Отчет по проекту по курсу «Теория информации и теория чисел»

Введем некоторые обозначения, которые будут использованы при оценке трудоемкости алгоритмов:

n – размер множества A; в условии задания $n = 10000$

q – верхняя граница диапазона выбора простых чисел p; в условии задания $q = 300$

Так же отметим тот факт, что количество простых чисел меньших q ассимптотически равняется $O\left(\frac{q}{\ln q}\right)$. Это будет использовано при оценке количества перебираемых простых чисел p.

Задание 1.

Для каждого значения p из заданного промежутка при помощи метода `get_residue_class_sizes()` посчитаем размерности классов вычетов $0..p-1$ для множества A. Далее по указанной формуле посчитаем «неравномерность» покрытия поля Z_p и выберем p с минимальным значением неравномерности.

Трудоемкость алгоритма $T(n,q) = O\left(\frac{q}{\ln q}(n+q)\right)$.

Задание 2.

Заметим, что размерность множества A превышает количество элементов поля Z_p, поэтому нет необходимости проверять на принадлежность кривой каждый элемент множества A. Достаточно проверить принадлежность каждого x в промежутке $[0;p)$ и, в случае принадлежности его кривой, прибавить к ответу размерность класса вычетов x. Для того, чтоб использовать этот факт, для каждого p мы один раз посчитаем размерности его классов вычетов для множества A с помощью метода `get_residue_class_sizes()` из задания 1.

Так же следует заметить, что для того чтобы проверить, найдется ли y при заданных p, a, b, x, нет необходимости проверять все значения y. Нам достаточно знать, является ли значение выражения $x^3 + ax + b$ квадратичным вычетом в поле Z_p. Для применения этого факта для каждого p один раз найдем все квадратичные вычеты по модулю p.

Тогда для поиска максимального $N(p,a,b)$ переберем параметры эллиптической кривой 3-мя вложенными циклами: p может принимать значения простых чисел в заданном интервале, а параметры a и b - все целые числа в промежутке $[0;p)$, т.к. все операции совершаются в поле Z_p. Для каждой тройки параметров переберем x в промежутке $[0;p)$, посчитаем значение выражения $x^3 + ax + b$, и, если результат является квадратичным вычетом в поле Z_p, прибавим к ответу размерность класса вычетов x.

Трудоемкость полученного алгоритма равняется: $T(q,n) = O\left(\frac{q}{\ln q}(n+q^3)\right)$.

На практике такой алгоритм работает недостаточно быстро (около 5 минут). Его можно ускорить, заменив перебор b и x на одно применение FFT. Тогда трудоемкость алгоритма уменьшится до $O\left(\frac{qn}{ln\,q}+q^3\right)$, что на практике будет работать достаточно быстро. Это будет реализовано на следующем этапе.

Задание 3.

С помощью метода `get_curve_points()` получим все точки, принадлежащие эллиптической кривой с параметрами a, b в поле Z_p. Полученные точки вместе с нейтральным элементом (точкой O) образуют группу. Поэтому порядком группы будет количество точек кривой, увеличенное на 1.

По определению, порядком элемента g аддитивной группы называется такое наименьшее число m, что $gm = O$, где $O -$ нейтральный (нулевой) элемент группы. Поэтому для определения порядка элемента g, будем в некоторую переменную прибавлять g, накапливая результат, пока он не станет равным нулевому элементу. Проделаем эту операцию для каждого элемента полученной группы при помощи метода `generate_cyclic_subgroup()`.

Для реализации сложения двух точек эллиптической кривой был реализован класс `ElCurvePoint` с перегруженным оператором сложения, а так же некоторыми другими операторами для удобства работы с элементами группы.

Трудоемкость описанного алгоритма $T(q) = O(q^2)$.

Задание 4.

В качестве алфавита возьмем пересечение множества абсцисс точек эллиптической кривой и множества A в поле Z_p. В качестве текста T возьмем множество элементов A, таких, которые в поле Z_p равняются одному из символов алфавита.

Генерацию алфавита вместе с частотами символов в T выполняет метод `get_alphabet()`. Трудоемкость этого метода равняется $O(n)$, или $O(q)$ в случае, если заранее посчитаны размеры классов вычетов поля Z_p для множества A.

Далее мы строим дерево Хаффмана для полученного алфавита при помощи метода `build_haffman_code()`. Дерево строится стандартным алгоритмом, используя приоритетную очередь для хранения поддеревьев в порядке возрастания их частоты. Для построения дерева используются объекты класса `Node`, представляющие узлы(вершины). Метод возвращает массив двоичных кодов для символов алфавита. Трудоемкость метода $T(s) = O(s\log s)$, где $s = O(q)$ - размер алфавита. Метод так же можно ускорить до $O(s)$, если вместо приоритетной очереди использовать 2 обычных, однако при заданных ограничениях в этом нет необходимости.

Далее вычисляется энтропия текста T с помощью метода `calculate_entropy()`, а так же длина кода текста и средняя длина кода символа по стандартным формулам.

Задание 5.

Для начала заметим, что множество A формировалось из равновероятно выбранных значений в диапазоне $[0; 10^9]$. Этот диапазон гораздо больше, чем размеры рассматриваемых полей. Так же мощность самого множества A в несколько десятков раз больше размера полей Z_p. Поэтому для любого p из заданного промежутка можно сказать, что элементы множества A равномерно распределены по полю Z_p, т.е. размеры всех классов вычетов поля равны между собой. На самом деле это распределение стремится к равномерному, полностью равномерным оно было бы при $n \to \infty$ и $p|U$, где U - размер диапазона выбора значений. Однако для практического исследования будем считать, что оно равномерное, т.к. мера неравномерности очень мала (как показывают результаты задания 1).

Если размерности всех классов вычетов поля Z_p равны, то, очевидно, они равны и для любого подмножества классов вычетов. Напомним, что алфавитом является множество различных чисел x из поля Z_p, т.е. подмножество его классов вычетов. Следовательно, все символы алфавита имеют одинаковые частоты, равные $\frac{1}{s}$, где s - размер алфавита. Тогда энтропия Шеннона для этого распределения будет равна

$$E = -\sum_{i=1}^{s} p_i \log_2 p_i = -\sum_{i=1}^{s} \frac{1}{s} log_2 \frac{1}{s} = -s \frac{1}{s} log_2 \frac{1}{s} = -log_2 \frac{1}{s} = \log_2 s.$$

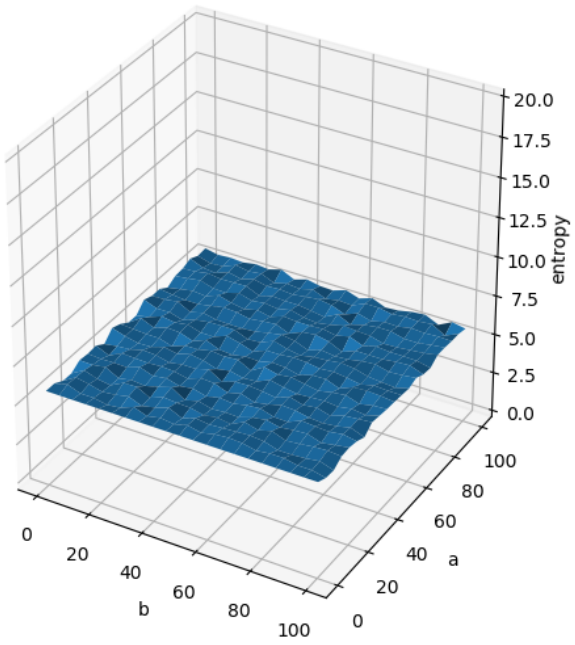
Кодирование Хаффмана предназначено для оптимизации по памяти текста, в котором некоторые символы алфавита встречаются чаще других. Применение кодирования Хаффмана для алфавитов с равными частотами символов бесполезно. Результатом такого кодирования будет назначение всем символам кодов одинаковой длины. Существует 2^l различных двоичных кодов длины l. Т.к. алгоритм Хаффмана выбирает оптимальные длины символов, будет выбрано такое l, при котором количество различных кодов будет равно s, т.е. количеству символов в алфавите. Из уравнения $2^l = s$ получаем, что $l = \log_2 s$. Если длина кода каждого символа равна $\log_2 s$, то средняя длина кода будет, очевидно, тоже равна $\log_2 s$.

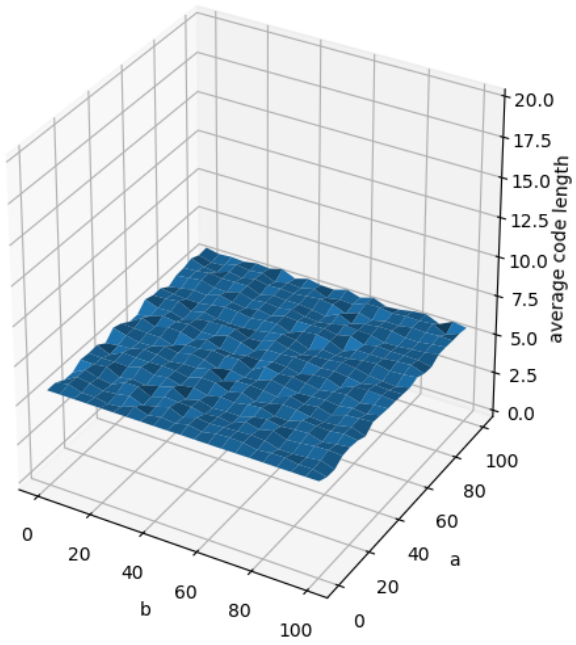
Получается, что как энтропия, так и средняя длина кода равняются $\log_2 s$ и, очевидно, зависят только от s - размера алфавита. Вернувшись к пункту 4, вспомним, что алфавитом является множество различных абсцисс точек эллиптической кривой. Количество точек эллиптической кривой в поле Z_p не имеет определенной зависимости от параметров кривой a и b, можно сказать, что эта зависимость практически случайная. Это количество на любой эллиптической кривой в поле Z_p равняется $\approx \frac{p}{2}$.

Вывод: в условии данного задания энтропия Шеннона и средняя длина кода не имеют определенной зависимости от параметров кривой a, b, а зависят только от размерности поля p, в котором эта кривая находится, и равняются $\approx log_2 \frac{p}{2}$.

Это подтверждается полученными на практике графиками:

Энтропия Шеннона и средняя длина кода для p = 101





Энтропия Шеннона и средняя длина кода для p = 293