}$, где $\lceil x \rceil$ - наибольшее целое, не превосходящее x .
11. При простом p и любом $k, p > k \geq 1$, число C_p^k кратно p . |
|---|---|

Лабораторная работа № 15. Полиномиальная и биномиальная теоремы

Задание

- Используя биномиальную теорему (бином Ньютона) решить первую задачу Вашего варианта.
- Используя полиномиальную теорему решить вторую задачу Вашего варианта.

В отчет включить:

- формулировку биномиальной теоремы;
- решение первой задачи;
- формулировку полиномиальной теоремы;
- решение второй задачи.

Варианты заданий

- Определить, содержит ли разложение $(a+b)^n$ рациональные члены, если содержит, то подсчитать их количество и найти номер k и значение наибольшего коэффициента C_n^k рациональных членов:

1в. $a = \sqrt{2}, b = \sqrt[3]{3}, n = 20$.	6в. $a = 2\sqrt{2}, b = \sqrt[4]{5}, n = 24$.
2в. $a = \sqrt{3}, b = \sqrt[4]{5}, n = 50$.	7в. $a = \sqrt{6}, b = \sqrt[4]{15}, n = 30$.
3в. $a = \sqrt[3]{6}, b = \sqrt[4]{2}, n = 100$.	8в. $a = \sqrt[3]{9}, b = \sqrt[4]{12}, n = 50$.
4в. $a = \sqrt[3]{12}, b = \sqrt[6]{3}, n = 30$.	9в. $a = \sqrt[3]{2}, b = \sqrt[6]{6}, n = 20$.
5в. $a = \sqrt[3]{6}, b = \sqrt[4]{2}, n = 100$.	10в. $a = \sqrt[3]{9}, b = \sqrt[4]{12}, n = 100$.

- Найти коэффициент при t^k в разложении для данного k , либо доказать их отсутствие.

1в. $(1+2t-3t^2)^8, k=9.$	6в. $(1+4t^3-3t^4)^8, k=9.$
2в. $(1-t+2t^2)^{10}, k=7.$	7в. $(1+3t-3t^3)^{10}, k=9.$
3в. $(2+t-2t^3)^{10}, k=5.$	8в. $(1-2t-3t^3)^7, k=5.$
4в. $(2+t^4+t^7)^{15}, k=17.$	9в. $(1+2t^2-5t^2)^{10}, k=8.$
5в. $(1+2t-3t^2)^8, k=9.$	10в. $(1+t-3t^3)^9, k=7.$

Лабораторная работа № 16. Формула включения-исключения

Задание. Используя формулу включения-исключения решить задачу Вашего варианта.

В отчет включить:

- 1) формулировку формулы включения-исключения;
- 2) решение первой задачи.

Варианты заданий

1. Найти число целых положительных чисел, не превосходящих 1000 и не делящихся ни на одно из чисел 3, 5 и 7.
2. Найти число целых положительных чисел, не превосходящих 1000 и не делящихся ни на одно из чисел 6, 10 и 15.
3. Показать, что если $n = 30m$, то количество целых положительных чисел, не превосходящих n и не делящихся ни на одно из чисел 6, 10, 15, равно $22m$.
4. Найти число пар (X, Y) таких подмножеств множества $U, |U| \geq 3$, что $X \cap Y = \emptyset$.
5. Найти число пар (X, Y) таких подмножеств множества $U, |U| \geq 3$, что $|X \Delta Y| = 1$.
6. Найти число троек (X, Y, Z) , таких подмножеств множества $U, |U| \geq 3$,
что $X \cup (Y \cap \bar{Z}) = \bar{X} \cup \bar{Y}$.
7. Найти число пар (X, Y) таких подмножеств множества $U, |U| \geq 3$,
что $X \cap Y = \emptyset, |X| \geq 2, |Y| \geq 3$.
8. Найти число пар (X, Y) таких подмножеств множества $U, |U| \geq 3$,
что $|X \Delta Y| = 1, |X| \geq 2, |Y| \geq 2$.
9. Найти число троек (X, Y, Z) , таких подмножеств множества $U, |U| \geq 3$,
что $X \cup (Y \cap \bar{Z}) = \bar{X} \cup \bar{Y}, |X| \geq 1, |Y| \geq 1, |Z| \leq 1$.
10. *Задача о супружеских парах.* Сколькими способами можно расположить за круглым столом шесть супружеских пар так, чтобы мужчины и женщины чередовались и никакие двое супругов не сидели рядом?

Лабораторная работа № 17. Производящие функции

Задание

1. Используя производящие функции, доказать свойство биномиальных коэффициентов Вашего варианта.
2. Создать программу на языке Python для проверки доказанного свойства для трех значений n : 10, 25, 50.

В отчет включить:

- 1) определение производящей функции;
- 2) доказательство свойства;
- 3) текст программы,
- 4) скриншот с результатами выполнения программы

Варианты заданий

- | | |
|---|--|
| 1. $\sum_{0 \leq k \leq \frac{n}{2}} C_n^{2k} = 2^{n-1}.$ | 6. $\sum_{k=0}^n k C_n^k = n 2^{n-1}.$ |
| 2. $\sum_{k=0}^n k(k-1) C_n^k = n(n-1) 2^{n-2}.$ | 7. $\sum_{k=0}^n \frac{1}{k+1} C_n^k = \frac{1}{n+1} (2^{n+1} - 1).$ |
| 3. $\sum_{k=0}^n (2k+1) C_n^k = (n+1) 2^n.$ | 8. $\sum_{k=0}^n (-1)^k \frac{1}{k+1} C_n^k = \frac{1}{n+1}.$ |
| 4. $\sum_{k=1}^n \frac{(-1)^{k-1}}{k} C_n^k = 1 + \frac{1}{2} + \frac{1}{3} + \dots + \frac{1}{n}.$ | 9. $\sum_{k=0}^n (C_n^k)^2 = C_{2n}^n.$ |
| 5. $\sum_{\substack{k \geq 0 \\ 2k+1 \leq n}} C_n^{2k} = 2^{n-1}.$ | 10. $\sum_{k=0}^n \frac{(2n)!}{(k!)^2 ((n-k)!)^2} = (C_{2n}^n)^2.$ |

Лабораторная работа № 18. Решение рекуррентных соотношений

Задание

1. Решить рекуррентное соотношение Вашего варианта, используя производящие функции.
2. Создать программу на языке Python для проверки найденного решения $A(n)$:
 - объявить рекурсивную функцию для вычисления a_n , используя заданное рекуррентное соотношение;
 - вывести на экран для $n = 30, 31, \dots, 40$ пары чисел $(a_n, A(n))$.

В отчет включить:

- 1) схему решения рекуррентных соотношений, используя производящие функции;
- 2) решение рекуррентного соотношения;
- 3) текст программы,
- 4) скриншот с результатами выполнения программы

Варианты заданий

1. $a_n = a_{n-1} + n, a_1 = 1.$
2. $a_n = -2a_{n-1} + 8a_{n-2} + 27 \cdot 5^n, a_1 = 0, a_2 = -9.$
3. $a_n = 2a_{n-1} - 2a_{n-2} + 2^n, a_1 = 1, a_2 = 2.$
4. $a_n = -a_{n-1} + 2a_{n-2} + n, a_1 = 1, a_2 = -2.$
5. $a_n = 4a_{n-1} - 4a_{n-2} + 2^n, a_1 = 1, a_2 = 2.$
6. $a_n = -a_{n-1} + 6a_{n-2} + 5 \cdot 2^{n+1}, a_1 = 2, a_2 = 1.$
7. $a_n = 4a_{n-1} - 3a_{n-2}, a_1 = 10, a_2 = 16.$
8. $a_n = 3a_{n-1} - a_{n-2} - 3a_{n-3}, a_1 = 3, a_2 = 7, a_3 = 27.$
9. $a_n = 3a_{n-1} - 2a_{n-2}, a_1 = a, a_2 = b, a_3 = c.$
10. $a_n = 2 \cos \alpha a_{n-1} + a_{n-2}, a_1 = 1, a_2 = \cos \alpha.$

Лабораторная работа № 4. Графы и их представление в ЭВМ

Лабораторная работа № 5. Маршруты, цепи, циклы. Эйлеровы и гамильтоновы графы. Деревья и их свойства

Лабораторная работа № 6. Алгоритмы на графах. Алгоритмы Дейкстры, Краскала и Прима

Лабораторная работа № 7. Прикладные задачи на графах. Планарность. Раскраски графа

Приложение 1. Требования к оформлению отчетов по лабораторным работам

Отчет выполняется на бумажном носителе в соответствии с приведенным в приложении 2 образцом. Каждый раздел отчета должен содержать заголовок, страницы должны быть пронумерованы.

Параметры форматирования:

Размер бумаги – А4.

Поля: левое – 2 см., правое – 1 см, верхнее – 2 см., нижнее – 2 см.

Тип шрифта: Times New Roman.

Размер шрифта – «14».

Междустрочный интервал – «одинарный».

Абзацный отступ – 1 см.

Выравнивание: по ширине.

Требования к заголовкам. Текст заголовка должен быть выделен относительно основного текста, например, выполнен в полужирном стиле, также отделен от основного текста пустой строкой.

Формулы следует набирать в Microsoft Equation 3.0 или аналогичных программных продуктах.

Блок-схемы программ выполняются в специализированных программных продуктах, например, Microsoft Visio.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет «Вычислительная техника»

Кафедра «Компьютерные технологии»

Направление подготовки 01.03.02 Прикладная математика и информатика

Операции над множествами

(Наименование выполняемой работы)

по дисциплине «Дискретная математика»

ОТЧЕТ

по лабораторной работе № номер

(Обозначение документа)

Студент
гр. 14ВГ1

(Подпись)

Петров П.П.
(Фамилия И.О.)

Преподаватель
к.ф.-м.н, доцент

(Подпись)

Горюнов Ю.Ю.
(Фамилия И.О.)

ПЕНЗА, 2017

П1.1 Содержание отчета по лабораторной работе № 2

Цель работы

Знакомство с операциями над множествами и их программной реализации на языке Python.

Задание 1. Доказать свойство коммутативности объединения $A \cup B = B \cup A$.

Доказательство. По определению равенства двух множеств необходимо доказать два включения: $A \cup B \subset B \cup A$ и $B \cup A \subset A \cup B$.

1) Доказательство $A \cup B \subset B \cup A$.

$$\forall x \in A \cup B \xRightarrow{\text{опр. } \cup} \underset{p}{x \in A} \vee \underset{q}{x \in B} \xRightarrow{p \vee q = q \vee p} x \in B \vee x \in A \xRightarrow{\text{опр. } \cup} x \in B \cup A,$$

следовательно, по определению включения: $A \cup B \subset B \cup A$.

2) Доказательство $B \cup A \subset A \cup B$.

$$\forall x \in B \cup A \xRightarrow{\text{опр. } \cup} \underset{p}{x \in B} \vee \underset{q}{x \in A} \xRightarrow{p \vee q = q \vee p} x \in A \vee x \in B \xRightarrow{\text{опр. } \cup} x \in A \cup B,$$

следовательно, по определению включения: $B \cup A \subset A \cup B$.

Доказанные два включения доказывают $A \cup B = B \cup A$.

Задание 2. Программа для проверки свойства $A \cup B = B \cup A$.

```
import random
random.seed()
def rndSet(k, a, b): # список из k случайных целых чисел из отрезка [a; b],
    L = []
    for i in range(0, k):
        x = random.randint(a, b)
        while x in L:
            x = random.randint(a, b)
        L += [x]
    return L
# задание множеств:
cA = int(input()); A = rndSet(cA, 0, 51) # множество A, |A| = cA,
cB = int(input()); B = rndSet(cB, 0, 51) # множество B, |B| = cB,
print('A =', A); print('B =', B) # вывод элементов множеств на экран,
# построение множеств точек:
import pylab # подключили необходимый модуль,
pylab.figure(figsize = (3, 3)) # размеры графического окна,
pylab.grid(color = 'k', linestyle='-.') # параметры решетки,
```

```

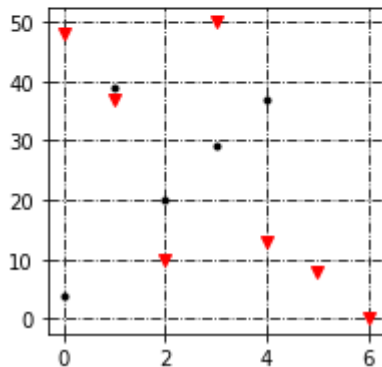
xA = range(0, cA); xB = range(0, cB) # абсциссы точек для графиков,
pylab.plot(xA, A, 'rv') # строим красным цветом точки множества A,
pylab.plot(xB, B, 'k.') # строим черным цветом точки множества B,
# проверка свойства  $A \cup B = B \cup A$ 
def subset(A, B): #  $A \subset B$  ?
    if A == []: return True
    for a in A:
        if a not in B: return False
    return True
def equal(A, B): #  $A = B$  ?
    if subset(A, B) and subset(B, A): return True
    return False
def union(A,B): #  $A \cup B$ ,
    if A == []: return B
    if B == []: return A
    for a in A:
        if a not in B: B +=[a]
    return B
print(A)
AB = union(A,B) #  $AB = A \cup B$ ,
BA = union(B,A) #  $BA = B \cup A$ ,
print("AvB = ", AB); print("BvA = ", BA)
if equal(AB,BA):
    print("Свойство выполняется")
else:
    print("Свойство не выполняется")

```


Скриншот результата выполнения программы

```
In [59]: runfile('C:/Users/User/.spyder-py3/temp.py', wdir='C:/Users/User/.spyder-py3')
```

```
7
5
A = [48, 37, 10, 50, 13, 8, 0]
B = [4, 39, 20, 29, 37]
A∪B = [4, 39, 20, 29, 37, 48, 10, 50, 13, 8, 0]
B∪A = [48, 37, 10, 50, 13, 8, 0, 4, 39, 20, 29]
Свойство выполняется
```



Вывод

В результате выполнения лабораторной работы:

- 1) ознакомились с
- 2) были доказаны два свойства операций над множествами: $A \cup B = B \cup A$, ...
- 3) разработана программа проверки свойства $A \cup B = B \cup A$, результаты тестирования которой показали правильность ее работы.

Задание на лабораторную работу выполнено полностью.

П1.2 Рекомендации по выполнению лабораторной работы № 3

Задание. Создать программу на языке Python для нахождения во множестве A двух элементов с наибольшей суммой.

Требования к программе

- множество A , $|A| = 5$, задать списком;
- множество A инициализировать случайными целыми числами из универсума $[-10; 10]$;
- для получения всех подмножеств использовать функции `nextGr`, `kSubsets`, либо объявить свои функции.

Программа

инициализация множества случайными числами:

```
import random # подключение модуля,
random.seed() # инициализация генератора случайных чисел,
```

```

def rndSet(k, a, b):
# вход:  $k \in \mathbb{N}$ ;  $a, b \in \mathbb{Z}$ ,  $a < b$ 
# выход: множество из  $k$  случайных целых чисел отрезка  $[a; b]$ 

L = []
for i in range(0, k):
    x = random.randint(a, b)
    while x in L:
        x = random.randint(a, b)
    L += [x]
return L

A=rndSet(5, -10, 10) # инициализация множества A,
print('A=',A)

# Получение  $i$ -го числа кода Грея:
def nextGr(B, i):
    # вход: список B, натуральное  $i$ ,
    # выход:  $i$ -е число кода Грея,
    def Q(i):
        # вход: натуральное  $i$ ,
        # выход: max степень 2, делящая  $i$ ,
        q = 1; j = i
        while j%2 == 0:
            j = int(j/2); q = q+1
        q = q - 1
        return q
    B[Q(i)] = 1- B[Q(i)] #1
    return B

# Количество 1 в числе кода Грея:
def card(B):
# вход: множество B из 0 и 1
# вход: количество 1 во множестве B

S = 0
for i in range(len(B)):
    if B[i] == 1: S +=1
return S

k=2
print("Все ",k," элементные подмножества A:")

```

```

def kSubsets(k, A): # Находим все 2-х элементные подмножества A
                    # и наибольшую сумму их элементов:
    n = len(A)
    m = -20 # т.к.  $A \subset [-10; 10]$ , то наибольшая сумма двух элементов из  $A \geq -20$ ,
    B = [0 for t in range(n)]
    for t in range(1, 2**n):
        B = nextGr(B, t)
        if card(B) == k:
            C = []
            for q in range(n):
                if B[q] == 1: C = C+[A[q]]
            # очередное  $C \subset A$ ,  $|C| = k$ , получено,
            # выводим его на экран:
            print(C)
            # сравниваем сумму элементов  $C$  с  $m$ :
            if C[0] + C[1] > m:
                m = C[0] + C[1]
                a1 = C[0]; a2 = C[1]
    return m, a1, a2 # результат: максимальная сумма и соответствующие элементы.

m, a1,a2 = kSubsets(k, A)
print("max=",m," a1=",a1, " a2=",a2)

```

Скриншот результата выполнения программы

```

runfile('C:/Users/User/.spyder-py3/temp.py', wdir='C:/Users/User/.spyder-py3')
A= [10, 9, 6, 0, -8]

```

Все 2 элементные подмножества A:

```
[10, 9] [9, 6] [10, 6] [6, 0] [9, 0] [10, 0] [0, -8] [6, -8] [9, -8] [10, -8]
```

```
max= 19 a1= 10 a2= 9
```

Программа дает правильный результат.

П1.3 Рекомендации по выполнению лабораторной работы № 6

Задание.

1. В тексте программы в виде списков задать множества: $A \subset \mathbb{Z}$, $|A| = 5$, $B \subset \{ 'a', 'b', \dots, 'z' \}$, $|B| = 5$.
2. В тексте программы в виде списка списков задать бинарные отношения: $\delta \subset A \times B$, $|\delta| = 5$, $\rho \subset A \times B$, $|\rho| = 5$, графически их изобразить (в MS Visio).
3. Вывести на экран матрицы: M_δ , M_ρ , $M_{\delta \cup \rho}$.

Выполнение задания

Программа

задание множеств A, B:

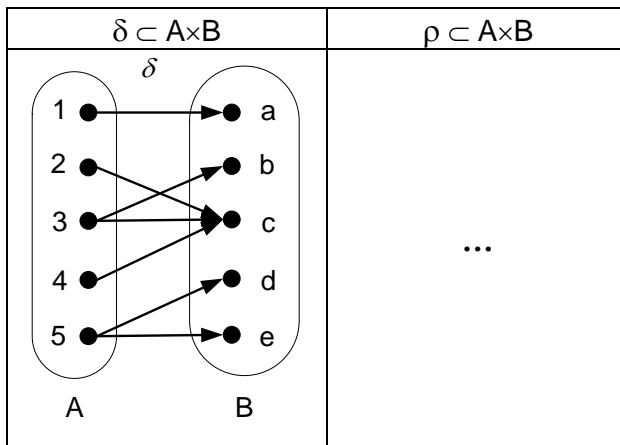
A = [1, 2, 3, 4, 5]; B = ['a', 'b', 'c', 'd', 'e']

задание бинарных отношений δ и ρ :

delAB = [[1, 'a'], [2, 'c'], [3, 'b'], [3, 'c'], [4, 'c'], [5, 'd'], [5, 'e']] # $\delta \subset A \times B$

roAB = [[2, 'a'], [3, 'c'], [3, 'b'], [3, 'c'], [4, 'a'], [5, 'a'], [5, 'b']] # $\rho \subset A \times B$

""" Графическое изображение бинарных отношений



"""

нахождение матрицы бинарного отношения:

def matr(rAB, A, B): # матрица $rAB \subset A \times B$

M = [[0 for x in range(len(B))] for y in range(len(A))]

for row in range(len(A)):

for col in range(len(B)):

M[row][col]=int([A[row], B[col]] in rAB)

return M

def outM(M, cRows): # построчный вывод матриц,

for i in range(cRows):

print(M[i])

матрица M_δ

Mdel = matr(delAB, A,B) # Mdel = M_δ

print("Mdel="); outM(Mdel,5)

''' на экране

$M_{\delta} =$

```
[1, 0, 0, 0, 0]
[0, 0, 1, 0, 0]
[0, 1, 1, 0, 0]
[0, 0, 1, 0, 0]
[0, 0, 0, 1, 1]
```

'''

матрица M_{ρ}

$M_{ro} = \text{matr}(\text{roAB}, A, B)$

$\text{print}("M_{ro}="); \text{outM}(M_{ro}, 5)$

''' на экране

$M_{\rho} =$

```
[0, 0, 0, 0, 0]
[1, 0, 0, 0, 0]
[0, 1, 1, 0, 0]
[1, 0, 0, 0, 0]
[1, 1, 0, 0, 0]
```

'''

матрица объединения бинарных отношений

$\text{def mUnInterRel}(M_s, M_r, \text{row}, \text{col}, \text{what}):$

 # what = 1 - объединение

$M = [[0 \text{ for } x \text{ in range}(\text{col})] \text{ for } y \text{ in range}(\text{row})]$

 for i in range(row):

 for j in range(col):

 if what == 1: $M[i][j] = \text{int}(M_s[i][j] \text{ or } M_r[i][j])$

 else: $M[i][j] = \text{int}(M_s[i][j] \text{ and } M_r[i][j])$

 return M

$mUnDelRo = \text{mUnInterRel}(M_{del}, M_{ro}, 5, 5, 1)$

$\text{print}("mUnDelRo="); \text{outM}(mUnDelRo, 5)$

''' на экране

$M_{\delta \cup \rho} =$

```
[0, 0, 0, 0, 0]
[1, 0, 0, 0, 0]
[0, 1, 1, 0, 0]
[1, 0, 0, 0, 0]
[1, 1, 0, 0, 0]
```

'''

П1.4 Рекомендации по выполнению лабораторной работы № 14

Задание

1. Доказать свойство

$$C_n^k = C_{n-1}^k + C_{n-1}^{k-1}$$

2. Проверить это свойство с помощью программы на языке Python.

Выполнения задания

1. Доказательство. Используя формулу $C_n^k = \frac{n!}{(n-k)!}$, получим

$$\begin{aligned} C_{n-1}^k + C_{n-1}^{k-1} &= \frac{(n-1)!}{k!(n-1-k)!} + \frac{(n-1)!}{(k-1)!(n-k)!} = \frac{(n-1)!}{(k-1)!(n-1-k)!} \left(\frac{1}{k} + \frac{1}{n-k} \right) = \\ &= \frac{(n-1)!}{(k-1)!(n-1-k)!} \cdot \frac{n}{k(n-k)} = \frac{n!}{k!(n-k)!} = \\ &= C_n^k. \end{aligned}$$

2. Программа

```
def P(n): # n!
    if n==0 or n==1: return 1
    else: return n*P(n-1)
def C(n, k): # C_n^k
    if n == 0 or n == 1: return 1
    else: return int(P(n)/P(k)/P(n-k))
n = 10
for k in range(1, 10):
    print('k=', k, ':', C(n-1,k), '+', C(n-1,k-1), '...', C(n,k))
```

Скриншот

```
k= 1 : 9 + 1 ... 10
k= 2 : 36 + 9 ... 45
k= 3 : 84 + 36 ... 120
k= 4 : 126 + 84 ... 210
k= 5 : 126 + 126 ... 252
k= 6 : 84 + 126 ... 210
k= 7 : 36 + 84 ... 120
k= 8 : 9 + 36 ... 45
k= 9 : 1 + 9 ... 10
```

Вывод: Свойство выполняется для $n=10$ и $k=1, 2, \dots, 9$.

П1.5 Рекомендации по выполнению лабораторной работы № 15

Задание 1. Определить, содержит ли разложение $(a+b)^n$ рациональные члены, если содержит, то подсчитать их количество и найти номер k и значение наибольшего коэффициента C_n^k для рациональных членов.

Вариант 0. $a = \sqrt{6}, b = \sqrt[4]{3}, n = 150$.

Решение. По биномиальной теореме имеем

$$\begin{aligned} (\sqrt{6} + \sqrt[4]{3})^{150} &= \sum_{k=0}^{150} C_{150}^k (\sqrt{6})^k (\sqrt[4]{3})^{150-k} = \sum_{k=0}^{150} C_{150}^k 6^{\frac{k}{2}} 3^{\frac{150-k}{4}} = \sum_{k=0}^{150} C_{150}^k 2^{\frac{k}{2}} 3^{\frac{150-k}{4} + \frac{k}{2}} = \\ &= \sum_{k=0}^{150} C_{150}^k 2^{\frac{k}{2}} 3^{\frac{150+k}{4}}. \end{aligned}$$

Так как биномиальные коэффициенты есть числа целые, то произведение $C_{150}^k 2^{\frac{k}{2}} 3^{\frac{150+k}{4}}$ будет числом рациональным (точнее целым) тогда и только тогда, когда числа $2^{\frac{k}{2}}$ и $3^{\frac{150+k}{4}}$ будут числами целыми, то есть когда k делится на 2 и $150+k$ делится на 4.

Если k делится на 2, то $k = 2t, t \in \mathbb{Z}$.

$150+k = 150+2t$ будет делиться на 4, если $75+t$ делится на 2, что возможно при нечетном t , то есть при $t = 2q+1, q \in \mathbb{Z}$.

Таким образом, каждому $k = 2(2q+1) = 4q + 2$ и только им соответствует рациональный член в данном разложении.

Так $0 \leq k \leq 150$, то для нахождения количества рациональных членов находим наибольшее q из неравенства $4q + 2 \leq 150$, то есть из $q \leq 37$.

Следовательно, количество рациональных чисел в данном разложении равно 37.

Программа нахождения значения k наибольшего C_{150}^k рациональных членов

<pre>def P(n): # n! if n==0 or n==1: return 1 else: return n*P(n-1) def C(n,k): if n==0 or n==1: return 1 else: return int(P(n)/P(k)/P(n-k)) def maxC(N): # C_n^k t=1; m=1 # t – номер наибольшего for k in range(2, N+1, 4): if C(150, k) > t: t=C(150,k); m=k return [t, m] N=150 k = maxC(N)[1]; MC = maxC(N)[0] print('k=', k, 'C(', N, ',', k, ')=', MC)</pre>	<pre># результат на экране k = 74 C(150 , 74) = 91604674082278411105849730017829166690336768</pre>
--	---

Ответ. 1) Количество рациональных чисел в разложении $(\sqrt{6} + \sqrt[4]{3})^{150}$ равно 37.

2) Значение наибольшего коэффициента C_{150}^k для рациональных членов достигается при $k = 74$ и равно 91604674082278411105849730017829166690336768.

Задание 2. 2. Найти коэффициент при t^k в разложении, либо доказать их отсутствие для данного k .

Вариант 0. $(1+t-2t^3)^7$, $k = 9$.

Решение. Применяя полиномиальную теорему, получим

$$(1+t-2t^3)^7 = \sum_{\substack{d_1+d_2+d_3=7 \\ d_1 \geq 0, d_2 \geq 0, d_3 \geq 0}} \frac{7!}{d_1!d_2!d_3!} t^{d_2} (-2t^3)^{d_3} = \sum_{\substack{d_1+d_2+d_3=7 \\ d_1 \geq 0, d_2 \geq 0, d_3 \geq 0}} \frac{7!}{d_1!d_2!d_3!} (-2)^{d_3} t^{d_2+3d_3}.$$

Так как по условию $d_2 + 3d_3 = 9$, а для каждого слагаемого $d_1 + d_2 + d_3 = 7$, то получаем систему линейных уравнений

$$\begin{cases} d_1 + d_2 + d_3 = 7 \\ d_2 + 3d_3 = 9 \\ d_1 \geq 0, d_2 \geq 0, d_3 \geq 0 \end{cases}$$

Второе уравнение имеет два неотрицательных решения: $d_2 = 0, d_3 = 3$ и $d_2 = 9, d_3 = 0$.

Так как второе решение не удовлетворяет первому уравнению, то система имеет единственное решение: $d_1 = 4, d_2 = 0, d_3 = 3$.

Следовательно, коэффициент при t^7 равен

$$\frac{7!}{4!3!}(-2)^3 = -8^4 \frac{7 \cdot 5 \cdot 4 \cdot 3 \cdot 2 \cdot 1}{4 \cdot 3 \cdot 2 \cdot 1 \cdot 3 \cdot 2 \cdot 1} = -\frac{140}{3} = -46\frac{2}{3}.$$

Ответ. Коэффициент при t^7 в разложении $(1+t-2t^3)^7$ равен $-46\frac{2}{3}$.

П1.6 Рекомендации по выполнению лабораторной работы № 16

Задание. В группе студентов 25 человек. Среди них 20 сдали сессию успешно, 12 занимаются в спортивных секциях, причем 10 из них сдали сессию успешно. Сколько неуспевающих студентов не посещает спортивных секций?

Решение. Обозначим:

$N = 25$ – количество студентов в группе,

$N_1 = 20$ – количество студентов успешно сдавших сессию,

$N_2 = 12$ – количество студентов, занимающихся в спортивных секциях,

$N_{1,2} = 10$ – количество студентов успешно сдавших сессию и занимающихся в спортивных секциях,

N_0 – количество неуспевающих студентов, которые не посещают спортивных секций.

Тогда по формуле включения-выключения

$$N_0 = N - (N_1 + N_2) + N_{1,2} = 25 - (20 + 12) + 10 = 3.$$

Ответ. 3 неуспевающих студента не посещают спортивные секции.

П1.7 Рекомендации по выполнению лабораторной работы № 17

1. Доказательство свойства

$$\sum_{k=0}^n k C_n^k = n 2^{n-1}.$$

Выберем $f_k(x) = x^k$ и положим $a_k = \begin{cases} C_n^k, & k \leq n, \\ 0, & k > n, \end{cases}$ получим производящую функцию

$$A(x) = \sum_{k=0}^n a_k x^k = \sum_{k=0}^n C_n^k x^k.$$

По биномиальной теореме $A(x) = (1+x)^n$.

Почленно дифференцируя, получим

$$A'(x) = \sum_{k=0}^n k C_n^k x^{k-1} = n(1+x)^{n-1}.$$

Отсюда при $x = 1$ следует

$$A'(1) = \sum_{k=0}^n k C_n^k = n 2^{n-1}.$$

2. Программа и результаты проверки

```
def P(n): # n!
    if n == 0 or n == 1: return 1
    else: return n*P(n-1)
def C(n, k): # C_n^k
    if n==0 or n==1: return 1
    else: return int(P(n)/P(k)/P(n-k))
def Test(n): # проверка свойства
    S=0
    for k in range(n+1):
        S += k*C(n,k)
    print('n=',n,'слева: ',S,'справа: ',n*2**(n-1))
Test(10)
Test(15)
Test(50)
```

```
n= 10 слева: 5120 справа: 5120
n= 15 слева: 245760 справа: 245760
n= 50 слева: 28147497671065516
        справа: 28147497671065600
```

П1.8 Рекомендации по выполнению лабораторной работы № 18

1. Решить рекуррентное соотношение

$$T_n = 2T_{n-1} + 1, T_0 = T_1 = 1,$$

используя производящие функции.

Решение

1) Умножим обе части рекуррентного соотношения на $\frac{x^n}{n!}$, суммируя по n , получим

$$\begin{aligned} \sum_{n \geq 1} T_n \frac{x^n}{n!} &= 2 \sum_{n \geq 1} T_{n-1} \frac{x^n}{n!} + \sum_{n \geq 1} \frac{x^n}{n!} = \sum_{n \geq 0} T_n \frac{x^{n+1}}{(n+1)!} + e^x - 1 = \\ &= 2 \sum_{n \geq 1} T_n \frac{x^{n+1}}{(n+1)!} + e^x - 1 \quad \text{м.к. } T_0=1 \quad = 2 \sum_{n \geq 0} T_n \frac{x^{n+1}}{(n+1)!} + e^x - 1. \end{aligned} \quad (*)$$

Обозначим

$$A(x) = \sum_{n \geq 1} T_n \frac{x^n}{n!}.$$

Возьмем производную от обеих частей равенства (*):

$$A'(x) = 2 \left(\sum_{n \geq 1} T_n \frac{x^{n+1}}{(n+1)!} \right)' + e^x \quad \text{или} \quad A'(x) = 2A(x) + e^x.$$

2) Решаем полученное линейное неоднородное дифференциальное уравнение:

а) $A'(x) = 2A(x)$ или $d \ln(A(x)) = 2$, следовательно, $A(x) = Ce^{2x}$;

b) считая C функцией от x , получим

$$A'(x) = C' e^{2x} + 2Ce^{2x};$$

c) подставляем $A'(x)$ в данное неоднородное уравнение, получим

$$C' e^{2x} + 2Ce^{2x} = 2Ce^{2x} + e^x;$$

следовательно,

$$C' = e^{-x};$$

отсюда $C = -e^{-x} + C_1$, поэтому

$$A(x) = Ce^{2x} = (-e^{-x} + C_1)e^{2x} = -e^x + C_1e^{2x}.$$

Так как $A(0) = 0$, то $C_1 = 1$ и

$$A(x) = e^{2x} - e^x.$$

3) Раскладываем $A(x)$ в ряд

$$A(x) = \sum_{n \geq 0} \frac{2^n x^n}{n!} - \sum_{n \geq 0} \frac{x^n}{n!} = \sum_{n \geq 0} \underbrace{\frac{(2^n - 1)x^n}{n!}}_{2^0 - 1 = 0} = \sum_{n \geq 1} \frac{(2^n - 1)x^n}{n!} = \sum_{n \geq 1} \frac{T_n x^n}{n!}.$$

4) Приравнявая коэффициенты, получим: $T_n = 2^n - 1$.

Ответ. $T_n = 2^n - 1$.

2. Программа для проверки найденного решения рекуррентного соотношения

<pre>def t(n): # рекуррентное соотношение if n == 0 or n == 1: return 1 else: return 2*t(n-1) + 1 for n in range(30, 41): print('n= ', n, '(', t(n), ', ', 2**n-1, ')')</pre>	<pre># результат на экране n= 30 (1073741823, 1073741823) n= 31 (2147483647, 2147483647) n= 32 (4294967295, 4294967295) n= 33 (8589934591, 8589934591) n= 34 (17179869183, 17179869183) n= 35 (34359738367, 34359738367) n= 36 (68719476735, 68719476735) n= 37 (137438953471, 137438953471) n= 38 (274877906943, 274877906943) n= 39 (549755813887, 549755813887) n= 40 (1099511627775, 1099511627775)</pre>
---	---

Вывод. Для всех натуральных n из отрезка $[30, 40]$ значения рекурсивной функции $t(n)$ и решения рекуррентного соотношения T_n совпадают.

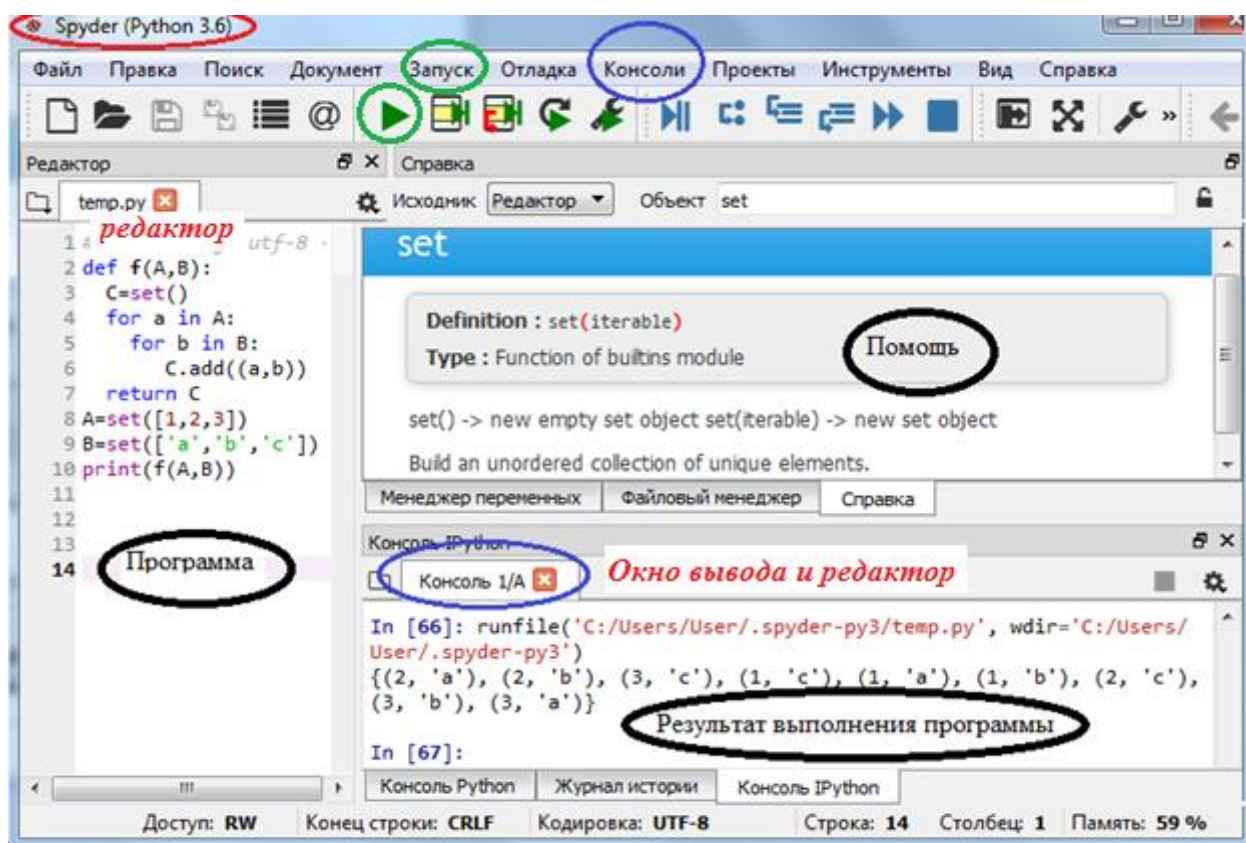
Приложение 2. Интегрированная среда Spyder

Spyder – свободная, кроссплатформенная, интерактивная интегрированная среда разработки для научных расчетов на языке Python, обеспечивающая простоту использования функциональных возможностей и «легковесность» программной части.

Возможности Spyder

- Редактор с подсветкой синтаксиса.
- Динамическая интроспекция кода: автодополнение, переход к определению объекта по щелчку мыши.
- Нахождение ошибок «на лету».
- Поддержка одновременного использования нескольких консолей Python (включая оболочку IPython).
- Встроенные средства доступа к документации.
- Гибко настраиваемый интерфейс.
- Интеграция с модулями Python – NumPy, SciPy, Matplotlib, Pandas.

Интерфейс Spyder



Приложение 3. Язык программирования Python

Руководство по написанию кода на Python можно найти по адресу

<https://pythonworld.ru/osnovy/pep-8-rukovodstvo-po-napisaniyu-koda-na-python.html>


«Python – это *свободный*, интерпретируемый, объектно-ориентированный, ..., язык программирования очень высокого уровня.

- ✓ свободный – все исходные тексты интерпретатора и библиотек доступны для любого использования;
- ✓ интерпретируемый – использует «позднее связывание»;
- ✓ объектно-ориентированный – классическая ОО модель;
- ✓ очень высокого уровня – динамическая типизация, встроенные типы данных высокого уровня, классы, модули, механизм исключений».

Python - активно развивающийся язык программирования.

На Python отсутствуют стандарты (ANSI, ISO, другие официальные стандарты), их роль выполняет CPython¹.

«Python – это *свободный*, интерпретируемый, объектно-ориентированный, ..., язык программирования очень высокого уровня».

	Автор языка Python Гвидо ван Россум (род. 1956, Нидерланды)	Пример программы на языке Python	
		текст в редакторе	результат в консоле
		<pre>def f(x): return x**2 print(f(8))</pre>	64

П3.1 Программа на языке Python

- ✓ Программа на языке Python – это совокупность *логических* строк.
- ✓ Логическая строка – это одна или несколько объединенных экранных строк (в интерактивном режиме с использованием символа «\» в конце экранной строки, в среде Spyder посредством отступов).

✓ *Комментарии:*

однострочечный комментарий

'''

многострочечный - строки между двумя тройными апострофами.

'''

- ✓ *Отступы: Python «чувствителен» к отступам* – продолжение конструкции на новой физической строке должно начинаться правее первого символа конструкции.

¹ CPython - наиболее распространённая, эталонная реализация языка программирования Python. Разработка CPython ведётся группой разработчиков под руководством создателя языка Python Гвидо ван Россума.

✓ *Идентификаторы* (имена, переменные): регистр важен, кириллица не допускается, длина не ограничена.

✓ *Выражения*: состоят из атомов, объединенных операторами; перед применением бинарных операторов операнды приводятся к общему типу; результат применения операторов зависит от типа операндов.

П3.2 Встроенные числовые типы (int, float)

int	float
-123	2.7182818284545
123456789101112131415	-3.14e-45
имеют неограниченную точность	

Арифметические операторы	Условные операторы
x**y +x -x x*y x/y x%y x+y x-y	x<y x>y x==y x<=y x>=y x!=y

При выполнении арифметических или логических операторов «меньший» тип операнда приводится к «большему» типу (int < float).

П3.3 Встроенные типы-последовательности (кортежи, строки, списки, диапазоны)

Кортежи, диапазоны и строки являются *неизменяемыми* объектами.

Операции, определенные для всех встроенных последовательностей

len(s)	длина последовательности s,
x in s	True, если $x \in s$, иначе False,
x not in s	False, если $x \in s$, иначе True,
s + t	конкатенация (объединение) последовательностей s и t,
s*n	конкатенация n копий последовательности s (тот же результат дает n*s),
s*0	пустая последовательность (0*s - тот же результат),
s += t	то же, что и $s = s + t$,
s *= t	то же, что и $s = s * n$,
s[i]	i-й элемент последовательности s, отсчет с 0;
s[i: j]	срез – подпоследовательность от i-го до (j-1)-го элементов (для $0 \leq i \leq j$).
min(s)	наименьший элемент последовательности s,
max(s)	наибольший элемент последовательности s.

П3.3.1 Кортежи

Кортеж – последовательность элементов через запятую, заключенных в круглые скобки.

Кортеж из одного элемента должен содержать завершающую запятую.

Примеры кортежей: (1, 2, 3); ("1", 2, 3); (); (1,)

- `s = input()` интерактивный ввод кортежа;
- *Преобразование последовательности в кортеж:* `tuple(последовательность);`

П3.3.2 Строки

Строка – последовательность символов, заключенная в апострофы или кавычки.

Примеры строк: `'qwerty'`, `"-12.09"`.

- `s = input()` интерактивный ввод строки;
- *Преобразование числа в строку:* `str(числовое выражение);`
- *Преобразование строки в целое число:* `int(строка); # int('12');`
- *Преобразование строки в вещественное число:* `float(строка); # float('-12.89');`

П3.3.3 Списки

Список – последовательность элементов через запятую, заключенная в квадратные скобки, например, `[1, "2", 3]`.

Примечание. Тип массив в языке Python используется редко (только для увеличения скорости выполнения программы); вместо массива часто используют тип список.

Создание списков

- 1) `[]` – пустой список,
- 2) перечисление элементов: `m = [1, 2, "список", [1, 5]]`,
- 3) применение выражения к каждому элементу последовательности, например,

<pre>c = [c*2 for c in 'list'] print(c) # на экране: ['mm', 'aa', 'mm', 'aa']</pre>	<pre>c = [c + d for c in 'list' if c != 'i' for d in 'spam' if d != 'a'] print(c) # на экране: ['ls', 'lp', 'lm', 'ss', 'sp', 'sm', 'ts', 'tp', 'tm']</pre>
--	--

Дополнительные операции над списками (x - объект, s и t- однотипные списки)

`s[i] = x` – замена i-го элемента списка s на x,

`del s[i]` – удаление элемента i-го элемента из списка s,

`s[i: j] = t` - замена части списка от i до j на t,

`s[i: j] = []` – удаление из списка подсписка от i до j.

Некоторые методы списков² (x - объект, s и t – однотипные последовательности)

`s.append(x)` - добавляет x в конец списка s,

`s.extend(t)` - добавляет в конец списка s список t,

`s.count(x)` - возвращает число вхождений элемента x в список s,

² Методы применимы к любым изменяемым последовательностям.

`s.sort()` – сортировка списка `s` по возрастанию (если, элементы списка можно сравнивать).

- Преобразование диапазона в список: `list(диапазон)`

Список выражений – это перечисление выражений через запятую.

name – переменная, *expr* – выражение, *expr_seq* – выражение типа последовательность.

П3.6 Объявление именованных функции пользователя (def)

`def имя_функции(список_параметров) : блок_кода`

Вызов функций

`имя_функции([список_выражений_через_запятую])`

Инструкция return [список_выражений] прерывает выполнение функции и возвращает значение функции.

Пример.

```
def f(x, y = 1.2): # y – параметр по умолчанию,
    return x**y
print(f(2, 2), f(2))
```

П3.6.1 Рекурсивные именованные функции

```
def f(x):
    if x == 1.0: return 1.0
    else: return x*f(x - 1)
print(f(100.0))
```

П3.7 Встроенные модули

Для Python создано более 500 встроенных модулей.

Перед использованием модуль следует подключить инструкцией `import`:

`import список_имен_модулей_через_запятую` # импорт указанных модулей,
`from имя_модуля import список_объектов_через_запятую` # импорт объектов из модуля

П3.7.1 Модуль math

Справка по функции модуля: `print(help('math.имя_функции'))`

Функции вещественного аргумента

acos	atan2	degrees	fmod	modf	sqrt
acosh	atanh	e	log	pow	tan
asin	ceil	exp	log10	radians	tanh
asinh	cos	fabs	log1p	sin	trunc
atan	cosh	floor	log2	sinh	

Функции целочисленного аргумента

factorial	gcd
-----------	-----

П3.7.2 Модуль random (случайные числа)

В модуле `random` объявлены функции для генерации случайных чисел и букв.

```
import random # подключение модуля,
random.seed() # инициализация генератора случайных чисел,
x = random.randint(a, b) # x = случайное целое число из отрезка [a; b], a и b – целые числа,
```

`z = random.random()` # `z` = случайное число из `[0; 1)`.

П3.8 Графические методы

Графические методы python реализованы в модуле `pylab`.

Справка по модулю: http://matplotlib.org/api/pyplot_api или `help('pylab.plot')`

- Построение линий и/или маркеров:

`pylab.plot(список_абсцисс, список_ординат, параметры)`

- Создание списков абсцисс и ординат точек:

✓ если абсциссы и ординаты являются целыми числами, то для создания из них списков можно использовать метод `range`;

✓ для вещественных абсцисс и/или ординат точек можно использовать метод `frange`:

`from matplotlib import pylab` # используем метод `frange`,

`xL = pylab.frange(a, b, h)` # `xL` – арифметическая прогрессия вещественных чисел

`yL = [f(x) for x in xL]` # `yL` – список значений функции `f` от элементов списка `xL`

✓ можно самим создавать списки, например,

`A = [1,2,3,4]; L = []`

for `a` in `A`: `L += [a**2.5]`

- Задание типа и цвета позиционными параметрами '`тип`' '`цвет`'

тип линии	тип маркера	цвет
- непрерывная,	. точка, , пиксель,	b = blue, g = green, r = red
-- пунктир,	o вида ● v вида ▼,	c = cyan, m = magenta, y = yellow
-. штрих-пунктир,	^ вида ▲ < вида ◀,	k = black, w = white ...
: из точек, ...	> вида ▶ * вида ★, ...	

- Задание типа и цвета именованными параметрами

<code>color = 'цвет_линии',</code> <code>linestyle = 'тип',</code> <code>linewidth = толщина_линии</code>	<code>marker = 'тип'</code> <code>markersize = размер</code> <code>markerfacecolor = 'цвет_маркера',</code>
---	---

- Задание размера окна `W×H` в дюймах:

`pylab.figure(figsize = (W, H))` # задает.

- Задание решетки на окне:

`pylab.grid(color='цвет', linestyle='стиль', linewidth=ширина)`

- Вывод текста, начиная с точки `(x, y)`, кегля `N`:

`pylab.text(x, y, 'текст', size = N)`

П8.1 Построение графика $y = f(x)$

Построение графика функции $y = \sqrt{x}$ на отрезке `[0; 16]` с шагом `0,5`.

```

from math import sqrt, sin
a = 0.0; b = 16.0; h = 0.5

from matplotlib import mlab
import pylab

xL = mlab.frange(a, b, h)
yL = [sqrt(x) for x in xL]

pylab.figure(figsize = (3, 3))
pylab.grid(color = 'k', linestyle = '-.',
           linewidth = 1)

# рисуем график:
pylab.plot(xL, yL, 'k-')

pylab.text(0.9, 3.6,
           "График функции", size = 12)

```

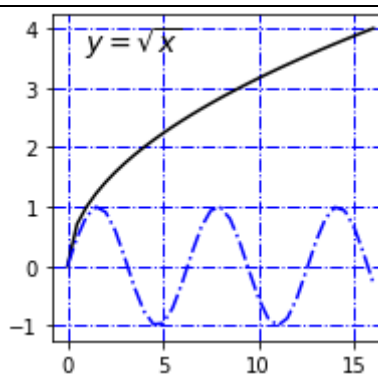


Примечание. Построение двух графиков в одном окне:

```

y2L = [sin(x) for x in xL]
pylab.plot(xL, yL, 'k-', # первый график,
           xL, y2L, 'b-', # второй график.
)

```



Пример. Построение множества точек с целочисленными координатами.

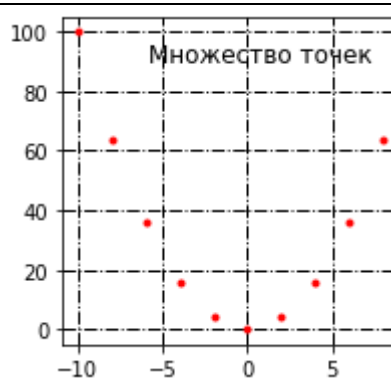
```

import pylab
xL = range(-10, 10, 2)
yL = [x**2 for x in xL]
pylab.figure(figsize = (3, 3))
pylab.grid(color = 'k', linestyle='-.')

# рисуем график:
pylab.plot(xL, yL, 'r.')

pylab.text(-5.9, 90,
           "Множество точек", size = 12)

```



Приложение 4. Библиотека функций для компьютерной поддержки курса «Дискретная математика»

П4.1 Представление множеств в ЭВМ (на языке Python)

Примечание. При копировании объявленных функций в редактор Python отступы могут измениться, об этом «подскажет» интерпретатор восклицательным знаком:

В этом случае следует восстановить отступы.

П4.1.1 rndSet(k, a, b): инициализация множества случайными числами

вход: $k \in \mathbb{N}$, $a, b \in \mathbb{Z}$

выход: множество из k случайных целых чисел из [a; b]

```
def rndSet(k, a, b):  
    L = []  
    for i in range(0, k):  
        x = random.randint(a, b)  
        while x in L:  
            x = random.randint(a, b)  
        L += [x]  
    return L
```

П4.1.2 inpSetI(k): интерактивная инициализация множества числами

вход: $k \in \mathbb{N}$

выход: множество из k целых чисел, введенных с клавиатуры

```
def inpSetI(k):  
    S = []  
    for i in range(k):  
        a = int(input()); S += [a]  
    return S
```

П4.1.3 inpSetC(k): интерактивная инициализация множества символами

вход: $k \in \mathbb{N}$

выход: множество из k символов, введенных с клавиатуры

```
def inpSetC(k):  
    S = []  
    for i in range(k):  
        a = input(); S = S+[a]  
    return S
```

П4.1.4 subset(A, B): являться подмножеством

вход: множества A, B

выход: True, если $A \subset B$

```
def subset(A, B):  
    if A == []: return True  
    for a in A:  
        if a not in B: return False  
    return True
```

П4.1.5 equal(A, B): равенство множеств

вход: множества A, B
выход: True, если $A = B$

```
def equal(A, B):  
    if subset(A, B) and subset(B, A): return True  
    return False
```

П4.1.6 union(A, B): объединение множеств

вход: множества A, B
выход: $A \cup B$

```
def union(A, B):  
    if A == []: return B  
    if B == []: return A  
    C = []  
    for a in A: C += [a]  
    for b in B:  
        if b not in A: C += [b]  
    return C
```

П4.1.7 diff(A, B): разность множеств

вход: множества A, B
выход: $A \setminus B$

```
def diff(A, B):  
    if A == []: return []  
    C = []  
    for a in A:  
        if a not in B: C += [a]  
    return C
```

П4.1.9 symDiff(A, B): симметрическая разность множеств

вход: множества A, B;
выход: $A \Delta B$

```
def symDiff(A, B):  
    return compl(A, B)+compl(B, A)
```

П4.1.10 kSubsets(k, A): все k-элементные подмножества множества A

вход: $k \in \mathbb{N}$, множество A

выход: все $B \subset A$, $|B| = k$

```
def kSubsets(k, A):  
    def nextGr(B, i):  
        # вход: список B, натуральное i,  
        # выход: i-е число кода Грея,
```

```

def Q(i):
    # вход: натуральное i,
    # выход: max степень 2, делящая i,
    q = 1; j = i
    while j%2 == 0:
        j = int(j/2); q = q+1
    q = q - 1
    return q
B[Q(i)] = 1- B[Q(i)] #1
return B
def card(B):
    S = 0
    for i in range(len(B)):
        if B[i] == 1: S +=1
    return S
n = len(A)
B = [0 for t in range(n)]
for t in range(1, 2**n):
    B = nextGr(B, t)
    if card(B) == k:
        C = []
        for q in range(n):
            if B[q] == 1: C = C+[A[q]]
        # очередное  $C \subset A$ ,  $|C| = k$ , получено,
        # что-то с ним делаем, например,
        print(C)

```

П4.2 Представление бинарных отношений в ЭВМ (на языке Python)

П4.2.1 dirPr(A, B): прямое произведение множеств

вход: множества A, B

выход: $A \times B$

```

def dirPr(A, B):
    if A == [] or B == []: return []
    C = []
    for a in A:
        for b in B:
            C = C + [[a, b]]
    return C

```

П4.2.2 revRel(ro): обратное бинарное отношение

вход: $ro \subset A \times B$

выход: ro^{-1}

```
def revRel(ro):  
    S = []  
    if ro == []: return S  
    for [a, b] in ro:  
        S += [[b, a]]  
    return S
```

П4.2.3 compRel(delAB, roBC, C): композиция бинарных отношений

вход: $delAB \subset A \times B$, $roBC \subset B \times C$, множество C

выход: композиция отношений $delAB \circ roBC$

```
def compRel(delAB, roBC, C):  
    X = []  
    if delAB == [] or roBC == []: return X  
    for [a, b] in delAB:  
        for c in C:  
            if [b, c] in roBC and not([a, c] in X):  
                X += [[a, c]]  
    return X
```

П4.2.4 matr(roAB, A, B): матрица бинарного отношения

вход: $roAB \subset A \times B$, множества A, B

выход: матрица бинарного отношения $roAB$

```
def matr(roAB, A, B):  
    M = [[0 for x in range(len(B)) for y in range(len(A))]  
    for row in range(len(A)):  
        for col in range(len(B)):  
            M[row][col] = int([A[row], B[col]] in roAB)  
    return M
```

П4.2.5 outM(M, cRows): построчный вывод матрицы

вход: матрица M , $cRows$ – количество строк в матрице

выход: построчный вывод на экран матрицы

```
def outM(M, cRows):  
    for i in range(cRows):  
        print(M[i])
```

П4.2.6 matrT(M, row, col): транспонирование матрицы

вход: матрица M порядка $row \times col$, $cRows$ – количество строк в матрице

выход: M^T

```
def matrT(M, row, col):
    mT = [[0 for x in range(row)] for y in range(col)]
    for i in range(row):
        for j in range(col):
            mT[j][i] = M[i][j]
    return mT
```

П4.2.7 mUnInterRel(Ms, Mr, row, col, what): матрица объединения/пересечения бинарных отношений

вход: Ms, Mr матрицы бинарных отношений порядка $row \times col$, $what \in \mathbb{Z}$

выход: если $what = 1$, то матрица объединения бинарных отношений, иначе матрица пересечения бинарных отношений

```
def mUnInterRel(Ms, Mr, row, col, what):
    M = [[0 for x in range(col)] for y in range(row)]
    for i in range(row):
        for j in range(col):
            if what == 1: M[i][j] = int(Ms[i][j] or Mr[i][j])
            else: M[i][j] = int(Ms[i][j] and Mr[i][j])
    return M
```

П4.2.8 matrComp(delAB, roBC, C): матрица композиции бинарных отношений

вход: $delAB \subset A \times B$, $roBC \subset B \times C$, множество C

выход: матрица композиции $delAB \circ roBC$ бинарных отношений

```
def matrComp(delAB, roBC, C):
    sigAC = compRel(delAB, roBC, C)
    return matr(sigAC, A, C)
```

П4.3 Вычисления количеств комбинаторных конфигураций на языке Python

$P(n) = P_n$	$A(n, k) = A_n^k$	$A_r(n, k) = \overline{A}_n^k$
<pre>def P(n): if n==0 or n==1: return 1 else: return n*P(n-1)</pre>	<pre>def A(n, k): return int(P(n)/P(n-k))</pre>	<pre>def A_r(n, k): return n**k</pre>
	$C(n, k) = C_n^k$	$C_r(n, k) = \overline{C}_n^k$
	<pre>def C(n, k): return int(P(n)/P(k)/P(n-k))</pre>	<pre>def C_r(n, k): return C(n+k-1, k)</pre>

П4.4 Теория графов

П.4.4.1 Инициализация матрицы смежности / весов ребер в тексте программы

```
#   0   1   2   3
M = [[0,  1,  4, 10], # 0
      [1,  0,  2, -1], # 1
      [4,  2,  0,  3], # 2
      [10, -1,  3,  0], # 3
      ]
```

П.4.4.2 Построчный вывод элементов матрицы

```
def outM(M, cRows): # построчный вывод матрицы
#вход: матрица M, cRows – количество строк в матрице
    for i in range(cRows): print(M[i])
def outM(M, cRows): # построчный вывод матрицы
```

П.4.4.2 Графическое представление графа

```
def drawGr(n, M, R):
#вход: n - порядок графа, M - матрица смежности/весов ребер графа, R - радиус
    import pylab
    from math import sin, cos, pi
    pylab.figure(figsize = (5, 5))
# строим вершины графа и помечаем их:
    def outL(x,y,k): # вывод метки вершины графа,
        return pylab.text(x, y, str(k), size = 12)
    for k in range(n):
        for j in range(k,n):
            xP = [R*cos(2*pi*p/n) for p in [k, j]]
            yP = [R*sin(2*pi*q/n) for q in [k, j]]
            pylab.plot(xP, yP, 'k.')
            C=cos(2*pi*k/n); S=sin(2*pi*k/n)
            # в какой четверти вершина?
            if C>=0 and S>=0: outL(xP[0] + 0.15, yP[0], k)    # I-я четверть,
            if C<=0 and S>=0: outL(xP[0] - 0.75, yP[0], k)    # II-я четверть,
            if C<=0 and S<=0: outL(xP[0] - 0.75, yP[0] - 0.4, k) # III-я четверть,
            if C>=0 and S<0: outL(xP[0] + 0.4, yP[0] - 0.1, k) # IV-я четверть,
# проводим ребра графа:
    for k in range(n):
        for j in range(k, n):
```

```

if M[k][j] > 0:
    xL = [R*cos(2*pi*p/n) for p in [k, j]]
    yL = [R*sin(2*pi*q/n) for q in [k, j]]
    pylab.plot(xL, yL, 'k-')

```

П.4.4.3 Алгоритм Дейкстры

```

def algDijkstra(pBeg, n, M):
# Вход: pBeg - начальная вершина, n - порядок графа,
#       М - матрица весов ребер.
# Выход: [d, From], где
#       d – список кратчайших расстояний из начальной до всех остальных вершин,
#       From – список, From[i] – номер вершины, из которой попали в i-ю вершину.
# Инициализация:
    d = [-1 for i in range(n)] # расстояния от начальной вершины,
    From = [-1 for i in range(n)] # из какой вершины пришли,
    W = [0 for i in range(n)] # все вершины поместили, как не посещённые,
    d[pBeg] = 0 # вес начальной вершины,
    #####
    def Env(k, n, M): # Окружение вершины,
#Вход: k - вершина, n - порядок графа, М - матрица смежности/весов графа.
#Выход: окружение вершины k, в которых не были
        en = []
        for i in range(n):
            if M[k][i] > 0 and W[i] == 0: en += [i]
        return en
    #####
    def popQue(Q): # удалить первый элемент из очереди
# Выход: [1-й элемент, [остаток очереди]]
        if Q == []: return [-1, Q]
        x = Q[0]; del Q[0]
        return [x, Q]
    #####
    Q = [pBeg] # начальную вершину в очередь,
    while Q != []: # пока очередь непустая,
        [x, Q] = popQue(Q) # вершина x была первой в очереди,
        W[x] = 1 # поместили ее, как посещённую,
        E = Env(x, n, M) # окружение x, в которых не были,

```

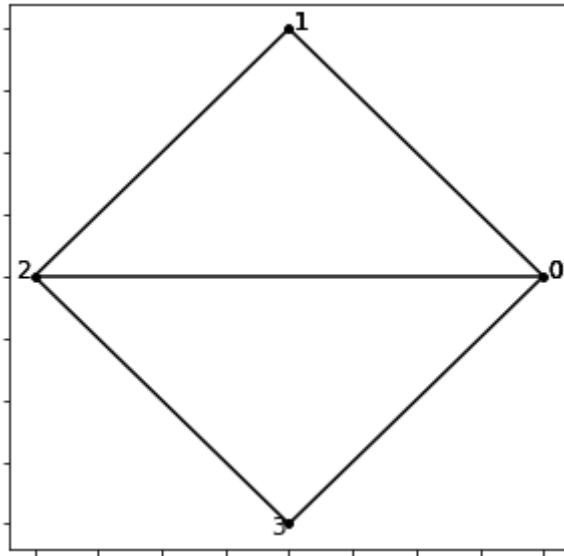
```

if E != []:
    for k in E:
        if M[x][k]>0:
            d0 = d[x] + M[x][k]
            if d[k] == -1: d[k]=d0; From[k]=x
        else:
            if d0<d[k]: d[k] = d0; From[k] = x
    for k in E: Q +=[k]
return [d,From]

# Тестирование функции algDijkstra
n=4
#   0   1   2   3
M = [[0,  1,  4, 10], # 0
      [1,  0,  2, -1], # 1
      [4,  2,  0,  3], # 2
      [10, -1,  3,  0], # 3
      ]
drawGr(n, M, R) # рисуем граф,
Res = algDijkstra(pBeg, n, M)
print('Кратчайшие расстояния из ',pBeg,'-й вершины до:')
for k in range(n): print(k,'-й =',Res[0][k])
pEnd = 3
print('Кратчайший путь от ',pBeg,'-й до ',pEnd,'-й вершины:')
x = pEnd; path = [x]
From = Res[1]
while x != -1:
    x=From[x]
    if x != -1: path = [x] + path
print(path)

```

Граф на экране:



Результат обхода:

Кратчайшие расстояния из 0-й вершины до:

0 -й = 0

1 -й = 1

2 -й = 3

3 -й = 6

Кратчайший путь от 0 -й до 3 -й вершины:

[0, 1, 2, 3]

П.4.4.4 Алгоритм Краскала

def algKruskal(n, mWGr):

Вход: n - порядок графа, mW - матрица весов графа.

Выход: матрица смежности наикратчайшего остова графа,

длина наикратчайшего остова.

1. Создаем список L ребер с положительными весами:

L = []

for i in range(n):

 for k in range(i, n):

 if mWGr[i][k] > 0: L += [[i, k]]

2. Сортируем список L по не убыванию весов ребер:

ag = True

while ag:

 ag = False

 for i in range(len(L) - 1):

 [a1, b1] = L[i]

```

[a2, b2] = L[i + 1]
if mWGr[a1][b1] > mWGr[a2][b2]:
    z = L[i]; L[i]=L[i + 1];L[i + 1] = z
    ag=True

```

3. Создаем наикратчайший остов:

Идея создания остова: при добавлении очередного ребра, количество вершин
не изменилось, то получили цикл.

Проверка наличия цикла:

```
def isCycle(pBeg, n, M):
```

Вход: pBeg - начальная вершина, n - порядок графа,

M - матрица весов ребер.

Выход: True, если есть цикл в компоненте связности,

содержащим вершину pBeg.

Обход графа в ширину:

```
W = [0 for i in range(n)] # все вершины пометили, как не посещённые,
```

```
T = [pBeg] # начальную вершину в очередь T,
```

```
W[pBeg] = 1 # пометили ее как посещённую,
```

```
E = []; V = []
```

```
while T != []: # пока очередь непустая,
```

```
    x = T[0]
```

```
    del T[0] # первую вершину в очереди удалили,
```

```
    E += [x]
```

```
    # получаем окружение вершины x:
```

```
    Env = []
```

```
    for i in range(n):
```

```
        if M[x][i] > 0:
```

```
            Env += [i]
```

```
            if [i, x] not in V: V += [[x, i]]
```

```
# те вершины из окружения x, в которых не были, ставим в очередь:
```

```
    for y in Env:
```

```
        if W[y] == 0: T += [y]; W[y] = 1
```

```
if len(E) == len(V): return True
```

```
else: return False
```

3.1 Инициализация матрицы остова:

```
mWOst = [[0 for i in range(n)] for k in range(n)]
```

3.2 Создаем остов

for [a,b] in L:

mWOst[a][b] = mWGr[a][b]; mWOst[b][a] = mWGr[a][b] # добавили очередное ребро,

if isCycle(a, n, mWOst): # если получили цикл,

mWOst[a][b] = 0; mWOst[b][a] = 0 # то удаляем его.

3.3 Длина наикратчайшего остова:

LenOst = 0

for a in range(n - 1):

for b in range(a+1, n): LenOst += mWOst[a][b]

return [mWOst, LenOst]

Тестирование функции algKruskal

n = 6; порядок графа,

0 1 2 3 4 5 - матрица весов ребер графа,

mWGr = [[0, -1, 7, -1, -1, 5], # 0 [0, 2] = 7, [0, 5] = 5

[-1, 0, -1, 15, -1, 17], # 1 [1, 3] = 15, [1, 5] = 17

[7, -1, 0, 11, -1, -1], # 2 [2, 3] = 11,

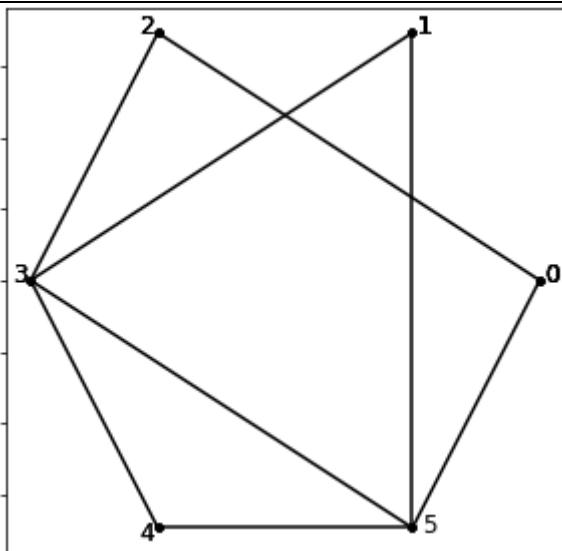
[-1, 15, 11, 0, 4, 5], # 3 [3, 4] = 4, [3, 5] = 5

[-1, -1, -1, 4, 0, 12], # 4 [4, 5] = 12

[5, 17, -1, 5, 12, 0], # 5

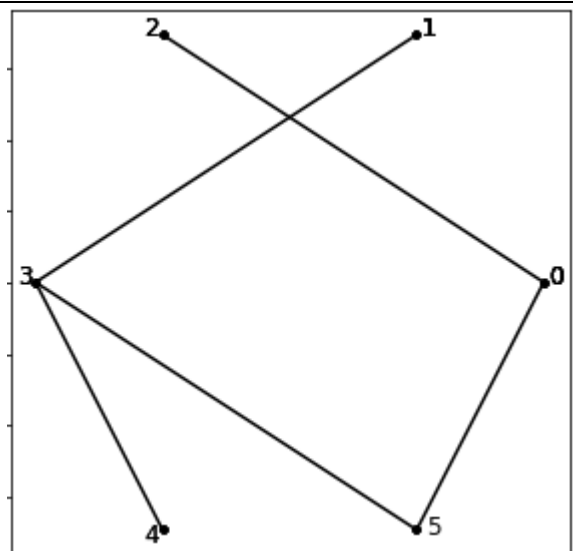
]

drawGr(n, mWGr, 10)



[M, L] = algKruskal(n, mWGr)

print('\n Длина = ',L)



Длина= 36

Ответ правильный

```
drawGr(n, M, 10)
```

```
'''
```

Литература

1. Асанов М.О. Дискретная математика: графы, матроиды, алгоритмы. СПб. : Лань, 2010. ЭБС Лань http://e.lanbook.com/books/element.php?pl1_id=536
2. Бондаренко Л. Н. Дискретная математика. Методические указания для выполнения лабораторных работ. Пенза: Изд-во ПГУ, 2007.
3. Гаврилов Г. П., Сапоженко А.А. Задачи и упражнения по дискретной математике. М.: Физматлит, 2005.
4. Копылов В. И. Курс дискретной математики. СПб. : Лань, 2011. ЭБС Лань http://e.lanbook.com/books/element.php?pl1_id=1798
5. Кузнецов О.П. Дискретная математика для инженера. — СПб.: Лань, 2009. ЭБС Лань http://e.lanbook.com/books/element.php?pl1_id=220
6. Мальцев И.А. Дискретная математика. СПб. : Лань, 2011. ЭБС Лань http://e.lanbook.com/books/element.php?pl1_id=638
7. Новиков Ф. А. Дискретная математика для программистов. СПб.: Питер, 2009.
8. Шевелев Ю.П. Дискретная математика. — СПб.: Лань, 2008. ЭБС Лань http://e.lanbook.com/books/element.php?pl1_id=437
9. Яблонский С. В. Введение в дискретную математику: учебное пособие. М.: Высш. шк., 2008.

дополнительная литература:

1. Белоусов, А. И. Дискретная математика. М.: Изд-во МГТУ им. Н. Э. Баумана, 2006.
2. Харари, Ф. Теория графов. М.: КомКнига, 2006.
3. Берж, К. Теория графов и ее применения. М.: Изд. иностр. литер., 1962.
4. Грэхем, Р. Конкретная математика. Основание информатики. М.: Мир, 1998.
5. Кирсанов, М. Н. Графы в Maple. М.: Физматлит, 2007.
6. Стенли, Р. Перечислительная комбинаторика. М.: Мир, 1990.
7. Хаггарты, Р. Дискретная математика для программистов. М. Техносфера, 2012.
8. Ху, Т. Ч. Комбинаторные алгоритмы. Нижний Новгород: Изд-во НГУ им. Н. И. Лобачевского, 2004.


```

from math import sin, cos, pi
import pylab, random

def matrGr_I(N): # интерактивная инициализация матрицы графа
# Вход: N - порядок графа
# Выход: матрица инцидентности графа
    M = [[0 for x in range(N)] for y in range(N)]
    for row in range(N):
        for col in range(N):
            if row == col: M[row][col] = 0
            else:
                print('M(',row, ',',col,')=')
                M[row][col]=int(input())
    return M

def matrGr_R(N): # инициализация матрицы графа случайными числами
# Вход: N - порядок графа
# Выход: матрица инцидентности графа
    M = [[0 for x in range(N)] for y in range(N)]
    for row in range(N):
        for col in range(row,N):
            if row==col:M[row][col]=0
            else:
                M[row][col]=random.randint(1, 1000)%2
                M[col][row]=M[row][col]
    return M

```