Theoretical Computer Science 636 (2016) 95–100

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Note

Batcher's odd–even exchange revisited: A generating functions approach

Helmut Prodinger [1]

Department of Mathematics, University of Stellenbosch, 7602 Stellenbosch, South Africa

ARTICLE INFO

Article history:
Received 23 November 2015
Received in revised form 7 April 2016
Accepted 6 May 2016
Available online 24 May 2016
Communicated by L.M. Kirousis

Keywords:
Batcher's odd–even exchange
Generating function
Singularity analysis
Mellin transform

ABSTRACT

In the celebrated odd–even exchange algorithm by Batcher, the quantity *average number of exchanges* is one of the fundamental quantities of interest. It was a mystery a few years ago and is still tricky today. We provide an approach that is purely based on generating functions to provide an explicit expression. The asymptotic analysis was done several years ago but never published in a journal and is thus provided here as well in condensed form. It is a combination of singularity analysis of generating functions and Mellin transform techniques.

1. Introduction

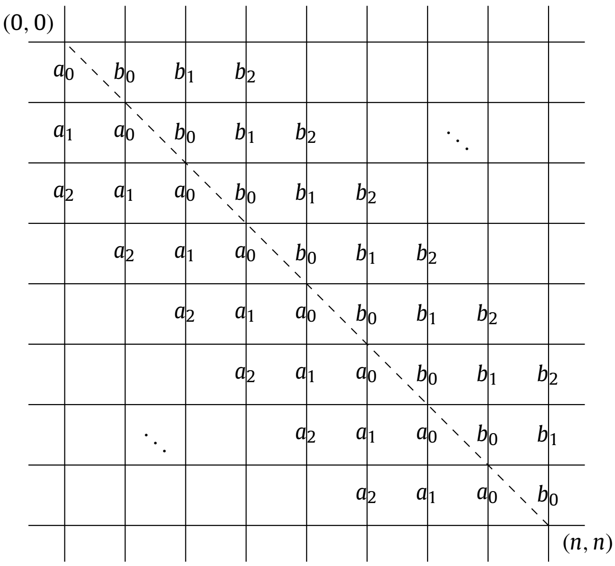
Batcher's odd–even merge is a sorting method that is well documented in books and papers, notably in Knuth's monumental work *The Art of Computer Programming*, Volume 3 [10,11]. For completeness, we provide the original reference [1].

Since it is a bit complicated and long to describe, we refrain from doing this and just mention that its analysis (average number of exchanges, provided that a random permutation is given) boils down to a *lattice path counting problem*, as described in [10]: All $\binom{2n}{n}$ lattice paths from $(0,0)$ to (n,n) are considered; for each path, the sum of the vertical weights that it traverses is recorded. The total sum of these counts, divided by the number of all paths $\binom{2n}{n}$ is denoted by B_n, the average number of exchanges.

E-mail address: hproding@sun.ac.za.

[1] The author was supported by an incentive grant (29584) of the National Research Foundation of South Africa.

http://dx.doi.org/10.1016/j.tcs.2016.05.014



The weights $a_0, a_1, \ldots$ and $b_0, b_1, \ldots$ will be discussed in a minute.

The analysis of this quantity was posed as an open problem in [10]; however, Knuth came very close to the solution, as we will soon see. A first complete answer was given by Sedgewick [16], who showed that

$$B_n = \frac{1}{\binom{2n}{n}} \sum_{k\ge 1} \binom{2n}{n-k} (2F(k) + k), \tag{1}$$

where $F(k)$ is the summatory function of $f(j)$, which is the number of ones in the Gray code representation of j:

$$F(k) := \sum_{0\le j<k} f(j).$$

More important than the sequence $f(k)$ itself is the sequence of its differences $\theta(k) = f(k) - f(k-1)$, since these numbers have a number theoretic significance, as discussed later.

The weights in our problem are $a_k = f(k)$ and $b_k = f(k) + 1$.

Sedgewick also provided asymptotics for B_n, by a technique, that was called *Gamma function method* at the period, which is today known as *Mellin transform*; see the survey [3].

Several researchers started from the formula (1) and discussed alternative asymptotic methods, see, e.g., [5,13,14,12,9]. Sedgewick derived this formula by skillful manipulations of binomial coefficients and sums involving them. A more modern approach would be through generating functions, a point of view that is emphasized in the important book [6], see also [7,15]. Knuth himself considered this problem through generating functions in the first edition [10] and came very close to this formula. However, in the second edition [11], he switched to Sedgewick's approach.

Here, I want to present such a generating function approach, perhaps close to what Knuth had in mind. In a final section, an asymptotic evaluation will be presented, which is probably the best one that exists today; it was already reported in [14], but the approach deserves to be better known.

2. A generating function approach

Instead of summing over all labels of one path, one splits such a path into n copies, where each one carries exactly one of the n labels, and sums these. We call such a path with exactly one vertical label a *decorated path*.

A decorated path goes from $(0,0)$ to (n,n) and carries exactly one (vertical) label. Let $\mathscr{W}$ be the family of all paths $(0,0)$ to (n,n), $\mathscr{D}$ be the family of all paths $(0,0)$ to (n,n), staying on one (prescribed) side of the diagonal, $\mathscr{R}_p$ the family of paths with vertical label a_p, and $\mathscr{S}_p$ the family of paths with vertical label b_p. We write d for a down-step, and h for a horizontal-step.

With Roman letters we write the associated ordinary generating functions (only the down-steps, say, are counted). We treat a_p either as a formal symbol or a number, depending on the context. When deriving a recursion for $\mathscr{R}_p$, we treat a_p as a *fixed* symbol (not depending on p). This convenient *abuse of language* allows us to avoid to write double indices.