Министерство образования Республики Беларусь

Учреждение образования

«Брестский государственный технический университет»

Кафедра ИИТ

Лабораторная работа №2

По дисциплине «Теоретические и множественные основы интеллектуальных систем»

Тема: «Комбинаторика»

Выполнил:

Студент 1 курса

Группы ИИ-21(1)

Кирилович А.А.

Проверил:

Глущенко Т. А.

Брест 2022

1. Сколько положительных целых чисел, меньших *1001*, делятся на *2*, *3* или *5*?

Пусть множество А – множество целых положительных чисел меньших 1001, которые делятся на 2 и 3; В – множество целых положительных чисел меньших 1001, которые делятся на 5. Вычислим мощность множества А: число, входящее в это множество должно делится и на 2, и на 3, то есть оно должно делится на 6; 1001/6 = 166 (ост. 5); мощность множества А равна 166. Вычислим мощность множества В: 1001/5 = 200 (ост. 1); мощность множества В равна 200. Также вычислим количество всех чисел, которые делятся и на 2, и на 3, и на 5: число, входящее в это множество должно делится и на 2, и на 3, и на 5, то есть оно должно делится на 30; 1001/30 = 33 (ост. 11); количество таких чисел равно 33. Используя формулу включений и исключений, найдем мощность пересечения множеств А и В: 166 + 200 – 33 = 333. Ответ: 333.

1. Сколькими способами можно расставить в ряд для фотографирования пять мальчиков и шесть девочек, если ни две девочки, ни два мальчика не должны стоять рядом?

Возможна только одна схема расстановки мальчиков и девочек: по очереди ставить девочек и мальчиков, причем девочки занимают первое и последние места в ряду, в противном случае девочки будут стоять возле девочек или мальчики будут стоять возле мальчиков, что невозможно по условию. Таким образом, чтобы посчитать общее количество вариантов расстановок, нужно посчитать отдельно количество перестановок девочек (6!) и мальчиков (5!), а потом применить правило умножения (6! \* 5!). Ответ: 6! \* 5!.

1. В зоомагазине продаются *5* черепах, *7* ящериц и *12* мышей. Сколько существует способов выбрать себе *2* черепахи, *3* ящерицы и *5* мышей?

Использую формулы сочетания и правила умножения найдем искомый результат:

Ответ:

1. Решить линейное однородное рекуррентное соотношение *2-го* порядка:  
   

Код:

def a(n):

    if n == 0:

        return 2

    if n == 1:

        return 6

    return -2\*a(n-1) - a(n-2)

print(a(5))

Вывод: 38.

1. Разложить 

Программа 1 (1):

import time

start\_time = time.time()

def lex(count, pos, p, used):

    if pos == count:

        for i in range(count):

            print(str[p[i]], end='')

        print('')

    for i in range(count):

        if not used[i]:

            used[i] = True

            p[pos] = i

            lex(count, pos + 1, p, used)

            p[pos] = 0

            used[i] = False

def antylex(count, pos, p, used):

    if pos == -1:

        for i in range(count - 1, -1, -1):

            print(str[p[i]], end='')

        print('')

    for i in range(count - 1, -1, -1):

        if not used[i]:

            used[i] = True

            p[pos] = i

            antylex(count, pos - 1, p, used)

            p[pos] = 0

            used[i] = False

str = input()

count = len(str)

p = [0 for i in range(count)]

used = [False for i in range(count)]

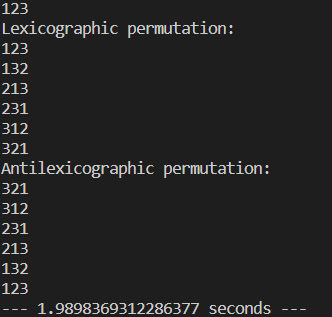
print("Lexicographic permutation:")

lex(count, 0, p, used)

print("Antilexicographic permutation:")

antylex(count, count-1, p, used)

print("--- %s seconds ---" % (time.time() - start\_time))



Программа 1 (1):

import time

start\_time = time.time()

def next\_permutation(sequence, compare) -> bool:

    count = len(sequence)

    i = count

    # Этап № 1

    while True:

        if i < 2:

            return False # Перебор закончен

        i -= 1

        if compare(sequence[i - 1], sequence[i]):

            break

    # Этап № 2

    j = count

    while j > i and not compare(sequence[i - 1], sequence[j - 1]):

        j -= 1

    sequence[i - 1], sequence[j - 1] = sequence[j - 1], sequence[i - 1]

    # Этап № 3

    j = count

    while i < j - 1:

        j -= 1

        sequence[i], sequence[j] = sequence[j], sequence[i]

        i += 1

    return True

def less(value\_0, value\_1) -> bool:

    return value\_0 < value\_1

def greater(value\_0, value\_1) -> bool:

    return value\_0 > value\_1

count = int(input())

sequence = list(range(1, count + 1))

print("Lexicographic permutation:")

permutation\_found = True

while permutation\_found:

    print(sequence)

    permutation\_found = next\_permutation(sequence, less)

print("Antilexicographic permutation:")

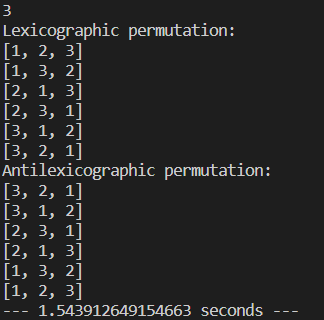
permutation\_found = True

while permutation\_found:

    print(sequence)

    permutation\_found = next\_permutation(sequence, greater)

print("--- %s seconds ---" % (time.time() - start\_time))



Программа 2:

permutations = []

def lex(count, pos, p, used):

    global permutations

    if pos == count:

        s = ''

        for i in range(count):

            s += str[p[i]]

        permutations.append(s)

    for i in range(count):

        if not used[i]:

            used[i] = True

            p[pos] = i

            lex(count, pos + 1, p, used)

            p[pos] = 0

            used[i] = False

print("Enter the string: ")

str = input()

count = len(str)

p = [0 for i in range(count)]

used = [False for i in range(count)]

lex(count, 0, p, used)

print("Enter the subset power: ")

k = int(input())

for i in range(len(permutations)):

    permutations[i] = permutations[i][:k]

permutations\_k = []

for item in permutations:

    if item not in permutations\_k:

        permutations\_k.append(item)

print(\*permutations\_k)

