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 №10**

**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:*

Составить программный продукт для ПЭВМ, который составляет и выводит в файл все слова алфавита A = { a, b, c, d, e, f, g, h, j, k } длины n, в которых ровно две буквы повторяются по 2 раза, и одна буква k раз, остальные буквы не повторяются.

*Описание программы:*

Для начала определим класс Combinatorica. Данный класс содержит метод Се для генерации всех возможных комбинаций длиной k с повторениями из заданного алфавита alfabet, который содержит n символов. Также данный метод содержит рекурсивный метод Backtracking для генерации комбинаций с повторениями. Данный метод позволяет перебирать все возможные варианты и добавляет на каждом шаге новый символ.

Суть алгоритма Backtracking (с возвратом):

1) поиск решения путем перебора: Алгоритм перебирает различные варианты для нахождения всех возможных решений;

2) построение решений по шагам: Решение строится шаг за шагом, добавляя элемент к текущему решению;

3) отмена последнего шага: если текущее частичное решение не приводит к полному решению, то последний шаг отменяется, и алгоритм пробует другой вариант;

4) рекурсия для управления процессом, здесь она используется для управления процессом перебора и отката;

Далее в main опишем обрабатку списка строк allCombinations и оставляем только те, которые удовлетворяют определенным условиям. Цикл foreach проходит по всем значениям (количествам вхождений) cnt в словаре map.

Суть заключается в том, что мы берем всевозможные комбинации символов и оставляет только те, которые соответствуют следующим требованиям:

1) содержит два символа, которые встречаются ровно два раза;

2) содержит ровно один символ, который встречается povtor раз;

3) содержит любое количество символов, которые встречаются ровно один раз;

4) не содержит символов, которые встречаются какое-либо другое количество раз;

*Ход решения:*

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

*Выберем повторяющиеся буквы:*

1) Нам нужно выбрать 2 буквы из 10 для повторения 2 раза, это можно сделать C(10, 2) способами.

2) Затем нам нужно выбрать 1 букву из оставшихся 8 для повторения k раз: C(8, 1).

*Размещение букв:*

1) Разместим 2 буквы, которые повторяются 2 раза: С(n, 2) – выбор места для первой позиции, С(n – 2, 2) – для второй позиции.

2) Разместим k одинаковых букв: C(n – 4, k)

*Общее число перестановок:* n! / (2! \* 2! \* k! \* (n – 4 – k )!). Далее разместим другие символы. Оставшиеся 7 букв нужно разместить на n – 4 – k позиций: 7! / (11 – n + k)! В данном случае если n – 4 – k > 7, то решения не будет. Таким образом получим общую формулу:

C(10, 2) \* C(8, 1) \* n! / (2! \* 2! \* k! \* (n – 4 – k )!) \* 7! / (11 – n + k)!

|  |  |  |
| --- | --- | --- |
|  | Ручной счет | Программный счет |
| Пример 1: n = 7, k = 3 | 75 600 | 75 600 |
| Пример 2: n = 6, k = 1 | 226 800 | 226 800 |
| Пример 3: n = 7, k = 1 | 529 200 | 529 200 |

**Задача 2**

*Условие:*

Составить программный продукт для ПЭВМ, который составляет и выводит в файл все слова алфавита A = { a, b, c, d, e, f, g, h, j, k } длины n, в которых одна буква повторяется не более k раз, одна буква повторяется ровно k+1 раз, одна буква повторяется или k+2 или k+3 раз, остальные буквы не повторяются.

*Описание программы:*

Определим класс Combinatorica. Данный класс содержит метод Се для генерации всех возможных комбинаций длиной k с повторениями из заданного алфавита alfabet, который содержит n символов. Также данный метод содержит рекурсивный метод Backtracking для генерации комбинаций с повторениями. Данный метод позволяет перебирать все возможные варианты и добавляет на каждом шаге новый символ.

Суть алгоритма Backtracking (с возвратом):

1) поиск решения путем перебора: Алгоритм перебирает различные варианты для нахождения всех возможных решений;

2) построение решений по шагам: Решение строится шаг за шагом, добавляя элемент к текущему решению;

3) отмена последнего шага, а именно если текущее частичное решение не приводит к полному решению, то последний шаг отменяется и алгоритм пробует другой вариант;

4) рекурсия для управления процессом: Рекурсия используется для управления процессом перебора и отката;

Теперь опишем то, как происходит в main фильтрация строк allCombinations на основе количества вхождений каждого символа в строке. А именно опишем суть работы foreach. Берем каждую строку из списка allCombinations и считаем, сколько раз там встречаются определенные символы. А затем проверяем, чтобы в строке находилось:

1) ровно один символ, встречающийся не более чем k раз;

2) ровно один символ, встречающийся ровно k+1 раз;

3) ровно один символ, встречающийся либо k+2, либо k+3 раз;

4) остальные символы должны встречаться не более 1 раза;

Если все эти условия, описанные выше, соблюдены, то строка добавляется в список validWords.

*Ход решения:*

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

Для начала выберем 3 различные буквы: С(10, 3)

Имеем 6 *вариантов* распределения повторений:

|  |  |  |
| --- | --- | --- |
| <= k | k + 1 | k + 2 или k + 3 |
| a | b | c |
| a | c | b |
| b | a | c |
| b | c | a |
| c | a | b |
| c | b | a |

Пусть есть выбранные буквы, например, a, b, c. Тогда

1) а может повторяться i раз, где 0 <= i <= k

2) b может повторяться k раз

3) c может повторяться k + 2 или k + 3 раза

*Теперь выберем места для каждого случая:*

1) a – i повторений, b – k + 1 повторение, c – k + 2 повторений:

n! / (i! \* (k + 1)! \* (k + 2)! \* (n – i – 2k – 3)!)

2) a – i повторений, b – k + 1 повторение, c – k + 3 повторений:

n! / (i! \* (k + 1)! \* (k + 3)! \* (n – i – 2k – 4)!)

Количество оставшихся букв: 7, их нужно разместить на (n – i – 2k - 3) или на (n – i – 2k – 4) позиции.

*Количество вариантов перестановок:*

1) P(7, n – i – 2k - 3) = 7! / (10 – n + i + 2k)!, при n – i – 2k – 3 <= 7

2) P(7, n – i – 2k - 4) = 7! / (11 – n + i + 2k)!, при n – i – 2k – 4 <= 7

*Получим итоговую формулу:*

C(10, 3) \* sum(n! / (i! \* (k + 1)! \* (k + 2)! \* (n – i – 2k – 3)!) \* 7! / (10 – n + i + 2k)!) + sum(n! / (i! \* (k + 1)! \* (k + 3)! \* (n – i – 2k – 4)!) \* 7! / (11 – n + i + 2k)!), 0 <= i <= 1

|  |  |  |
| --- | --- | --- |
|  | Ручной счет | Программный счет |
| Пример 1: n = 6, k = 1 | 43 200 | 43 200 |
| Пример 2: n = 8, k = 2 | 201 600 | 201 600 |
| Пример 3: n = 7, k = 3 | 0 | 0 |

**Задача 3**

*Условие:*

Составить программный продукт для ПЭВМ, который составляет и выводит в файл все слова алфавита A = { a, b, c, d, e, f, g, h, j, k} длины 10, состоящих из 4 различных букв.

*Описание программы:*

Метод GenerateWords предназначен для генерации всех возможных слов определенной длины из заданного алфавита, при этом учитывается, что в каждом слове должно быть определенное количество уникальных букв. Создается список validWords, который будет содержать все удовлетворяющие условиям слова. Используем также очередь queue для хранения текущих строк, начиная с пустой строки. Метод выполняет цикл, который будет продолжатся до тех пор, пока в очереди есть элементы. Извлекаем currentWord из очереди, а затем определяется его длина. А азтем необходимо проверить длину слова, а затем создать множество, чтобы подсчитать в него уникальные слова. Если же их количество совпадет с uniqueLetters, слово добавится в список. А затем происходит генерация новых слов. В конце проверяем количество уникальных символов. Если оно не превышает uniqueLetters, слово добавляется в очередь для обработки. После того как очередь опустеет возращаем список найденных слов.

*Ход решения:*

Для начала выберем 4 буквы из 10: С(10, 4). Подсчитаем все возможные расположения букв в слове: 4^10.

*Воспользуемся формулой включений-исключений и рассчитаем количество слов с менее чем 4-мя одинаковыми буквами:*

1) выберем 3 буквы из 4 возможных и разместим на 10 позициях: C(4, 3) \* 3^10

2) выберем 2 буквы из 4 возможных и разместим на 10 позициях: C(4, 2) \* 2^10

3) выберем 1 букву из 4 возможных и разместим на 10 позициях: C(4, 1) \* 1^10

*Получим:* 4^10 – C(4, 3) \* 3^10 – C(4, 2) \* 2^10 – C(4, 1) \* 1^10

*Разместим буквы на места и получим:* С(10, 4) \* (4^10 – C(4, 3) \* 3^10 – C(4, 2) \* 2^10 – C(4, 1) \* 1^10)

Ручной счет: 171 889 200

Программный счет: 171 889 200

**Задача 4**

*Условие:*

Составить программный продукт для ПЭВМ, который составляет и выводит в файл все целые неотрицательные решения уравнения x1 + x2 + x3 + x4 + x5 + x6 + x7 = 60, такие, что x1 < 9 , x2 > 8 , x3 < 7 , x4 > 6 , x5 < 5 , x6 > 3 , x7 < 3.

*Описание программы:*

Разберем метод для генерации всех возможных решений для определенного набора условий. Суть метода заключается в том, чтобы найти все комбинации 7 целых неотрицательных чисел, сумма которых равна 60 и удовлетворяющих ограничениям. Метод GenerateSolutions начинает решение задачи, создавая пустой массив x и вызывая рекурсивный метод Generate. Данный метод, в свою очередь, строит все возможные комбинации целых чисел от 0 до 60 рекурсивно в массиве currentX. На каждом уровне рекурсии, выбирается значение для текущего элемента массива, с учетом диапазона ограничений, который зависит от номера уровня. Если же на каком-то из уровней комбинация не подходит, возвращаемся на предыдущий уровень, для выбора других значений (backtracking). После того как все элементы массива заполнены, а также если сумма всех элементов равна 60, и все дополнительные условия выполнены, то массив сохраняется в список решений. Затем продолжаем перебирать все комбинации и проверять их рекурсивно. В итоге нам вернется список всех корректных комбинаций.

*Ход решения:*

*Для начала изменим интервалы условий и выполним замену:*

x1 < 9 => x1 <= 8 => y1 = x1

x2 > 8 => x1 >= 9 => y2 = x2 - 9

x3 < 7 => x3 <= 6 => y3 = x3

x4 > 6 => x4 >= 7 => y4 = x4 - 6

x5 < 5 => x5 <= 4 => y5 = x5

x6 > 3 => x1 >= 4 => y6 = x6 - 4

x7 < 3 => x7 <= 2 => y7 = x7

*Получим уравнение:* y1 + y2 + y3 + y4 + y5 + y6 + y7 = 40

*Введем обозначения:*

S – множество всех решений, без ограничений

A1 – множество решений, где y1 >= 9

A2 – множество решений, где y3 >= 7

A3 – множество решений, где y5 >= 5

A4 – множество решений, где y7 >= 3

*По формуле включений-исключений, нужно найти:* S \ (A1 \/ A2 \/ A3 \/ A4) = S – sum(Ai) + sum(Ai /\ Aj) – sum(Ai /\ Aj /\ Ak) + (A1 /\ A3 /\ A5 /\ A7)

*Сделаем подсчеты для каждого слагаемого:*

1) sum(Ai) = C(37, 6) + C(39, 6) + C(41, 6) + C(43, 6)

2) sum(Ai /\ Aj) = C(30,6) + C(26,6) + C(28,6) + C(28,6) + C(30,6) + C(32,6)

3) sum(Ai /\ Aj /\ Ak) = C(21,6) + C(23,6) + C(25,6) + C(23,6)

4) A1 /\ A3 /\ A5 /\ A7 = C(16, 6)

5) S = C(46, 6)

*Итог:* C(46, 6) – C(37, 6) – C(39, 6) – C(41, 6) – C(43, 6) + C(30,6) + C(26,6) + C(28,6) + C(28,6) + C(30,6) + C(32,6) – C(21,6) – C(23,6) – C(25,6) – C(23,6) + C(16, 6)

Ручной счет: 475 020

Программный счет: 475 020

**Задача 5**

*Условие:*

Составить программный продукт для ПЭВМ, который составляет и выводит в файл все слова русского алфавита, составленные перестановкой букв слова ПРЕДОПРЕДЕЛЕННОСТЬ.

*Описание программы:*

Метод GeneratePermutations предназначен для генерации всех уникальных перестановок заданного слова и записи этих перестановок в файл. Преобразуем входную строку в массив. Отсортируем его, чтобы начать с наименьшей комбинации, а также для гарантии уникальности перестановок.

Сохраним только уникальные перестановки используя uniquePermutations.

Затем происходит генерация перестановок. Метод NextSet используется для поиска следующей перестановки. Уникальная перестановка, которую получаем из массива символов, добавляется в множество, и процесс продолжается до тех пор, пока новые перестановки будут существовать. Затем результат необходимо записать в файл. Таким образом, метод позволяет генерировать и сохранять все уникальные перестановки заданного слова, используя алгоритм следующей перестановки, а также позволяет избегать дубликатов

*Ход решения:*

Количество символов: 18

Определим количество букв в слове:

П – 2

Р – 2

Е – 4

Д – 2

О – 2

Л – 1

Н – 2

С – 1

Т – 1

Ь – 1

Воспользуемся формулой перестановок с повторениями:

P(n, n1, n2, …, nm) = n! / (n1! \* n2! \* … \* nm)

Подставим значения: 18! / (2! \* 2! \* 4! \* 2! \* 2! \* 2!)

Ручной счет: 8 336 424 096 000

Программный счет: 8 336 424 096 000