В статье [1] рассматривается метод построения минимального множества полиномов двух переменных для данной конечной двумерной последовательности, дано математическое обоснование полученных результатов, однако формальное описание алгоритма, которое было бы пригодно для получения программной реализации, отсутствует: имеется лишь длинная цепочка ссылок на полученный набор теорем и лемм. Таким образом, в данной работе, кроме создания программной системы, возмещен этот недостаток.

Что касается самой реализации, то она выполнена на языке программирования C++, в соответствии с ныне действующим Стандартом 1998 года, и является хорошо переносимой: в частности, были подготовлены сборки под платформы win32 (операционных систем семейства Windows) и Linux. Был использован ряд программных библиотек, как-то: NTL («A Library for doing Number Theory» за авторством Victor Shoup для обеспечения арифметики конечных полей), Boost[3] (коллекция библиотек, поддерживающих создание переносимых программ, ориентированных на современные концепции C++-дизайна).

При создании программного продукта потребовалась разработка ряда структур данных и арифметического блока, оперирующего ими. Структуры данных представлены классами

- полинома,
- точки в четверть-плоскости над конечным полем,
- множества полиномов, составляющих «Δ-множество».

Следует отметить некоторые специальные свойства перечисленных структур данных. Отличительной особенностью алгоритма является то, что арифметические операции с полиномами ограничиваются сложением, умножением на скаляр и умножением на моном. Если первые две являются традиционными, то третья операция, реализованная специальным образом (отличным от общей операции умножения двух полиномов), предоставила возможность для оптимизации.

Точка — суть массив координат фиксированной длины (в данном, двумерном случае — длины два), особенность этой абстракции заключается в попеременном использовании различных способов упорядочения её объектов. А именно, в работе использовано одно полное («мономиальное») упорядочение, которое условно можно назвать градуированным антилексикографическим, и частичный порядок, основанный на покомпонентном сравнении. Во многих стандартных алгоритмах C++ есть возможность удобного использования перегруженной операции класса <, которая должна соответствовать концепции Строго Слабого Упорядочения [4]. Полное упорядочение является частным случаем указанной концепции, так что операция < реализует именно его. Покомпонентное сравнение остается свободной функцией.

Множество полиномов, старшие (относительно введённого полного порядка) степени которых образуют т.н. Δ-множество, и само Δ-множество в исходной работе являются вполне независимыми сущностями. При реализации показалось естественным совместить их в одном классе.

Основными арифметическими операциями являются

- $(\cdot)[\cdot]_{\cdot}$ — операция умножения полинома на последовательность в данной точке,
- BP<·,·> — «процедура Берлекэмпа», строящая по двум полиномам текущего минимального множества (на вход фактически принимает индексы этих полиномов в указанном множестве) новый полином, способный генерировать большее количество элементов исходной последовательности,
- SP<·,·> — «вспомогательная процедура», выполняющая задачу, аналогичную BP, вызываемая в некоторых вырожденных случаях,
- $n \in \Delta + s[i]$, где n, s[i] точки (см. выше) — проверки принадлежности данной точки к «сдвинутому» Δ-множеству.

Стоит упомянуть об отличительной особенности части перечисленных операций, которая состоит в том, что их нельзя представить «чистыми функциями», то есть одни из них активно используют контекст алгоритма, а другие имеют побочные эффекты. Так BP, помимо

текущего минимального множества, использует множество вспомогательных полиномов и соответствующее им множество невязок. А проверка принадлежности, если дала положительный ответ, должна предоставить информацию о том, за счет какого элемента Δ-множества условие выполнилось. Эти и некоторые другие соображения позволили выделить сам алгоритм в самостоятельный класс.

Литература.

1. S. Sakata, «Finding a minimal set of linear recurring relations capable of generating a given finite two–dimensional array,» J. Symb. Comp., vol. 5, pp. 321–337, 1988.
2. http://www.shoup.net/ntl/
3. http://www.boost.org/
4. Остерн М. Г., «Обобщенное программирование и STL: Использование и наращивание стандартной библиотеки шаблонов C++,» пер. с англ. — СПб.: Невский диалект, 2004.