Алгоритмы и структуры данных
Домашняя работа 4

Мирзоев Денис

1

(a) Сортируем выбором. Для i-ой позиции будем искать патрон в следующих k позициях.

(b) Сортируем вставками. Каждый патрон будет погружаться в отсортированную часть массива на такую глубину, сколько инверсий он образует.

(c) Оценим число перестановок, которые могли возникнуть. На каждую позицию в неотсортированном массиве претендуют не менее k элементов. Число перестановок, которые могли возникнуть равно $(k+1)^{n-k}k!$. Для того, чтобы различить $(k+1)^{n-k}k!$ перестановок алгоритму нужно будет выполнить сравнений минимум $\log((k+1)^{n-k}k!) \geq \log(k^{n-k}k!) \geq \log(k^{n-k}\sqrt{2\pi k}(\frac{k}{e})^k e^{\frac{1}{12k+1}}) \geq \log(k^{n-k}k^k) = \log(k^n) = n\log k = \Omega(n\log k)$.

(d) Сортируем слиянием. Сначала отсортируем каждые два соседних патрона, потом каждый четыре, потом каждые восемь, пока каждые $2k$ патронов не будут отсортированы. Это случится через $\lceil \log 2k \rceil + 1$ повторений. На каждом повторении мы совершаем n действий. Теперь сделаем ещё один этап сортировки для массивов длины $2k$ со сдвигом k от начала массива.

2

- **Лемма 1**. Если длина числа a в скошенной СС(ССС) равна n, то $2^n - 1 \leq a \leq 2^{n+1} - 2$.

  Доказательство:

  Наименьшее число длины n в ССС имеет вид: $10\dots0$. Его значение равно $2^n - 1$.

  Наибольшее число длины n в ССС должно иметь вид: $1\dots120\dots0$ (j единиц и k нулей так, что $j + 1 + k = n$), потому что уменьшение любого разряда в данном числе уменьшит его значение.

Но тогда $a = \sum_{i=k+2}^{n}(2^i - 1) + 2(2^{k+1} - 1) = 2^{n+1} - 1 - (n-k) \le 2^{n+1} - 2$. Из последней формулы видно, что число вида $20\dots0 (k = n-1)$ максимально.

Теорема о существовании Для неотрицательного числа a существует его представление в ССС.

Доказательство:

Докажем индукцией по n, что если $a \le 2^{n+1} - 2$, то представление существует. Так как для любого a существует n, для которого это выполняется, то это докажет теорему.

При $n = 1$: $a \le 2$, $a \in 0, 1, 2$.

Предположим утверждение индукции верно при $n = k$ и $a \le 2^{k+2} - 2$. Если $a \le 2^{k+1} - 2$, то представление существует по предположению индукции. Если $a = 2^{k+2} - 2$, то представление будет $20\dots0$. Остался случай, когда $2^{k+1} - 1 \le A \le 2^{k+2} - 3$. Рассмотрим $b = a - 2^{k+1} + 1 \le 2^{k+1} - 2$. Для b существует представление по предположению индукции. По лемме 1 "длина b" $= L$ не превосходит k. Тогда рассмотрим число в ССС вида $10\dots0b$, где ноль повторяется $k - L$ раз. Его значение равно $2^{k+1} - 1 + b$, что равно a.

Теорема о единственности Для неотрицательного числа a существует единственное представление в ССС.

Доказательство:

Достаточно показать, что если $a \ne b$, то их представления неодинаковы.

Если $a \ne b$, тогда либо $a \ge b + 1$, либо существует ρ такое что $a = \rho\alpha\tau$ и $b = \rho\beta\pi$ и значение $\tau =$ значение π и значение $\alpha \ge$ значение $\beta + 1$.

Если $a \ge b + 1$, то по лемме 1, $a \ge 2^{\text{длина a}} - 1 \ge 2^{\text{длина b+1}} - 1 > 2^{\text{длина b+1}} - 2 \ge b$

В противном случае предположим $n =$ значение $\tau(=$ значение π, тогда по лемме 1,

$a0\dots0(n\text{нулей})$
$\ge (b+1)0\dots0(n\text{нулей}) = b0\dots0(n\text{нулей}) + (2^{n+1} - 1)$
$> b0\dots0(n\text{нулей}) + \pi = b\pi$

Тогда

$a = \rho\alpha\tau \ge \rho0\dots0(n+1\text{раз}) + a0\dots0(n\text{раз})$
$> \rho0\dots0(n+1\text{раз}) + \beta\pi = \rho\beta\pi = b.$

Доказательство из `https://publications.mpi-cbg.de/Myers_1983_6328.pdf`

- Если число 0, то заменим 0 на 1.

  Предположим мы знаем, где находится последний ненулевой разряд. Если это 1, установим её в 2. Это можно сделать, потому что двойка может стоять только на месте последнего ненулевого разряда и это как раз оно. Если это 2, то установим её в 0, а следующий разряд увеличим на 1: если он 0, то установим в 1, если 1 то установим в 2. Двойки там быть не может, потому что мы уже встретили её в следующем разряде, а в числе она может быть только одна.

Чтобы знать где находится последний ненулевой разряд можно хранить число в виде списка пар ("разряд", "позиция") отсортированного по увеличению позиции, где "разряд" не равен нулю. Тогда последний ненулевой разряд будет первым элементом этого списка.

3

(a) После добавленя элемента к списку в нём будет $n + 1$ элементов. Это число отличается от n максимум в дух последних ненулевых разрядах. в скошенной системе счисления. Значит нам нужно будет модифицировать максимум два последних дерева.

Если последний ненулевой разряд был 2, то значит все последующие разряды были 0. После добавления элемента 2 станет 1, а следующий разряд увеличится на 1. В списке в этом случае в начале находятся два полных двоичных дерева высоты i, где i - номер разряда двойки. После добавления элемента два дерева высоты i должны будут исчезнуть, и вместо них должно появиться одно дерево высоты (i+1). Получим его, из двух имеющихся деревьев высоты i, поставив в корень новый элемент.

Если двоек не было и последний разряд был равен 0 или 1, то после добавления элемента он станет 1 или 2 соответственно. В этом случае добавим одноэлементное дерево в начало списка.

(b) По представлению n в скошенной СС линейным поиском найдём дерево, в котором находится элемент с индексом i. Представление нужно для вычисления количества элементов в обозреваемом дереве. Когда мы его найдём мы будем знать элементы с какими индексами в него входят, пусть это будут индексы от l до r. В правом поддереве у него будут индексы от l до $(l + r - 1)/2$, в левом от $(l + r - 1)/2$ до r. Рекурсивно спускаясь в этом дереве найдём элемент с нужным индексом.

(c) Деревья элементы которых все имеют индекс больше k перенесём без изменений в конец нового списка. Дерево элемента с индексом k придётся распилить, то есть разбить на список деревьев, которые мы добавим в начало списка-резуьтата. Легко показать, что в результате данного процесса у нас получится список из полных бинарных деревьев неубывающей высоты такой, что два дерева одной высоты могут находиться только в начале. Понятно, что такой список является скошенным. Рассмотрим процесс распила:

```
# Распил дерева {x, A, B}, где x - корень, A, B - поддеревья
# l, r - индексы, находящиеся в текущем дереве
Saw({x, A, B}, l, r):
    m = (l+r-1)/2
    if k == m:
        return k, B
    else if k in [m, r]: # если k в правом поддереве
        return Saw(B, m, r)
    else # если k в левом
        return Saw(A, l, m - 1), B
```

4

```
Fix({x, {y, A, B, r1}, {z, C, D, r2}, r3})
    if r1 > f2: # меняем местами левое и правое поддерево
        return {x, {z, C, D, r2}, {y, A, B, r1}, r2}
    else        # оставляем как есть
        return {x, {y, A, B, r1}, {z, C, D, r2}, r2}

# x, y - корни
# A, C - левые поддеревья корней
# B, D - правые поддеревья корней
# r1, r2 - ранги
Merge({x, A, B, r1}, {y, C, D, r2})
    if x < y:
        return Fix({x, A, Merge(B, {y, C, D, r2}), r1)
    else:
        return Fix({y, C, Merge(D, {x, A, b, r1}), r2)
```

- Делаем merge с одноэлементной кучей [x].
- Пусть A и B - левое и правое поддерево корня, тогда вернём merge от A и B.
- Clone будет возвращать указатель на саму кучу. Все операции не изменяют кучу, а создают новую, поэтому саму кучу можно вернуть в качестве копии.

5

Рассмотрим отсотрированный по возрастанию массив. Будем изменять его. В начале будем хранить макс. кучу. Дальше будет отсортированная часть. Для всех элементов от второго до последнего вызовем siftUp. В результате получим искомую перестановку.

Мы выполним $\mathcal{O}(n \log n)$ действий. Сортировка устроена таким образом, что она будет выполнять те же действия в обратном порядке, поэтому её сложность тоже $\mathcal{O}(n \log n)$.