Содержание

1 Основные термины и определения

Все формулировки следуют конспекту лекций «Введение в коммуникационную сложность» В.В. Подольского и А.Е. Ромащенко [1].

• *Коммуникационный протокол* для вычисления некоторой функции из $X \times Y$ в Z – это ориентированное двоичное дерево с корнем и со следующей разметкой на вершинах и рёбрах. Каждая внутренняя вершина дерева помечена буквой A или B. Каждой вершине с пометкой A приписана некоторая функция $g_i : X \in \{0,1\}$ в каждой вершине с пометкой B приписана некоторая функция $h_j : Y \to \{0,1\}$ (разным вершинам соответствуют разные функции). Каждому листу дерева сопоставлен некоторый элемент из Z. Каждое ребро в графе помечено нулём или единицей. Из каждой вершины, не являющейся листом, выходит по одному ребру с пометкой 0 и одному ребру с пометкой 1.

• *Сложность коммуникационного протокола* – его глубина, то есть максимальное расстояние от корня до листа дерева.

• *Коммуникационная сложность* функции f – минимальная сложность протокола, вычисляющего f. Обозначается как $CC(f)$.

• *Комбинаторный прямоугольник* (прямоугольное множество): (1) это множество $S \subseteq X \times Y$ для которого существуют такие $A \subseteq X$ и $B \subseteq Y$, что $S = A \times B$. (2) это множество $S \subseteq X \times Y$, такое что для любых двух пар (x, y) и (x', y') из S пара (x, y') также принадлежит S.

• M_f - матрица размера $|X| \times |Y|$, представляющая функцию f. Будем считать, что строки матрицы соответствуют элементам X, а столбцы соответствуют элементам Y. На пересечении x-ой строки и y-ого столбца будет стоять соответствующее значение функции $f(x, y)$. Таким образом, матрица заполнена нулями и единицами.

• $cov_z(f)$ – минимальное число комбинаторных прямоугольников, покрывающих все элементы матрицы M_f со значением z.

• $cov(f)$ – минимальное число одноцветных комбинаторных прямоугольников, покрывающих всю матрицу M_f.

• Для функции $f : X \times Y \to Z$ будем называть множество $S \subseteq X \times Y$ *трудным*, если найдется такое $z \in Z$, что (1) $\forall\ (x, y) \in S$ имеем $f(x, y) = z$, (2) для любых двух (несовпадающих) пар $(x, y), (x', y') \in S$ имеем $f(x, y') = z$ или $f(x', y) = z$.

• *Недетерминированный коммуникационный протокол* для вычисления функции $f : X \times Y \to Z$ – это двоичное дерево с выделенным корнем, со следующей разметкой. Каждая внутренняя вершина дерева помечена буквой A или B. Каждой вершине с пометкой A приписана некоторая функция $g_i : X \times Adv_a \to \{0,1\}$, а каждой вершине с пометкой B приписана некоторая функция $h_i : Y \times Adv_b \to \{0,1\}$ (разным вершинам соответствуют разные функции). Каждому листу дерева сопоставлен либо некоторый элемент из Z, либо знак «?» (неопределённость).

• Недетерминированный коммуникационный протокол *вычисляет* некоторую функцию $f : X \times Y \to Z$, если выполнены следующие условия: (1) $\forall x \in X,\ y \in Y$ найдутся такие «советы» $n_a \in Adv_a$ и $n_b \in Adv_b$, что следуя ветви дерева, соответствующей данной четвёрке (x, n_a, y, n_b) Алиса и Боб попадают в лист, помеченный значением $f(x, y)$, (2) $\forall x \in X,\ y \in Y$и для любых «советов» $n_a \in Adv_a$ и $n_b \in Adv_b$, следуя ветви протокола для данной четвёрки (x, n_a, y, n_b), Алиса и Боб могут попасть либо в некоторый лист, помеченный значением $f(x, y)$, либо в некоторый лист с пометкой неопределённость (но не могут попасть в лист, помеченный значением $z \in Z$ отличным от $f(x, y)$.

• *Сложность недетерминированного протокола* – это глубина его дерева (максимальное расстояние от корня до листа).

• *Недетерминированная коммуникационная сложность* функции f – минимальная сложность недетерминированного протокола для вычисления f. Обозначается $NCC(f)$.

• Для $\varepsilon > 0$ *вероятностная коммуникационная сложность* $RCC_\varepsilon(f)$ – это минимальная глубина вероятностного протокола, который для любой пары входов $x,\ y$ имеет вероятность ошибки не больше ε.

• *Средняя коммуникационная сложность* заданного вероятностного протокола для данных входов $x,\ y$ – это математическое ожидание числа пересылаемых битов.

• *Средняя коммуникационная сложность* данного протокола *в худшем случае* – максимум средней коммуникационной сложности по всем парам входов $x,\ y$.

• В *вероятностных коммуникационных протоколах* А и Б разрешается использовать случайность. Формально это означает, что Алиса и Боб независимо выбирают случайные последовательности битов $r_a \in \{0,1\}^{l_a}$ и $r_b \in \{0,1\}^{l_b}$ соответственно. Далее на каждом шаге протокола действия Алисы зависят от её входа x и от r_a, а действия Бобы зависят от его входа y и r_b. $\forall\ (x,y)$ возникает распределение вероятностей на множестве листьев, в которые Алиса и Боб могут попасть. Причем в вероятностных протоколах каждому листу прописано некоторое значение $z \in Z$ и для некоторых $x,\ y$ и некоторых r_a, r_b Алиса и Боб могут получить неправильное значение $f(x,y)$.

• *Вероятностной коммуникационной сложностью с нулевой ошибкой* $RCC_0(f)$ называется минимальная средняя коммуникационная сложность вероятностных протоколов с нулевой ошибкой.

• *Вероятностный коммуникационный протокол с общими случайными битами* – вероятностный протокол в котором мы требуем, чтобы что в начале работы протокола случайно выбиралась строка $r \in \{0,1\}^l$ фиксированной длины l, и далее на каждом шаге протокола действия Алисы зависят от x и r, а действия Боба зависят от y и r. Более того, мы будем считать, что от значения случайных битов r может зависеть, кто из участников посылает сообщение на очередном шаге.

• *Вероятностная коммуникационная сложность для протокола с общими случайными битами* обозначается как $RCC_\varepsilon^{pub}(f)$ и имеет почти идентичное с вероятностной коммуникационной сложностью определение.

• *Коммуникационная сложность функции с распределением на аргументах*. Пусть дана функция $f : X \times Y \to Z$ и распределение вероятностей μ на множестве $X \times Y$. Коммуникационной сложностью f для распределения μ с ошибкой ε называется минимум глубины среди всех детерминированных протоколов π, которые правильно вычисляют значение f для пар $(x,y) \in X \times Y$ общей μ-меры не менее $(1-\varepsilon)$, то есть:

$$P_\mu[\pi(x,y) = f(x,y)] \geq 1 - \varepsilon$$

Обозначается как $CC_\varepsilon^\mu(f)$.

• Неоднородность функции при заданном распределении. Пусть дана функция $f : X \times Y \to \{0,1\}$ и распределение вероятностей μ на $X \times Y$. Для комбинаторного прямоугольника $R \subseteq X \times Y$ будем называть *неоднородностью f на R* разницу между μ-мерами таких пар $(x,y) \in R$, на которых функция f равна единице, и таких пар $(x,y) \in R$ на которых функция f равна нулю. Будем обозначать неоднородность на заданном комбинаторном прямоугольнике $Disc_\mu(R,f)$.

$$Disc_\mu(R,f) = |Prob_\mu[(x,y) \in R \text{ and } f(x,y) = 1] - Prob_m u[(x,y) \in R \text{ and } f(x,y) = 0]|$$

Неоднородностью f по мере μ называется максимум неоднородности по всем комбинаторным прямоугольникам:

$$Disc_\mu(f) = \max_R [Disc_\mu(R,f)]$$

2 Введение

2.1 Предмет исследования. Актуальность

Предметом исследования данного проекта является коммуникационная сложность. Коммуникационная сложность – это раздел теоретической информатики, занимающийся изучением количества коммуникации, необходимого для решения определенной задачи, при том, что исходные данные, необходимые для её решения, распределены между двумя и более сторонами.

Впервые коммуникационная сложность была упомянута в работе Эндрю Яо «Some Complexity Questions Related to Distributed Computing» [3] в 1979 году. В данной работе рассматривалась классическая коммуникационная модель для двух участников. Прошло много времени с момента первого упоминания коммуникационной сложности, и за это время наше общество претерпело множество изменений. Естественно, прогресс не обошел и теоретическую информатику. На данный момент в коммуникационной сложности рассматриваются не только модели с двумя участниками, но и с большим их количеством. Также со временем коммуникационная сложность нашла свое применение в различных науках. Её объектами исследования являются самого разного рода задачи. Таким образом, коммуникационная сложность полезна не только во многих разделах теоретической информатики, например при оптимизации компьютерных сетей, при изучении алгоритмов и структур данных, в теории вычислительной сложности и сложности доказательств, но и в совершенно разных областях компьютерных наук: например, при проектировании больших интегральных схем требуется минимизировать используемую энергию, путём уменьшения количества электрических сигналов между различными компонентами во время распределенных вычислений [7].

2.2 Цель и задачи

Основной целью любого исследовательского проекта является проведение исследований в той или иной области математики или компьютерных наук. В рамках данного проекта для изучения были предложены на выбор конспект лекций по коммуникационной сложности – В.В. Подольский, А.Е. Ромащенко «Введение в коммуникационную сложность» [1] и книга на английском языке – E. Kushilevitz, N. Nisan [2]. «Communication complexity». Я же склонила свой выбор в сторону конспектов лекций. Таким образом моей основной задачей в данном проекте являлось подробное ознакомление с материалами лекций и в процессе знакомство с основными идеями коммуникационной сложности. Следующим этапом выполнения данного проекта стало применение полученных мною знаний для решения задачи по коммуникационной сложности.

Конкретно можно выделить следующие задачи проекта:

1. Получение базовых знаний в области коммуникационной сложности.
2. Применение полученных знаний для решения конкретной задачи по теме.

3 Основная часть

3.1 Обзор и сравнительный анализ источников

Говоря об имеющихся источниках, повествующих о коммуникационной сложности и её различных приложениях, нельзя сказать, что информация в них полностью идентична. Зачастую определения в них, хоть и являются синонимичными, но все-таки отличаются между собой. В качестве основного источника для получения знаний в рамках данного проекта были предложены на выбор конспект лекций по коммуникационной сложности – В.В. Подольский, А.Е. Ромащенко «Введение в коммуникационную сложность» [1] и книга на английском языке – Kushilevitz, E., N. Nisan. «Communication complexity» [2]. Для себя я четко понимала, что изучение материалов на английском будет для меня куда менее продуктивны, чем на русском. Так что основным источником, откуда я и черпала новые знания, стал конспект лекций дисциплины «Введение в коммуникационную сложность», а в книге «Communication complexity» я в основном смотрела различные примеры, так как в отличие от конспекта лекций, там имеется довольно много разных иллюстраций, помогающих лучше понять и освоить материал. Далее я постараюсь проанализировать полученные мною знания и сжать их в краткую выдержку.

Для начала для того, чтобы повествовать о коммуникационной сложности, хочется ввести определение её основной, простейшей модели, которая и являлась предметом изучения в прочитанном мною конспекте лекций. А выглядит она следующим образом. Имеется два участника, их принято называть Алиса (А) и Боб (Б), которые совместно хотят решить определенную задачу, то есть вычислить значение какой-то функции, зависящей от двух аргументов. Но при этом ни один из них не может вычислить значение самостоятельно, так как и Алисе, и Бобу известна лишь часть данных (только первый или только второй аргумент функции соответственно). Таким образом для того, чтобы решить данную задачу Алисе и Бобу необходимо общаться, то есть обмениваться некоторым количеством битов информации. Притом, мы предполагаем, что Алиса и Боб заранее договариваются о коммуникационном протоколе, то есть наборе соглашений, какие именно данные и в каком порядке они будут пересылать друг другу при тех или иных значениях исходных данных (x и y). Очевидно, что в данном случае приоритетной задачей является подсчёт количества переданных бит, необходимых для вычисления функции. Решением именно этой задачи и занимается коммуникационная сложность.

Вернемся к коммуникационному протоколу, согласно которому и передается информация между Алисой и Бобом. Именно он предопределяет, сколько именно битов информации потребуется передать для вычисления исходной функции. Если быть конкретнее, имеется три основных вида коммуникационных протоколов:

1) Детерминированный
2) Недетерминированный
3) Вероятностный

Все они принципиально отличаются между собой. Но отличаются они не только в принципе своей работы, но и, что очевидно, в результате. То есть если рассматривать какую-то конкретную функцию, то оценка её коммуникационной сложности может значительно отличаться в зависимости от выбранного протокола. Но также есть функции, детерминированная, недетерминированная и вероятностная (с нулевой ошибкой) коммуникационная сложность которых совпадают.

Одним из самых простых примеров такой функции является функция $EQ : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$, принимающая на вход два аргумента. То есть $EQ(x,y) = 1$, если $x = y$ и $EQ(x,y) = 0$,

если $x \neq y$. У данной функции $CC(EQ) = NCC(EQ) = RCC_0(EQ) = n + 1$. При этом $\forall\ \varepsilon > 0$: $RCC_\varepsilon(EQ) = \Omega(\log n),\ RCC_\varepsilon^{pub}(EQ) = O(log\frac{1}{\varepsilon})$.

С функцией же $MCE : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ (Matrix Column Equality) дела обстоят иначе. Для начала объясню, в чем же заключается суть этой функции. Мы будем рассматривать $x,\ y \in \{0,1\}^n$ как матрицы размера $\sqrt{n} \times \sqrt{n}$ и $MCE(x,y) = 1$, если найдется такой номер $i \in \{1, ..., \sqrt{n}\}$, что i-ые столбцы в матрицах $x,\ y$ совпадают. Соответственно иначе $MCE(x,y) = 0$. У данной функции имеются следующие оценки на коммуникационную сложность: $CC(MCE) = \Omega(n),\ NCC(MCE) = O(\sqrt{n}\log n),\ RCC_0^{pub}(MCE) = O(\sqrt{n})$. Несложно заметить, что оценки довольно сильно отличаются. К примеру, недетерминированная сложность задачи почти квадратично меньше детерминированной, а вероятностная сложность с нулевой ошибкой для протокола с общими случайными битами в точности квадратично меньше детерминированной сложности.

Также, хочется добавить, что разница есть и между вероятностной коммуникационной сложностью и вероятностной коммуникационной сложностью для протокола с общими случайными битами. В отличие от предыдущих рассматриваемых случаев, может показаться, что разница между этими двумя не настолько велика. Но на практике выходит, что если разрешить Алисе и Бобу пользоваться общим источником случайности, то вычисление функции может облегчиться довольно сильно. Хорошим примером этого является функция от двух аргументов GT : $\{1, ..., 2n\} \times \{1, ..., 2n\} \to \{0,1\}$. Причем $GT(x,y) = 1$, если $x > y$ и $GT(x,y) = 0$, если $x \leq y$. Доказано, что вероятностная коммуникационная сложность данной функции $RCC_\varepsilon(GT) = O(\log^2 n)$, а $RCC_\varepsilon^{pub}(GT) = O(\log n \cdot \log(\log n))$, и если быть более точными, то $RCC_\varepsilon(GT) = RCC_\varepsilon^{pub}(GT) + O(\log n)$. Что является немаленькой разницей при достаточно больших n.

Помимо того, что я своими глазами смогла увидеть и проанализировать разницу между различными коммуникационными протоколами, также я узнала о том, как именно они связаны. Эти факты довольно полезны для анализа тех или иных функций. К примеру, если необходимо оценить определенную коммуникационную сложность заданной функции, но другая коммуникационная сложность считается куда проще, то можно оценить одну коммуникационную сложность через другую, тем самым решив поставленную изначально задачу. Мною были изучены и проанализированы следующие оценки:

$$1)\ CC(f) = O\left(NCC(f)^2\right)$$

$$2)\ NCC(f) \leq RCC_0(f)$$

$$3)\ RCC_\varepsilon(f) = \Omega\left(\log CC(f)\right)$$

$$4)\ RCC_\varepsilon^{pub}(f) \leq RCC_\varepsilon(f)$$

$$5)\ RCC_0^{pub}(f) \leq RCC_0(f)$$

$$6)\ RCC_{\varepsilon+\delta}(f) \leq RCC_\varepsilon^{pub}(f) + O\left(\log n + \log\frac{1}{\delta}\right)$$

$$7)\ RCC_\varepsilon^{pub}(f) = \max_\mu\left[CC_\varepsilon^\mu(f)\right]$$

Также мне стали известны некоторые общие оценки на коммуникационную сложность. Их можно с легкостью получить, если известны какие-то дополнительные сведения о рассматриваемой функции или же просто если мы будем рассматривать какие-то дополнительные элементы. К примеру, вот некоторые из них:

1) Если для функции $f : X \times Y \to Z$ имеется трудное множество размера K, то $CC(f) > \log K$.

2) Для любой функции $f : X \times Y \to Z : CC(f) \leq 3\log_2[$минимальное число листьев в протоколе для вычисления $f]$.

3) Для любой функции $f : X \times Y \to \{0,1\} : CC(f) \geq \log rk(M_f),\ CC(f) \geq \log(2rk(M_f) - 1)$

4) Для любой функции $f : X \times Y \to Z$: минимальное число одноцветных прямоугольников в разбиении $M_f \leq$ минимальное число листьев в протоколе для вычисления $f \leq 2^{CC(f)}$.

5) Для любой функции $f : X \times Y \to Z : NCC(f) \leq \lceil \log(cov(f)) \rceil + 1,\ cov(f) \leq 2^{CC(f)}$.

6) Для любой функции $f : X \times Y \to \{0,1\}$, для любой вероятностной меры μ на $X \times Y$ и любого вещественного $\varepsilon > 0$ выполнено $CC^{\mu}_{\frac{1}{2}-\varepsilon}(f) \geq \log \dfrac{2\varepsilon}{Disc_{\mu}(f)}$

7) Для любого $\varepsilon > 0$ и всех достаточно больших n для значительного большинства функций $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ выполнено: $RCC^{pub}_{\frac{1}{2}-\varepsilon}(f) \geq 0.1n$

Каждая из этих оценок полезна, ведь комбинируя их можно получить довольно точную оценку на коммуникационную сложность для конкретной задачи.

3.2 Исследование сложности функции скалярного произведения. Детерминированная коммуникационная сложность.

Следующим этапом выполнения проекта после подробного ознакомления с теорией стало проведение вычислительного эксперимента, то есть в моем случае – решение задачи по теме. Мне была предложена следующая задача:

Дана функция $IP_3(x, y)$. Где x и y – это векторы длины n, каждая координата которых – это какой-то остаток по модулю 3. Функция равна скалярному произведению этих векторов над полем F_3, то есть:

$$IP_3(x, y) = \sum_{i=1}^{n} x_i y_i (mod\ 3)$$

Необходимо оценить коммуникационную сложность данной функции.

Перейдём непосредственно к решению задачи. Для начала попробуем найти верхнюю оценку на детерминированную коммуникационную сложность для общего случая IP_p. Рассмотрим самый очевидный протокол: пусть Алиса передает свое значение x Бобу, а он в свою очередь вычисляет скалярное произведение и передает результат Алисе. Так как $x \in \{0, 1, ..., p-1\}^n,\ y \in \{0, 1, ..., p-1\}^n$, а $\forall x\ \forall y\ IP_p(x, y) \in \{0, 1, ..., p-1\}$, то наша функций определена так: $IP_p(x, y) : X \times Y \to Z$, где $X,\ Y$ - это множества всех слов длины n над полем из p элементов, то есть $X = Y = \{0, 1, ..., p-1\}^n$, а $Z = \{0, 1, ..., p-1\}$. Тогда рассмотрим одну из самых простых оценок на детерминированную коммуникационную сложность, а именно:

$$\forall f : X \times Y \to Z : CC(f) \leq \lceil \log |X| \rceil + \lceil \log |Z| \rceil$$

В нашем случае $|X| = p^n,\ |Z| = p$, таким образом, $CC(IP_p) \leq \lceil \log p^n \rceil + \lceil \log p \rceil$. Это и будет являться верхней оценкой на детерминированную коммуникационную сложность анализируемой функции.

Теперь попробуем оценить коммуникационную сложность снизу. Рассмотрим случай с $p = 3$. То есть найдём коммуникационную сложность функции IP_3, где:

$$IP_3(x, y) = \sum_{i=1}^{n} x_i y_i (mod\ 3),\ x_i, y_i \in \{0, 1, 2\}$$

Если мы рассматриваем случай с $p = 3$, значит работаем с полем F_3. Из алгебры мы знаем, что любое конечное поле простого порядка p может быть представлено в виде кольца вычетов по модулю p (т.е. $F_p \simeq \mathbb{Z}/(p) = \mathbb{Z}_p$). А как мы знаем, $\mathbb{Z}_p = \{0, 1, ..., p-1\}$, значит и мы можем сказать, что пусть для определенности $F_p = \{0, 1, ..., p-1\}$ (это в дальнейшем упростит рассуждения и вычисления). Таким образом в текущем случае, рассматриваемом нами, $F_3 = \{0, 1, 2\}$. Для того, чтобы дать нижнюю оценку на детерминированную коммуникационную сложность этой функции, воспользуемся следующим утверждением:

$$CC(f) \geq \log(rk(M_f))$$

Где M_f – это матрица размера $|X| \times |Y|$, представляющая функцию f. Так как мы работаем с полем F_3, то элементами матрицы M_{IP_3} являются числа 0, 1, 2. Рассмотрим подробнее эту матрицу и опишем то, как она устроена.

В данной матрице хранятся все возможные значения скалярного произведения двух векторов $x = (x_1, x_2, ..., x_n)$ и $y = (y_1, y_2, ..., y_n)$, где $x_i,\ y_i \in \{0, 1, 2\}$. Значит размер такой матрицы будет $3^n \times 3^n$, так как каждый из векторов может принимать ровно 3^n различных значений (каждая из n координат может принимать 3 значения).

Далее попробуем удобно пронумеровать столбцы и строки данной матрицы. Отсортируем векторы x и y в лексикографическом порядке (то есть сравнение происходит покоординатно) и назовем $x^1 = (0, ..., 0)$ (то есть самый маленький x), $x^2 = (0, ..., 0, 1)$ и так далее, $x^{3^n} = (2, ..., 2)$ (то есть самый большой x), аналогично с y: $y^1 = (0, ..., 0),\ y^2 = (0, ..., 0, 1), ...,\ y^{3^n} = (2, ..., 2)$. Пускай строки матрицы будут отвечать за x, а столбцы за y. То есть получим следующее:

$$M_{IP_3} = \begin{array}{c} \\ x^1 \\ x^2 \\ \vdots \\ x^{3^n} \end{array} \begin{array}{c} \begin{array}{cccc} y^1 & y^2 & \dots & y^{3^n} \end{array} \\ \begin{pmatrix} IP_3(x^1, y^1) & IP_3(x^1, y^2) & \dots & IP_3(x^1, y^{3^n}) \\ IP_3(x^2, y^1) & IP_3(x^2, y^2) & \dots & IP_3(x^2, y^{3^n}) \\ \vdots & \vdots & \ddots & \vdots \\ IP_3(x^{3^n}, y^1) & IP_3(x^{3^n}, y^2) & \dots & IP_3(x^{3^n}, y^{3^n}) \end{pmatrix} \end{array}$$

К сожалению, оценить ранг матрицы в таком виде довольно сложно, поэтому для удобства рассмотрим квадрат данной матрицы (назовём её P):

$$P = \begin{array}{c} \\ x^1 \\ x^2 \\ \vdots \\ x^{3^n} \end{array} \begin{array}{c} \begin{array}{cccc} y^1 & y^2 & \dots & y^{3^n} \end{array} \\ \begin{pmatrix} \sum\limits_{i=1}^{n} IP_3(x^1, y^i) IP_3(x^i, y^1) & \sum\limits_{i=1}^{n} IP_3(x^1, y^i) IP_3(x^i, y^2) & \dots & \sum\limits_{i=1}^{n} IP_3(x^1, y^i) IP_3(x^i, y^{3^n}) \\ \sum\limits_{i=1}^{n} IP_3(x^2, y^i) IP_3(x^i, y^1) & \sum\limits_{i=1}^{n} IP_3(x^2, y^i) IP_3(x^i, y^2) & \dots & \sum\limits_{i=1}^{n} IP_3(x^2, y^i) IP_3(x^i, y^{3^n}) \\ \vdots & \vdots & \ddots & \vdots \\ \sum\limits_{i=1}^{n} IP_3(x^{3^n}, y^i) IP_3(x^i, y^1) & \sum\limits_{i=1}^{n} IP_3(x^{3^n}, y^i) IP_3(x^i, y^2) & \dots & \sum\limits_{i=1}^{n} IP_3(x^{3^n}, y^i) IP_3(x^i, y^{3^n}) \end{pmatrix} \end{array}$$

Так как $x^1 = (0, ..., 0)$, то $IP_3(x^1, z) = 0,\ \forall z = (z_1, z_2, ..., z_n)$. Аналогично с y: так как $y^1 = (0, ..., 0)$, то $IP_3(z, y^1) = 0,\ \forall z = (z_1, z_2, ..., z_n)$. Таким образом у матрицы P первый столбец и

первая строка будут нулевыми. То есть матрица имеет такой вид:

$$P = \begin{array}{c} \\ x^1 \\ x^2 \\ \vdots \\ x^{3^n} \end{array} \begin{array}{c} \begin{array}{cccc} y^1 & y^2 & \dots & y^{3^n} \end{array} \\ \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & \sum\limits_{i=1}^{n} IP_3(x^2, y^i) IP_3(x^i, y^2) & \dots & \sum\limits_{i=1}^{n} IP_3(x^2, y^i) IP_3(x^i, y^{3^n}) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \sum\limits_{i=1}^{n} IP_3(x^{3^n}, y^i) IP_3(x^i, y^2) & \dots & \sum\limits_{i=1}^{n} IP_3(x^{3^n}, y^i) IP_3(x^i, y^{3^n}) \end{pmatrix} \end{array}$$

Обозначим за $(P)_{i,j}$ элемент матрицы P, находящийся на пересечении i-ой строчки и j-ого столбца. А также заметим, что $\forall i \ x^i = y^i$ (потому что это одинаковым образом отсортированные векторы из одних и тех же элементов (надо полем F_3), то получим такую матрицу:

$$P = \begin{array}{c} \\ x^1 \\ x^2 \\ \vdots \\ x^{3^n} \end{array} \begin{array}{c} \begin{array}{cccc} y^1 & y^2 & \dots & y^{3^n} \end{array} \\ \begin{pmatrix} 0 & 0 & \dots & 0 \\ 0 & \sum\limits_{z \in \{0,1.2\}^n} IP_3(x^2, z) IP_3(z, y^2) & \dots & \sum\limits_{z \in \{0,1,2\}^n} IP_3(x^2, z) IP_3(z, y^{3^n}) \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \sum\limits_{z \in \{0,1,2\}^n} IP_3(x^{3^n}, z) IP_3(z, y^2) & \dots & \sum\limits_{z \in \{0,1,2\}^n} IP_3(x^{3^n}, z) IP_3(z, y^{3^n}) \end{pmatrix} \end{array}$$

Рассмотрим значения оставшихся элементов.

$$1) \ (P)_{x^i, y^j}, \ x^i = y^j$$

То есть рассмотрим элементы матрицы P, находящиеся на главной диагонали.

$$(P)_{x^i, y^j} = (P)_{x^i, x^i} = \sum_{z \in \{0,1,2\}^n} \left[IP_3(x^i, z) \right]^2$$

Очевидно, что «вес» в данную сумму вносят только ненулевые слагаемые. Причем если $IP_3(x^i, z) = 1$, то в сумму входит слагаемое $1^2 = 1$, а если $IP_3(x^i, z) = 1$, то в сумму входит слагаемое $2^2 = 4$. Таким образом нам нужно найти количество слагаемых обоих видов.

- $IP_3(x^i, z) = 1$. Значит нам необходимо найти количество решений такого линейного уравнения:

$$x_1^i z_1 + x_2^i z_2 + ... + x_n^i z_n = 1$$

Поскольку уравнение невырождено ($x \neq (0, ..., 0)$), то размерность пространства решений над полем из трёх элементов равна $n-1$, а число таких решений равно 3^{n-1} (Также так как вероятность 0, 1 или 2 в той или иной координате вектора одинакова, то интуитивно ясно, что скалярное произведение будет равновероятно принимать значения 0, 1 и 2. То есть его матожидание будет в точности единица. Так как всего 3^n вариантов значения вектора z, то ровно в $\frac{3^n}{3} = 3^{n-1}$ случаях скалярное произведение x с z будет равно 1 и т.д.). Значит вес, вносимый слагаемыми такого типа в общую сумму, будет равен 3^{n-1}.

- $IP_3(x^i, z) = 2$. Значит нам необходимо найти количество решений такого линейного уравнения:

$$x_1^i z_1 + x_2^i z_2 + ... + x_n^i z_n = 2$$

Поскольку уравнение невырождено ($x \neq (0, ..., 0)$), то размерность пространства решений над полем из трёх элементов равна $n-1$, а число таких решений равно 3^{n-1}. Значит вес, вносимый слагаемыми такого типа в общую сумму, будет равен $4 \cdot 3^{n-1}$.

Значит в итоге $(P)_{x^i,y^j} = 3^{n-1} + 4 \cdot 3^{n-1} = 5 \cdot 3^{n-1}$. Таким образом мы нашли значение каждого диагонального элемента матрицы P.

$$2)\ (P)_{x^i,y^j},\ x^i \neq y^j$$

То есть рассматриваем элементы вне главной диагонали.

$$(P)_{x^i,y^j} = \sum_{z \in \{0,1,2\}^n} \left[IP_3(x^i, z) IP(z, y^j) \right]$$

Очевидно, что «вес» в данную сумму вносят только ненулевые слагаемые. Причем если $IP_3(x^i, z) = 1, IP(z, y^j) = 1$, то в сумму входит слагаемое $1 \cdot 1 = 1$, если $IP_3(x^i, z) = 1,\ IP(z, y^j) = 2$, то в сумму входит слагаемое $1 \cdot 2 = 2$, если $IP_3(x^i, z) = 2,\ IP(z, y^j) = 1$, то в сумму входит слагаемое $2 \cdot 1 = 2$, и наконец если $IP_3(x^i, z) = 2,\ IP(z, y^j) = 2$, то в сумму входит слагаемое $2 \cdot 2 = 4$. Таким образом нам нужно найти количество слагаемых всех видов.

- $IP_3(x^i, z) = 1,\ IP(z, y^j) = 1$. Значит нам необходимо найти количество решений такой системы линейных уравнений:

$$\begin{cases} x_1^i z_1 + x_2^i z_2 + ... + x_n^i z_n = 1 \\ y_1^j z_1 + y_2^j z_2 + ... + y_n^j z_n = 1 \end{cases}$$

Так как оба линейных уравнения невырождены, а векторы $x,\ y$ точно линейно независимы (так как случай с $x = y$ был рассмотрен выше), то размерность пространства решений над полем из трёх элементов равна $n-2$, а число таких решений равно 3^{n-2}. Значит вес, вносимый слагаемыми такого типа в общую сумму, будет равен 3^{n-2}.

- $IP_3(x^i, z) = 1,\ IP(z, y^j) = 2$. Значит нам необходимо найти количество решений такой системы линейных уравнений:

$$\begin{cases} x_1^i z_1 + x_2^i z_2 + ... + x_n^i z_n = 1 \\ y_1^j z_1 + y_2^j z_2 + ... + y_n^j z_n = 2 \end{cases}$$

Заметим, что так как мы работаем с полем $\mathbb{Z}_3$, то векторы x и y могут быть как линейно независимы, так и линейно зависимы. Рассмотрим каждый из случаев. Если векторы линейно независимы, то всего решений данной системы ровно 3^{n-2}. Значит вес, вносимый слагаемыми такого типа в общую сумму, будет равен $2 \cdot 3^{n-2}$.

Если же векторы линейно зависимы, то стоит рассмотреть 3 случая. Так как мы работаем с полем, в котором только 0, 1 и 2, то могут возникнуть только такие линейные зависимости: $x = 2y,\ y = 2x,\ \forall v = v_1, ..., v_k : x^i[v] = 2y^j[v],\ \forall w = w_1, ..., w_m : y^j[w] = 2x^i[w],\ k + m = n$. Так как случай $x = y$ мы рассматривали ранее, а случай с нулевым вектором x или y нам нет смысла рассматривать.

Отдельно поясню третий случай линейной зависимости. В данном случае мы рассматриваем такие векторы $x^i,\ y^j$, что каждая из их координат находится либо в соотношении 1 к 2, либо 2 к 1 соответственно. К примеру, при $n = 2$ подходят следующие векторы: $(1, 1, 2)$ и $(2, 2, 1)$ и т.д. Почему

же они являются линейно зависимыми? Потому что в поле $\mathbb{Z}_3$ выполнено следующее равенство: $4 = 1$. Таким образом, выполнено соотношение $x^i = 2y^j$ $(y^j = 2x^i)$.

a) $x = 2y$
Получаем следующую систему:

$$\begin{cases} 2(y_1^j z_1 + y_2^j z_2 + ... + y_n^j z_n) = 1 \\ y_1^j z_1 + y_2^j z_2 + ... + y_n^j z_n = 2 \end{cases}$$

Так как вычисление скалярного произведения происходит по модулю 3, то $y_1^j z_1 + ... + y_n^j z_n = 3c_1 + 2$ для некоторой константы c_1 (неотрицательное целое число). Таким образом, мы получим, что $x_1^i z_1 + ... + x_n^i z_n = 2(y_1^j z_1 + ... + y_n^j z_n) = 2(3_1 + 2) = 6_1 + 4 = 1(mod\ 3)$. Таким образом количество решений данной системы равно количеству решений уравнения $y_1^j z_1 + ... + y_n^j z_n = 2$. А как мы уже знаем, решений такого уравнения ровно 3^{n-1}. Мы получили, что в таком случае вес, вносимый слагаемыми такого типа в общую сумму, будет равен $2 \cdot 3^{n-1}$.

b) $y = 2x$
Получаем следующую систему:

$$\begin{cases} x_1^i z_1 + x_2^i z_2 + ... + x_n^i z_n = 1 \\ 2(x_1^i z_1 + x_2^i z_2 + ... + x_n^i z_n) = 2 \end{cases}$$

Так как вычисление скалярного произведения происходит по модулю 3, то $x_1^i z_1 + ... + x_n^i z_n = 3c_2 + 1$ для некоторой константы c_2 (неотрицательное целое число). Таким образом, мы получим, что $y_1^j z_1 + ... + y_n^j z_n = 2(x_1^i z_1 + ... + x_n^i z_n) = 2(3_2 + 1) = 6_2 + 2 = 2(mod\ 3)$. Таким образом количество решений данной системы равно количеству решений уравнения $x_1^i z_1 + ... + x_n^i z_n = 1$. А как мы уже знаем, решений такого уравнения ровно 3^{n-1}. Мы получили, что в таком случае вес, вносимый слагаемыми такого типа в общую сумму, будет равен $2 \cdot 3^{n-1}$.

c) $\forall v = v_1, ..., v_k : x^i[v] = 2y^j[v],\ \forall w = w_1, ..., w_m : y^j[w] = 2x^i[w],\ k + m = n$
Так как в поле $\mathbb{Z}_3$ выполнено соотношение $x^i = 2y^j$ $(y^j = 2x^i)$, так как $4 = 1$, то данный случай сводится к любому из предыдущих (вообще говоря, они эквиваленты, так как в поле $\mathbb{Z}_3$ условия $x^i = 2y^j$ и $y^j = 2x^i$ всегда выполняются одновременно, опять таки из-за равенства $4 = 1$, а значит случаи *a*) и *b*) по сути являются одним случаем.) Значит в таком случае вес, вносимый слагаемыми такого типа в общую сумму, также будет равен $2 \cdot 3^{n-1}$.

- $IP_3(x^i, z) = 2$, $IP(z, y^j) = 1$. Значит нам необходимо найти количество решений такой системы линейных уравнений:

$$\begin{cases} x_1^i z_1 + x_2^i z_2 + ... + x_n^i z_n = 2 \\ y_1^j z_1 + y_2^j z_2 + ... + y_n^j z_n = 1 \end{cases}$$

Также, как и в пункте выше, рассмотрим 3 случая линейной зависимости:

a) $x = 2y$
Получаем следующую систему:

$$\begin{cases} 2(y_1^j z_1 + y_2^j z_2 + ... + y_n^j z_n) = 2 \\ y_1^j z_1 + y_2^j z_2 + ... + y_n^j z_n = 1 \end{cases}$$

Так как вычисление скалярного произведения происходит по модулю 3, то $y_1^j z_1 + ... + y_n^j z_n = 3c_3 + 1$ для некоторой константы c_3 (неотрицательное целое число). Таким образом, мы получим, что $x_1^i z_1 + ... + x_n^i z_n = 2(y_1^j z_1 + ... + y_n^j z_n) = 2(3c_3 + 1) = 6_3 + 2 = 2(mod\ 3)$. Таким образом количество решений данной системы равно количеству решений уравнения $y_1^j z_1 + ... + y_n^j z_n = 1$. А как мы уже знаем, решений такого уравнения ровно 3^{n-1}. Мы получили, что в таком случае вес, вносимый слагаемыми такого типа в общую сумму, будет равен $2 \cdot 3^{n-1}$.

b) $y = 2x$

Получаем следующую систему:

$$\begin{cases} x_1^i z_1 + x_2^i z_2 + ... + x_n^i z_n = 2 \\ 2(x_1^i z_1 + x_2^i z_2 + ... + x_n^i z_n) = 1 \end{cases}$$

Так как вычисление скалярного произведения происходит по модулю 3, то $x_1^i z_1 + ... + x_n^i z_n = 3c_4 + 2$ для некоторой константы c_4 (неотрицательное целое число). Таким образом, мы получим, что $y_1^j z_1 + ... + y_n^j z_n = 2(x_1^i z_1 + ... + x_n^i z_n) = 2(3c_4 + 2) = 6_4 + 4 = 1(mod\ 3)$. Таким образом количество решений данной системы равно количеству решений уравнения $x_1^i z_1 + ... + x_n^i z_n = 2$. А как мы уже знаем, решений такого уравнения ровно 3^{n-1}. Мы получили, что в таком случае вес, вносимый слагаемыми такого типа в общую сумму, будет равен $2 \cdot 3^{n-1}$.

c) $\forall v = v_1, ..., v_k : x^i[v] = 2y^j[v],\ \forall w = w_1, ..., w_m : y^j[w] = 2x^i[w],\ k + m = n$

Идея вычислений точно такая же, как в пунктах a) и b), таким образом вес, вносимый слагаемыми такого типа в общую сумму, будет равен $2 \cdot 3^{n-1}$.

- $IP_3(x^i, z) = 2,\ IP(z, y^j) = 2$. Значит нам необходимо найти количество решений такой системы линейных уравнений:

$$\begin{cases} x_1^i z_1 + x_2^i z_2 + ... + x_n^i z_n = 2 \\ y_1^j z_1 + y_2^j z_2 + ... + y_n^j z_n = 2 \end{cases}$$

Так как оба линейных уравнения невырождены, а векторы $x,\ y$ точно линейно независимы (так как случай с $x = y$ был рассмотрен выше), то размерность пространства решений над полем из трёх элементов равна $n - 2$, а число таких решений равно 3^{n-2}. Значит вес, вносимый слагаемыми такого типа в общую сумму, будет равен $4 \cdot 3^{n-2}$.

Резюмируя вычисления, рассмотрим, какой же вид имеет каждый из элементов матрицы P.

1) $x^i = y^j \Rightarrow (P)_{x^i, y^j} = 3^{n-1} + 4 \cdot 3^{n-1} = 5 \cdot 3^{n-1}$

2) $y^j = 2x^i \Rightarrow (P)_{x^i, y^j} = 2 \cdot 3^{n-1} + 2 \cdot 3^{n-1} = 4 \cdot 3^{n-1}$

3) $x^i = 2y^j \Rightarrow (P)_{x^i, y^j} = 2 \cdot 3^{n-1} + 2 \cdot 3^{n-1} = 4 \cdot 3^{n-1}$

4) $\forall v = v_1, ..., v_k : x^i[v] = 2y^j[v],\ \forall w = w_1, ..., w_m : y^j[w] = 2x^i[w],\ k + m = n \Rightarrow (P)_{x^i, y^j} = 2 \cdot 3^{n-1} + 2 \cdot 3^{n-1} = 4 \cdot 3^{n-1}$

5) $else \Rightarrow (P)_{x^i, y^j} = 3^{n-2} + 2 \cdot 3^{n-2} + 2 \cdot 3^{n-2} + 4 \cdot 3^{n-2} = 9 \cdot 3^{n-2} = 3^n$

Представим эти данные более удобно в виде системы:

$$(P)_{x^i,y^j} = \begin{cases} 5 \cdot 3^{n-1} & \text{if } x^i = y^j \\ 4 \cdot 3^{n-1} & \text{if } y^j = 2x^i \\ 4 \cdot 3^{n-1} & \text{if } x^i = 2y^j \\ 4 \cdot 3^{n-1} & \text{if } \forall v = v_1, ..., v_k : x^i[v] = 2y^j[v],\ \forall w = w_1, ..., w_m : y^j[w] = 2x^i[w],\ k + m = n \\ 3^n & \text{else} \end{cases}$$

Для наглядности, самостоятельно построим матрицу P в более тривиальных случаях, а именно при $n = 1,\ n = 2$.

$$n = 1 : P = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 5 \cdot 3^{n-1} & 4 \cdot 3^{n-1} \\ 0 & 4 \cdot 3^{n-1} & 5 \cdot 3^{n-1} \end{pmatrix} \end{array} = \begin{array}{c} \\ 0 \\ 1 \\ 2 \end{array} \begin{array}{c} \begin{array}{ccc} 0 & 1 & 2 \end{array} \\ \begin{pmatrix} 0 & 0 & 0 \\ 0 & 5 & 4 \\ 0 & 4 & 5 \end{pmatrix} \end{array}$$

$$n = 2 : P = \begin{array}{c} \\ (0,0) \\ (0,1) \\ (0,2) \\ (1,0) \\ (1,1) \\ (1,2) \\ (2,0) \\ (2,1) \\ (2,2) \end{array} \begin{array}{c} \begin{array}{ccccccccc} (0,0) & (0,1) & (0,2) & (1,0) & (1,1) & (1,2) & (2,0) & (2,1) & (2,2) \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 \cdot 3^{n-1} & 4 \cdot 3^{n-1} & 3^n & 3^n & 3^n & 3^n & 3^n & 3^n \\ 0 & 4 \cdot 3^{n-1} & 5 \cdot 3^{n-1} & 3^n & 3^n & 3^n & 3^n & 3^n & 3^n \\ 0 & 3^n & 3^n & 5 \cdot 3^{n-1} & 3^n & 3^n & 4 \cdot 3^{n-1} & 3^n & 3^n \\ 0 & 3^n & 3^n & 3^n & 5 \cdot 3^{n-1} & 3^n & 3^n & 3^n & 4 \cdot 3^{n-1} \\ 0 & 3^n & 3^n & 3^n & 3^n & 5 \cdot 3^{n-1} & 3^n & 4 \cdot 3^{n-1} & 3^n \\ 0 & 3^n & 3^n & 4 \cdot 3^{n-1} & 3^n & 3^n & 5 \cdot 3^{n-1} & 3^n & 3^n \\ 0 & 3^n & 3^n & 3^n & 3^n & 4 \cdot 3^{n-1} & 3^n & 5 \cdot 3^{n-1} & 3^n \\ 0 & 3^n & 3^n & 3^n & 4 \cdot 3^{n-1} & 3^n & 3^n & 3^n & 5 \cdot 3^{n-1} \end{pmatrix} \end{array}$$

$$P = \begin{array}{c} \\ (0,0) \\ (0,1) \\ (0,2) \\ (1,0) \\ (1,1) \\ (1,2) \\ (2,0) \\ (2,1) \\ (2,2) \end{array} \begin{array}{c} \begin{array}{ccccccccc} (0,0) & (0,1) & (0,2) & (1,0) & (1,1) & (1,2) & (2,0) & (2,1) & (2,2) \end{array} \\ \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 15 & 12 & 9 & 9 & 9 & 9 & 9 & 9 \\ 0 & 12 & 15 & 9 & 9 & 9 & 9 & 9 & 9 \\ 0 & 9 & 9 & 15 & 9 & 9 & 12 & 9 & 9 \\ 0 & 9 & 9 & 9 & 15 & 9 & 9 & 9 & 12 \\ 0 & 9 & 9 & 9 & 9 & 15 & 9 & 12 & 9 \\ 0 & 9 & 9 & 12 & 9 & 9 & 15 & 9 & 9 \\ 0 & 9 & 9 & 9 & 9 & 12 & 9 & 15 & 9 \\ 0 & 9 & 9 & 9 & 12 & 9 & 9 & 9 & 15 \end{pmatrix} \end{array}$$

Примем за гипотезу то, что ранг матрицы в общем случае равен $3^n - 1$. Далее это будет доказано.

Для того, чтобы наглядно увидеть, как выглядит матрица P при разных n, и чему равен её ранг, мною был написан небольшой код (приложение 1). Заодно можно убедиться, что значения матрицы P посчитаны мною верно. Уточню, что n во всех функциях – это длина векторов x и y.

1) Для начала я написала функцию $fill\_nums(n)$, которая создает массив из всех возможных векторов длины n над полем $\mathbb{Z}_3$, и при этом все векторы в масиве отсортированы в лексикографическом порядке.

```
def fill_nums(n):
  size = pow(3, n)
  nums = [[0 for x in range (0, n)] for y in range (0, size)]
  for i in range (0, size):
    el = i
    count = n - 1
    while i > 2:
      res = i % 3
      nums[el][count] = res
      i = i // 3
      count = count - 1
    nums[el][count] = i

  return nums
```

2) Также функцию, которая вычисляет скалярное произведение двух векторов по модулю 3.

```
def IP_3(x, y):
  return np.sum(np.dot(x, y)) % 3
```

3) Затем создала непосредственно функцию $fill\_matrix(n)$, заполняющую матрицу P по правилу.

```
def fill_matrix(n):
   size = pow(3, n)
   matrix = [[0 for x in range (0, size)] for y in range (0, size)]
   nums = fill_nums(n)
   for i in range(0, size):
     for j in range(0, size):
       for z in range(0, size):
         ip_1 = IP_3(nums[i], nums[z])
         ip_2 = IP_3(nums[z, nums[j]])
         matrix[i][j] = matrix[i][j] + ip_1 * ip_2

   return matrix
```

4) Также я написала функцию $rank\_check(max)$, вычисляющую ранг матрицы P от $n = 1$ до заданного числа max и проверяющую, что ранг действительно равен $3^n - 1$.

```
def rank_check(max):
  count = 0
  for n in range(1, max + 1):
    matr = fill_matrix(n)
    rk = np.linalg.matrix_rank(matr)
    if rk == pow(3, n) - 1:
      count = count + 1
      print("Ранг матрицы P при n =", n, "равен", rk, "= 3^n - 1")
    else:
      print("Ранг матрицы P при n =", n, "равен", rk, "!= 3^n - 1")

  print()

  if count == max:
    print("Для всех предложенных n наше утверждение подтвердилось. rk(P) = 3^n - 1")
  else:
    print("Утверждение верно не для всех предложенных n. rk(P) != 3^n - 1")
```

Ниже можно видеть результат работы данной функции для $n = 8$.

```
n = 8
rank_check(n)

Ранг матрицы P при n = 1 равен 2 = 3^n - 1
Ранг матрицы P при n = 2 равен 8 = 3^n - 1
Ранг матрицы P при n = 3 равен 26 = 3^n - 1
Ранг матрицы P при n = 4 равен 80 = 3^n - 1
Ранг матрицы P при n = 5 равен 242 = 3^n - 1
Ранг матрицы P при n = 6 равен 728 = 3^n - 1
Ранг матрицы P при n = 7 равен 2186 = 3^n - 1
Ранг матрицы P при n = 8 равен 6560 = 3^n - 1
```

Теперь докажем, что $rk(P) = 3^n - 1$. Рассмотрим ненулевые строки матрицы P и убедимся, что они линейно независимы. Заметим, что в нашей матрице есть всего 3 различных вида значений: 3^n, $5 \cdot 3^{n-1}$, $4 \cdot 3^{n-1}$. Также заметим, что на главной диагонали матрицы находятся значения $5 \cdot 3^{n-1}$. Для начала сразу скажем, что одинаковых строк у данной матрицы нет, что очевидно из построения, как минимум потому что элементы вида $5 \cdot 3^{n-1}$ находятся только на главной диагнали и больше нигде. Также нет пропорциональных строк, что тоже ясно из видов элементов матрицы. В каждой из строк ровно 1 элемент имеет вид $4 \cdot 3^{n-1}$ и ровно 1 элемент имеет вид $5 \cdot 3^{n-1}$ (диагональный), все остальные элменты равны 3^n. Причем заметим, что элементы вида $4 \cdot 3^{n-1}$ и $5 \cdot 3^{n-1}$ находятся в каждой строчке на новой позиции. Для элментов вида $5 \cdot 3^{n-1}$ это очевидно, так как они находятся на главной диагонали. Объясню для элементов вида $4 \cdot 3^{n-1}$. К примеру, если в i-ой строчке j-ый элемент равен $4 \cdot 3^{n-1}$, то в k-ой строчке j-ый элемент точно не равен $4 \cdot 3^{n-1}$, где $i \neq k$. Это верно из-за того, что для конкретного фиксированного номера строчки, то есть вектора длины n над полем $\mathbb{Z}_3$), назовем её x^i, существует только один другой вектор длины n над полем $\mathbb{Z}_3$ – y^j, для которых выполнено $x^i = 2y^j$ (или $y^j = 2x^i$, что эквивалентно). Таким образом только элемент матрицы $(P)_{x^i,x^j} = 4 \cdot 3^{n-1}$.

Значит в итоге мы имеем матрицу, у которой на главной диагонали стоят элементы $5 \cdot 3^{n-1}$ и в каждой строчке имеется элемент вида $4 \cdot 3^{n-1}$, находящийся на разных позициях для разных строчек. Причем если есть строчка, у которой на i-ой позиции $4 \cdot 3^{n-1}$, а на j-ой $5 \cdot 3^{n-1}$, то найдётся строчка, у которой на j-ой $4 \cdot 3^{n-1}$, а на i-ой $5 \cdot 3^{n-1}$, так как матрица симметрична по построению (из-за нумерации строк и столбцов). Тогда интуитивно ясно, что никаким образом представить конкретную строчку матрицы в виде линейной комбинации других строк не выйдет.

Расмотрим строку, у которой $4 \cdot 3^{n-1}$ находится на i-ой позиции, а $5 \cdot 3^{n-1}$ на j-ой и попробуем представить её в виде линейной комбинации других строк. Если наша строка имеет такой вид, то это значит, что в других строках на i-ой позиции в одной из строк находится элемент $5 \cdot 3^{n-1}$, а во всех остальных ($3^n - 3$ строчках) 3^n, а на j-ой позиции в одной из строк находится элемент $4 \cdot 3^{n-1}$, а во всех остальных ($3^n - 3$ строчках) 3^n. Причем строчка, у которой на i-ой позиции находится элемент $5 \cdot 3^{n-1}$ и строчка, у которой на j-ой позиции находится элмент $4 \cdot 3^{n-1}$ – одна и та же строчка. То есть для того, чтобы выразить данную строку через другие нужно как минимум, чтобы у данной системы существовало решение (пока не рассматриваем представление элементов 3^n):

$$\begin{cases} a \cdot 5 \cdot 3^{n-1} + \sum\limits_{i=1}^{3^n-3} b_i \cdot 3^n = 4 \cdot 3^{n-1} \\ a \cdot 4 \cdot 3^{n-1} + \sum\limits_{i=1}^{3^n-3} b_i \cdot 3^n = 5 \cdot 3^{n-1} \end{cases} \Rightarrow \left[\sum_{i=1}^{3^n-3} b_i \cdot 3^n = c \right] \Rightarrow \begin{cases} a \cdot 5 \cdot 3^{n-1} = 4 \cdot 3^{n-1} - c \\ a \cdot 4 \cdot 3^{n-1} = 5 \cdot 3^{n-1} - c \end{cases}$$

$$\Rightarrow \begin{cases} 3^{n-1}(5a-4) = -c \\ 3^{n-1}(4a-5) = -c \end{cases} \Rightarrow 5a-4 = 4a-5 \Rightarrow a = -1$$

$$\Rightarrow 3^{n-1} \cdot (-9) = -c = -3 \cdot 3^n \Rightarrow c = 3 \cdot 3^n = \sum_{i=1}^{3^n-3} b_i \cdot 3^n = 3^n \sum_{i=1}^{3^n-3} b_i \Rightarrow$$

$$\Rightarrow \sum_{i=1}^{3^n-3} b_i = 3$$

А теперь вспомним, что также нам нужно представить элементы (3^n) на других позициях в нашей строке в виде линейной комбинации. Рассмотрим k-ую позицию строки. Тогда в других строках на k-ой позиции в одной из строк будет находиться $4 \cdot 3^{n-1}$, в одной $5 \cdot 3^{n-1}$, а во всех остальных 3^n.

Тогда:

$$b_{i_{1k}} \cdot 4 \cdot 3^{n-1} + b_{i_{2k}} \cdot 5 \cdot 3^{n-1} + \sum_{i=1}^{3^n-5} b_i \cdot 3^n + a \cdot 3^n = 3^n$$

Так как выше мы нашли, что коэффициент a должен быть -1, а $c = 3$, то это значит, что:

$$b_{i_{1k}} \cdot 4 \cdot 3^{n-1} + b_{i_{2k}} \cdot 5 \cdot 3^{n-1} + \sum_{i=1}^{3^n-5} b_i \cdot 3^n = 2 \cdot 3^n, \ \sum_{i=1}^{3^n-5} b_i + b_{i_{1k}} + b_{i_{2k}} = 3$$

$$\left[b_{i_{1k}} \cdot 4 \cdot 3^{n-1} = \frac{4b_{i_{1k}}}{3} \cdot 3^n, \ b_{i_{2k}} \cdot 5 \cdot 3^{n-1} = \frac{5b_{i_{2k}}}{3} \cdot 3^n \right]$$

$$\Rightarrow \frac{4b_{i_{1k}}}{3} \cdot 3^n + \frac{5b_{i_{2k}}}{3} \cdot 3^n + \left(\sum_{i=1}^{3^n-5} b_i \right) \cdot 3^n = 2 \cdot 3^n$$

$$\Rightarrow \left(\frac{4b_{i_{1k}}}{3} + \frac{5b_{i_{2k}}}{3} + \sum_{i=1}^{3^n-5} b_i - 2 \right) 3^n = 0$$

$$\Rightarrow \frac{4b_{i_{1k}}}{3} + \frac{5b_{i_{2k}}}{3} + \sum_{i=1}^{3^n-5} b_i = 2$$

Значит в итоге нам нужно понять, есть ли решения у такой системы:

$$\begin{cases} \frac{4b_{i_{1k}}}{3} + \frac{5b_{i_{2k}}}{3} + \sum\limits_{i=1}^{3^n-5} b_i = 2 \\ \sum\limits_{i=1}^{3^n-5} b_i + b_{i_{1k}} + b_{i_{2k}} = 3 \end{cases} \Rightarrow \left[\sum_{i=1}^{3^n-5} b_i = 3 - b_{i_{1k}} - b_{i_{2k}} \right]$$

$$\Rightarrow \begin{cases} b_{i_{1k}} + 2b_{i_{2k}} = -3 \\ \sum\limits_{i=1}^{3^n-5} b_i + b_{i_{1k}} + b_{i_{2k}} = 3 \end{cases} \Rightarrow \begin{cases} b_{i_{1k}} = -3 - 2b_{i_{2k}} \\ b_{i_{2k}} = \sum\limits_{i=1}^{3^n-5} b_i - 6 \end{cases}$$

И так далее, при рассмотрении других элементов нашей исходной строки, которые равны 3^n (пускай на r-ой позиции в строчке) накладываются точно такие же условия на $b_{i_{1r}}$ и $b_{i_{2r}}$, где $b_{i_{1r}}$

– коэффициент в линейной комбинации при строчке, в которой элемент $5 \cdot 3^{n-1}$ стоит на той r-ой позиции, $b_{i_{2r}}$ – коэффициент в линейной комбинации при строчке, в которой элемент $4 \cdot 3^{n-1}$ стоит на r-ой позиции. Значит для каждой из оставшихся строчек, кроме той, где элементы $5 \cdot 3^{n-1}$ и $4 \cdot 3^{n-1}$ располагаются на тех же позициях (симметрично), что и в нашей строчке, будет накладываться такое условие на $b_{i_{1r}}$ и $b_{i_{2r}}$. Очевидно, что одновременно все эти условия выполняться не могут. Как минимум потому что на одну и ту же конкретную строчку накладывается и условие на $b_{i_{1l}}$, в случае когда мы рассматриваем элемент 3^n в исходной строчке на той позиции, что и $5 \cdot 3^{n-1}$ в этой (на l-ой), и условие на $b_{i_{2s}}$, в случае когда мы рассматриваем элемент 3^n в исходной строчке на той позиции, что и $4 \cdot 3^{n-1}$ в этой (на s-ой). Несложно заметить, что одновременно эти условия выполняться не могут. Значит любая из строчек матрицы не представима в виде линейной комбинации остальных. Значит строчки матрицы (не рассматриваем первую) линейно независимы.

Таким образом:

$$rk(P) = 3^n - 1$$

Наконец, перейдём к вычислению ранга исходной матрицы M_{IP_3}. Ясно, что при возведении в степень ранг матрицы не мог возрасти и при этом ранг исходной матрицы точно не был 3^n, так как у нее, как и у её квадрата, были нулевые первая строка и первый столбец. Таким образом:

$$rk(M_{IP_3}) = 3^n - 1$$

Для того, чтобы наглядно понять, как выглядит сама матрица M_{IP_3} (также это помогло на начальном этапе решения задачи) и убедиться, что её ранг равен $3^n - 1$, мною был написан еще один простой код (приложение 1). Функции $fill\_nums(n)$ и $IP_3(x, y)$ аналогичны функциям из предыдущего кода.

1) Я написала функцию $fill\_matrix2(n)$, заполняющую матрицу M_{IP_3}, согласно правилам, по которым заполняется матрица значений функции.

```
def fill_matrix2(n):
   size = pow(3, n)
   matrix = [[0 for x in range (0, size)] for y in range (0, size)]
   nums = fill_nums(n)
   for i in range(0, size):
     for j in range(0, size):
       matrix[i][j] = IP_3(nums[i], nums[j])
   return matrix
```

2) Затем создала функцию $rank\_check2(max)$, которая вычисляет ранг матрицы M_{IP_3} для $n = 1$ до заданного числа max. И также эта функция проверяет, что ранг M_{IP_3} действительно равен $3^n - 1$.

```
def rank_check2(max):
  count = 0
  for n in range(1, max + 1):
    matr = fill_matrix2(n)
    rk = np.linalg.matrix_rank(matr)
    if rk == pow(3, n) - 1:
      count = count + 1
      print("Ранг матрицы M_f при n =", n, "равен", rk, "= 3^n - 1")
    else:
      print("Ранг матрицы M_f при n =", n, "равен", rk, "!= 3^n - 1")

  print()

  if count == max:
    print("Для всех предложенных n наше утверждение подтвердилось. rk(M_f) = 3^n - 1")
  else:
    print("Утверждение верно не для всех предложенных n. rk(M_f) != 3^n - 1")
```

Результат её работы для $n = 8$ можно видеть ниже.

```
n = 8
rank_check2(n)

Ранг матрицы M_f при n = 1 равен 2 = 3^n - 1
Ранг матрицы M_f при n = 2 равен 8 = 3^n - 1
Ранг матрицы M_f при n = 3 равен 26 = 3^n - 1
Ранг матрицы M_f при n = 4 равен 80 = 3^n - 1
Ранг матрицы M_f при n = 5 равен 242 = 3^n - 1
Ранг матрицы M_f при n = 6 равен 728 = 3^n - 1
Ранг матрицы M_f при n = 7 равен 2186 = 3^n - 1
Ранг матрицы M_f при n = 8 равен 6560 = 3^n - 1
```

Итак, мы вычислили ранг матрицы M_{IP_3}, теперь применим утверждение, которое хотели применить, а именно:

$$CC(f) \geq \log(rk(M_f))$$

Значит в нашем случае $CC(IP_3) \geq \log(3^n - 1)$. В самом начале мы получили следующую верхнюю оценку: $CC(IP_3) \leq \lceil \log 3^n \rceil + \lceil \log 3 \rceil = \lceil \log 3^n \rceil + 2$. График 1 визуализирует верхние и нижние оценки на коммуникационную сложность функции в зависимости от n. Ось x отвечает за значения n, ось y за значения $CC(IP_3)$.

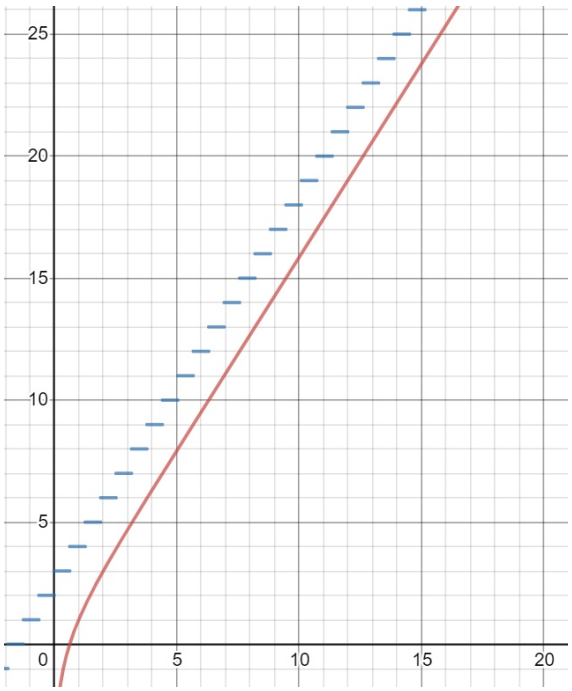

График 1. Красный график – $\log(3^n - 1)$, *синий* – $\lceil \log 3^n \rceil + 2$

Для более хорошего понимания скорости роста коммункационной сложности функции сравним её с некоторыми стандартными – а именно с линейной, квадратичной и $n \ln n$ (График 2).

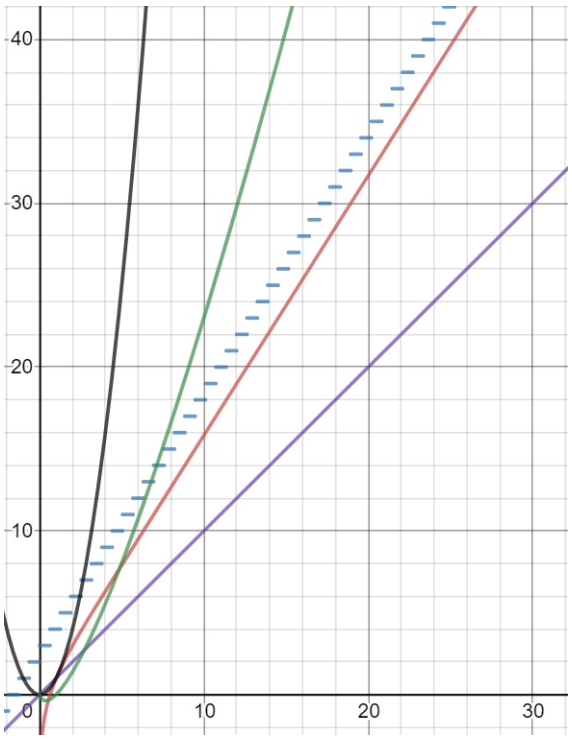

График 2. Красный график – $\log(3^n - 1)$, *синий* – $\lceil \log 3^n \rceil + 2$, *фиолетовый* – n, *зелёный* – $n \ln n$, *чёрный* – n^2

Таким образом мы можем заметить, что скорость роста детерменированной коммуникационной сложности при всех n больше линейной, при $n \geq 7$ меньше, чем $n \ln n$ и при $n \geq 3$ меньше квадратичной.

Резюмируя, мы имеем такую оценку на детерминированную коммуникационную сложность функции IP_3:

$$\log(3^n - 1) \leq CC(IP_3) \leq \lceil \log 3^n \rceil + 2$$

3.3 Недетерминированная коммуникационная сложность функции скалярного произведения

Теперь попробуем дать оценки на недетерминированную сложность, рассматриваемой нами функции. Напомню, функция имеет следующий вид:

$$IP_3(x, y) = \sum_{i=1}^{n} x_i y_i (mod\ 3)$$

Для того, чтобы дать верхнюю оценку на недетрминированную коммуникационную сложность функции IP_3 воспользуемся следующим утверждениям:

$$\text{Для любой функции } f : X \times Y \to Z:$$
$$NCC(f) \leq \lceil \log(cov(f)) \rceil + 1,\ cov(f) \leq 2^{CC(f)}.$$

Ранее мы выяснили, что

$$CC(IP_3) \leq \lceil \log 3^n \rceil + 2$$

Значит

$$2^{CC(IP_3)} \leq 2^{\lceil \log 3^n \rceil + 2} = 2^{\lceil \log 3^n \rceil} \cdot 4 \Rightarrow cov(IP_3) \leq 4 \cdot 2^{\lceil \log 3^n \rceil}$$

Таким образом мы можем сделать вывод, что

$$NCC(IP_3) \leq \lceil \log(cov(IP_3)) \rceil + 1 \leq \lceil \log(4 \cdot 2^{\lceil \log 3^n \rceil}) \rceil + 1 = \lceil \log 4 + \log(2^{\lceil \log 3^n \rceil}) \rceil + 1 =$$

$$= \lceil 2 + \lceil \log 3^n \rceil \rceil + 1 = \lceil \log 3^n \rceil + 3$$

В конечном итоге мы получили следующую верхнюю оценку на недетерминированную коммуникационную сложность функции:

$$NCC(IP_3) \leq \lceil \log 3^n \rceil + 3$$

3.4 Вероятностная коммуникационная сложность функции скалярного произведения

Наконец, попробуем оценить вероятностную коммуникационную сложность функции IP_3.

Для того, чтобы дать нижнюю оценку на вероятностую коммуникационную сложность, воспользуемся следующим утверждением:

$$RCC_{\varepsilon}(f) = \Omega(\log CC(f))$$

То есть в нашем случае:

$$RCC_{\varepsilon}(IP_3) = \Omega(\log CC(IP_3))$$

Теперь разберёмся, что это значит. По определению:

$$f(n) = \Omega(g(n)), \text{ если существуют действительные константы } c > 0 \text{ и } n_0 > 0, \text{ такие, что}$$
$$f(n) \geq c \cdot g(n) \text{ для всех } n > n_0.$$

Ранее нами была получена следующая нижняя оценка на детерминированную коммуникационную сложность:

$$CC(IP_3) \geq \log(3^n - 1) \Rightarrow \log CC(IP_3) \geq \log(\log(3^n - 1))$$

Тогда из определения Ω ясно, что

$$f(n) = \Omega(\log CC(IP_3)) \Rightarrow f(n) = \Omega(\log(\log(3^n - 1)))$$

Так как очевидно, что если найдётся константа c, для которой при достаточно больших n выполнено, что $f(n) \geq c \cdot \log CC(IP_3)$, то для той же константы выполнено $f(n) \geq c \cdot \log(\log(3^n - 1)$, так как $\log CC(IP_3) \geq \log(\log(3^n - 1))$

Таким образом мы получаем следующее:

$$RCC_\varepsilon(IP_3) = \Omega(\log(\log(3^n - 1)))$$

4 Заключение

Подводя итог, хочу сказать, что работа над проектом стала довольно большим и полезным опытом для меня. За время работы над ним мною были получены базовые теоретические знания о вычислении коммуникационной сложности. Также я увидела, как именно эти знания применяются на практике для решения конкретных задач. Еще, что немаловажно, я смогла узнать о применении коммуникационной сложности для решения реальных задач из разных разделов компьютерных наук и убедиться в том, насколько данный раздел теоретической информатики актуален.

Кульминацией проекта стало самостоятельное примение некоторых полученных знаний для вычисления коммуникационной сложности функции скалярного произведения. При решении поставленной задачи возникло немало трудностей, но это стало отличным опытом проведения исследовательской работы, который, я уверена, будет полезен мне в будущем.

5 Список используемых источников

[1] В.В. Подольский, А.Е. Ромащенко. Конспект лекций «Введение в коммуникационную сложность», 2012.

[2] E. Kushilevitz, and N. Nisan. «Communication complexity». Cambridge University Press, 1997.

[3] Yao, A. C., «Some Complexity Questions Related to Distributed Computing», 1979.

[4] Александр Александрович Разборов. Коммуникационная сложность. Серия Летняя школа «Современная математика». МЦНМО, 2012.

[5] Верещагин Н. К., Щепин Е. В. «Информация, кодирование и предсказание» – М.: ФМОП, МЦНМО, 2012.

[6] Курс «Коммуникационная сложность» – Н. К. Верещагин, 2017.

[7] Википедия. Коммуникационная сложность. URL: `https://clck.ru/PArHf`

6 Приложения

1. Ссылка на google collab с кодом.
(https://colab.research.google.com/drive/1g_ao7Fso7yXt1Gnh6_x8VPKTKBzwiv6B?usp=sharing)