МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ИНДИВИДУАЛЬНАЯ РАБОТА №11**

**Дисциплина: Комбинаторный анализ**

**Вариант №3**

Работу выполнил: Вишняков Д. И.

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: Руденко О. В.

Краснодар

2024

**Задача 1**

*Условие:*

Дано множество A = { a, b, c, d, e, f, g, h, j, k}. Построить все возможные способы разбить его на 5 неименованных множеств.

*Описание программы:*

Для решения задачи опишем одну функцию generate\_partitions() для генерации всех разбиений множества s на k непустых множеств

В качестве параметров зададим s – исходное множество, k – количество непустых множеств, current – текущее разбиение.

Также описана функция to\_file() для записи разбиений в файл.

Создаем список текущего разбиения и проверяем условия выхода из функции, и сортируем исходный кортеж. Далее перебираем возможные индексы элементов для добавления элемента в подмножества или создаем новое.

*Ход решения:*

Для начала решим задачу математически и выведем формулу для того, чтобы реализовать ее программно:

Введем понятие неименного множества. Это множество, в котором порядок не важен. Получим, что данная задача будет решаться через число Стирлинга 2-го рода. Имеем рекурсивную формулу: S(n, k) = S(n – 1, k - 1) + kS(n – 1, k). Нам необходимо посчитать S(10, 5). Принимаем факт, что S(k, k) = 1. Рекурсивно просчитав эти значения, получим: 42 525.

Ручной счет: 42 525

Программный счет: 42 525

**Задача 2**

*Условие:*

Дано множество A = { a, b, c, d, e, f, g, h, j, k}. Построить все возможные способы разбить его на 5 именованных множеств

*Описание программы:*

Для решения задачи опишем одну функцию generate\_partitions() для генерации всех разбиений множества s на k непустых множеств

В качестве параметров зададим s – исходное множество, k – количество непустых множеств.

Также описана функция to\_file() для записи разбиений в файл.

Если исходное множество пусто, то возвращаем пустой кортеж. Если же количество множеств равно 1, то вернем кортеж из кортежа. Далее создаем возможные комбинации и получаем разницу между исходным множеством и комбинации.

Также в решении были применен 1 метода библиотеки itertools: combinations: метод, который генерирует всевозможные сочетания элементов списка.

*Ход решения:*

Для начала решим задачу математически и выведем формулу для того, чтобы реализовать ее программно:

Введем понятие неименного множества. Это множество, в котором порядок важен. Следовательно, получим исходное множество, в котором все элементы различны. Задача сводится к предыдущей, за исключением того, что нам надо найти еще и перестановки элементов: 5! Получим, что данная задача будет решаться через число Стирлинга 2-го рода. Имеем рекурсивную формулу: S(n, k) = S(n – 1, k - 1) + kS(n – 1, k). Нам необходимо посчитать S(10, 5). Принимаем факт, что S(k, k) = 1. Рекурсивно просчитав эти значения, получим: 42 525. Также домножим на 5!

Ручной счет: 5 103 000

Программный счет: 5 103 000

**Задача 3**

*Условие:*

Дано множество A = { a, a, a, a, a, a, a, a, a, a}. Построить все возможные

способы разбить его на 5 неименованных множеств

*Описание программы:*

Для решения задачи опишем функцию partitions() для генерации всех разбиений числа n на k положительных слагаемых (в данном случае состав множества не имеет значения, нас интересует только количество).

В качестве параметров зададим n – исходное число, k – количество частей, current – текущее разбиение, start – минимальное значение слагаемых.

Также описана функция to\_file() для записи разбиений в файл.

Создаем список текущего разбиения и возвращаем текущее разбиение, преобразованное в кортеж. Далее получаем текущее разбиение и само число, которые являются решением.

Опишем функцию generate\_partitions() для генерации всех разбиений множества с повторениями на неименованные множества. В качестве параметров зададим n – общее количество элементов, subsets – количество подмножеств, на которые надо разбить. Функция возвращает разбиение в виде кортежа чисел, показывающее количество элементов в каждом подмножестве.

*Ход решения:*

Наше множество имеет неразличимые элементы. Переберем все варианты вручную:

1) 1 + 1 + 1 + 1 + 6 = 10

2) 1 + 1 + 1 + 2 + 5 = 10

3) 1 + 1 + 1 + 3 + 4 = 10

4) 1 + 1 + 2 + 2 + 4 = 10

5) 1 + 1 + 2 + 3 + 3 = 10

6) 1+ 2 + 2 + 2 + 3 = 10

7) 2 + 2 + 2 + 2 + 2 = 10

Здесь каждое слагаемое означает количество элементов в подмножестве, а 10 – итоговое количество элементов в множестве.

Ручной счет: 7

Программный счет: 7

**Задача 4**

*Условие:*

Дано множество A = { a, a, a, a, a, a, a, a, a, a}. Построить все возможные

способы разбить его на 5 именованных множеств

*Описание программы:*

Для решения задачи опишем функцию partitions() для генерации всех разбиений числа n на k положительных слагаемых с учетом их порядка (в данном случае состав множества не имеет значения, нас интересует только количество).

В качестве параметров зададим n – исходное число, k – количество частей.

Также описана функция to\_file() для записи разбиений в файл.

Вернем пустой кортеж, если количество частей равно 0. В случае 1 вернем кортеж, содержащий n. Далее получим все разбиения числа n на k положительных частей.

Опишем функцию generate\_partitions() для генерации всех разбиений множества с повторениями на неименованные множества. В качестве параметров зададим n – общее количество элементов, subsets – количество подмножеств, на которые надо разбить. Для каждого элемента из разбиений создает кортеж, состоящий из кортежей, содержащих символ «а», количество которых равно количеству элементов для данного подмножества.

*Ход решения:*

Нам необходимо решить задачу Муавра: x1 + x2 + x3 + x4 + x5 = 10, где каждый x\_i >= 1. Сделаем замену y\_i = x\_i – 1 => y\_i >= 0. Получим: y1 + y2 + y3 + y4 + y5 = 5. Общее количество найдем: N = C(n – 1, k – 1) = C(10 – 1, 5 – 1) = 126

Ручной счет: 126

Программный счет: 126

**Задача 5**

*Условие:*

Построить все подстановки на множестве цифр, содержащих 5 независимых

цикла.

*Описание программы:*

Для решения задачи опишем функцию int\_partitions() для генерации всех разбиений числа n на k положительных слагаемых с учетом их порядка.

В качестве параметров зададим n – исходное число, k – количество частей.

Также описана функция to\_file() для записи разбиений в файл.

Вернем пустой кортеж, если количество частей равно 0. В случае 1 вернем кортеж, содержащий n. Далее получим все разбиения числа n на k положительных частей.

Опишем функцию generate\_cycles() для генерации всех циклов для заданных размеров. В качестве параметров зададим sizes – размеры циклов, digits – массив цифр. Перебираем все комбинации текущего цикла и ищем оставшиеся цифры, которые не вошли в цикл. Далее перебираем все перестановки и возвращаем список из циклов.

Опишем функцию part\_cycles() для генерации всех циклов для заданных размеров. В качестве параметров зададим cycles – набор циклов. Определяем максимальную цифру в цикле и создаем список перестановок. Каждому элементу цикла присваиваем следующий.

Опишем функцию generate\_permutations() для генерации всех подстановок с заданным количеством независимых циклов. В качестве параметров зададим digits – массив цифр, count\_cycles – количество циклов. Перебираем циклы разных размеров. Прогнав все циклы для соответствующих размеров, получаем перестановки.

Также в решении были применен 1 метода библиотеки itertools: combinations: метод, который генерирует всевозможные сочетания элементов списка.

*Ход решения:*

Для начала разобьем число 10 на 5 слагаемых. Это было сделано в 3 задаче. Выберем 1 символ для первого цикла. Поскольку цикл с одним элементом, то можно менять его местами. Имеем для каждого из вариантов:

1) (С(10, 1) \* С(9, 1) \* С(8, 1) \* С(7, 1) \* 5!) / 4! = 25 200

2) (С(10, 1) \* С(9, 1) \* С(8, 1) \* С(7, 2) \* (5 – 1)!) / 3! = 60 480

Продолжая так до 7 и сложив значения, получим: 269 325

Ручной счет: 269 325

Программный счет: 269 325

**Задача 5**

*Условие:*

Построить все связные графы из 8 вершин

*Описание программы:*

Для решения задачи опишем функцию combinations() для вычисления сочетаний из n по k.

Функция возвращает значение, которое считается как n! / ((n – k) ! \* k!)

Опишем функцию power() для вычисления выражения 2^(С(p, 2)).

Функция возвращает значение которые считается как 2^(p \* (p - 1) / 2)

Опишем функцию С() для вычисления C\_p по рекуррентной формуле. Вводим начальные элементы для рекурсии и далее вычисляем рекуррентную формулу.

Также в решении были применен 1 метода библиотеки math: factorial: метод, который считает факториал числа и возвращает значение после вычисления.

*Ход решения:*

Для нахождения числа связных помеченных графов воспользуемся формулой: C\_p = 2^C(p, 2) – 1/p \* sum\_{k=1}^{p-1}k \* C(p, k) \* 2^C(p – k, 2) \* C\_k. При этом нам известен C\_1 = 1, C\_2 = 1. Нам нужно найти все связные графы с 8 вершинами. Значит нам необходимо посчитать C\_8. Выполним это по рекуррентной формуле.

Ручной счет: 251 548 592

Программный счет: 251 548 592

**Задача 6**

*Условие:*

Построить все связные графы из 8 вершин

*Описание программы:*

Для решения задачи опишем функцию combinations() для вычисления сочетаний из n по k.

Функция возвращает значение, которое считается как n! / ((n – k) ! \* k!)

Опишем функцию power() для вычисления выражения 2^(С(p, 2)).

Функция возвращает значение которые считается как 2^(p \* (p - 1) / 2)

Опишем функцию С() для вычисления C\_p по рекуррентной формуле. Вводим начальные элементы для рекурсии и далее вычисляем рекуррентную формулу.

Также в решении были применен 1 метода библиотеки math: factorial: метод, который считает факториал числа и возвращает значение после вычисления.

*Ход решения:*

Для нахождения числа связных помеченных графов воспользуемся формулой: C\_p = 2^C(p, 2) – 1/p \* sum\_{k=1}^{p-1}k \* C(p, k) \* 2^C(p – k, 2) \* C\_k. При этом нам известен C\_1 = 1, C\_2 = 1. Нам нужно найти все связные графы с 8 вершинами. Значит нам необходимо посчитать C\_8. Выполним это по рекуррентной формуле.

Ручной счет: 251 548 592

Программный счет: 251 548 592

**Задача 7**

*Условие:*

Построить все эйлеровы графы из 8 вершин

*Описание программы:*

Для решения задачи опишем функцию combinations() для вычисления сочетаний из n по k.

Функция возвращает значение, которое считается как n! / ((n – k) ! \* k!)

Опишем функцию power() для вычисления выражения 2^(С(p – 1, 2)).

Функция возвращает значение которые считается как 2^(p \* (p - 1) / 2)

Опишем функцию С() для вычисления C\_p по рекуррентной формуле. Вводим начальные элементы для рекурсии и далее вычисляем рекуррентную формулу. Добавляем проверку на эйлеров граф.

Также в решении были применен 1 метода библиотеки math: factorial: метод, который считает факториал числа и возвращает значение после вычисления.

*Ход решения:*

Для нахождения числа связных помеченных графов воспользуемся формулой: U\_p = 2^C(p – 1, 2) – 1/p \* sum\_{k=1}^{p-1}k \* C(p, k) \* 2^C(p – k – 1, 2) \* U\_k. При этом нам известен U\_1 = 1, U\_2 = 1, U\_3 = 0. Нам нужно найти все эйлеровы графы с 8 вершинами. Значит нам необходимо посчитать U\_8. Выполним это по рекуррентной формуле.

Ручной счет: 2 097 152

Программный счет: 2 097 152

**Задача 8**

*Условие:*

Построить все графы из 8 вершин, которые можно раскрасить в 8 цветов

*Описание программы:*

Для решения задачи опишем функцию permutations() для вычисления перестановок.

Опишем функцию power() для вычисления выражения 2^(С(p – 1, 2)).

Функция возвращает значение которые считается как 2^(p \* (p - 1) / 2)

В ответе перемножим количество перестановок и результат функции power().

Также в решении были применен 1 метода библиотеки math: factorial: метод, который считает факториал числа и возвращает значение после вычисления.

*Ход решения:*

В задаче нет условия, связанного со структурой графа. Значит имеем общее количество: 2^C(8, 2) = 2^28. Пусть выбраны 8 цветов, которые не повторяются. То есть красим первую вершина в 1 из 8 цветов, вторую в один из 7 цветов и так далее. Таким образом получим формулу 2^28 \* P(8).

Ручной счет: 10 823 317 585 920

Программный счет: 10 823 317 585 920

**Задача 9**

*Условие:*

Построить все эйлеровы графы из 8 вершин и 13 ребер

*Описание программы:*

Для решения задачи опишем функцию combinations() для вычисления сочетаний из n по k.

Опишем функцию wp\_x(), в которой мы раскладываем наш полученный многочлен в ряд Тейлора для получения коэффициента при x в степени количества ребер.

Также в решении были применен 1 метода библиотеки math: factorial: метод, который считает факториал числа и возвращает значение после вычисления.

*Ход решения:*

Для нахождения числа связных помеченных графов воспользуемся формулой: W\_p(x) = 1/(2^p) \* (1 + x)^C(p, 2) \* sum\_{n=0}^p C(p, n) \* ((1 – x) / (1 + x))^(n(p – n)). В результате раскрытия этой рекуррентной формулы, получим многочлен. Для того, чтобы получить ответ, необходимо найти коэффициент при x^13, так как по условию дано 13 ребер.

Ручной счет: 287 600

Программный счет: 287 600