2 задача: Дерево отрезков (2)

4 модуль, 2 семестр

ФИВТ МФТИ, 2019

Описание by Илья Белов

1. Текст задачи:

Последовательность единиц

Дан массив из нулей и единиц $a_0, a_1, ..., a_{n-1}$. Для каждого запроса *[left, right]* найдите такой подотрезок $a_l, a_{l+1}, ..., a_r$ *этого* массива (*0 <= left <= l <= r <= right < n*), что числа $a_l, a_{l+1}, \ldots, a_r$ являются максимально возможной последовательностью единиц.

Требуемое время ответа на запрос - O(log n).

Формат входных данных

Описание каждого теста начинается с двух чисел *n* и *m* - длины массива и числа интересующих подотрезков.

В следующей строке содержится *n* нулей и единиц.

Далее следуют описания подотрезков, каждое описание состоит из двух чисел *left* и *right*, обозначающих левый и правый конец подотрезка (0 <= *left* <= *right* < *n).*

Формат выходных данных

Для каждого примера выведите *m* чисел: искомую максимальную длину последовательности единиц для каждого из подотрезков.

Пример

in	out
10 4 0 1 0 1 1 1 1 0 1 1 2 2 1 5 0 9 9 9	0 3 4 1

2. Описание алгоритма

Для каждого отрезка строки a в дереве будем хранить 4 параметра:

1) Длину отрезка len

2) Длину префикса единиц $pref$

3) Длину суффикса единиц suf

4) Максимальную длину последовательности единиц на отрезке seq

Для каждой вершины при построении будем вычислять эти параметры рекурсивно следующий образом:

Самый нижний слой:

$len[i] = 1,\ pref = suf\ =\ seq\ =\ 1\ if\ a[i] == 1\ else\ 0$

Остальные слои:

Пусть $left$ и $right$ - левый и правый ребёнок вершины i соответственно

$len[i]\ = len[left] + len[right]$

$pref[i]\ = len[left] + pref[right]\ if\ len[left] == pref[left]\ else\ pref[left]$

$suf[i]\ = suf[left] + len[right]\ if\ len[right] == suf[right]\ else\ suf[right]$

$seq[i] = max(seq[left],\ seq[right],\ suf[left] + pref[right])$

При запросе вершины будут возвращать $seq[i]$ если отрезок в запросе совпадает с отрезком в вершине и $max(seq[left],\ seq[right])$ иначе

3. Доказательство корректности

Доказательство корректности работы дерева отрезков смотри на https://e-maxx.ru/algo/segment_tree

4. Время работы и дополнительная память

$T = O(n + mlogn)$

$M = O(n)$

5. Доказательство времени работы и дополнительной памяти

Докажем доп память.

Мы должны хранить каждый слой бинарного дерева, а это в общей сложности $\frac{n}{2} + \frac{n}{4} + ... + 2 + 1 = n$ вершин (в случае $n = 2^k$, иначе асимптотика не изменится так как в худшем случае кол-во вершин будет $n + logn$). Получаем $M = O(n)$

Докажем время работы.

1) Построение

Как указано выше, вершин в дереве будет $O(n)$. При построении дерева значение каждой вершины вычисляется один раз за $O(1)$. Следовательно, время построения всего дерева будет $O(n)$

2) Запросы

На каждой высоте дерева рекурсией затрагивается не более чем 4 вершины, высота дерева $O(logn)$, следовательно один запрос работает на $O(logn)$

Итоговое время работы алгоритма с m запросами получаем $T = O(n + mlogn)$