Journal of Computer and System Sciences 81 (2015) 1556–1574

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Pancake Flipping is hard ☆

Laurent Bulteau, Guillaume Fertin, Irena Rusu

Laboratoire d'Informatique de Nantes-Atlantique (LINA), UMR CNRS 6241, Université de Nantes, 2 rue de la Houssinière, 44322 Nantes Cedex 3, France

ARTICLE INFO

Article history:
Received 18 February 2013
Received in revised form 30 July 2014
Accepted 31 July 2014
Available online 11 April 2015

Keywords:
Pancake problem
Permutations
Prefix reversals
Computational complexity

ABSTRACT

Pancake Flipping is the problem of sorting a stack of pancakes of different sizes (that is, a permutation), when the only allowed operation is to insert a spatula anywhere in the stack and to flip the pancakes above it (that is, to perform a prefix reversal). In the burnt variant, one side of each pancake is marked as burnt, and it is required to finish with all pancakes having the burnt side down. Computing the optimal scenario for any stack of pancakes and determining the worst-case stack for any stack size have been challenges for over more than three decades. Beyond being an intriguing combinatorial problem in itself, it also yields applications, e.g. in parallel computing and computational biology. In this paper, we show that the Pancake Flipping problem, in its original (unburnt) variant, is NP-hard, thus answering the long-standing question of its computational complexity.

1. Introduction

The pancake problem was stated in [10] as follows:

The chef in our place is sloppy, and when he prepares a stack of pancakes they come out all different sizes. Therefore, when I deliver them to a customer, on the way to the table I rearrange them (so that the smallest winds up on top, and so on, down to the largest at the bottom) by grabbing several from the top and flipping them over, repeating this (varying the number I flip) as many times as necessary. If there are n pancakes, what is the maximum number of flips (as a function of n) that I will ever have to use to rearrange them?

Stacks of pancakes are represented by permutations, and a flip consists in reversing a prefix of any length. The previous puzzle yields two entangled problems:

- Designing an algorithm that sorts any permutation with a minimum number of flips (this optimization problem is called MIN-SBPR, for Sorting By Prefix Reversals).
- Computing $f(n)$, the maximum number of flips required to sort a permutation of size n (the diameter of the so-called *pancake network*).

☆ A preliminary version of this article appeared in the proceedings of the 37th International Symposium on Mathematical Foundations of Computer Science (MFCS 2012) [6].

E-mail addresses: Laurent.Bulteau@univ-nantes.fr (L. Bulteau), Guillaume.Fertin@univ-nantes.fr (G. Fertin), Irena.RUsu@univ-nantes.fr (I. Rusu).

http://dx.doi.org/10.1016/j.jcss.2015.02.003
0022-0000/

Gates and Papadimitriou [12] introduced the *burnt* variant of the problem: the pancakes are two-sided, and an additional constraint requires the pancakes to end with the unburnt side up. The diameter of the corresponding *burnt pancake network* is denoted $g(n)$. A number of studies [7–9,12,15–17] have aimed at determining more precisely the values of $f(n)$ and $g(n)$, with the following results:

- $f(n)$ and $g(n)$ are known exactly for $n \leq 19$ and $n \leq 17$, respectively [8].
- $15n/14 \leq f(n) \leq 18n/11 + O(1)$ [16,7].
- $\lfloor (3n+3)/2 \rfloor \leq g(n) \leq 2n - 6$ [8] (upper bound for $n \geq 16$).

Considering MIN-SBPR, 2-approximation algorithms have been designed, both for the burnt and unburnt variants [9,11]. Moreover, Labarre and Cibulka [17] have characterized a subclass of signed permutations, called *simple permutations*, that can be sorted in polynomial time.

The pancake problems have various motivations. For instance, the pancake network, having both a small degree and diameter, is of interest in parallel computing [1,19,18]. A more distant motivation concerns a variant of the problem, called Sorting By Reversals [2,3], which has applications in comparative genomics. In Sorting By Reversals, *any* subsequence can be flipped at any step (not only prefixes), and reversals are possible elementary modifications that can affect a genome during evolution. The Sorting By Reversals problem is now well-known, with a polynomial-time exact algorithm [13,14] for the signed case, and a 1.375-approximation [4] for the APX-hard unsigned case [5]. Although prefix reversals are less realistic, any improvement in this setting may have some impact on the more general Sorting By Reversals problem.

In this paper, we prove that the MIN-SBPR problem is NP-hard (in its unburnt variant), thus answering a question which has remained open for several decades. We in fact prove a stronger result: it is known that the number of breakpoints of a permutation (that is, the number of pairs of consecutive elements that are not consecutive in the identity) is a lower bound on the number of flips necessary to sort a permutation. We show that deciding whether this bound is tight is already NP-hard.

2. Notations

We denote by $[\![a\,;\,b]\!]$ the interval $\{a, a+1, \ldots, b\}$ (for $b < a$, we have $[\![a\,;\,b]\!] = \emptyset$). Let n be an integer. Input sequences are permutations of $[\![1\,;\,n]\!]$, that is we consider only sequences where all elements are unsigned, and there cannot be duplicates. When there is no ambiguity, we use the same notation for a sequence and the set of elements it contains. We use upper case letters for sets and sequences, and lower case letters for elements.

Consider a sequence S of length n, $S = \langle x_1, x_2, \ldots, x_n \rangle$. Element x_1 is said to be the *head element* of S. Sequence S has a *breakpoint* at position r, $1 \leq r < n$ if $x_r \notin \{x_{r+1} - 1, x_{r+1} + 1\}$, and a *breakpoint* at position n if $x_n \neq n$. We write $d_b(S)$ the number of breakpoints of S. Note that having $x_1 \neq 1$ does not directly count as a breakpoint, and that $d_b(S) \leq n$ for any sequence of length n. For any $p \leq q \in \mathbb{N}$, we write $\mathcal{I}_q^p$ the sequence $\langle p, p+1, p+2, \ldots, q \rangle$; $\mathcal{I}_n^1$ is the *identity*. For a sequence of any length $S = \langle x_1, x_2, \ldots, x_k \rangle$, we write $^\star S$ the sequence obtained by reversing S: $^\star S = \langle x_k, x_{k-1}, \ldots, x_1 \rangle$. Given an integer p, we write $p + S = \langle p + x_1, p + x_2, \ldots, p + x_k \rangle$.

The *flip of length* r is the operation that consists in reversing the r first elements of the sequence. It transforms

$$S = \langle x_1, x_2, \ldots, x_r, \quad x_{r+1}, \ldots, x_n \rangle$$
$$\text{into} \quad S' = \langle x_r, x_{r-1}, \ldots, x_1, \quad x_{r+1}, \ldots, x_n \rangle.$$

Note that the flip of length 1 does not modify S, and the flip of length n transforms S into $^\star S$. Moreover, since a flip of length r cannot add or remove breakpoints other than in position r, we have the following easy property.

Property 1. *Given a sequence S' obtained from a sequence S by performing one flip, we have $d_b(S') - d_b(S) \in \{-1, 0, 1\}$.*

A flip from S to S' is said to be *efficient* if $d_b(S') = d_b(S) - 1$, and we reserve the notation $S \to S'$ for such flips. A sequence of size n, different from the identity, is a *deadlock* if it yields no efficient flip, and we write $S \to \bot$. By convention, we place a specific separator $\downarrow$ in a sequence at the positions corresponding to possible efficient flips: there are at most two of them, and at least one if the sequence is neither a deadlock nor the identity. A *path* is a series of flips, it is *efficient* if each flip it contains is efficient. A sequence S is *efficiently sortable* if there exists an efficient path from S to the identity (equivalently, if it can be sorted in $d_b(S)$ flips). See for example Fig. 1.

Let S be a sequence different from the identity, and $\mathbb{T}$ be a set of sequences. We write $S \Longrightarrow \mathbb{T}$ if both following conditions are satisfied:

1. for each $T \in \mathbb{T}$, there exists an efficient path from S to T.
2. for each efficient path from S to the identity, there exists a sequence $T \in \mathbb{T}$ such that the path goes through T.