Разбор задачи 3. Река

Региональный этап Всероссийской олимпиады школьников по информатике
Республика Татарстан. Казань

24 января 2015

Условие задачи

- Задан целочисленный массив $a[1], a[2] \dots a[n]$
- Требуется поддерживать два вида запросов:
  - Разделить равномерно $a[k]$ на два элемента и вставить их на место k и $k+1$, после запроса n увеличится на единицу
  - Удалить $a[k]$ и равномерно прибавить его значение соседям в массиве, после запроса n уменьшится на единицу. Если соседей два, то элемент делится так же, как и в первом запросе
- Нечетный элемент $a[k] = 2x + 1$ разделится на два: $a[k] = x$ и $a[k+1] = x + 1$
- После каждого запроса требовалось вывести $\sum_{i=1}^{n} a[i]^2$

Общая идея решения

- Чтобы промоделировать запросы достаточно поддерживать следующие операции:
  1. Получить и изменить элемент по номеру k
  2. Вставить элемент в позицию k
  3. Удалить элемент с номером k
- Подзадачи будут отличаться структурой данных, которая используется для этих операций
- Кроме этой структуры данных будем поддерживать переменную $answer$, которая будет пересчитываться после каждого изменения в структуре данных

Подзадача 1

- Здесь изначально не больше чем 100 элементов и не больше чем 100 запросов
- Можно все сохранить в массив
- Для вставки или удаления элемента, нужно все следующие элементы сдвинуть на один вправо или на один влево, соответственно. Время работы $O(n)$
- Нахождение $a[k]$ в массиве происходит за $O(1)$
- Итоговое время работы: $O(nm)$

Подзадача 2

- Здесь элементов и запросов не больше 10^5, но индексы, в которых происходят соседние запросы расположены близко: $|v_i - v_{i+1}| \leqslant 10$
- Здесь нельзя просто реализовать на массиве, потому что вставка и удаление не позволительна за линейное время
- Реализуем двусвязный список: добавление и удаление по элементу происходит за $O(1)$
- Если поддерживать текущий элемент, с которым произошел запрос, и его индекс, то можно найти следующий за $O(1)$
- Итоговое время работы: $O(n + m)$

Подзадача 3

- Здесь элементов и запросов не больше 10^5, но не происходит запросов разделения
- В этой подзадаче не нужно поддерживать вставку элемента, только удаление
- Поэтому можно реализовать структуру данных дерево отрезков
- Итоговое время работы: $O(m \log n)$

Подзадача 4

- Здесь отрезков и запросов не больше 10^5
- В отличие от предыдущей подзадачи, дерево отрезков не умеет вставлять элемент между существующими элементами
- Нужно реализовать более сильную структуру данных: двоичное дерево поиска или корневую оптимизацию
- Итоговое время работы: $O(m\sqrt{n})$ или $O(m \log n)$

Корневая оптимизация

- Разделим все элементы на блоки размера B
- Каждый блок храним в виде массива длины $2B$
- Как найти и изменить k-й элемент: идем слева направо по блокам, находим блок, в котором элемент и где он в этом блоке. Время работы $O(\frac{n}{B})$
- Как вставить или удалить элемент: находим место, куда вставить или удалить, и вставляем или удаляем из массива за $O(B)$. Время работы $O(\frac{n}{B} + B)$
- Есть проблема: в блоке может стать больше $2B$ элементов
- Решаем так: после каждых B запросов можно вывести все блоки в массив и разделить на блоки заново
- Итоговое время работы: $O(m(\frac{n}{B} + B) + \frac{m}{B}n)$, минимум достигается при $B = \sqrt{n+m}$