Алгоритмы и модели вычислений.
Задание 9: сортировки

Сергей Володин, 272 гр.

задано 2014.04.10

(каноническое) Задача 38

(Идея обсуждалась с Дашей Решетовой)

1. Фиксируем алгоритм A, $n \in \mathbb{N}$ D_A — разрешающее дерево. $P = \{(k_i, l_i)\}_{i=1}^m$ — путь. $G_A^P(V, E)$ — граф, $E = \overline{1, m}$. $(i, j) \in E \Leftrightarrow \exists t \in \overline{1, m} \colon (i, j) = (k_t, l_t)$
   (a) Обозначим $\mathcal{A}$ — множество корректных алгоритмов нахождения минимума
   (b) $A \overset{\text{def}}{=} \left[\forall A \in \mathcal{A} \forall P \text{ — путь от корня к листу в } D_A \hookrightarrow |P| \geqslant n-1\right]$
   (c) $B \overset{\text{def}}{=} \left[\forall A \in \mathcal{A} \forall P \text{ — путь от корня к листу в } D_A \hookrightarrow G_A^P \text{ — связен}\right]$
   (d) Фиксируем $A \in \mathcal{A}$, P — путь от корня к листу в D_A. Пусть B. Тогда $G_A^P = (V, E)$ — связен. Докажем, что $|P| \geqslant n - 1$. Действительно, $|V| = n$, G_A^P — связен $\Rightarrow |E| \geqslant n-1$. Но $|E| = \left|\{(i, j) \middle| \exists t \in \overline{1, m} \colon (i, j) = (k_t, l_t)\}\right| \leqslant m$ ($\leqslant$, т.к. сравнение может происходить дважды). Получаем, что $|P| = m \geqslant |E| \geqslant n - 1$ ■
   (e) Пусть $\neg B \Rightarrow \exists A \in \mathcal{A} \exists P$ — путь от корня к листу в $D_A \colon G_A^P$ — не связен. Фиксируем вход $a_{i\,i=1}^{\,n} \subset \mathbb{R}$, на котором достигается путь P. Пусть $V = V_1 \cup ... \cup V_f$ — компоненты связности G_A^P. Пусть на этом пути минимум достигается на элементе с идексом b: $b = \arg\min\{a_i\}_{i=1}^n$. Без ограничения общности, $b \in V_1$ (в первой компоненте связности). Рассмотрим другую компоненту связности V_2 (существует, по предположению граф не связен). Пусть c — индекс в этой компоненте, элемент с этим индексом a_c минимален. Рассмотрим другой вход $\{a'_i\}_{i=1}^n$, совпадающий с исходным кроме $a'_c = a'_b - 1$. Тогда результаты всех сравнений не изменятся: a_c сравнивается только с элементами x с индексами из V_2, и: было $x \geqslant a_c$, теперь $x \geqslant a_c \geqslant a_b > a_b - 1$. Поэтому в разрешающем дереве этому входу соответствует также путь P, значит, A на новом входе вернет ответ a'_b, что неверно, так как $a'_c = a'_b - 1 < a'_b$. Значит, $A \notin \mathcal{A}$ — противоречие. Значит, B ■
2. (a) Утверждение может быть неверно, если в массиве есть повторяющиеся элементы. Пусть это $a_i = a_j$, до текущего шага они не участововали в сравнениях, а на текущем шаге они сравниваются между собой. Тогда $\Delta a = -2$, $\Delta c = 0$ (ни один из них не меньше другого), и $\Delta f = \Delta a + \Delta c = -2$
   (b) Считаем, что элементы не повторяются (иначе утверждение неверно, см. выше). Пусть $\Delta f < -1 \Leftrightarrow \Delta f \leqslant -2 \Leftrightarrow \Delta a + \Delta c \leqslant -2$. c — количество элементов, меньших во всех сравнениях. Значит, $\Delta c \geqslant 0$. Значит, $\Delta a + \Delta c \geqslant \Delta a$. Получаем $\Delta a \leqslant -2$. Очевидно-хуевидно, $\Delta a \geqslant -2$, т.к. за один раз сравниваются два элемента, поэтому количество еще не сравнивающихся элементов не может уменьшиться больше, чем на 2. Получаем $\Delta a = -2$, откуда оба сравнивающихся элемента не сравнивались до этого. Значит, $-2 + \Delta c \leqslant -2$, откуда $\Delta c \leqslant 0$, откуда $\Delta c = 0$. Получаем, что было произведено сравнение элементов, и ни один из них не меньше другого, значит, они равны — противоречие.

(каноническое) Задача 39

$\varnothing$

(каноническое) Задача 40

(Модифицируем алгоритм merge sort. Задачу давал Пименов)

1. n — размер массива $(a_1, ..., a_n)$.
2. Если $n \leqslant 1$, то нет ни одной пары элементов, и количество инверсий равно 0.
3. Разобьем массив X на две части: $A = \overline{1, l}$, $B = \overline{l+1, n}$. Пусть посчитаны количества инверсий для элементов с индексами из L и из R, и элементы в L и R отсортированы. Тогда осталось посчитать количество инверсий для пар (i, j): $i \in A$, $j \in B$, или наоборот. Очевидно, что сортировка частей не изменила количество инверсий для таких пар, т.к. порядок элементов такой пары в массиве не изменился после сортировки: если изначально $A \ni i_0 < j_0 \in B$, то $i < l + 1 \leqslant j$. Обозначим L и R — подмассивы с индексами A и B соответственно
   Выполним модифицированный алгоритм слияния двух упорядоченных подмассивов L и R. На каждой итерации цикла считаем количество инверсий текущего элемента с последующими из другого множества (так обойдем все возможные пары). Результаты суммируем (каждая пара рассматривается только один раз).
   Пусть получены $k-1$ элементов итогового массива, выбраны первые i элементов из L, j из R. Инверсии для $k-1$ также посчитаны.
   k-й равен L_{i+1}, если $L_{i+1} < R_{j+1}$ или R_{j+1} иначе.
   Найдем число инверсий для каждого случая:

- Если добавляем L_{i+1}, то последующими элементами из другого множества (из другой части) могут быть $R_{j+1} < R_{j+2} < \ldots$. Но (условие случая) $L_{i+1} < R_{j+1} < \ldots \Rightarrow$ инверсий с последующими нет (т.к. "более левый" в исходном массиве элемент меньше правых из другого множества).
- Если добавляем R_{j+1}, то последующими элементами из другого множества будут $L_{i+1} < L_{i+2} < \ldots$. Но (условие случая) $R_{j+1} < L_{i+1} < \ldots$. Получаем, что число инверсий равно количеству оставшихся элементов из левой части, т.е. $|L| - i$ (т.к. "более правый" в исходном массиве элемент меньше такого количества "более левых" из другого множества).

Таким образом, найдено количество инверсий в X, и X отсортирован.

4. Значит, сортировку частей и поиск количеств инверсий в них можно делать рекурсивно, $l \stackrel{\text{def}}{=} \lfloor \frac{n}{2} \rfloor$
5. Очевидно, что время работы такого алгоритма равно времени работы merge sort, т.е. $T(n) = O(n \log n)$