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

Обычное умножение многочленов имеет сложность O(n2), быстрое преобразование Фурье позволяет перемножить два полинома длины n со сложностью O(nlogn), что значительно лучше сложности O(n2).

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 нашем случае под быстрым преобразованием Фурье будем иметь в виду конкретно дискретное преобразование Фурье (ДПФ).

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.

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\_pw). 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. Затем для простоты написания дальнейшего кода заранее генерируются все степени корня Wn и записываются в массив roots\_array.
5. Происходит вычисление ДПФ для каждого из коэффициентов векторов многочлена vector1 и vector2.
   1. Используется функция calc\_fft, которая в качестве параметров принимает указатель на массив fft\_vec (массив, в который будут записываться значения ДПФ), указатель на массив vec (вектор коэффициентов многочлена, по которому будет считаться ДПФ) и указатель на массив roots\_array, который будет использоваться для вычисления ДПФ.
   2. Вычисление ДПФ происходит по следующей формуле: . Каждый коэффициент рассчитывается независимо друг от друга, что позволяет распараллелить эту операцию.
   3. Все коэффициенты ДПФ записываются в массив fft\_vec.
6. После расчета ДПФ для каждого из векторов происходит их поэлементное умножение в функции multiply\_vectors, которая принимает на вход два вектора, которые необходимо перемножить, и вектор, в который необходимо записать результат. Коэффициенты векторов перемножаются независимо друг от друга, что позволяет распараллелить эту операцию.
7. Затем необходимо выполнить операцию, обратную операции ДПФ. Выполняется она по следующей формуле: , где – элемент, обратный к элементу root по модулю mod. Операция выполняется с помощью функции calc\_revert\_fft, которая аналогичная функции calc\_fft. Полученный вектор, записанный в массив rev\_fft\_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 |