УДК 519.159+656.212.9 Вестник СПбГУ. Сер. 1. Т. 2 (60). 2015. Вып. 2

ОЦЕНКА СЛОЖНОСТИ АЛГОРИТМА СОРТИРОВКИ ЖЕЛЕЗНОДОРОЖНОГО СОСТАВА

И. В. Романовский, Е. Э. Ржевская

Санкт-Петербургский государственный университет,
Российская Федерация, 199034, Санкт-Петербург, Университетская наб., 7-9

Рассматривается задача сортировки грузового железнодорожного состава и ее математическая формулировка. Описывается предлагаемый алгоритм, использующий сортировочную горку. Эта горка состоит из небольшого возвышения, с которого спускается дерево путей, начинающихся на верхушке холма и разделяющихся на k путей, с возможностью провести вагон с верхушки возвышения в любой заданный конец. На каждом прогоне вагоны распределяются по путям, а затем состав собирается из получившихся фрагментов. Находятся максимальное число прогонов всего поезда через горку для сортировки состава и общая трудоемкость алгоритма. Некоторыми чертами идея этого метода похожа на основную идею алгоритма сортировки перфокарт в классическом счетно-аналитическом комплекте. Исходные данные — это перестановка, сопоставляющая каждому вагону его номер от конца в требуемом расположении вагонов. В решении используется разбиение перестановки $1:n$ на *монотонные сегменты*. Например, перестановка $3, 8, 2, 6, 1, 4, 5, 7$ с $n = 8$ разбивается на сегменты $(3, 2, 1)$, (4), $(6, 5)$, $(8, 7)$. Эти сегменты легче определить в терминах перестановки, обратной исходной. Она считается строкой из чисел, и каждому сегменту разбиения исходной перестановки соответствует максимальная подстрока из монотонно убывающих чисел. В примере обратная перестановка $5, 3, 1; 6, 7, 4; 8, 2$ разбивается на 4 подстроки (разделенные знаками «;»). Пусть p — число частей в этом разбиении, а k — число путей на горке. Доказано, что число прогонов состава через горку для получания $n, n-1, \ldots, 2, 1$ не превосходит $\lceil \log_k p \rceil \leqslant \lceil \log_k n \rceil$, и эта оценка достигается предложенным в статье алгоритмом. Библиогр. 2 назв.

Ключевые слова: упорядочение перестановки, сортировочная горка.

1. Введение. В статье [1] одним из авторов был описан метод сортировки железнодорожного состава. Рассматривается сортировочная горка с k путями, с помощью которой требуется расставить n вагонов состава в требуемом порядке. В работе охарактеризован метод, имеющий некоторое сходство с идеями устройства для сортировки перфокарт, которое применялось в начале «компьютерной эры» [2], и описана компьютерная реализация прогона состава на горке. Здесь мы намерены оценить число прогонов и общую трудоемкость численного метода.

2. Математическая формулировка. Полезно перейти от железнодорожных терминов к математическим. То, что задан железнодорожный состав, означает, что у нас имеется перестановка из чисел от 1 до n, каждое число описывает место, которое соответствующий вагон должен занять в упорядоченном поезде, эти места отсчитываются от хвоста поезда. На каждом шаге сортировки (спуске с горки или прогоне) вагоны распределяются по k железнодорожным путям, ведущим от верхушки горки вниз. Под действием силы тяжести вагоны поочередно спускаются вниз, и с помощью стрелок диспетчер выбирает путь для этого вагона. Когда весь состав распределен таким образом по путям, состав собирается заново из получившихся k частей, что завершает один спуск. Мы считаем, что на каждом спуске состав распускается полностью. Так как спуски с сортировочной горки — действия достаточно трудоемкие, хочется минимизировать число спусков, для чего необходимо найти наименьшее возможное число шагов s.

Исходными данными являются число вагонов в составе n, число путей в горке k и перестановка $P = \{p_1, \ldots, p_n\}$ чисел $1, \ldots, n$. Переформирование состава — это

получение из P последовательности $n, n-1, \ldots, 1$ с помощью описанной операции «спуска состава»: последовательного переноса чисел $p_n, \ldots, p_1$ в k стековых накопителей (путей горки). Эту операцию, чтобы отдалиться от «вагонной» терминологии, мы будем в дальнейшем называть *прогоном*.[1]

Определение. *Монотонным сегментом* будем называть наибольший диапазон чисел от a до b ($n \geqslant a \geqslant b \geqslant 1$), такой, что любое число i из этого диапазона, кроме b, стоит в перестановке P правее $i+1$.

Например, в перестановке $2, 3, 5, 1, 4$ есть три *монотонных сегмента*: $\Gamma_1 = (2, 1), \Gamma_2 = (3), \Gamma_3 = (5, 4)$.[2] Легко видеть, что разбиение перестановки на монотонные сегменты однозначно. Его легко вычислить (за линейное по n время), если найти перестановку Q, обратную P: каждый монотонный сегмент P соответствует фрагменту убывания чисел в перестановке Q.

Для нашего примера $Q = (pos_1, pos_2, pos_3, pos_4, pos_5) = (4, 1, 2, 5, 3)$ с тремя фрагментами убывания $(4, 1)$, (2), $(5, 3)$. Они и задают выписанные выше сегменты Γ_i.

Дальнейшее лучше иллюстрировать на примере большего размера. Пусть $n = 15$, $k = 3$, и перестановка P выглядит так:

#	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
P	9	5	7	6	1	2	10	4	3	12	15	11	14	13	8
Q	5	6	9	8	2	4	3	15	1	7	12	10	14	13	11

Здесь же показана и обратная перестановка Q. Получаются следующие монотонные сегменты $\Gamma_1 = (1)$, $\Gamma_2 = (2)$, $\Gamma_3 = (5, 4, 3)$, $\Gamma_4 = (7, 6)$, $\Gamma_5 = (9, 8)$, $\Gamma_6 = (10)$, $\Gamma_7 = (12, 11)$, $\Gamma_8 = (15, 14, 13)$.

Сегменты по накопителям (стекам) распределяются в циклическом порядке

стек 1	$\Gamma_1 = (1),$	$\Gamma_4 = (7, 6),$	$\Gamma_7 = (12, 11),$	(7,6,1,12,11)
стек 2	$\Gamma_2 = (2),$	$\Gamma_5 = (9, 8),$	$\Gamma_8 = (15, 14, 13),$	(9,2,15,14,13,8)
стек 3	$\Gamma_3 = (5, 4, 3),$	$\Gamma_6 = (10),$		(5,10,4,3)

— сначала выделяется по одному сегменту каждому накопителю, затем снова по одному сегменту, затем снова, пока не закончатся сегменты. В правом столбце таблицы показаны множества чисел, попадающих в каждый накопитель. В этих множествах порядок записи чисел наследуется из исходной перестановки (так: $\ldots, 7, 6, 1, \ldots, 12, \ldots, 11, \ldots$). Эта операция выполняется также за линейное время — объединение сегментов в одно множество с учетом порядка элементов, т. е. в

$$7, 6, 1, 12, 11\,,\ 9, 2, 15, 14, 13, 8\,,\ 5, 10, 4, 3$$

лишнего времени не занимает.

3. Оценка трудоемкости метода. Теперь для оценки трудоемкости метода осталось оценить количество прогонов s. Это можно сделать, оценивая эффект шагов.

Лемма. $s \leqslant \lceil \log_k n \rceil$.

[1] Слово *прогон* в смысле «однократное выполнение некоторой сложной процедуры или алгоритма» еще не вошло в словари, но используется уже достаточно часто. Например, «генеральный прогон спектакля».

[2] Построение монотонных сегментов очень просто: Число 1 входит в первый сегмент обязательно. Если число 2 стоит до 1 (где угодно), оно добавляется к сегменту. А и число 3 добавилось бы, но стоит после 2 и начинает новый сегмент. И 4 тоже начинает новый сегмент, в который входит и 5.

Доказательство. Докажем, что на каждом прогоне количество сегментов уменьшается почти в k раз. Сегменты получают свои номера в порядке возрастания наименьших чисел в них. Рассмотрим сегменты $\Gamma_1, \ldots, \Gamma_k$. Они располагаются соответственно в накопителях $1, \ldots, k$. После прогона эти сегменты будут располагаться в перестановке в правильном порядке, т. е. Γ_i будет расположен правее Γ_{i+1}. Таким образом, все числа этих сегментов образуют в следующей перестановке монотонный сегмент. Аналогично для $\Gamma_{l\cdot k+1}, \ldots, \Gamma_{(l+1)\cdot k}$ при каждом следующем l, кроме, возможно, последнего. В последней группе сегментов может быть меньше k, от чего тоже образуется новый сегмент. Таким образом, после каждого прогона количество сегментов уменьшается примерно в k раз, округляясь вверх до ближайшего целого, так что $s' \leqslant \lceil s/k \rceil$, где s' — число сегментов в получающейся перестановке.

Пусть в начальной перестановке было p групп, $p \leqslant n$. Выберем такое m, чтобы $k^m \leqslant p < k^{m+1}$, тогда $m+1 = \lceil \log_k p \rceil$. Отсюда следует, что $r = p - k^m < (k-1)\cdot k^m$. После первого прогона получим количество сегментов

$$\left\lceil \frac{k^m + r}{k} \right\rceil = k^{m-1} + \left\lceil \frac{r}{k} \right\rceil \leqslant k^{m-1} + \left\lceil \frac{(k-1)\cdot k^m}{k} \right\rceil \leqslant k^{m-1} + (k-1)\cdot k^{m-1}.$$

Аналогично после второго прогона получим, что количество сегментов не превосходит $k^{m-2}+(k-1)\cdot k^{m-2}$, …, через m прогонов не превосходит $1+(k-1) = k$, а k сегментов легко на $m+1$-м пропуске окажутся просто в правильном порядке. Таким образом, $s \leqslant \lceil \log_k p \rceil \leqslant \lceil \log_k n \rceil$. □

Видно, что наибольшее количество шагов (прогонов) потребуется при перестановке с n сегментами, а именно для $1, \ldots, n$. А так как оценка количества прогонов зависит от количества сегментов, интересно посмотреть, сколько же сегментов в перестановке для различных n.[3]

Из доказанной леммы и линейности времени выполнения одного прогона следует справедливость следующей теоремы

Теорема. *Существует такое λ, что трудоемкость алгоритма не превосходит $\lambda \cdot n \cdot \lceil \log_k n \rceil$.*

Литература

1. *Романовский И. В.* Алгоритм для железнодорожной сортировочной горки // Компьютерные инструменты в образовании. 2014. Вып. 3. С. 56–59.

2. *Брустейн Д. П.* Сортировальная машина // Большая сов. энциклопедия. Изд. 3-е. Т. 24, кн. 1. 1976. С. 197.

Статья поступила в редакцию 25 декабря 2014 г.

Сведения об авторах

Романовский Иосиф Владимирович — профессор; josephromanovsky@gmail.com

Ржевская Екатерина Эдуардовна — студент; katrin8794@yandex.ru

[3] Описанный выше удобный способ вычисления числа монотонных сегментов в перестановке позволил увидеть близость распределения этого числа как функции от случайной перестановки к нормальному распределению. Соответствующие вычисления, точные для малых n и статистические для больших, проводились при активной помощи студента Н. М. Мирошниченко.

COMPLEXITY OF A CAR SORTING ALGORITHM

Joseph V. Romanovskiy, Ekaterina E. Rzhevskaya

St.Petersburg State University, Universitetskaya nab., 7-9, St.Petersburg, 199034, Russian Federation;
josephromanovsky@gmail.com, katrin8794@yandex.ru

In this article a problem of sorting cars in a freight train is considered and its mathematical formulation is given. The sorting is executed on a so called sorting yard hump. The yard hump consists of a small hump with a tree of railroads starting on the top of the hump and branching out to k rail ends with controls that allow to direct a car moving from the top to a given end. On each run, the train is distributed to the end paths and after that collected back from fragments on the paths. The maximum number of runs of the whole train to sort it and overall complexity of the algorithm are found. Some features of this method are similar to the main idea of the well know procedure of sorting perfocards in the classical Hollerith set of hardware. The initial data is a permutation of the set $1 : n$, where each car is associated with its position in the desired train sequence. In solution we use a partition of the permutation of $1 : n$ to *monotone segments*. For example, the permutation $3, 8, 2, 6, 1, 4, 5, 7$ with $n = 8$ is divided to segments $(3, 2, 1)$, (4), $(6, 5)$, $(8, 7)$. These segments are easier defined in terms of the permutation inverse to the given one. It is treated as a string consisting of numbers, and each part of the partition corresponds to a maximal monotonically decreasing set of numbers in the list of elements. In our example the inverse permutation $5, 3, 1; 6; 7, 4; 8, 2$ is divided to 4 substrings by separators "**;**". Let p be the number of parts in this partition, and k is the number of roads in the hump. Then the number of runs of the train through the hump to get $n, n-1, \ldots, 2, 1$ does not exceed $\lceil \log_k p \rceil \leqslant \lceil \log_k n \rceil$, and this estimation is reached by the proposed algorithm. Refs 2.

Keywords: ordering of permutation, sorting hump.

References

1. Romanovsky J. V., "An algorithm for the railroad sorting hump", *Computational instruments in education* 3, 56–59 (2014) [in Russian].

2. Brustein D. P., "Sorting mashine", *Great Sov. Enc.* 3-rd ed., **24** part 1, 197 (1976) [in Russian].