Алгоритмы и Структуры Данных. Лекция 2

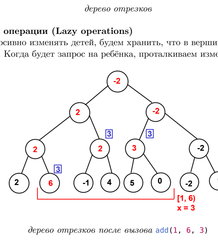
14.02.2024

__scarleteagle **ikochelorov**

Дерево отрезков. Отложенные операции

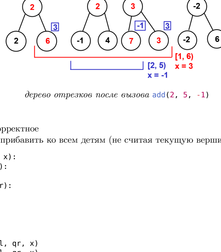
Операции:
- `get(l, r)` `#min(a_l, a_l+1, ..., a_r+1)`
- `add(l, r, x)` `#O(log n)`
  ```
  #a_l += x
  #a_l+1 += 1
  #...
  #a_r-1 += x
  ```

Как выполнить операцию над отрезком за $O(\log n)$, если отрезок в худшем случае имеет длину $O(n)$

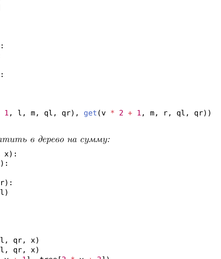


дерево отрезков

Ленивые/отложенные операции (Lazy operations)
Вместо того, чтобы рекурсивно изменять детей, будем хранить, что в вершине результат правильный, а в детях нужно добавить N. Когда будет запрос на ребёнка, проталкиваем изменения в обоих детей



дерево отрезков после вызова `add(1, 6, 3)`



дерево отрезков после вызова `add(2, 5, -1)`

Инвариант дерева:
1. На выходе значение корректное
2. `pr[v]` хранит, сколько прибавить ко всем детям (не считая текущую вершину)

```
def add(v, l, r, ql, qr, x):
  if (qr <= l or r <= ql):
    return
  if (ql <= l and r <= qr):
    tree[v] += x
    pr[v] += x
    return
  m = (l + r) // 2
  push(v)
  add(2 * v + 1, l, m, ql, qr, x)
  add(2 * v + 2, m, r, ql, qr, x)
  tree[v] = min(tree[2 * v + 1], tree[2 * v + 2])

def push(v):
  if (pr[v] == 0):
    return
  tree[2 * v + 1] += pr[v]
  tree[2 * v + 2] += pr[v]
  pr[2 * v + 1] += pr[v]
  pr[2 * v + 2] += pr[v]
  pr[v] = 0

MAX_INT = 1e12
def get(v, l, r, ql, qr):
  if ql >= r or qr <= l:
    return MAX_INT
  if ql <= l and r <= qr:
    return tree[x]
  push(v)
  m = (l + r) // 2
  return min(get(v * 2 + 1, l, m, ql, qr), get(v * 2 + 1, m, r, ql, qr))
```

Наше дерево легко превратить в дерево на сумму:

```
def add(v, l, r, ql, qr, x):
  if (qr <= l or r <= ql):
    return
  if (ql <= l and r <= qr):
    tree[v] += x * (r - l)
    pr[v] += x
    return
  m = (l + r) // 2
  push(v)
  add(2 * v + 1, l, m, ql, qr, x)
  add(2 * v + 2, m, r, ql, qr, x)
  tree[v] = sum(tree[2 * v + 1], tree[2 * v + 2])

def push(v):
  if (add[v] == 0):
    return
  tree[2 * v + 1] += pr[v] * (m - l)
  tree[2 * v + 2] += pr[v] * (r - m)
  pr[2 * v + 1] += pr[v]
  pr[2 * v + 2] += pr[v]
  pr[v] = 0

MAX_INT = 1e12
def get(v, l, r, ql, qr):
  if ql >= r or qr <= l:
    return MAX_INT
  if ql <= l and r <= qr:
    return tree[x]
  push(v)
  m = (l + r) // 2
  return sum(get(v * 2 + 1, l, m, ql, qr), get(v * 2 + 1, m, r, ql, qr))
```

Суть отложенных операций в том, чтобы выполнять их не когда они пришли, а когда возникла необходимость

> Зачем готовиться к экзамену по матану с сентября, если можно начать готовиться за 2 дня до него
>
> — Первеев Михаил Валерьевич

Попробуем сделать более тяжёлую операцию. Например, прибавление арифметической прогрессии

```
add(l, r, x, d)
  #a_l += x
  #a_l+1 += x+d
  #a_l+2 += x+2d
  #...
  #a_r-1 += x+d*(...)
```

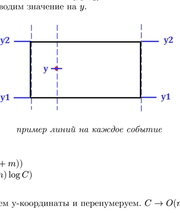
Сумма арифметических прогрессий — арифметическая прогрессия

$x_1 + x_2, d_1 + d_2 \Leftrightarrow \begin{cases} x_1 & d_1 \\ x_2 & d_2 \end{cases}$

отрезок добавления арифметической прогрессии

Дерево отрезков на сумму с добавлением арифметической прогрессии возможно. На минимум — нет.

Задача:
Даны прямоугольники и точки. Найти скольким прямоугольниками покрывается точка



пример прямоугольников и точек

Точки: (x, y)
Прямоугольники: $(x_1, y_1), (x_2, y_2)$
$0 \le x_i, y_i \le c$
Сканирующая прямая:

для каждой координаты линии по y вычисляем скольками прямоугольниками она покрывается

События: границы прямоугольника, точки.
Если дано n прямоугольников, m точек, то событий — 2n+m.
Сортируем события по x-координате, затем в порядке "начало, точка, конец"

Проходим по событиям
- Если встречаем левую границу, то на отрезке $[y_1, y_2]$ добавляем 1.
- Если встречаем левую границу, то на отрезке $[y_1, y_2]$ вычитаем 1.
- Если встречаем точку, то выводим значение на y.

пример линий на каждое событие

Время работы:
- Сортировка: $O((n+m)\log(n+m))$
- Обработка событий: $O((n+m)\log C)$

Память: $O(C)$
Сжатие координат: отсортируем y-координаты и перенумеруем. $C \to O(n+m)$