Theoretical Computer Science 609 (2016) 60–75

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Successor rules for flipping pancakes and burnt pancakes

J. Sawada [a,1], A. Williams [b,*]

[a] School of Computer Science, University of Guelph, Canada
[b] Division of Science, Mathematics, and Computing, Bard College at Simon's Rock, Great Barrington, USA

ARTICLE INFO

Article history:
Received 28 January 2015
Received in revised form 12 July 2015
Accepted 6 September 2015
Available online 10 September 2015
Communicated by G.F. Italiano

Keywords:
Pancake sorting
Greedy algorithm
Permutations
Signed-permutations
Prefix-reversal
Symmetric group
Cayley graph
Hamilton cycle
Human problem solving

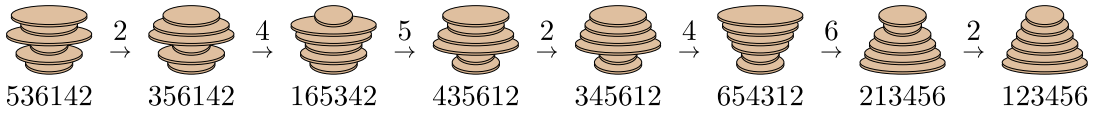
ABSTRACT

A stack of n pancakes can be rearranged in all $n!$ ways by a sequence of $n! - 1$ flips, and a stack of n 'burnt' pancakes can be rearranged in all $2^n n!$ ways by a sequence of $2^n n! - 1$ flips. In both cases, a computer program can efficiently generate suitable solutions. We approach these tasks instead from a human perspective. *How can we determine the next flip directly from the current stack? How can we flip the minimum or maximum number of (burnt) pancakes overall? What if we are only allowed to flip the top $n-2$, $n-1$, or n (burnt) pancakes?* We answer the first question with simple successor rules that take worst-case $O(n)$-time and amortized $O(1)$-time. Then we answer the second question exactly for minimization, and provide conjectures for maximization. For the third question, we prove that solutions almost certainly exist for pancakes and burnt pancakes using only these three flips. More broadly, we discuss how efficiency and optimality can shape iterative solutions to Hamilton cycle problems in highly symmetric graphs.

1. Introduction

1.1. Harried waiter problems

Jacob Goodman, writing under the name Harry Dweighter ("*harried waiter*"), introduced the original pancake problem: Given a stack of n pancakes of distinct sizes, what is the minimum number of flips required to sort the pancakes from smallest to largest? In this problem, the individual pancakes are numbered $1, 2, \ldots, n$ by increasing size, and a stack of pancakes can be represented by a permutation in one-line notation. Each 'flip' of the topmost i pancakes corresponds to a *prefix-reversal* of length i in the permutation. For example, the following illustration shows how the stack 536142 can be sorted in 7 flips:

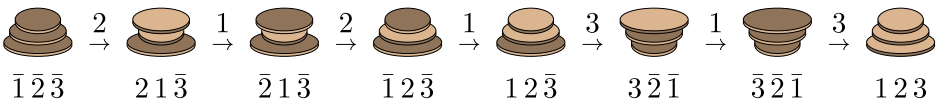


* Corresponding author.
E-mail addresses: jsawada@uoguelph.ca (J. Sawada), haron@uvic.ca (A. Williams).

[1] Research supported by NSERC RGPIN 298335-2012.

http://dx.doi.org/10.1016/j.tcs.2015.09.007
0304-3975/

A well-studied variation features 'burnt' pancakes, which have two distinct sides. In this problem, a stack is represented by a *signed permutation* in one-line notation, with i and $\bar{i}$ being used when the burnt side of pancake i is facing down or up, respectively. Each 'flip' of the topmost i pancakes corresponds to a *sign-complementing prefix-reversal* in the signed permutation. For example, the stack $\bar{3}\,\bar{2}\,\bar{1}$ can be sorted in 7 flips as follows:



Goodman was interested in worst-case stacks (i.e. the largest number of flips required to sort any stack) and the previous illustrations are examples of such stacks for $n = 6$ pancakes and $n = 3$ burnt pancakes. Currently the best-known lower-bounds and upper-bounds are $\frac{15}{14}n$ and $\frac{18}{11}n$ for pancakes (see Heydari and Sudborough [12], and Chitturi et al. [4]), and $2n - 2$ and $\lfloor \frac{3n+2}{2} \rfloor$ for burnt pancakes (see Cohen and Blum [6], and Cibulka [5]), respectively. Determining the minimum number of flips for an arbitrary stack was recently shown to be NP-hard for pancakes (see Bulteau, Fertin, and Rusu [3]) while the complexity of the burnt variation is unknown. If arbitrary substacks are allowed to be flipped, then the unburnt sorting problem is APX-hard (see Berman and Karpinski [2]) and the burnt sorting problem can be solved in polynomial-time (see Hannenhalli and Pevzner [10]).

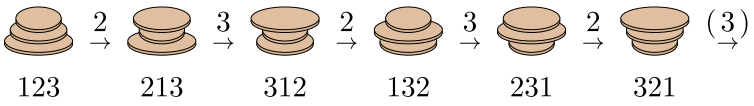
Research on pancake sorting had humble beginnings — Goodman formulated the problem while sorting a stack of towels — but has a number of interesting applications including genomics (see Fertin et al. [9]) and in vivo computing (see Haynes [11] for an introduction to the 'e.Hop' restaurant), and has been discussed by the media (see Singh [27]).

1.2. Harassed waitress problems

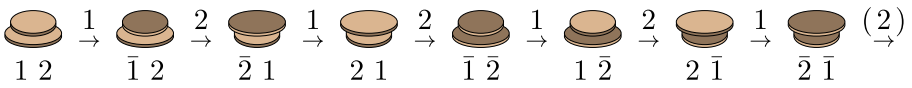
Goodman introduced his problem using an overzealous waiter who sorts his customers' pancakes to hide the chef's sloppiness. On the other hand, Zaks [35] solved a problem that could be posed by a demanding obsessive-compulsive customer at the same restaurant:

> Using our algorithms the poor waiter will be able to generate, in $n!$ such steps, all possible $n!$ stacks (returning to the original one) ... Moreover, in $1/2$ of these steps he will reverse the top 2 pancakes, in $1/3$ of them the top 3, and, in general, in $(k-1)/k!$ of them he will reverse the top k pancakes, which amounts to less than 2.8 pancakes reversed on the average.

To differentiate this problem from Goodman's, we replace the waiter by a waitress, and name it a *harassed waitress problem*. For example, Zaks's solution for $n = 3$ is as follows:



This solution is *cyclic* since one additional flip transforms the last stack into the first stack. Cyclic solutions to the analogous 'burnt' problem were found by Suzuki, N. Sawada, and Kaneko [30]. For example, their solution for $n = 2$ (using $\mathsf{HC}(\mathbf{s})$ for $\mathbf{s} = 1\,2\,\cdots\,n$) is:



Both solutions can be generated by efficient algorithms that are difficult for humans to run in practice. In particular, the algorithm in [35] maintains two additional arrays of n integers, while the algorithm in [30] is recursive. Our goal is to create efficient algorithms that are practical for humans. We focus on three questions that are illustrated in Fig. 1. The first question is the following:

How efficiently can we compute the next flip from the current stack?

As the above question indicates, we want a *successor rule* that determines the next flip to apply directly from the current stack, without any additional memory or algorithmic state. This will allow our clever waitress to suspend and resume her task without devoting any memory to her progress. We also strive for 'optimal' solutions by counting the total number of (burnt) pancakes that are flipped throughout the order: