**Умножение многочленов в Zp с помощью быстрого преобразования Фурье и параллелизма с использованием технологии CUDA**

Пусть имеется многочлен n-ой степени:

Не теряя общности, можно считать, что n является степенью 2. Если в действительности n не является степенью 2, то мы просто добавим недостающие коэффициенты, положив их равными нулю.

В поле Zp необходимо найти такой элемент, порядок которого будет равен n. Обозначим этот элемент Wn. Если такой элемент существует, то выполнение умножения многочленов с помощью ДПФ возможно. Также необходимо выполнение следующего условия для умножения многочленов A(x) и B(x):

Дискретным преобразованием Фурье многочлена A(x) (или, что то же самое, ДПФ вектора его коэффициентов (a0, a1,…,an-1)) называются значения этого многочлена в точках , то есть это вектор

Тогда обратное дискретное преобразование Фурье для вектора значений многочлена (y0, y1, …, yn-1) – это вектор коэффициентов многочлена (a0, a1,…,an-1).

Таким образом, для умножения двух многочленов A(x) и B(x) необходимо найти ДПФ каждого из них, затем умножить каждый элемент одного вектора на соответствующий ему элемент другого вектора и применить обратное ДПФ.

Быстрое преобразование Фурье – это метод, позволяющий вычислять ДПФ за время . Этот метод основывается на свойствах комплексных корней из единицы (а именно, на том, что степени одних корней дают другие корни).

Логика работы программы:

1. Выбирается простой модуль (MOD), для него устанавливается элемент поля, порядок которого равен степени двойки (ROOT). Степень двойки должна быть не меньше, чем сумма степеней умножаемых многочленов (ROOT\_ORDER). Примеры таких значений приведены в таблице после описания алгоритма.
2. В коде задается размер блоков (blockSize) и считается количество блоков (numBlocks) для параллелизации с помощью CUDA.
3. Пользователю предлагается выбрать входные данные (вектор коэффициентов многочленов): либо сгенерировать рандомно, либо использовать захардкоженные значения (небольшие многочлены, чтобы удобно было вручную проверить). В конечном итоге значения записываются в массивы vector1 и vector2.
4. Происходит вычисление БПФ для каждого из векторов коэффициентов многочленов vector1 и vector2.
   1. Используется функция parallel\_fft, которая в качестве параметров принимает указатель на массив vec (вектор коэффициентов многочлена, по которому будет считаться БПФ и куда будет записываться результат), размер массива size, флаг invert (если установлен в true, то будет вычисляться обратное БПФ), количество блоков numBlocks и размер блока blockSize.
   2. Вычисление БПФ происходит следующим образом: сначала выполняется поразрядная обратная перестановка входных значений vec с помощью функции bit\_reversal. Действия по перестановке элементов выполняются над каждым элементом массива независимо, что позволяет распараллелить эту операцию. Затем следуют этапов, на каждом из которых параллельно выполняется набор преобразований бабочки с помощью функции fft\_butterflies. В итоге в массиве vec получаются коэффициенты прямого БПФ.
5. После расчета БПФ для каждого из массивов vector1 и vector2 происходит их поэлементное умножение в функции multiply\_vectors, которая принимает на вход два массива, которые необходимо перемножить, массив, в который необходимо записать результат, и размер массивов. Коэффициенты массивов перемножаются независимо друг от друга, что позволяет распараллелить эту операцию.
6. Затем выполняется обратное БПФ от результирующего массива с помощью той же функции parallel\_fft, но с установленным в true флагом invert. Вместо ROOT, используемого для вычисления поворачивающих множителей преобразований бабочки, используется обратный к ROOT по модулю MOD. Также в конце алгоритма каждый элемент массива vec умножается на обратный к n по модулю MOD в функции invert\_fft\_result. Данные действия выполняются независимо друг от друга, что позволяет распараллелить эту операцию.
7. Полученные коэффициенты массива vec есть набор коэффициентов многочлена, который получается в результате умножения двух многочленов, поданных на вход алгоритма.

Параметры, на которых была протестирована работа программы:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| mod | root | root\_pw | Многочлен 1 | Многочлен 2 |
| 17 | 3 | 24 | 7x3+8x2+3x+4 | 9x2+5x+16 |
| 137 | 10 | 23 | 107x3+126x2+32x+41 | 40x3+106x2+59x+109 |
| 1889 | 31 | 25 | 279x3+283x2+314x+41 | 279x3+283x2+314x+41 |
| 54401 | 114 | 27 | 279x3+283x2+314x+41 | 279x3+283x2+314x+41 |
| 5393 | 14 | 24 | 4568x7+1709x6+312x5+  5390x4+692x3+2990x2+941x+41 | 491x7+648x6+1180x5+2892x4+  2393x3+4938x2+4928x+2288 |