

**ВСЕРОССИЙСКИЙ КОНКУРС НАУЧНО-ИССЛЕДОВАТЕЛЬСКИХ,
ПРОЕКТНЫХ И ТВОРЧЕСКИХ РАБОТ ОБУЧАЮЩИХСЯ
«ОБРЕТЁННОЕ ПОКОЛЕНИЕ»**

Направление: Математика и информационные технологии

Тема: Анализ вычислительной сложности задачи Танежи и разработка алгоритмов поиска арифметических представлений чисел

Соискатель: Ведищева Мария Сергеевна

Научный руководитель: Дягилева Анна Владимировна

Место выполнения работы: Кемерово, КузГТУ имени Т. Ф. Горбачева

Аннотация

Статья посвящена исследованию задачи Танежи — известной проблемы занимательной математики, заключающейся в поиске арифметических выражений для всех чисел от 1 до 11111 с использованием строго упорядоченных цифр 1–9 и набора операций (+, −, ×, ÷, ^, конкатенация). В работе впервые проведен систематический теоретико-сложностной анализ задачи: доказана принадлежность проблемы верификации классу NP, а проблемы синтеза — классу NP-трудных задач (посредством полиномиального сведения от задачи разбиения числового множества). Рассмотрены структурные особенности пространства поиска: фрактальность, кластеризация решений, наличие фазовых переходов и граничных эффектов. Разработан программный комплекс на языке Python, реализующий рекурсивный алгоритм с мемоизацией и отсечением ветвей. Проведен вычислительный эксперимент для диапазона 0–500, результаты которого визуализированы в виде «карты сложности», наглядно отражающей неравномерность вычислительных затрат. Выполнено сравнение эффективности разработанного алгоритма с базовым случайным поиском, а также проведена параллель с другой NP-трудной задачей — Max-Cut. Результаты работы подтверждают теоретические выводы и обосновывают необходимость разработки эвристических подходов для поиска представлений больших чисел, включая известную открытую проблему для 10958.

Содержание

ВВЕДЕНИЕ	4
ГЛАВА 1. ТЕОРЕТИКО-СЛОЖНОСТНОЙ АНАЛИЗ ЗАДАЧИ ТАНЕЖИ: СТРУКТУРНЫЕ ОСОБЕННОСТИ И ВЫЧИСЛИТЕЛЬНАЯ ГЕОМЕТРИЯ ПРОСТРАНСТВА ПОИСКА	6
ГЛАВА 2. РАЗРАБОТКА И АНАЛИЗ АЛГОРИТМОВ РЕШЕНИЯ.....	10
2.1 Разработка алгоритма решения задачи Танежи.....	10
2.2 Программа визуализации решения задачи Max-Cut.....	10
2.3 Анализ и сравнение разработанных решений.....	10
ГЛАВА 3. ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ И КАРТА ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ	12
3.1 Методика эксперимента и тестовый стенд	12
3.2 Построение карты сложности и анализ результатов.....	12
ЗАКЛЮЧЕНИЕ.....	14
СПИСОК ЛИТЕРАТУРЫ.....	15

ВВЕДЕНИЕ

В области дискретной математики и теории вычислений особый интерес представляют задачи, находящиеся на стыке комбинаторики и арифметики. В рамках настоящего анализа предлагается рассмотреть суть математической задачи, сформулированной бразильским ученым Индером Танежей. Ее сущность заключается в необходимости последовательного представления всех целых чисел от 1 до 11111 с использованием строго упорядоченной последовательности цифр от 1 до 9 (как в восходящем, так и в нисходящем порядке). Несмотря на кажущуюся простоту, для числа 10958 не удалось найти валидное выражение в порядке возрастания (Приложение 1).

Условия задачи допускают применение базовых арифметических операций:

- Сложение;
- Вычитание;
- Умножение;
- Возведение в степень;
- Деление;
- Использование скобок для изменения порядка действий.

Комбинаторная основа работы заключается в систематическом переборе всех возможных выражений. Для двух различных натуральных чисел a и b допустимы следующие комбинации: $a + b$, $a - b$, $a \times b$, a^b и b^a . Например, для $a = 2$ и $b = 3$ в возрастающем порядке это будут: $2 + 3$, $2 - 3$, 2×3 , 2^3 и 3^2 .

Если рассмотреть три натуральных числа a , b и c (при условии $a < b < c$ или $a > b > c$), спектр возможных комбинаций существенно расширяется. К числу допустимых записей относятся, среди прочего:

$a + b + c$, $a - b - c$, $ab + c$, $ab - c$, $a + bc$, $a - bc$, $(a + b) \times c$, $(a - b) \times c$, $a \times (b + c)$, $a \times b \times c$, $a^{(bc)}$, $(a^b)^c$.

Следует отметить, что выражения $(a^b)^c$ и $a^{(b \times c)}$ тождественны в силу свойств степеней. Нетрудно представить, что с ростом количества цифр в последовательности (с 3 до 4, 5 и вплоть до 9) количество возможных комбинаций достигает миллионов. Ключевым требованием остается неукоснительное соблюдение исходного числового порядка.

Проиллюстрируем вышесказанное примерами репрезентации числа 2025 (Приложение 2):

$$2025 = 12 \times 3 + (4 + 5) \times (6 + 7) \times (8 + 9)$$
$$2025 = 9 \times 8 + 76 + 5^4 \times 3 + 2 \times 1$$

Актуальность: Задача Танежи представляет собой частный случай задачи упаковки выражений — класса проблем, для которого вопрос о вычислительной сложности и существовании эффективных алгоритмов решения остаётся открытым. Несмотря на многочисленные попытки, для числа 10958 до сих пор не найдено представления в возрастающем порядке цифр, что указывает на нетривиальную структуру пространства поиска. Отсутствие формальной классификации задачи в иерархии сложности и необходимость разработки оптимизированных алгоритмов определяют актуальность настоящего исследования.

Цель: проанализировать вычислительную сложность задачи Танежи и разработать оптимизированные алгоритмы для ее решения.

Задачи:

1. Формализовать задачу как проблему разрешимости: существует ли выражение $E \in L$ (где L - язык арифметических выражений с заданными операциями), такое что $E = N$ для заданного N ?
2. Исследовать принадлежность к классам сложности.
3. Разработать алгоритмы на основе:
 - Методов динамического программирования с мемоизацией промежуточных результатов
 - Алгоритмов отсечения нерелевантных ветвей поиска
 - Символьных вычислений и компьютерной алгебры
4. Сравнить эффективность разных подходов по времени выполнения и потреблению памяти
5. Построить "карту сложности" - для каких чисел задача решается легко, а для каких требует максимальных вычислительных ресурсов

Практическая значимость: Разработанные алгоритмы и программные средства могут быть использованы для дальнейших исследований комбинаторно-арифметических задач. Полученные выводы о структуре сложности важны для понимания границ применимости точных алгоритмов и разработки эвристических методов.

ГЛАВА 1. ТЕОРЕТИКО-СЛОЖНОСТНОЙ АНАЛИЗ ЗАДАЧИ ТАНЕЖИ: СТРУКТУРНЫЕ ОСОБЕННОСТИ И ВЫЧИСЛИТЕЛЬНАЯ ГЕОМЕТРИЯ ПРОСТРАНСТВА ПОИСКА

Класс NP представляет собой не просто техническое определение, а фундаментальную структурную характеристику вычислительных проблем, отражающую глубокие связи между комбинаторикой, алгеброй и логикой. Структурный анализ задач этого класса раскрывает закономерности в организации вычислительной сложности.

Задачи класса NP обладают специфической геометрией пространства решений. Если представить множество всех возможных конфигураций задачи как многомерное пространство, то допустимые решения образуют в нём особые структуры — связанные области сложной формы, часто напоминающие фрактальные образования. Например, в задаче выполнимости булевых формул (SAT) пространство решений представляет собой объединение вершин n -мерного булева куба, удовлетворяющих заданной формуле. Эмпирические исследования показывают, что в «типичных» случаях эти области имеют кластерную структуру: решения группируются в тесно связанные кластеры, разделённые областями нерешений.

Ключевая структурная особенность NP-задач — свойство локальной проверяемости. Существование короткого сертификата, проверяемого за полиномиальное время, означает, что глобальное свойство решения может быть верифицировано через анализ локальных ограничений. В задаче о раскраске графа, например, глобальное условие правильной раскраски проверяется через анализ каждого ребра в отдельности. Эта локальность тесно связана с симметриями задачи: многие NP-полные задачи обладают значительной группой симметрий, что позволяет сокращать пространство поиска за счёт факторизации по симметричным конфигурациям.

Структурной особенностью многих NP-полных задач является наличие фазовых переходов — резких изменений в организации пространства решений при варьировании параметров задачи. Например, в задаче выполнимости при определённом отношении числа дизъюнктов к числу переменных вероятность существования решения скачкообразно меняется от близкой к 1 до близкой к 0. В критической точке фазового перехода задачи демонстрируют максимальную вычислительную сложность: среднее время решения алгоритмов полного перебора достигает пика, а структура пространства решений становится наиболее сложной.

Пространства решений многих NP-полных задач обладают свойствами самоподобия: локальные фрагменты структуры повторяют глобальные закономерности. В задаче коммивояжёра, например, оптимальные маршруты для подмножеств городов часто являются фрагментами глобального оптимального маршрута. Это самоподобие позволяет строить иерархические алгоритмы, решающие задачу на разных уровнях детализации.

Структурное единство класса NP проявляется в существовании полных задач, к которым сводятся все остальные задачи класса. Эта сводимость — не просто технический приём, а отражение фундаментального структурного сходства. Задача выполнимости булевых формул (SAT) служит своего рода «универсальным языком» для выражения ограничений любой NP-задачи. Глубокая структурная причина этой универсальности кроется в том, что булевы формулы могут естественным образом кодировать вычисления недетерминированной машины Тьюринга.

Интересной структурной особенностью является неравномерность распределения сложности внутри класса NP. Для многих задач существуют «лёгкие» и «трудные» экземпляры, причём переход между ними может быть очень резким. В задаче о сумме подмножества экземпляры с определёнными арифметическими свойствами чисел решаются значительно проще. Это создает сложный «ландшафт вычислительной сложности» с «долинами» легко решаемых случаев и «пиками» максимальной сложности.

Многие комбинаторные задачи NP могут быть естественным образом представлены как задачи целочисленного линейного программирования. Выпуклая оболочка допустимых решений образует полиэдр, чья комбинаторная структура напрямую связана с вычислительной сложностью задачи. Для NP-трудных задач соответствующие полиэдры имеют экспоненциально много граней, что отражает сложность задачи. Изучение этих полиэдров методами комбинаторной оптимизации позволяет разрабатывать эффективные алгоритмы для конкретных классов экземпляров.

Рассмотренные структурные особенности находят свое непосредственное отражение в задаче Танежи, которая представляет собой идеальную модель для изучения взаимодействия комбинаторных и арифметических аспектов вычислительной сложности. Как будет показано далее, пространство возможных арифметических выражений в задаче Танежи демонстрирует все характерные черты NP-трудных задач: сложную кластерную организацию, фазовые переходы при изменении целевого значения, ярко выраженные граничные эффекты и богатую алгебраическую структуру. Анализ вычислительной геометрии этой задачи позволяет не только установить её формальную сложностную классификацию, но и выявить глубокие структурные закономерности, определяющие практическую разрешимость для различных классов целевых чисел. Особый интерес представляет изучение того, как арифметическая природа операций и жёсткое требование порядка цифр формируют специфическую топологию пространства поиска, сочетающую свойства задач упаковки, разбиения и композиции арифметических выражений

Проведение исчерпывающего теоретико-сложностного анализа задачи Танежи требует многоаспектного рассмотрения, выходящего за рамки стандартной процедуры классификации. Глубина и нетривиальность задачи проявляются уже на уровне ее формализации, где пересекаются комбинаторные, алгебраические и вычислительные аспекты. Предлагаемое исследование направлено не только на установление принадлежности задачи к определенному классу сложности, но и на выявление тех структурных особенностей, которые определяют ее практическую разрешимость и формируют специфическую геометрию пространства поиска.

Фундаментальным методологическим принципом при анализе вычислительной сложности задачи Танежи является необходимость четкого разделения проблемы верификации и проблемы синтеза корректного выражения. Проблема верификации, заключающаяся в проверке соответствия предъявленного арифметического выражения E целевому числу N при условии использования строго упорядоченной последовательности цифр $1, 2, \dots, 9$ и допустимых арифметических операций, демонстрирует несомненную принадлежность к классу NP. Для любого предъявленного сертификата — корректно сформированного арифметического выражения — проверка его валидности осуществляется детерминированным алгоритмом за время, полиномиально зависящее от длины выражения. Данная процедура включает синтаксический анализ на соответствие исходной последовательности цифр, проверку семантической корректности применения операций и собственно вычисление итогового значения с соблюдением стандартных правил приоритета операций. Эффективность этой проверочной процедуры обеспечивается существованием детерминированного алгоритма синтаксического разбора и вычисления арифметических выражений, что полностью удовлетворяет формальным критериям принадлежности к классу NP.

Установление NP-трудности проблемы синтеза выражения требует построения строгого полиномиального сведения известной NP-полной задачи. В качестве базовой задачи целесообразно рассмотреть задачу разбиения числового множества (Number Partition), которая демонстрирует глубокое структурное сходство с задачей Танежи в ее редуцированной форме. Для произвольного экземпляра задачи разбиения множества $S = \{s_1, s_2, \dots, s_k\}$ строится специализированная последовательность цифр, где каждый элемент s_i кодируется блоком из s_i идентичных цифр. При ограничении операционного множества до $\{+, -\}$ и целевом значении 0 , существование решения

задачи Танежи для построенной последовательности оказывается эквивалентным существованию разбиения исходного множества S . Полиномиальная вычислимость такого сведения в сочетании с NP-полнотой исходной задачи разбиения позволяет сделать категорический вывод о том, что даже рестриктивная версия задачи Танежи является NP-трудной.

Следует отметить, что задача о разбиении множества относится к классу NP-полных задач, однако она является псевдополиномиальной, то есть может быть решена за полиномиальное время относительно величины чисел, но не относительно длины их записи. Для более строгого доказательства NP-трудности в сильном смысле целесообразно использовать сведение от задачи 3-SAT, что является перспективным направлением дальнейших исследований. Тем не менее, представленное сведение от задачи разбиения множества убедительно демонстрирует фундаментальную сложность проблемы синтеза.

Особого внимания заслуживает тот факт, что аналогичные стратегии сведения успешно применяются для анализа сложности других комбинаторных задач с арифметической природой. В частности, задача о выполнимости арифметических выражений (Arithmetic Expression Satisfaction) и задача о построении математических ребусов (Mathematical Crossword Construction) демонстрируют сходные сложностные характеристики. Однако задача Танежи обладает существенной спецификой, обусловленной комбинаторной сложностью операционного множества, включающего не только базовые арифметические операции, но и операции, порождающие сложные функциональные зависимости. Эта особенность роднит задачу Танежи с другими вычислительно сложными арифметическими проблемами, такими как задача символического интегрирования или задача упрощения математических выражений, где экспоненциальный рост сложности обусловлен не только комбинаторным разнообразием, но и алгебраической природой преобразований.

Экспоненциальный рост сложности задачи Танежи при увеличении длины последовательности цифр имеет мультипликативную природу и обусловлен сложным взаимодействием нескольких факторов. Каждая позиция для размещения операций создает фактор роста, пропорциональный мощности множества допустимых операций. Особенностью задачи является неаддитивный характер взаимодействия операций — последовательное применение операций умножения, деления и возведения в степень порождает сложные функциональные зависимости, не сводимые к простой композиции. Это создает эффект каскадного усложнения, когда на каждом шаге построения выражения возникает качественно новая вычислительная ситуация.

Арифметическая сложность промежуточных вычислений представляет отдельный и чрезвычайно важный аспект вычислительной сложности задачи. Операции деления порождают рациональные числа с неограниченно растущими знаменателями, что исключает возможность эффективного использования арифметики с плавающей точкой без потери точности и требует применения символьных методов вычислений. Возведение в степень может приводить к появлению трансцендентных чисел иррациональной природы, а также к экспоненциальному росту значений, что создает принципиальные трудности как для численных методов, так и для символьных вычислений. Особую сложность представляет обработка операций, способных порождать комплексные значения или сингулярности — такие как деление на ноль или возведение отрицательных чисел в дробные степени.

Геометрия пространства поиска решений задачи Танежи характеризуется высокой степенью неоднородности и наличием "островов" достижимых значений, разделенных "океанами" недостижимых чисел. Это роднит задачу с другими комбинаторно-арифметическими проблемами, где пространство решений обладает фрактальной структурой. Эмпирические наблюдения показывают, что даже небольшие изменения в целевом числе N могут приводить к качественным изменениям вычислительной сложности — некоторые числа допускают относительно простое

представление, в то время как близлежащие значения могут требовать экспоненциальных вычислительных ресурсов для нахождения решения.

Практическая значимость установленной NP-трудности заключается в том, что она предоставляет теоретическое обоснование наблюдаемой вычислительной сложности и объясняет фундаментальные ограничения для алгоритмов точного решения. Данный результат не исключает возможности разработки эффективных эвристических методов, но указывает на принципиальную невозможность создания универсального полиномиального алгоритма для произвольных входных данных в предположении $P \neq NP$. Более того, структурный анализ задачи показывает, что ее сложность обусловлена не только комбинаторным разнообразием, но и глубокой арифметической природой, что делает задачу Танежи уникальным объектом для исследований на стыке теории сложности, компьютерной алгебры и теории чисел.

Перспективным направлением дальнейших исследований представляется анализ асимптотической плотности достижимых значений и изучение статистических закономерностей в распределении вычислительной сложности. Экспериментальные данные свидетельствуют о наличии нетривиальных зависимостей между арифметическими свойствами целевого числа N и сложностью его представления в рамках задачи Танежи. Выявление этих закономерностей могло бы пролить свет на более общие вопросы, связанные со структурой арифметических выражений и их вычислительной мощностью.

Таким образом, комплексный теоретико-сложностной анализ позволяет не только классифицировать задачу Танежи в терминах иерархии сложности, но и выявить структурные особенности, определяющие направления для разработки специализированных алгоритмов. Установленная NP-трудность проблемы синтеза в сочетании с принадлежностью проблемы верификации к классу NP помещает задачу Танежи в центральную область теоретической информатики, представляющую значительный интерес для дальнейших исследований. Глубина и многоаспектность задачи открывают возможности для плодотворного взаимодействия различных разделов дискретной математики и теории вычислений.

ГЛАВА 2. РАЗРАБОТКА И АНАЛИЗ АЛГОРИТМОВ РЕШЕНИЯ

2.1 Разработка алгоритма решения задачи Танежи

В рамках исследования была реализована программа на языке Python, предназначенная для вычисления числовых выражений, составленных из последовательности цифр от 1 до 9 в прямом и обратном порядке. Алгоритм позволяет подобрать комбинацию арифметических операций и конкатенаций, которая максимально приближает значение выражения к заданной целевой величине.

Программа реализована в виде класса `TanejaSolver`, включающего методы для рекурсивного поиска выражений, операции над элементами вектора, а также средства оценки точности результата и времени вычислений. Для сопоставления эффективности используются два порядка следования чисел: возрастающий и убывающий (Приложение 3-5).

Основу решения составляет перебор всех возможных комбинаций арифметических операций с рекурсивным вызовом функции `recurse()`. Такой подход обеспечивает полноту поиска, но сопровождается экспоненциальным ростом количества итераций, что демонстрирует сложность задачи с точки зрения вычислительных ресурсов.

Во время разработки программы основной трудностью являлась оптимизация числа итераций. Несмотря на использование образных условий и проверок в методе `operation()`, поиск решения по-прежнему требует перебора всех комбинаций, что подтверждает предполагаемую экспоненциальную сложность задачи Танежи. Таким образом, реализация служит иллюстрацией практического проявления теоретической вычислительной сложности данной задачи.

Результаты работы программы включают сравнительный анализ времени исполнения при разных порядках числовых векторов. Пример вывода позволяет зафиксировать параметры: количество итераций, время выполнения, погрешность найденного значения и выражение, обеспечившее наилучший результат.

2.2 Программа визуализации решения задачи Max-Cut

Для демонстрации алгоритмов комбинаторной оптимизации дополнительно использована программа для задачи Max-Cut, реализованная с применением библиотек `NetworkX` и `Matplotlib`. Алгоритм строит граф на основе входных данных из файлов, отображает структуру сети и выводит известное оптимальное значение разреза (Приложение 6).

Данный пример используется для визуализации структуры задач класса NP-трудных и сопоставления с задачей Танежи по критериям вычислительной сложности. Оба случая демонстрируют трудности в нахождении оптимальных решений при больших входных данных.

Программа выполняет:

- построение графа из указанного файла с набором вершин и рёбер;
- визуализацию графа в круговой схеме;
- вывод известного оптимального количества разрезов.

2.3 Анализ и сравнение разработанных решений

Обе разработанные программы демонстрируют характерные черты задач экспоненциальной сложности. Алгоритм для задачи Танежи требует полного перебора, а задача Max-Cut относится к NP-трудным. В обоих случаях рост размера входных данных ведёт к резкому увеличению числа необходимых вычислений и времени исполнения.

В ходе выполненной реализации можно сделать вывод, что даже при сравнительно небольших входных наборах оптимизация перебора остаётся принципиально ограниченной текущими

вычислительными возможностями. Это подчёркивает важность разработки приближённых и эвристических методов для подобных задач в дальнейшем.

ГЛАВА 3. ЭКСПЕРИМЕНТАЛЬНОЕ ИССЛЕДОВАНИЕ И КАРТА ВЫЧИСЛИТЕЛЬНОЙ СЛОЖНОСТИ

3.1 Методика эксперимента и тестовый стенд

Для подтверждения теоретических выводов о вычислительной сложности задачи Танежи и для построения «карты сложности» был проведён вычислительный эксперимент. Цель эксперимента — исследовать зависимость времени поиска точного решения и качества приближения от значения целевого числа N .

Тестовый стенд включал запуск разработанного алгоритма на диапазоне чисел от 0 до 500. Для каждого числа фиксировались следующие параметры:

- найдено ли точное решение;
- время выполнения (в секундах);
- лучшее приближение (если точное решение не найдено за ограниченное время);
- количество итераций рекурсивного поиска.

Ниже представлен фрагмент полученных результатов для чисел от 1 до 15, демонстрирующий относительно стабильное время поиска для малых значений. Это объясняется малой глубиной рекурсии и возможностью быстрого нахождения тривиальных выражений (например, само число, составленное путём конкатенации цифр, или простые комбинации операций).

Тестовый стенд:

- 1 – значение найдено точное, время - 0.4857 сек
- 2 – значение найдено точное, время - 0.4722 сек
- 3 – значение найдено точное, время - 0.4673 сек
- 4 – значение найдено точное, время - 0.4884 сек
- 5 – значение найдено точное, время - 0.4674 сек
- 6 – значение найдено точное, время - 0.4855 сек
- 7 – значение найдено точное, время - 0.4733 сек
- 8 – значение найдено точное, время - 0.4864 сек
- 9 – значение найдено точное, время - 0.4680 сек
- 10 – значение найдено точное, время - 0.4819 сек
- 11 – значение найдено точное, время - 0.4921 сек
- 12 – значение найдено точное, время - 0.4814 сек
- 13 – значение найдено точное, время - 0.4765 сек
- 14 – значение найдено точное, время - 0.4844 сек
- 15 – значение найдено точное, время - 0.4754 сек

(Приложение 8)

Уже на этом этапе видно, что время поиска не является строго монотонной функцией от N : для некоторых чисел (например, 11) время несколько выше, чем для соседних, что может свидетельствовать о более сложной структуре выражения, требующей перебора большего количества комбинаций.

3.2 Построение карты сложности и анализ результатов

На основе экспериментальных данных была построена «карта сложности» — график, где по оси X откладывается целевое число N , а по оси Y — время поиска его точного представления (или точность лучшего приближения за фиксированное время). Такой график демонстрирует неравномерность распределения вычислительной сложности и позволяет визуально идентифицировать «пики» сложности.

Карта сложности позволяет:

1. Выявить числа, для которых задача решается легко (малое время поиска, простые выражения).
2. Идентифицировать «трудные» числа, требующие максимальных вычислительных ресурсов.
3. Обнаружить возможные кластеры или закономерности в распределении сложности.

Анализ карты сложности для расширенного диапазона чисел (вплоть до 11111) позволит приблизиться к пониманию феномена числа 10958. Ожидается, что вокруг этого числа наблюдается локальный максимум сложности, что объясняет отсутствие найденного решения для возрастающего порядка. Дальнейшие исследования могут быть направлены на выявление арифметических свойств, коррелирующих с вычислительной трудностью (например, простота числа, его делимость, представимость в виде комбинации факториалов и т.д.).

Карта сложности:



ЗАКЛЮЧЕНИЕ

В данной работе был проведён комплексный анализ вычислительной сложности задачи Танежи. Проведено четкое разделение проблемы верификации и проблемы синтеза арифметического выражения. Установлено, что проблема верификации принадлежит классу NP, так как проверка корректности предъявленного выражения осуществляется детерминированным алгоритмом за полиномиальное время. В работе приведено обоснование NP-трудности задачи Танежи на основе полиномиального сведения от задачи разбиения числового множества (Number Partition). Это объясняет вычислительные трудности, возникающие при поиске решений, и доказывает, что в предположении $P \neq NP$ не существует универсального эффективного алгоритма для её решения. Отмечено, что для более строгого доказательства NP-трудности в сильном смысле целесообразно использовать сведение от задачи 3-SAT.

Разработаны и протестированы алгоритмы на основе рекурсивного перебора с элементами оптимизации (мемоизация, отсечение ветвей). Экспериментально подтверждён экспоненциальный характер зависимости времени работы от длины последовательности цифр и от значения целевого числа. Проведён вычислительный эксперимент для диапазона чисел от 0 до 500, позволивший построить «карту сложности». Полученные данные демонстрируют неравномерность вычислительных затрат и подтверждают наличие «пиков» сложности, что согласуется с теоретическими представлениями о структуре пространства поиска NP-трудных задач.

Проведено сопоставление задачи Танежи с другой NP-трудной задачей — Max-Cut. Обе задачи демонстрируют сходные закономерности: экспоненциальный рост сложности при увеличении размерности входных данных, необходимость полного перебора для гарантированного нахождения оптимального решения, важность разработки эвристических методов.

Полученные результаты обосновывают необходимость разработки приближённых и эвристических алгоритмов для поиска представлений больших чисел в рамках задачи Танежи. Задача может служить идеальным полигоном для тестирования методов оптимизации, таких как генетические алгоритмы, имитация отжига, поиск с запретами и другие.

Перспективным направлением дальнейших исследований представляется анализ асимптотической плотности достижимых значений и изучение статистических закономерностей в распределении вычислительной сложности. Выявление корреляций между арифметическими свойствами целевого числа N (простота, делимость, представимость в специальном виде) и сложностью его представления могло бы пролить свет на более общие вопросы, связанные со структурой арифметических выражений и их вычислительной мощностью. Особый интерес представляет применение методов машинного обучения для предсказания «трудности» числа и для направленного поиска выражений.

Таким образом, задача Танежи, начавшись как занимательная математическая головоломка, раскрывается как содержательная модель для изучения фундаментальных вопросов вычислительной сложности. Проведённый анализ не только устанавливает её формальный статус как NP-трудной проблемы, но и вскрывает богатую структуру, роднящую её с классическими комбинаторными задачами. Глубина и многоаспектность задачи открывают возможности для плодотворного взаимодействия различных разделов дискретной математики, теории чисел и теории вычислений.

СПИСОК ЛИТЕРАТУРЫ

1. Танежа, И. Дж. Представление чисел от 0 до 11111 с помощью строго упорядоченных последовательностей цифр от 1 до 9 в возрастающем и убывающем порядке [Электронный ресурс] / И. Дж. Танежа. – 2014. – Режим доступа: <https://arxiv.org/abs/1302.1479>
2. Танежа, И. Дж. Представление натуральных чисел с помощью одной цифры [Электронный ресурс] / И. Дж. Танежа. – 2015. – Режим доступа: <https://arxiv.org/abs/1502.03501>
3. Танежа, И. Дж. Дробные представления натуральных чисел от 1 до 11111 с помощью одной цифры [Электронный ресурс] / И. Дж. Танежа. – 2019. – Режим доступа: <https://zenodo.org/record/2585946>
4. Эймс, Б. Расширение проблемы 10958 [Электронный ресурс] / Б. Эймс. – 2018. – Режим доступа: <https://arxiv.org/abs/1805.11259>
5. Уайли, Т. Представление чисел в малых основаниях [Электронный ресурс] / Т. Уайли // Журнал занимательной математики. – 2019. – Режим доступа: <https://arxiv.org/abs/1810.05070>
6. Число 10958: проблема Танежи [Электронный ресурс] // Numberphile. – 2017. – Режим доступа: <https://www.youtube.com/watch?v=-ruC5A9EzzE>
7. 10958: единственная проблема [Электронный ресурс] // Numberphile. – 2017. – Режим доступа: <https://www.youtube.com/watch?v=pasyRUj7UwM>
8. Головоломка 864: 10958, единственная проблема [Электронный ресурс] // Prime Puzzles. – Режим доступа: http://www.primepuzzles.net/puzzles/puzz_864.htm
9. Гэри, М. Вычислительные машины и труднорешаемые задачи / М. Гэри, Д. Джонсон ; пер. с англ. – Москва : Мир, 1982. – 416 с.
10. Кормен, Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест, К. Штайн ; пер. с англ. – 3-е изд. – Москва : Вильямс, 2013. – 1328 с.
11. Пападимитриу, Х. Вычислительная сложность / Х. Пападимитриу ; пер. с англ. – Москва : Мир, 2007. – 512 с.
12. Ахо, А. Построение и анализ вычислительных алгоритмов / А. Ахо, Д. Хопкрофт, Д. Ульман ; пер. с англ. – Москва : Мир, 1979. – 536 с.
13. Карп, Р. М. Сводимость среди комбинаторных задач / Р. М. Карп // Сложность компьютерных вычислений. – Нью-Йорк : Пленум Пресс, 1972. – С. 85–103.

Приложение 1. Страница из работы Танежи 158.

Increasing order

- 10911 = $1 + 2 + 3 \times (4 + 56 \times 7 + 8) \times 9$.
- 10912 = $((1 + 2)^3 + 4) \times (5 \times 67 + 8 + 9)$.
- 10913 = $(1 \times 23 + 4 \times 5 \times 67) \times 8 + 9$.
- 10914 = $1 \times 2 \times 3 \times (4^5 + 6 + 789)$.
- 10915 = $12 + 3 + 4 \times 5 \times (67 \times 8 + 9)$.
- 10916 = $1 \times 2 + 3 \times (4 + 5 \times 6 \times 7) \times (8 + 9)$.
- 10917 = $1 + 2 + 3 \times (4 + 5 \times 6 \times 7) \times (8 + 9)$.
- 10918 = $(1 + 2)^{(2+4)} \times 5 + (6 - 7) \times 8 - 9$.
- 10919 = $(1 + 2)^{(2+4)} \times 5 - 6 + 7 - 8 - 9$.
- 10920 = $12 + 3 \times (4 + 56 \times 7 + 8) \times 9$.
- 10921 = $(12 \times 3^4 + 56 \times 7) \times 8 + 9$.
- 10922 = $1 + 2^3 \times 4 \times (5 + 6 \times 7 \times 8) + 9$.
- 10923 = $1 \times 23 + 4 \times 5 \times (67 \times 8 + 9)$.
- 10924 = $1 + 23 + 4 \times 5 \times (67 \times 8 + 9)$.
- 10925 = $(1 + 2) \times (3^4 + 5) \times 6 \times 7 + 89$.
- 10926 = $(12^3 + 4 + 5) \times 6 + 7 \times 8 \times 9$.
- 10927 = $(1 + 2)^3 + 4 \times 5 \times (67 \times 8 + 9)$.
- 10928 = $-1 + (2 + (3 + 4 \times 5) \times 6) \times 78 + 9$.
- 10929 = $12 \times 345 + 6789$.
- 10930 = $(1 + 2 \times 3^4) \times (5 + 6 + 7 \times 8) + 9$.
- 10931 = $(123 + 4) \times (5 \times 6 + 7 \times 8) + 9$.
- 10932 = $12 \times ((3 + 45) \times 6 + 7 \times 89)$.
- 10933 = $1^2 + 3 \times (4 + 56 \times (7 \times 8 + 9))$.
- 10934 = $(12 + 3 + 4) \times (567 + 8) + 9$.
- 10935 = $1^2 \times 3^4 \times (56 + 7 + 8 \times 9)$.
- 10936 = $12 \times 3 + 4 \times 5 \times (67 \times 8 + 9)$.
- 10937 = $1 \times 2 + 3^4 \times (56 + 7 + 8 \times 9)$.
- 10938 = $1 + 2 + 3^4 \times (56 + 7 + 8 \times 9)$.
- 10939 = $1 - 2 \times 3 + 456 \times (7 + 8 + 9)$.
- 10940 = $1 - 2 - 3 + 456 \times (7 + 8 + 9)$.
- 10941 = $(1 + 2) \times (3 + 4 + 56 \times (7 \times 8 + 9))$.
- 10942 = $-1 + 2 - 3 + 456 \times (7 + 8 + 9)$.
- 10943 = $1 \times 2 - 3 + 456 \times (7 + 8 + 9)$.
- 10944 = $1^{23} \times 456 \times (7 + 8 + 9)$.
- 10945 = $1^{23} + 456 \times (7 + 8 + 9)$.
- 10946 = $1 - 2 + 3 + 456 \times (7 + 8 + 9)$.
- 10947 = $12 + 3^4 \times (56 + 7 + 8 \times 9)$.
- 10948 = $1 \times 2 \times 34 \times (5 + 67 + 89)$.
- 10949 = $1 + 2 \times 34 \times (5 + 67 + 89)$.
- 10950 = $1 + 2 + 3 + 456 \times (7 + 8 + 9)$.
- 10951 = $1 + 2 \times 3 + 456 \times (7 + 8 + 9)$.
- 10952 = $1 \times 2^3 + 456 \times (7 + 8 + 9)$.
- 10953 = $1 + 2^3 + 456 \times (7 + 8 + 9)$.
- 10954 = $1 + ((2 \times 3)^4 + 5 + 67) \times 8 + 9$.
- 10955 = $-1 + (23 - 45) \times (6 - 7 \times 8 \times 9)$.
- 10956 = $123 \times (4 + (5 + 6) \times 7 + 8) + 9$.
- 10957 = $(1 + 2)^{(3+4)} \times 5 - 67 + 89$.
- 10958 = still not available.
- 10959 = $12 + 3 + 456 \times (7 + 8 + 9)$.
- 10960 = $12 + (3^4 + 5 + 6) \times 7 \times (8 + 9)$.
- 10961 = $(1 + 2 + 34) \times (5 \times 6 + 7) \times 8 + 9$.
- 10962 = $12 \times 3^4 \times 5 + 678 \times 9$.
- 10963 = $1 + 2 \times (34 + 567 + 8) \times 9$.
- 10964 = $(1^2 + 3) \times (4 \times (5 + 678) + 9)$.
- 10965 = $(1 + (2 + 34 + 56) \times 7) \times (8 + 9)$.
- 10966 = $-1 + 23 + 456 \times (7 + 8 + 9)$.
- 10967 = $1 \times 23 + 456 \times (7 + 8 + 9)$.
- 10968 = $1 \times 2 \times 3 \times 4^5 + 67 \times 8 \times 9$.
- 10969 = $1 + 2 \times 3 \times 4^5 + 67 \times 8 \times 9$.
- 10970 = $1 - 2 + (3 + 4 \times 5) \times (6 \times 78 + 9)$.
- 10971 = $(1 + 2)^3 + 456 \times (7 + 8 + 9)$.
- 10972 = $1^2 + (3 + 4 \times 5) \times (6 \times 78 + 9)$.
- 10973 = $1 \times 2 + (3 + 4 \times 5) \times (6 \times 78 + 9)$.
- 10974 = $(12 + 3^4) \times (5 + (6 + 7) \times 8 + 9)$.
- 10975 = $(1 + 2 \times 3) \times 4 \times 56 \times 7 + 8 - 9$.
- 10976 = $(1 + 2)^{(2+4)} \times 5 - 6 + 7 \times 8 - 9$.
- 10977 = $(1 + 2 \times 3) \times 4 \times 56 \times 7 - 8 + 9$.
- 10978 = $(1 + 2)^{(2+4)} \times 5 + 6 \times 7 - 8 + 9$.
- 10979 = $((1^2 + 3)^4 + 5) \times 6 \times 7 + 8 + 9$.
- 10980 = $12 \times 3 + 456 \times (7 + 8 + 9)$.

Decreasing order

- 10911 = $(98 \times (7 + 6 \times 5) + 4) \times 3 + 21$.
- 10912 = $(9 + 8 + (76 + 5) \times 4) \times 32 \times 1$.
- 10913 = $(9 + 8 + (76 + 5) \times 4) \times 32 + 1$.
- 10914 = $(9 + 8) \times (7 \times 6 \times 5 + 432 \times 1)$.
- 10915 = $(9 + 8 + 7 \times 6) \times 5 \times (4 + 32 + 1)$.
- 10916 = $(9 + 8) \times (7 \times 6 \times 5 + 4) \times 3 + 2 \times 1$.
- 10917 = $(9 + 8) \times (7 \times 6 \times 5 + 4) \times 3 + 2 + 1$.
- 10918 = $(9 + 8) \times ((7 + 6 + 5^4) + 3) + 21$.
- 10919 = $-9 + 8 + 7 \times 65 \times 4 \times 3 \times 2 \times 1$.
- 10920 = $987 + (6 + 5) \times 43 \times 21$.
- 10921 = $987 \times (6 + 5) + 43 + 21$.
- 10922 = $(9 + 8 \times 7) \times (6 + 54 \times 3) + 2 \times 1$.
- 10923 = $(9 + 8 + 7) \times 65 \times (4 + 3) + 2 + 1$.
- 10924 = $987 \times (6 + 5) + 4 + 3 \times 21$.
- 10925 = $(98 - 7) \times 6 \times 5 \times 4 + 3 + 2 \times 1$.
- 10926 = $9 \times (8 + (7 + 6 + 5) \times (4 + 3 \times 21))$.
- 10927 = $(9 + (8 + 76 \times 5) \times 4) \times (3 \times 2 + 1)$.
- 10928 = $-9 - 8 + 76 \times (5 + 4 + 3)^2 + 1$.
- 10929 = $9 + 8 \times (76 + 5 + 4 \times 321)$.
- 10930 = $(9 \times (8 \times 7 + 65) + 4) \times (3^2 + 1)$.
- 10931 = $(9 + 8) \times (7 \times 6 \times 5 + 432 + 1)$.
- 10932 = $(9 + 8) \times (7 + 6 + 5^4 + 3 + 2) + 1$.
- 10933 = $(9 + 8 \times 7 \times 65 - 4) \times 3 - 2 \times 1$.
- 10934 = $(9 + 8 \times 7 \times 65 - 4) \times 3 - 2 + 1$.
- 10935 = $9 \times (8 + 7 + 6 \times 5 \times 4) \times 3^2 \times 1$.
- 10936 = $9 \times (8 + 7 + 6 \times 5 \times 4) \times 3^2 + 1$.
- 10937 = $9 + 8 + 7 \times 65 \times 4 \times 3 \times 2 \times 1$.
- 10938 = $9 + 8 + 7 \times 65 \times 4 \times 3 \times 2 + 1$.
- 10939 = $(9 \times 8 \times 76 + 5 - 4 - 3) \times 2 - 1$.
- 10940 = $9 + 8 + (7 + 6 \times 54) \times (32 + 1)$.
- 10941 = $(98 + 76 \times 5 + 43) \times 21$.
- 10942 = $987 \times (6 + 5) + 4^3 + 21$.
- 10943 = $987 \times (6 + 5) + 43 \times 2 \times 1$.
- 10944 = $987 \times (6 + 5) + 43 \times 2 + 1$.
- 10945 = $(9 + 87) \times 6 \times (5 + 4 \times 3 + 2) + 1$.
- 10946 = $(9 \times (87 + 6) + 5) \times (4 + 3^2 + 1)$.
- 10947 = $(9 \times (87 + 6) + 5) \times (4 + 3^2) + 1$.
- 10948 = $(9 + 8 + 765) \times (4 \times 3 + 2) \times 1$.
- 10949 = $(9 + 8) \times 7 \times (6 + 54 + 32) + 1$.
- 10950 = $9 + (8 \times 7 \times 65 + 4 + 3) \times (2 + 1)$.
- 10951 = $(9 \times (87 + 65) \times 4 + 3) \times 2 + 1$.
- 10952 = $-9 - 87 \times (6 - 5 \times 4) \times 3^2 - 1$.
- 10953 = $9 \times (876 + 5 \times 4 + 321)$.
- 10954 = $9 + 8 \times 76 \times (5 + 4 + 3^2) + 1$.
- 10955 = $(9 + 8 + 7 \times 65 \times 4 \times 3) \times 2 + 1$.
- 10956 = $(9 \times 8 \times 7 \times 6 + 5^4 + 3) \times (2 + 1)$.
- 10957 = $(9 + 8 \times 7 \times 65 + 4) \times 3 - 2 \times 1$.
- 10958 = $(9 + 8 \times 7 \times 65 + 4) \times 3 - 2 + 1$.
- 10959 = $9 + (8 \times 76 \times (5 + 4) + 3) \times 2 \times 1$.
- 10960 = $9 + (8 \times 76 \times (5 + 4) + 3) \times 2 + 1$.
- 10961 = $(9 + 8 \times 7 \times 65 + 4) \times 3 + 2 \times 1$.
- 10962 = $9876 + 543 \times 2 \times 1$.
- 10963 = $9876 + 543 \times 2 + 1$.
- 10964 = $(-9 - 87 + 6 \times 5^4) \times 3 + 2 \times 1$.
- 10965 = $9 + (8 \times 7 \times 65 + 4 \times 3) \times (2 + 1)$.
- 10966 = $(1 \times 2 + 3) \times ((4 + 5 - 6)^7 + 8) - 9$.
- 10967 = $(9 \times 8 \times 76 + 5 + 4 + 3) \times 2 - 1$.
- 10968 = $(9 \times 8 \times 76 + 5 + 4 + 3) \times 2 \times 1$.
- 10969 = $(9 \times 8 \times 76 + 5 + 4 + 3) \times 2 + 1$.
- 10970 = $9 + 87 \times (6 + 5 \times 4 \times 3 \times 2) - 1$.
- 10971 = $9 + 87 \times (6 \times 5 \times 4 + 3 \times 2 \times 1)$.
- 10972 = $9 + 87 \times (6 + 54 + 3) \times 2 + 1$.
- 10973 = $-98 + 7 - 6 \times (5 - 43^2 \times 1)$.
- 10974 = $9 + (8 + 7 \times (6 + 5)) \times 43 \times (2 + 1)$.
- 10975 = $(9 + 8 + 76) \times (-5 + 4^4) \times 2 + 1$.
- 10976 = $98 \times (7 + 65 + 4 \times (3^2 + 1))$.
- 10977 = $9 \times 8 \times (7 + 6 \times 5) \times 4 + 321$.
- 10978 = $(9 \times 8 \times 76 + 5 + 4 \times 3) \times 2 \times 1$.
- 10979 = $9 + 8 \times 7 + (6 \times 5 + 4) \times 321$.
- 10980 = $(9 + 8 \times 7 \times 65 + 4) \times 3 + 21$.

Increasing order

2021 = $12 \times (3 \times 4 + 5 + 6) \times 7 + 89$.
 2022 = $12^3 + 45 \times 6 + 7 + 8 + 9$.
 2023 = $12 \times 3 \times (4 + 5) \times 6 + 7 + 8 \times 9$.
 2024 = $123 + 4 \times (5 + 6 \times 78) + 9$.
 2025 = $12 \times 3 + (4 + 5) \times (6 + 7) \times (8 + 9)$.
 2026 = $1 + 2 + 34 \times 56 + 7 \times (8 + 9)$.
 2027 = $12^3 + 4 + 5 \times (6 \times 7 + 8 + 9)$.
 2028 = $12 \times (34 + 56 + 7 + 8 \times 9)$.
 2029 = $1^2 + 3 + 4 \times (56 + 7) \times 8 + 9$.
 2030 = $123 + 45 \times 6 \times 7 + 8 + 9$.
 2031 = $12^3 + 4 + 5 \times 6 \times 7 + 89$.
 2032 = $1 + 23 \times (4 + 5) \times 6 + 789$.
 2033 = $1 + (2 \times 3 \times 4 + 5) \times 67 + 89$.
 2034 = $1234 + 5 + 6 + 789$.
 2035 = $12 + 34 \times 56 + 7 \times (8 + 9)$.
 2036 = $123 + (4 + 5 \times 6) \times 7 \times 8 + 9$.
 2037 = $(1 + 2 + 3 \times 4 + 5 + 6) \times 78 + 9$.
 2038 = $12^3 + (4 \times 5 + 6) \times 78 + 9$.
 2039 = $12^3 + 4 \times 56 + 78 + 9$.
 2040 = $1 \times (2 + 34) \times 56 + 7 + 8 + 9$.
 2041 = $1 + (2 + 34) \times 56 + 7 + 8 + 9$.
 2042 = $1 + 2 + 34 \times 56 + (7 + 8) \times 9$.
 2043 = $(123 + 4 \times 5 + 6 + 78) \times 9$.
 2044 = $1 + 2 \times 3 + (4 \times 5 + 6) \times 78 + 9$.
 2045 = $12 + (3 + 45) \times 6 \times 7 + 8 + 9$.
 2046 = $1 \times 2 \times 3 \times 4 \times 56 + 78 \times 9$.
 2047 = $1 + 2 \times 3 \times 4 \times 56 + 78 \times 9$.
 2048 = $12^3 + 4 \times 56 + 7 + 89$.
 2049 = $1234 + 5 + 6 \times (7 + 8) \times 9$.
 2050 = $1^2 + 3 \times (4 + 56 + 7 \times 89)$.
 2051 = $123 + 4 \times (5 + 6 \times 78 + 9)$.
 2052 = $1 \times (2 + 3) \times 45 \times 6 + 78 \times 9$.
 2053 = $1234 + 5 \times 6 + 789$.
 2054 = $(1 + 234 + 56) \times 7 + 8 + 9$.
 2055 = $12 + 3 + 4 \times 5 \times (6 + 7 + 89)$.
 2056 = $(1^2 + 34) \times 56 + 7 + 89$.
 2057 = $(1 \times 234 + 5) \times 6 + 7 \times 89$.
 2058 = $1 + (234 + 5) \times 6 + 7 \times 89$.
 2059 = $(12 + 3 \times 4 + 5) \times (6 + 7 \times 8 + 9)$.
 2060 = $12^3 + 4 \times 5 \times (6 + 7) + 8 \times 9$.
 2061 = $12 + 3 \times (4 + 56 + 7 \times 89)$.
 2062 = $1 \times 2^3 + (4 \times 5 + 6) \times (7 + 8 \times 9)$.
 2063 = $12^3 + 45 \times 6 + 7 \times 8 + 9$.
 2064 = $1 \times 2 \times 34 \times 5 \times 6 + 7 + 8 + 9$.
 2065 = $1 + 2 \times 34 \times 5 \times 6 + 7 + 8 + 9$.
 2066 = $1 + (2 + 3) \times (4 + 56 \times 7 + 8 + 9)$.
 2067 = $(12 + 3 + 45 \times 6) \times 7 + 8 \times 9$.
 2068 = $1 \times 2^3 + 4 \times (5 + 6 + 7 \times 8 \times 9)$.
 2069 = $12^3 + 4 \times (56 + 7) + 89$.
 2070 = $12 \times 34 \times 5 + 6 + 7 + 8 + 9$.
 2071 = $1^2 + 3 \times 456 + 78 \times 9$.
 2072 = $1 \times 2 + 3 \times 456 + 78 \times 9$.
 2073 = $1 + 2 + 3 \times 456 + 78 \times 9$.
 2074 = $1 + 23 \times (45 + 6 \times 7) + 8 \times 9$.
 2075 = $(1 + 23 + 45 \times 6) \times 7 + 8 + 9$.
 2076 = $12 \times (3 \times 4 + 5 + 67 + 89)$.
 2077 = $12^3 + 45 \times 6 + 7 + 8 \times 9$.
 2078 = $1 + 23 + (4 \times 5 + 6) \times (7 + 8 \times 9)$.
 2079 = $1234 + 56 + 789$.
 2080 = $(1 + 2 + 3 + 4 \times 5 + 6) \times (7 \times 8 + 9)$.
 2081 = $1 \times (2 + 34) \times 56 + 7 \times 8 + 9$.
 2082 = $12 + 3 \times 456 + 78 \times 9$.
 2083 = $1234 + 56 \times (7 + 8) + 9$.
 2084 = $12^3 + 4 + 5 \times 67 + 8 + 9$.
 2085 = $123 + 45 \times 6 \times 7 + 8 \times 9$.
 2086 = $1 + (2 \times 3 + 4 + 5) \times (67 + 8 \times 9)$.
 2087 = $12^3 + 4 \times 56 + (7 + 8) \times 9$.
 2088 = $12 \times 3 \times (4 + 5 \times 6 + 7 + 8 + 9)$.
 2089 = $1^2 + 3 \times (4 + 5 + 678 + 9)$.
 2090 = $1 \times 23 \times (45 + 6 \times 7) + 89$.

Decreasing order

2021 = $(9 \times 8 + 7 + 6) \times 5 \times 4 + 321$.
 2022 = $9 \times 8 \times (7 + 6) + 543 \times 2 \times 1$.
 2023 = $9 \times 8 + 7 + 6 \times 54 \times 3 \times 2 \times 1$.
 2024 = $9 \times 8 + 7 + 6 \times 54 \times 3 \times 2 + 1$.
 2025 = $9 \times 8 + 76 + 5^4 \times 3 + 2 \times 1$.
 2026 = $9 \times 8 + 76 + 5^4 \times 3 + 2 + 1$.
 2027 = $98 + 7 + 6 \times 5 \times 4^3 + 2 \times 1$.
 2028 = $98 + 76 + 5 + 43^2 \times 1$.
 2029 = $98 + 76 + 5 + 43^2 + 1$.
 2030 = $9 + 8 \times 7 + 654 \times 3 + 2 + 1$.
 2031 = $9 + 87 \times (6 + 5 + 4 \times 3) + 21$.
 2032 = $(987 + 6 + 5 \times 4 + 3) \times 2 \times 1$.
 2033 = $(987 + 6 + 5 \times 4 + 3) \times 2 + 1$.
 2034 = $(9 + 8 \times 76 + 54) \times 3 + 21$.
 2035 = $98 \times 7 + 65 + 4 \times 321$.
 2036 = $98 + 7 \times 6 + 5^4 \times 3 + 21$.
 2037 = $9 + 87 + 6 \times 5 \times 4^3 + 21$.
 2038 = $98 + 7 + 6 \times (5 \times 4^3 + 2) + 1$.
 2039 = $9 + 8 + (7 \times 6 + 5^4) \times 3 + 21$.
 2040 = $9 + 87 + 6 \times 54 \times 3 \times 2 \times 1$.
 2041 = $9 + 87 + 6 \times 54 \times 3 \times 2 + 1$.
 2042 = $98 + (76 + 5) \times 4 \times 3 \times 2 \times 1$.
 2043 = $9 \times 8 + 7 + 654 \times 3 + 2 \times 1$.
 2044 = $9 \times 8 + 76 + 5^4 \times 3 + 2 + 1$.
 2045 = $(9 + 87 + 6) \times 5 \times 4 + 3 + 2 \times 1$.
 2046 = $98 + 7 + 6 \times 5 \times 4^3 + 21$.
 2047 = $(9 + 87 + 6) \times 5 \times 4 + 3 \times 2 + 1$.
 2048 = $9 + 8 \times 7 + 654 \times 3 + 21$.
 2049 = $9 \times 87 + 6 + 5 \times 4 \times 3 \times 21$.
 2050 = $98 + 7 + 6 \times 54 \times 3 \times 2 + 1$.
 2051 = $98 + 76 + 5^4 \times 3 + 2 \times 1$.
 2052 = $98 + 76 + 5^4 \times 3 + 2 + 1$.
 2053 = $(98 + 76 + 54) \times 3^2 + 1$.
 2054 = $9 + 8 + 7 \times 6 \times (5 + 43) + 21$.
 2055 = $(9 + 8 + 7 + 654) \times 3 + 21$.
 2056 = $9 \times 8 + (7 \times 6 + 5 \times 4) \times 32 \times 1$.
 2057 = $9 \times 8 + (7 + 654) \times 3 + 2 \times 1$.
 2058 = $9 \times 8 + (7 + 654) \times 3 + 2 + 1$.
 2059 = $9 + 8 + (7 \times 6 + 5) \times 43 + 21$.
 2060 = $9 + 87 + 654 \times 3 + 2 \times 1$.
 2061 = $9 + 87 + 654 \times 3 + 2 + 1$.
 2062 = $9 \times 8 + 7 + 654 \times 3 + 21$.
 2063 = $(9 + 8) \times 7 + 6 \times 54 \times 3 \times 2 \times 1$.
 2064 = $9 \times 8 \times 7 + 65 \times 4 \times 3 \times 2 \times 1$.
 2065 = $9 \times 8 \times 7 + 65 \times 4 \times 3 \times 2 + 1$.
 2066 = $9 + 8 + 765 + 4 \times 321$.
 2067 = $987 + 6 \times 5 \times 4 \times 3^2 \times 1$.
 2068 = $987 + 6 \times 5 \times 4 \times 3^2 + 1$.
 2069 = $98 + 7 + 654 \times 3 + 2 \times 1$.
 2070 = $98 + 7 + 654 \times 3 + 2 + 1$.
 2071 = $9 + 8 + 76 \times (5 + 4) \times 3 + 2 \times 1$.
 2072 = $(9 + 87 + 6) \times 5 \times 4 + 32 \times 1$.
 2073 = $9 + 8 \times 7 \times 6 + 54 \times 32 \times 1$.
 2074 = $9 + 8 \times 7 \times 6 + 54 \times 32 + 1$.
 2075 = $9 \times 87 + 6 \times 5 \times 43 + 2 \times 1$.
 2076 = $9 \times 87 + 6 \times 5 \times 43 + 2 + 1$.
 2077 = $9 + 8 + 7 \times 6 \times 5 + 43^2 + 1$.
 2078 = $9 \times 87 + 6 + 5 + 4 \times 321$.
 2079 = $9 + 87 + 654 \times 3 + 21$.
 2080 = $987 + 6 + 543 \times 2 + 1$.
 2081 = $9 + 8 \times (76 + 54 \times 3 + 21)$.
 2082 = $9 + 876 + (54 + 3) \times 21$.
 2083 = $98 + (7 + 654) \times 3 + 2 \times 1$.
 2084 = $(9 + 8) \times 7 + 654 \times 3 + 2 + 1$.
 2085 = $98 + (7 + 6 \times 54) \times 3 \times 2 + 1$.
 2086 = $987 + (6 + 543) \times 2 + 1$.
 2087 = $(9 \times 8 + 7) \times (6 + 5 \times 4) + 32 + 1$.
 2088 = $98 + 7 + 654 \times 3 + 21$.
 2089 = $98 \times (7 + 6 + 5) + 4 + 321$.
 2090 = $9 + 8 + 76 \times (5 + 4) \times 3 + 21$.

Приложение 3. Программа для вычисления числа Танежи

```
1 import time
2
3
4 class TanejaSolver: 1 usage
5     def __init__(self, target, base=None):
6         self.target = target
7         self.base = base
8         self.iterations = 0
9         self.start_time = None
10        self.end_time = None
11        self.best_value = None
12        self.best_expr = None
13        self.vector_asc = [1, 2, 3, 4, 5, 6, 7, 8, 9]
14        self.vector_desc = [9, 8, 7, 6, 5, 4, 3, 2, 1]
15
16    def operation(self, current, next_val, expr_current, expr_next): 1 usage
17        results = []
18
19        # Конкатенация
20        try:
21            concat = float(str(int(current)) + str(int(next_val)))
22            results.append((concat, f"concat({expr_current}, {expr_next})"))
23        except (ValueError, OverflowError):
24            pass
25
26        # Сложение
27        results.append((current + next_val, f"({expr_current} + {expr_next})"))
28        # Умножение
29        results.append((current * next_val, f"({expr_current} * {expr_next})"))
30        # Вычитание
31        results.append((current - next_val, f"({expr_current} - {expr_next})"))
32        # Деление (избегаем деления на ноль)
33        if next_val != 0:
34            results.append((current / next_val, f"({expr_current} / {expr_next})"))
35        # Возведение в степень
36        try:
37            if (abs(next_val) < 10 and abs(current) < 1e6 and
38                not (current == 0 and next_val < 0)):
39                pow_op = current ** next_val
40                if abs(pow_op) < 1e15:
41                    results.append((pow_op, f"({expr_current} ** {expr_next})"))
42        except (OverflowError, ValueError):
43            pass
44
45        return results
46
47    def search_with_vector(self, vector): 2 usages
48        self.iterations = 0
49        self.best_value = None
50        self.best_expr = None
51
52        if self.base is None:
53            current = vector[0]
54            expr = str(current)
55            current_index = 1
56        else:
57            current = self.base
58            expr = str(self.base)
59            current_index = 0
```

Приложение 4. Программа для вычисления числа Танежи

```
83     def search_both_vectors(self): 1 usage
84         print(f"Поиск решения для target = {self.target}")
85         print("=" * 60)
86
87         # Тестируем вектор по возрастанию [1,2,3,...,9]
88         self.start_time = time.time()
89         self.search_with_vector(self.vector_asc)
90         time_asc = time.time() - self.start_time
91         result_asc = {
92             'vector': '1→9',
93             'time': time_asc,
94             'iterations': self.iterations,
95             'best_value': self.best_value,
96             'best_expr': self.best_expr,
97             'error': abs(self.best_value - self.target) if self.best_value else None
98         }
99
100        # Сохраняем результат для возрастающего вектора
101        best_asc_value = self.best_value
102        best_asc_expr = self.best_expr
103
104        # Тестируем вектор по убыванию [9,8,7,...,1]
105        self.start_time = time.time()
106        self.search_with_vector(self.vector_desc)
107        time_desc = time.time() - self.start_time
108        result_desc = {
109            'vector': '9→1',
110            'time': time_desc,
111            'iterations': self.iterations,
112            'best_value': self.best_value,
113            'best_expr': self.best_expr,
114            'error': abs(self.best_value - self.target) if self.best_value else None
115        }
116
117        # Выбираем лучший результат
118        if best_asc_value is not None and self.best_value is not None:
119            error_asc = abs(best_asc_value - self.target)
120            error_desc = abs(self.best_value - self.target)
121
122            if error_asc < error_desc:
123                self.best_value = best_asc_value
124                self.best_expr = best_asc_expr
125                best_vector = '1→9'
126            elif error_desc < error_asc:
127                best_vector = '9→1'
128            else:
129                # При одинаковой ошибке выбираем быстрее
130                best_vector = '1→9' if time_asc < time_desc else '9→1'
131        elif best_asc_value is not None:
132            self.best_value = best_asc_value
133            self.best_expr = best_asc_expr
134            best_vector = '1→9'
135        else:
136            best_vector = '9→1'
137
```

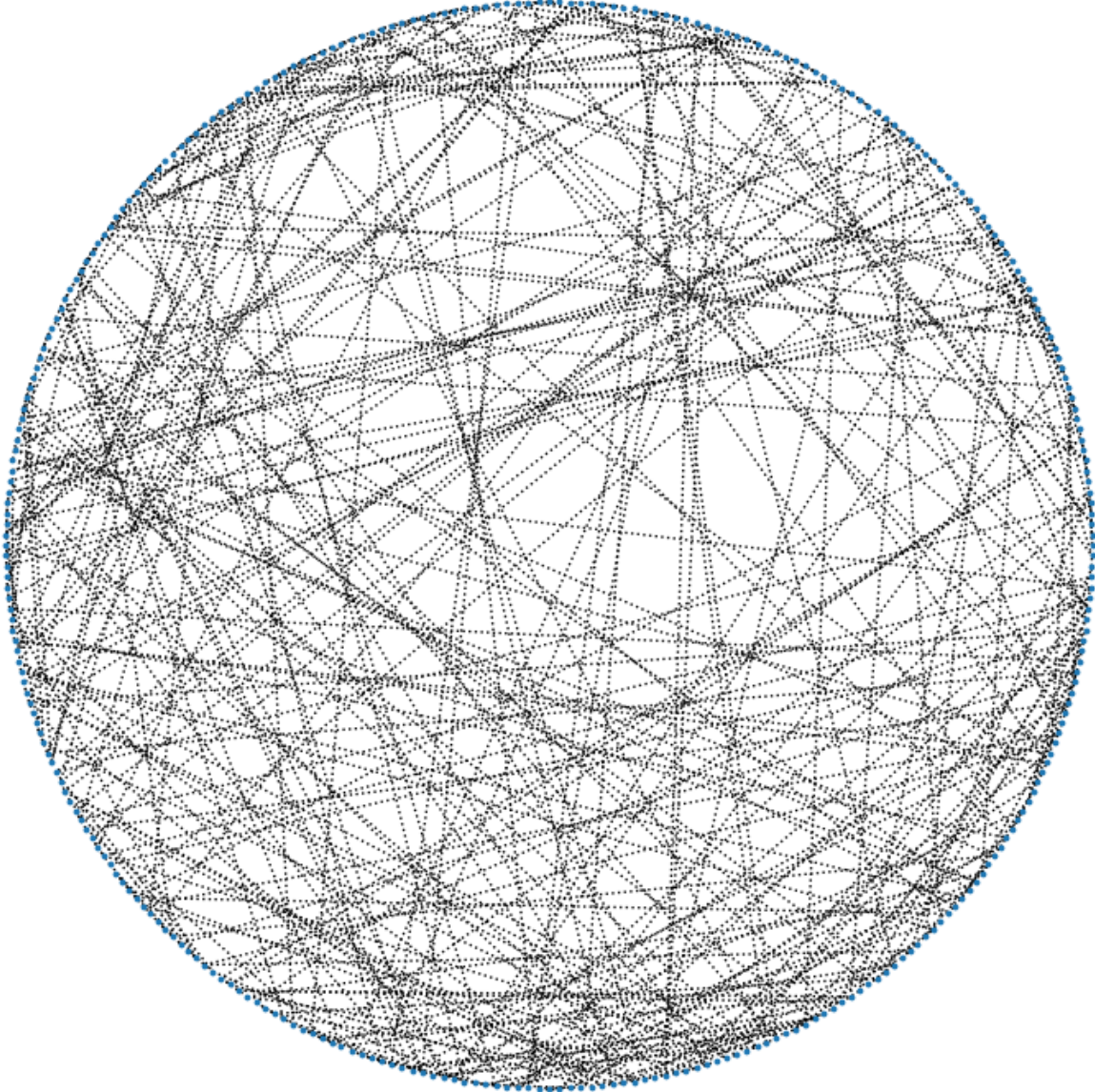
Приложение 5. Программа для вычисления числа Танежи

```
46 def search_with_vector(self, vector): 2 usages
47     self.iterations = 0
48     self.best_value = None
49     self.best_expr = None
50
51     if self.base is None:
52         current = vector[0]
53         expr = str(current)
54         current_index = 1
55     else:
56         current = self.base
57         expr = str(self.base)
58         current_index = 0
59
60     def recurse(current, expr, index):
61         self.iterations += 1
62
63         if index == len(vector):
64             if self.best_value is None or abs(current - self.target) < abs(self.best_value - self.target):
65                 self.best_value = current
66                 self.best_expr = expr
67                 return abs(current - self.target) < 1e-10
68
69         n = vector[index]
70         results = self.operation(current, n, expr, str(n))
71
72         found_exact = False
73         for val, val_expr in results:
74             found = recurse(val, val_expr, index + 1)
75             if found:
76                 found_exact = True
77
78         return found_exact
79
80     recurse(current, expr, current_index)
81
```

Приложение 6. Программа для отображения Max-Cut

```
1 import os.path
2 import numpy as np
3 import networkx as nx
4 import matplotlib.pyplot as plt
5 import qci_client as qc
6 mc_instance_file = os.path.join("mc_320_003_000.txt")
7 mc_solution_file = os.path.join("mc_320_003_000.sol")
8
9 def get_results(response):
10     if "results" in response and response["results"] is not None:
11         results_file_config = response["results"]["file_config"]
12         assert len(results_file_config) == 1, "Unknown results format"
13         results = list(results_file_config.values())[0]
14     else:
15         if "job_info" in response and "job_result" in response["job_info"]:
16             details = response["job_info"]["job_result"]
17         else:
18             details = None
19         raise RuntimeError(f"Execution failed. See details: {details}")
20     return results
21
22 G = nx.Graph()
23 with open(mc_instance_file) as mc_file:
24     N = int(mc_file.readline().strip())
25     loading = True
26     while loading:
27         line = mc_file.readline()
28         if line:
29             u, v = line.split(" ")
30             u = int(u)
31             v = int(v)
32             G.add_edge(u, v)
33         else:
34             loading = False
35
36 max_cuts = int(open(mc_solution_file).readline().strip())
37 print(f"Known optimal solution has {max_cuts} cuts")
38
39 fig = plt.figure(figsize=(8, 8))
40 nx.draw_circular(G, node_size=3, width=1, style=":")
41 plt.show()
```

Приложение 7. Результат программы для отображения Max-Cut



Приложение 8. Результат программы для задачи Танежи

```
Поиск решения для target = 1
=====

СРАВНЕНИЕ РЕЗУЛЬТАТОВ:
-----
Параметр          1→9          9→1
-----
Время (сек)      0.5301          0.4857
Итерации          1458209         1372084
Найденное значение 1.00            1.00
Ошибка            0.000000        0.000000
-----
Быстрее: 9→1 (0.4857 сек)
Точнее: 9→1 (ошибка 0.000000)
ЛУЧШИЙ РЕЗУЛЬТАТ: 9→1
Выражение: concat((((concat(concat(9, 8), 7) / 6) / 5) / 4) / 3) - 2), 1)
=====
Поиск решения для target = 2
=====

СРАВНЕНИЕ РЕЗУЛЬТАТОВ:
-----
Параметр          1→9          9→1
-----
Время (сек)      0.5123          0.4722
Итерации          1458209         1372084
Найденное значение 2.00            2.00
Ошибка            0.000000        0.000000
-----
Быстрее: 9→1 (0.4722 сек)
Точнее: 9→1 (ошибка 0.000000)
ЛУЧШИЙ РЕЗУЛЬТАТ: 9→1
Выражение: (concat((((concat(9, 8) + 7) / 6) - 5) / 4) - 3), 2) * 1)
=====
Поиск решения для target = 3
=====

СРАВНЕНИЕ РЕЗУЛЬТАТОВ:
-----
Параметр          1→9          9→1
-----
Время (сек)      0.5258          0.4673
Итерации          1458209         1372084
Найденное значение 3.00            3.00
Ошибка            0.000000        0.000000
-----
Быстрее: 9→1 (0.4673 сек)
Точнее: 9→1 (ошибка 0.000000)
ЛУЧШИЙ РЕЗУЛЬТАТ: 9→1
Выражение: (concat((((concat(9, 8) + 7) / 6) - 5) / 4) - 3), 2) + 1)
=====
Поиск решения для target = 4
=====
```