ДЗ2

Наумов Иван M34391

September 2022

Task1

$$\sum_{i=0}^{\log n} \log \frac{n}{2^i} = \sum_{i=0}^{\log n} \log n - \sum_{i=0}^{\log n} \log 2^i = \log^2 n - \log 2 \sum_{i=0}^{\log n} i = \log^2 n - \frac{\log n \cdot (\log n + 1)}{2} = \frac{log^2 n - \log n}{2}$$

$$C_1 \cdot \log^2 n \leq \frac{log^2 n - \log n}{2} \leq C_2 \cdot \log^2 n$$

Положим $C_1 = \frac{1}{4}$, $C_2 = 1$

$$\frac{\log^2 n}{4} \leq \frac{log^2 n - \log n}{2} \leq \log^2 n$$

$$\frac{\log^2 n}{2} \leq log^2 n - \log n \leq 2 \cdot \log^2 n$$

$$\frac{\log^2 n}{2} + \log n \leq log^2 n \leq 2 \cdot \log^2 n + \log n$$

Положим

$$\log n \leq \frac{\log^2 n}{2}, n \geq 1$$

$$\log n \geq 2$$

$$n \geq 4$$

Таким образом, левое неравенство выполняется при $n \geq 4$, правое - $n \geq 1$

$$\exists C_1 = \frac{1}{4}, C_2 = 1, N_0 = 4 \Longrightarrow \sum_{i=0}^{\log n} \log \frac{n}{2^i} = \Theta(\log^2 n)$$

Task2

```
function scan1(v, l, r, tree, a, f):
    if l + 1 == r:
        tree[v] = a[l]
        return
    m = (l + r) // 2
    fork2join({
        scan1(2 * v + 1, l, m, tree, a),
        scan1(2 * v + 2, m, r, tree, a)
    })
    tree[v] = f(tree[2 * v + 1], tree[2 * v + 2])
```

```
function scan2(v, l, r, tree, a, f, acc):
    if l + 1 == r:
        a[l] = f(acc, a[l])
        return
    m = (l + r) // 2
    fork2join({
        scan2(2 * v + 1, l, m, tree, a, f, acc),
        scan2(2 * v + 1, l, m, tree, a, f, f(acc, tree[2 * v + 1]))
    })

function scan(a, f):
    n = len(a)
    tree = [0] * (4 * n)
    scan1(0, 0, n, tree, a, f)
    scan2(0, 0, n, tree, a, f, id(f))
    return a
```

Task3

Будем строить решето Эратосфена *prime* блоками длины 2^i. На первом шаге алгоритма обозначим число 2 за простое, а все четные числа - за составные. Это значит, что в массиве до числа 2 * 2 $=$ 4 нет составных чисел, которые бы имели флаг в *prime* как простое. Значит, можно обойти отрезок [3,4] с помощью pfor, он найдет все простые числа верно. Обобщая для построенного префикса длины n: отрезок $[n{+}1, n^2]$ содержит верно выставленные флаги. Таким образом получается следующий алгоритм

```
// N is len(prime)
function sieve(n, prime):
    if n >= N:
        return
    pfor(n + 1, min(n * n, N),
    {
        if prime[i] == 1:
            pfor(i * i, N, { prime[j] = 0 })
    })
    sieve(n * n, prime)

function primes(n):
    prime = [1] * n
    pfor(0, n // 2, { prime[2 * i] = 0 })
    prime[1] = 0
    prime[2] = 1
    sieve(2, prime)
    return filter(prime, id)
```

Начиная с длины отрезка 2, показатель его степени будет удваиваться с каждым шагом рекурсии => глубина рекурсии $O(\log\log n)$. Внешний pfor порождает span глубины $O(\log(\texttt{offcut length})) = O(\log n)$, каждый из которых может, в свою очередь породить обход массива, внутренний pfor тоже порождает span $\log n$. Суммарный span: $O(\log\log n) \cdot 2 \cdot O(\log n) = O(\log n \cdot \log\log n) = O(polylog(n))$ Поскольку по индукции отрезок, переданный в функцию sieve содержит корректные флаги, внутренний pfor будет производить обход только при нахождении простыми тех чисел, которые в действительности простые. Это значит, что суммарный work такой же, как у решета Эратосфена - O($n \log\log n$).
Work: O($n \log\log n$)
Span: O($\log n \cdot \log\log n$)