**Документальний опис алгоритму та задачі "Trees Made to Order"**

Опис задачі: Задача "Trees Made to Order" полягає в нумерації бінарних дерев та виведенні дерева за його номером. Кожне бінарне дерево отримує унікальний номер на основі кількості вершин та номерів його лівого та правого піддерев. Номери дерев присвоюються за таким принципом:

* Пусте дерево має номер 0.
* Дерево з однією вершиною має номер 1.
* Усі бінарні дерева з однаковою кількістю вершин мають номери, менші за номери дерев з більшою кількістю вершин.
* Серед дерев з однаковою кількістю вершин, дерево з меншим номером лівого піддерева матиме менший номер, ніж дерево з більшим номером лівого піддерева.
* Серед дерев з однаковою кількістю вершин та однаковим номером лівого піддерева, дерево з меншим номером правого піддерева матиме менший номер, ніж дерево з більшим номером правого піддерева.

Алгоритм вирішення задачі базується на використанні чисел Каталана для підрахунку кількості можливих бінарних дерев з певною кількістю вершин та рекурсивному обчисленні номерів лівого та правого піддерев.

Структура проекту:

* constants.py: Файл для зберігання константи NODE\_NUM\_MAX, яка визначає максимальну кількість вершин у бінарному дереві.
* binary\_tree\_printer.py: Файл з класом BinaryTreePrinter, який відповідає за нумерацію бінарних дерев та виведення дерева за заданим номером.
* main.py: Файл з головною функцією main(), яка зчитує вхідні дані, створює об'єкт BinaryTreePrinter, викликає методи для виведення дерев та обробляє вихідні дані.
* test\_binary\_tree\_printer.py: Файл з тестами для класу BinaryTreePrinter, що перевіряє коректність його роботи.

Алгоритм:

1. Ініціалізація (\_initialize):
   * Обчислюємо кількість можливих дерев з певною кількістю вершин (tree\_num) на основі чисел Каталана.
     + Використовуємо рекурентну формулу для чисел Каталана: C(n) = sum(C(i) \* C(n-i-1)) для i від 0 до n-1, де C(n) - n-те число Каталана.
     + Числа Каталана дають кількість структурно унікальних бінарних дерев з n вершинами.
   * Обчислюємо сумарну кількість дерев до певної кількості вершин (tree\_num\_sum).
     + tree\_num\_sum[i] зберігає суму tree\_num[j] для всіх j від 0 до i.
     + Це дозволяє швидко визначати діапазон номерів дерев з певною кількістю вершин.
   * Ініціалізуємо tree\_num[0] та tree\_num\_sum[0] значенням 1, оскільки існує лише одне пусте дерево.
2. Виведення дерева за номером (print\_tree та \_print\_tree\_helper):
   * Визначаємо кількість вершин node\_num у дереві на основі заданого номера order\_num, використовуючи tree\_num\_sum.
     + Шукаємо найбільше значення i, для якого tree\_num\_sum[i] <= order\_num.
     + Це значення i відповідає кількості вершин у дереві з номером order\_num.
   * Рекурсивно обчислюємо номери лівого та правого піддерев:
     + order\_before - кількість дерев з node\_num вершинами, але з меншою кількістю вершин у лівому піддереві, ніж у поточного дерева.
       - Перебираємо всі можливі варіанти розбиття вершин на ліве та праве піддерева.
       - Для кожного варіанта обчислюємо кількість дерев з такою конфігурацією, використовуючи tree\_num.
       - Сумуємо кількості дерев для всіх варіантів, де кількість вершин у лівому піддереві менша, ніж у поточного дерева.
     + left\_node\_num - кількість вершин у лівому піддереві поточного дерева.
     + left\_order\_num - номер лівого піддерева серед дерев з left\_node\_num вершинами.
       - Обчислюємо як (order\_num - order\_before) // tree\_num[node\_num - 1 - left\_node\_num].
       - Це відповідає номеру рядка в уявній таблиці, де рядки - ліві піддерева, а стовпці - праві піддерева.
     + right\_order\_num - номер правого піддерева серед дерев з node\_num - 1 - left\_node\_num вершинами.
       - Обчислюємо як (order\_num - order\_before) % tree\_num[node\_num - 1 - left\_node\_num].
       - Це відповідає номеру стовпця в уявній таблиці, де рядки - ліві піддерева, а стовпці - праві піддерева.
   * Формуємо результуючий рядок, який представляє структуру дерева:
     + Якщо left\_node\_num не дорівнює 0 (ліве піддерево не пусте), рекурсивно викликаємо \_print\_tree\_helper для лівого піддерева та додаємо його в дужках перед вершиною.
     + Додаємо вершину X до результуючого рядка.
     + Якщо node\_num - 1 - left\_node\_num не дорівнює 0 (праве піддерево не пусте), рекурсивно викликаємо \_print\_tree\_helper для правого піддерева та додаємо його в дужках після вершини.

Обґрунтування алгоритму:

* Використання чисел Каталана для підрахунку кількості можливих дерев з певною кількістю вершин базується на комбінаторних властивостях бінарних дерев.
  + Числа Каталана дають кількість структурно унікальних бінарних дерев з n вершинами.
  + Це дозволяє ефективно обчислювати кількість дерев без явного перебору всіх можливих конфігурацій.
* Рекурсивний підхід для обчислення номерів піддерев та формування результуючого рядка дозволяє обробляти дерева з довільною структурою.
  + Рекурсія природно відображає структуру бінарного дерева та дозволяє обробляти підзадачі для лівого та правого піддерев.
* Попереднє обчислення та зберігання значень tree\_num та tree\_num\_sum оптимізує процес обчислень.
  + Значення tree\_num обчислюються один раз під час ініціалізації та використовуються для швидкого визначення кількості дерев з певною кількістю вершин.
  + Значення tree\_num\_sum дозволяють швидко визначати діапазон номерів дерев з певною кількістю вершин.