Корневая декомпозиция

Гафаров Т.Э. Б01-411

Мотивация и постановка задачи

Пусть у нас есть большой массив, и требуется быстро выполнять запросы на диапазонах: сумму, минимум, максимум, медиану, НОД и другие.

Простой перебор за O(n) может быть слишком медленным при большом числе запросов.

Корневая декомпозиция делит массив на блоки и позволяет ускорить такие операции до O(√n) на каждый запрос при этом она достаточна проста в реализации.

Математическая формулировка задачи

Пусть дано: массив целых чисел $a = [a_0, a_1, ..., a_{n-1}]$ и последовательность запросов к диапазонам $[l, r]$.

Нам нужно поддерживать, например, следующие операции:

Сумма на отрезкея: $a[l] + \ldots + a[r]$.

Минимум или максимум на отрезке: $a[l], \ldots a[r]$

Медиана на отрезке `a[l], ... a[r]`

А также любые операции, для которых:

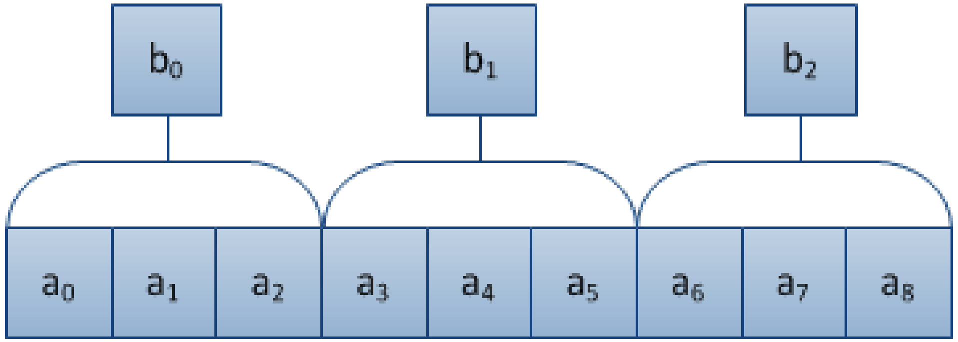
-результат можно заранее вычислить на полном блоке

-а остатки на границах легко пересчитываются вручную.

Наша цель сделать обработку каждого запроса за $O(\sqrt{n})$ после подготовки структуры за $O(n)$

Алгоритм применения корневой декомпозиции

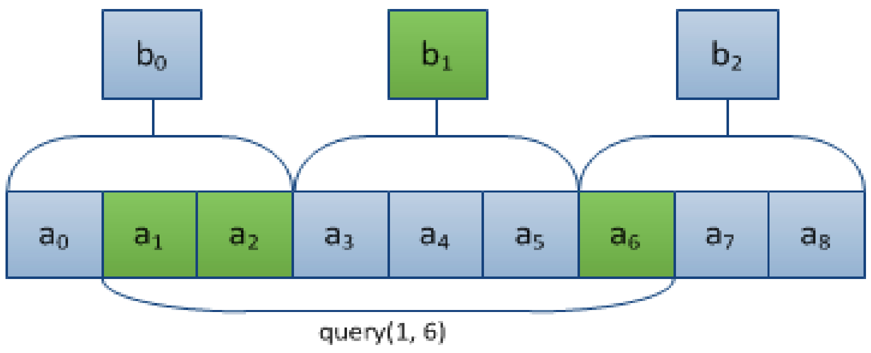
1) Разбиение массива на блоки. Разбиваем исходный массив длины n на примерно √n равных по длине блоков.
2) Предварительная обработка блоков. Для каждого блока заранее сохраняем агрегированные данные: сумма, минимум, максимум, отсортированная копия, частоты и т.п.



Алгоритм применения корневой декомпозиции

3)Обработка запроса на отрезке [l, r]

- элементы внутри целых блоков обрабатываются быстро
- элементы на краях считаются вручную



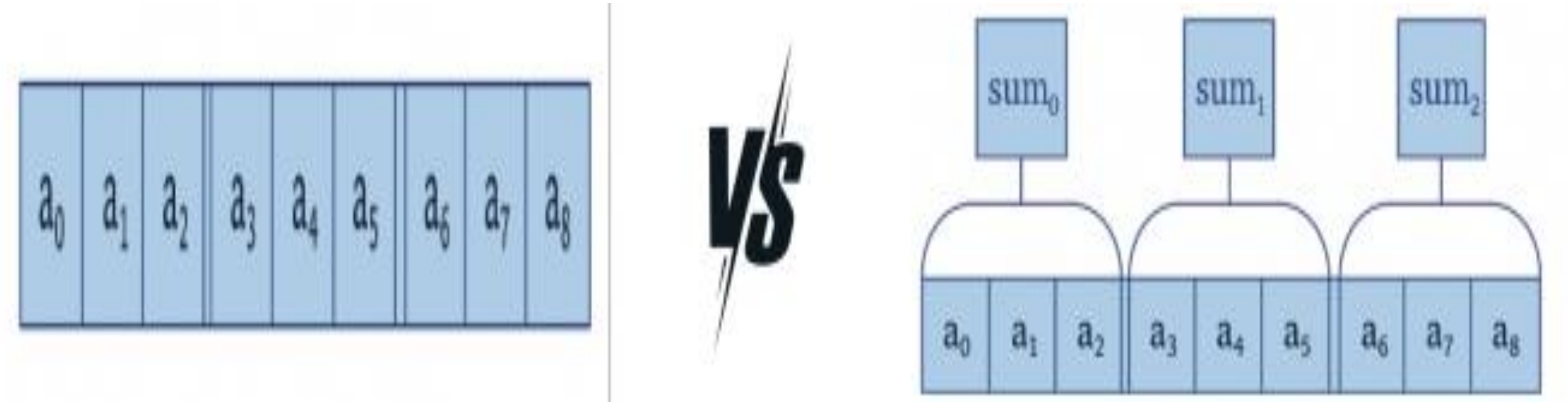
Оценка сложности операций

Операция	Лучшая	Средняя	Худшая
Сумма на отрезке	$O(1)$	$O(\sqrt{n})$	$O(\sqrt{n})$
Минимум или максимум на отрезке	$O(1)$	$O(\sqrt{n})$	$O(\sqrt{n})$
Обновление одного элемента	$O(1)$	$O(1)$	$O(\sqrt{n})$
Прибавление на отрезке	$O(1)$	$O(\sqrt{n})$	$O(\sqrt{n})$

Сравнение подходов на задаче: сумма на отрезке

Наивный подход - простой перебор всех элементов на отрезке (время работы O(n) на каждый запрос)

Корневая декомпозиция — массив разбит на блоки длиной √n с предрасчётом суммы каждого блока (время O(√n) на каждый запрос).



Сравнение подходов на задаче: сумма на отрезке

Кол-во элементов `n`	Кол-во запросов `q`	Наивный, сек	Корневая декомпозиция, сек
10 000	10 000	0.13	0.02
100 000	50 000	5.83	0.11
1 000 000	100 000	116.73	0.44
10 000 000	100 000	1168	1.381

А можно ли еще быстрее ?

Хотя корневая декомпозиция ускоряет наивный перебор до O(√n) существуют и более быстрые структуры данных, например, дерево отрезков , которое позволяет отвечать на теже самые запросы не за O(√n) , а за O(log n)

Структура	Время запроса	Cache-friendly
Перебор	O(n)	Да
Sqrt Decomposition	O(√n)	Да
Дерево отрезков	O(log n)	Нет

Особенности реализации на реальных вычислителях

- Простая реализация (легко реализовать без дополнительных сложных структур и алгоритмов)

- Cache-friendly(данные обрабатываются последовательно по блокам расположенным рядом в памяти)

- Ограничения по типу операций

Полезные ссылки

- neerc.ifmo.ru: Корневая эвристика
- e-maxx.ru: Корневая декомпозиция
- Habr: Статистики на отрезках
- algorithmica.org: Sqrt Decomposition
- Подробное описание алгоритма, с примерами реализации и задачами.
- Собственная реализация и тесты для всего этого