Домашнее задание на первую неделю

Бельденова Камила, 675

13 февраля 2018

1	2	3	4	5	6	Σ

Задача 1. Дан массив из n, на которых определено отношение равенства (например, речь может идти о массиве картинок или музыкальных записей). Постройте алгоритм, который в "потоковом режиме обработки данных"определяет, есть ли в массиве элемент, повторяющийся больше $\frac{n}{2}$ раз. Считается, что в вашем распоряжении есть память объемом $O(log\ n)$ битов.

Решение

Воспользуемся следующим фактом: последовательность чисел $\{a_1,a_2,\ldots,a_k\}$ можно закодировать одним числом: $\boxed{X = 2^{a_1}3^{a_2}5^{a_3}...}$.
Причем такое кодирование однозначно и обладает таким свойством, что по числу X можно однозначно восстановить исходную последовательность закодированных чисел.

Теперь используя этот факт, построим алгоритм:
1) Закодируем весь наш массив чисел в одно число $\boxed{X = 2^{a_1}3^{a_2}5^{a_3}...}$
2) Раскодируем первые 5 элементов. Найдем среди них медиану. Обозначим найденное число за b_1, формируя новое число $Y = 2^{b_1}$. Будем продолжать то же самое и для последующих пятерок из оставшихся чисел массива, продолжая вычислять $Y = Yp_k^{b_k}$, где p_k - k-ое простое число. Кроме медианы будем искать min и max среди всех чисел простым сравнением, изменяя соответсвенно значения min и max при находжении числа, меньшего, чем min, и большего, чем max.
3) В итоге в Y будет закодирован массив медиан. Будем повторять (2), пока не получим одно число - медиану медиан исходного массива. Затем распределим числа в "мешки"относительно этого числа и запустим алгоритм для "большего мешка".
4) После нахождения middle, max, min сравним эти числа, если middle совпадает с одним из чисел, то в массиве есть число, которое повторяется $n/2$ раз, иначе – такого числа нет.

Требуемая быстрая память для алгоритма: 5 элементов для нахождения медианы среди 5-рок чисел, 3 элемента для хранения чисел X, min, max. Т. е. константное значение для любого массива длины n.

Задача 2. Дано описание программы.

- ВХОД(x,y — натуральные числа).
- Пусть 2^{d_x} — максимальная степень 2, делящая x; d_y определяется аналогично.
- Положим $a = x2^{-d_x}$, $b = y2^{-d_y}$.
- Поменять местами b и a, если $b < a$.
- ПОКА $b > 1$ ВЫПОЛНИТЬ

- Положим r равным тому из чисел $a+b$, $a-b$, которое делится на 4.
- Положим $a = \max(b, r2^{-d_r})$, $b = \min(b, r2^{-d_r})$, где 2^{d_r}– максимальная степень 2, делящая r.
- КОНЕЦ ЦИКЛА ПОКА
- ВЫХОД($a2^{\min(d_x, d_y)}$).

Программа, **если в ней исправить неточности**, вычисляет известную функцию. Какую? Оцените трудоемкость (скорректированной) процедуры (число битовых операций), если x, y — n-битовые числа.

Решение

Если изменить в алгоритме 2 условия:
1) $b > a$ на $b < a$ в 4 строке;
2) $b > 1$ на $b \geq 1$ в 5 строке,

то получается что-то похожее на бинарный алгоритм Эвклида. Возьмем описание этого алгоритма с 372-й страницы книги Кнута:

1) Присвоить $k \leftarrow 0$, затем повторни присваивать $k \leftarrow k+1, x \leftarrow x/2, y \leftarrow y/2$ нуль или более раз до тех пор, пока оба числа x и y станут нечетными.
2) (Исходные значение чисел x, y уже разделены на 2^k и по крайней мере одно из текущих значений нечетно). Если нечетно x, то присвоить $t \leftarrow -x$ и перейти к шагу 4. Иначе присвоить $t \leftarrow x$.
3) (Это случай когда t нечетно) Присвоить $t \leftarrow t/2$
4) Если t четно, то вернутся к шагу 3.
5) Если $t > 0$, то присвоить $x \leftarrow t$, в противном случае $y \leftarrow -t$ (большее из чисел заменяется на $|t|$ за исключением, возможно первого выполнения этого шага).
6) Присвоить $t \leftarrow x - y$ Если $t \neq 0$, то вернуться к шагу 3. В противном случае алгоритм останавливает выполнение, а на выходе $x2^k$

Этот алгоритм аналогичен представленному в задании.

Задача 3. На вход подается описания n событий в формате (s,f) — время начала и время окончания. Требуется составить расписание для человека, который хочет принять участие в максимальном количестве событий. Например, события — это доклады на конфереции или киносеансы на фестивале, которые проходят в разных аудиториях. Предположим, что участвовать можно только с начала события и до конца. Рассмотрим три жадных алгоритма.

- Выберем событие кратчайшей длительности, добавим его в расписание, исключим из рассмотрения события, пересекающиеся с выбранным. Продолжим делать то же самое далее.
- Выберем событие, наступающее раньше всех, добавим его в расписание, исключим из рассмотрения события, пересекающиеся с выбранным. Продолжим делать то же самое далее.
- Выберем событие, завершающееся раньше всех, добавим его в расписание, исключим из рассмотрения события, пересекающиеся с выбранным. Продолжим делать то же самое далее.

Какой алгоритм вы выберете? В качестве обоснования для каждой процедуры проверьте, что она является оптимальной (т. е. гарантирует участие в максимальном числе событий) или постройте конкретный контрпример.

Решение

Первые 2 алгоритма неверны, приведем контрпример:
1) Пусть есть 3 события, заданные в формате (s_i, f_i): (0, 5); (6,10); (4,7). Правильным решением будет пойти на (0,5) и (6,10), т. е. можно пойти на 2 события.

Первый алгоритм выберет кратчайшее по длительности – (4,7) и пересекающиеся с этим уберет, поэтому человек пойдет только на одно событие, вместо двух ⇒ алгоритм неверен.
2) Пусть есть 3 события, заданные в формате (s_i, f_i): (0, 5); (1,2); (3,4). Правильным решением будет пойти на (1,2) и (3,4), т. е. можно пойти на 2 события.
Второй алгоритм выберет то событие, которое начинается раньше остальных – (0,5), т. к. событий после момента времени 5 нет, то человек пойдет на 1 событие ⇒ алгоритм неверен.
3) Сначала отсортируем список событий по возрастанию времени окончания события.
Покажем, что этот алгоритм дает оптимальное решение задачи о выборе события, содержащее событие, которое заканчивается раньше остальных (назовем его событие под номером 1). В самом деле, если в каком-то оптимальном множестве событий событие под номером 1 не содержится, то можно заменить в нем событие с самым ранним временем окончания на событие под номером 1, что не повредит совместимости событий и не изменит их общего количества. Значит, можно искать оптимальное множесто заявок А среди содержащих событие под номером 1: существует оптимальное решение, которое начинается с жадного выбора.

Исходя из написанного, можно выбросить события, несовместные с событием под номером 1, и задача сводится к выбору оптимального набора других событий из множества оставшихся событий. Т. е. задача свелась к аналогичной, только с меньшим числом событий. По индукции получаем, что, делая на каждом шаге подобный жадный выбор, алгоритм будет работать корректно.

Задача 4. Найдите явное аналитическое выражение для производящей функции чисел BR_{4n+2} правильных скобочных последовательностей длины $4n+2$ (ответ в виде суммы ряда не принимается).

Производящая функция чисел Каталана C_n : $\sum_{n=0}^{\infty} C_n z^n = \frac{1-\sqrt{1-4z}}{2z}$

Задача 5. Оцените трудоемкость рекурсивного алгоритма, разбивающего исходную задачу размера n на три задачи размером $\lceil \frac{n}{\sqrt{3}} \rceil - 5$, используя для этого $10\frac{n^3}{\log n}$ операций.

Решение

$$T(n) = 3T\left(\frac{n}{\sqrt{3}} - 5\right) + 10\frac{n^3}{log(n)}$$

$$3 > 0$$
$$0 < \frac{1}{\sqrt{3}} < 1$$
$$10\frac{n^3}{log(n)} \in O(n^3)$$
$$-5 \in O\left(\frac{n}{log^2(n)}\right)$$

Следовательно, можем воспользоваться теоремой Акра-Баззи:

$$T(n) = \Theta\left(n^p\left(1 + \int_1^n \frac{x^3}{x^{p+1}log(x)}dx\right)\right),$$

где p ищется из условия:

$$3 \cdot \frac{1}{\sqrt{3^p}} = 1 \Leftrightarrow p = 2$$

Получаем:

$$T(n) = \Theta\left(n^2\left(1 + \int_1^n \frac{x^3}{x^3log(x)}dx\right)\right)$$

Задача 6. Рассмотрим детерминированный алгоритм поиска медианы по кальке известного линейного алгоритма, где используется разбиение массива на четвёрки элементов, в каждой из которых определяется *нижняя* медиана, т. е. из в каждой четверки выбирается второй по порядку элемент (элементы можно считать различными). Приведите рекуррентную оценку числа сравнений в этой процедуре и оцените сложность такой модификации.

Решение

В нашем случае медиана медиан обладает таким свойством, что она больше, по крайней мере, 5/24 элементов, тогда трудоемкость рекурсии:

$$T(n) = T\left(\frac{n}{4}\right) + T\left(\frac{3n}{4}\right) + O(n),$$

где $T(\frac{n}{4})$ – время на рекурсивное нахождения медианы медиан (каждый раз откидываем 4-ю часть обьектов);
$T(\frac{3n}{4})$ – количество обьектов, которое в худшем случае остается для проверки после сравнений;
$O(n)$ - трудоемкость "сшивания в нашем случае это трудоемкость сравнения.

Докажем, что $\exists c : T(n) \leq c \cdot n \cdot log_2(g)$

$$\exists c : T(n) \leq \frac{n}{4} \cdot c \cdot log_2\left(\frac{n}{4}\right) + \frac{3n}{4} \cdot c \cdot log_2\left(\frac{3n}{4}\right) + n = c \cdot n \cdot log_2(n) + n\left(1 - c\left(2 + log_2\left(\frac{4}{3}\right)\right)\right) \leq c \cdot n \cdot log_2(n)$$

Докажем, что $\exists c : T(n) \geq c \cdot n \cdot log_2(g)$

$$\exists c : T(n) \geq \frac{n}{4} \cdot c \cdot log_2\left(\frac{n}{4}\right) + \frac{3n}{4} \cdot c \cdot log_2\left(\frac{3n}{4}\right) + n = c \cdot n \cdot log_2(n) + n\left(1 - c\left(2 + log_2\left(\frac{4}{3}\right)\right)\right) \geq c \cdot n \cdot log_2(n)$$

Следовательно $T(n) = \Theta(n \cdot log(n))$