При обработке большого количества запросов к статическим данным для удобства можно заранее производить некоторые несложные операции с массивами входных данных. Например, можно определять для них массивы так называемых префиксных сумм за O(n) [1]. Слово «префиксная» довольно полно раскрывает саму суть понятия: поскольку префиксом в информатике называют подстроку, начинающуюся с начала основной строки, по аналогии образуются и массивы префиксных сумм – они насчитываются с первого элемента массива входных данных.

**Пример 1.** Создание массива префиксных сумм.

Пусть дан массив A: [​, ​, ​, …, ​]. Для него можно подсчитать такие суммы, что k-тая сумма равна сумме первых k элементов данного массива, то есть с до . Такие префиксные суммы имеют вид = + *.* Их удобно хранить в массиве S: [​, ​, ​, …, ], причём его длина всегда будет на единицу больше, чем длина первоначального массива A, потому что нулевой элемент обязательно принимается равным нулю (= 0).

Для простоты понимания можно представить, что элементы массива находятся в ячейках, а префиксные суммы находятся между ними — на перегородках, и содержат в себе суммы всего того, что находится левее этой перегородки. Именно поэтому для нулевого элемента сумма ноль – левее неё ничего нет.

Для предварительной обработки массива входных данных можно применять не только операцию суммирования. Среди операций, подходящий для использования в данном алгоритме необходимым условием является только тот факт, что функция, которая считается на отрезке, обладает свойством обратимости, то есть для неё реализуема возможность по двум префиксам восстановить значение на отрезке. Потому простейшим примером и является суммирование, ведь операция суммы обратима, потому что, если было прибавлено лишнее, потом это можно отнять обратно. К таким операциям можно отнести сложение по модулю (обратное ему – вычитание, также как и у простых сумм), а ещё умножение и умножение по модулю, для которых обратной операцией является деление.

Помимо вышеперечисленных, одна из часто употребляемых в алгоритмах операций – это операция «побитового исключающего или», которая еще называется XOR и обозначается ⊕. При этом для неё выполнено тождество x ⊕ x = 0 для любого числа x, что означает, что операция XOR обратна сама себе.

Существуют и другие варианты обработки массивов входных данных, помимо образования префиксных сумм, например, иногда удобнее насчитывать суффиксные суммы, которые представляю из себя суммы концов, а не начал массива, а также можно работать с массивами разниц – префиксными разностями, и разностными массивами, которые нетрудно получить из массива префиксных сумм [3].

Рассмотрим подробнее понятие разностного массива. Работа с ним представляет собой представленный ранее алгоритм, только развёрнутый наоборот. То есть по массиву префиксных сумм S: [​, ​, ​, …, ] можно восстановить исходный A: [​, ​, ​, …, ​], который при этом будет являться *разностным массивом* для S и определяться следующим образом:

=

Таким образом получается, что формула= — это просто преобразованная рекуррентная формула для поиска префиксных сумм: . Проводя аналогию с математикой, можно сделать вывод о том, что переход к разностному массиву — это дискретное дифференцирование, а переход к массиву префиксных сумм — дискретное интегрирование.

Стоит различать разностные массивы и массивы разниц (или массивы разностей), ведь последние применяют несколько иначе.

**Пример 3.** Массив разниц.

Пусть дан массив входных данных A: [​, ​, ​, …, ​]. Массив разниц D: [*​*,,, …,] образуется на его основе путем вычитания из текущего значения предыдущего, причём первое значение такого массива , поскольку подразумевает разницу между «нулевым» и первым значением, а последнее значение

=

Важным для практического применения свойством алгоритма является возможность применения его не только на одномерных массивах данных, но и на многомерных.

**Пример 4.** Создание многомерного массива префиксных сумм.

Пусть имеется двумерный массив A. Тогда массив его префиксных сумм B можно найти, подсчитав сперва префиксные суммы на одномерных массивах, то есть провести сложение сперва по горизонталям, а после – сложив полученные суммы по вертикалям.

# 3 Возможности применения алгоритма

Префиксные суммы удобно брать за основу алгоритма в таких прикладных задачах программирования, когда, например, нужно найти сумму на полуинтервале или отрезке с позиции *l* до позиции *r*. У префиксных сумм есть замечательное свойство:

Таким образом, в префиксной сумме содержатся все интересующие нас элементы полуинтервала от до, однако вместе с ними в такую сумму неизбежно попадают и «лишние», оставшиеся при расчётах от начала массива – это элементы​, , …, ​. Сумма эти элементов в свою очередь равна уже посчитанной префиксной сумме ​. ​Тогда для ответа на вопрос задачи о поиске суммы на произвольном полуинтервале нужно просто вычесть друг из друга эти две предподсчитанные префиксные суммы (.

Использовать полуинтервалы предпочтительнее, потому что, исключив один из концов отрезка, формула для подсчёта суммы на полуинтервале получается проще, короче и красивее, например, сумма на отрезке [3,7] может быть определена как сумма на полуинтервале [3,8) и равна , в то время как если бы оба конца произвольного отрезка [*l, r*] были бы включены, то в формуле появились бы единицы *,* усложняющие как запись, так и понимание, к тому же случай, когда *l* = 0 пришлось бы разбирать отдельно.

Аналогично это будет работать и с многомерными массивами. Например, после подсчёта двумерного массива префиксных сумм можно будет находить суммы по произвольным полупрямоугольникам (по аналогии с полуинтервалами):

=

Другой пример постановки вопроса задачи: необходимо найти любой непустой подотрезок с нулевой суммой элементов. Поскольку суммы на отрезках — это разности префиксных сумм, то если сумма на отрезке равна нулю, это равносильно тому, что префиксные суммы его концов равны.

Таким образом, задача сводится к нахождению подотрезка нулевой суммы, то есть к задаче нахождения двух одинаковых элементов в массиве S префиксных сумм. Использовать здесь префиксные суммы удобно потому, что получается, что суммы на отрезках — это разности префиксных сумм, если массив в ходе запросов не меняется. Потому что в обратном случае, если какой-либо элемент из числа входных данных поменялся, нужно пересчитывать заново все префиксные суммы, в которые он входит, а это достаточно долго.

Массивы разниц рационально использовать для выполнения задач, где имеются запросы на обновление некоторого диапазона матрицы на значение x, то есть на добавление x к определенной подматрице. Изменив граничные значения диапазона на величину x в массиве разниц и пересчитав на его основе массив исходных данных заново, получим искомый обновлённый массив.

Среди прочих возможностей алгоритма стоит упомянуть, что разностные массивы используются в таких приложениях, как тестирование программного обеспечения, проверка подлинности и сжатие данных. [2]

После обработки