Maximum Number of Steps of Topswops on 18 and 19 Cards

Kento Kimura, Atsuki Takahashi, Tetsuya Araki, Kazuyuki Amano*

March 16, 2021

Abstract

Let $f(n)$ be the maximum number of steps of Topswops on $n$ cards. In this note, we report our computational experiments to determine the values of $f(18)$ and $f(19)$. By applying an algorithm developed by Knuth in a parallel fashion, we conclude that $f(18) = 191$ and $f(19) = 221$.

1 Introduction

Consider a deck of $n$ cards numbered 1 to $n$ arranged in random order, which can be viewed as a permutation on $\{1, 2, \ldots, n\}$. Continue the following operation until the top card is 1. If the top card of the deck is $k$, then turn over a block of $k$ cards at the top of the deck. This card game is called Topswops, which was originally invented by J.H. Conway in 1973. See e.g., the introduction of [4] for a short history of the game.

The problem is to find an initial deck that requires a maximum number of steps until termination, for a given number of cards. For a positive integer $n$, let $f(n)$ be the maximum number of steps until termination for Topswops on $n$ cards. We call an initial deck that needs $f(n)$ steps largest. For example, the deck $(3, 1, 4, 5, 2)$ is largest for $n = 5$. The game goes as

$$(3, 1, 4, 5, 2) \rightarrow (4, 1, 3, 5, 2) \rightarrow (5, 3, 1, 4, 2) \rightarrow (2, 4, 1, 3, 5)$$
$$\rightarrow (4, 2, 1, 3, 5) \rightarrow (3, 1, 2, 4, 5) \rightarrow (2, 1, 3, 4, 5) \rightarrow (1, 2, 3, 4, 5).$$

and terminates after $f(5) = 7$ steps.

The best known upper bound on $f(n)$ is $F(n + 1) - 1 = O(1.618^n)$, where $F(k)$ is the $k$-th Fibonacci number [2] Problems 107–109] and the best known lower bound is $\Omega(n^2)$ [4]. The gap is exponential. The exact values of $f(n)$ for $n \leq 17$ have been obtained by an exhaustive search with some pruning techniques. The sequence is $(0, 1, 2, 4, 7, 10, 16, 22, 30, 38, 51, 65, 80, 101, 113, 139, 159)$ for $n = 1, 2, \ldots, 17$. See the sequence A000375 of OEIS [5].

In this note, we describe our effort for extending this list for $n = 18$ and 19. Namely, by applying an algorithm developed by Knuth [3] in a parallel fashion, we conclude that $f(18) = 191$ and $f(19) = 221$. We also find that the number of initial decks that attain the maximum for

*All authors are at Department of Computer Science, Gunma University
Algorithm 1, the minus value $-i$ for $i$ can always create another deck $A_{1, 2, \ldots}.$ The sequence A123398 at OEIS [5].

The rest of this note is as follows. In Section 2, we give a brief explanation of Knuth’s algorithm [2]. Then, in Section 3, we describe our computational experiments for determining $f(18)$ and $f(19).$ The code used in our experiments can be viewed on GitLab at https://gitlab.com/kkimura/tswops.

## 2 Knuth’s algorithm

In this section, we explain an algorithm for finding a largest deck for Topswops used in our experiment, which was developed by Knuth [2] Solution of Problem 107] (see also [3] for the code itself). Three algorithms were described there and we use the most efficient one, which is referred to as a “better” algorithm.

For a natural number $n$, let $[n]$ denote the set $\{1, 2, \ldots, n\}$. For an initial deck $A$, let $S(A)$ be a list $(d_1, d_2, \ldots, d_k)$ ($k \leq n$) where $d_i$ is the $i$-th card that appeared at the top of the deck in the game starting from $A$. For example, $S((3, 1, 4, 5, 2)) = (3, 4, 5, 2, 1)$ (see Eq. (1)) and $S((3, 5, 4, 1, 2)) = (3, 4, 1)$. Notice that the length of $S(A)$ depends on $A$, but the last element of $S(A)$ is always 1. An important property is that if $A$ is largest, then the length of $S(A)$ must be $n$. This can be verified by seeing that if $S(A) = (d_1, \ldots, d_{k-1}, 1)$ for some $k < n$, then we can always create another deck $A'$ such that the first $k$ elements of $S(A')$ is $(d_1, \ldots, d_{k-1}, d')$ for $d' \in \{2, \ldots, n\}\{d_1, \ldots, d_{k-1}\}$ and that the game for $A'$ is strictly longer than the one for $A$.

Let $P$ be the set of all lists $p = (p_1, p_2, \ldots, p_n)$ such that $p$ is a permutation on $[n]$ and $p_n = 1$. Given a list $p \in P$, we can get an initial deck $S^{-1}(p)$ by the following algorithm. In Algorithm 1 the minus value $-i$ in $A$ means that the $i$-th card in a deck is not specified yet.

### Algorithm 1 Generate an Initial Deck

1: **procedure** GenInitDeck($p$)
2: Let $A$ be an array with $(-1, -2, \ldots, -n)$.
3: for $i = 1, 2, \ldots, n$ do
4: $a_{-A_1} \leftarrow p_i$
5: $A_1 \leftarrow p_i$
6: while $A_1 > 1$ do
7: Turn over a block of $A_1$ cards of $A$.
8: return $(a_{i})_{i \in [n]}$

The above arguments suggest that we can determine $f(n)$ by examining all $(n - 1)!$ lists in $P$ together with Algorithm 1. Essentially, Knuth’s algorithm enumerates these lists as well as corresponding decks in a depth-first fashion. Moreover, the algorithm applies two pruning criteria to reduce the size of the search tree.

The first pruning is based on the fact that a largest deck must be a derangement, i.e., the $k$-th card from the top is not $k$ for every $k \in [n]$. In order to explain the second pruning, we need some definitions. Let $A$ be an initial deck and let $A_0$ be the deck obtained from $A$ by executing $c$ steps of the game. Let $T(A_c)$ denote the largest integer $k$ such that the cards numbered $1, 2, \ldots, k$
are located at positions at 1, 2, . . . , k (in an arbitrary order) in the deck $A_e$. It is obvious that if $f(T(A_e)) + c < f(n)$, then $A$ is not largest. Although $f(n)$ is not known beforehand, we can use any lower bound $\ell(n)$ on $f(n)$ in the right hand side of inequalities for pruning.

Note that the depth of the search tree without pruning is $(n - 1)$ and each node at depth $k$ has $n - 1 - k$ children.

### 3 Experiments and Results

Since the search tree of Knuth's "better" algorithm is well-balanced, it is easy to be parallelized. First, we generate the search tree for the first few levels, which corresponds to the first few elements of the list $p$ explained in the last section. Then, distribute the leaves of the tree to many threads and resume the generation in parallel by letting a given leaf as a root of a subtree.

For $n = 18$, we truncate the tree at level two and divide it into 240 subtrees. For $n = 19$, we truncate the tree at level three and divide it into 3,952 subtrees. Each of these numbers is slightly smaller than the one in the original search tree, i.e., 272 (= 17 × 16) or 4,080 (= 18 × 17 × 16), because of the pruning.

In our experiments, we use up to 172 threads in parallel spreading out over nine standard PCs. The computation takes about 7 hours for $n = 18$ (using 132 threads), and about 6 days for $n = 19$ (using 172 threads). This means that, if we run the code on a single thread, then the computation would take approximately $10^3$ days for $n = 19$. The total numbers of traversed nodes are 43, 235, 268, 208, 065 for $n = 18$ and 933, 351, 108, 741, 643 for $n = 19$, respectively. The ratios to the number of nodes in the search tree without pruning, i.e., $\sum_{i=0}^{n-1} \prod_{j=1}^{i} (n - f)$, are 4.47% and 5.36%, respectively. The breakdown of the number of traversed nodes for $n = 19$ with respect to the levels of the tree is shown in Table 1.

By examining the result, we conclude that $f(18) = 191$ and $f(19) = 221$. The largest initial deck for $n = 18$ is unique. It is

$$(6 14 9 2 15 8 1 3 4 12 18 5 10 13 16 17 11 7),$$

which terminates at the sorted position (1 2 3 . . . 18). There are four largest initial decks for $n = 19$. These are

$$(9 4 19 17 10 1 11 15 12 8 5 2 18 13 16 7 3 14 6),$$
$$(12 15 11 1 10 17 19 2 5 8 9 4 18 13 16 7 3 14 6),$$
$$(12 1 18 11 3 14 2 6 8 16 5 4 15 10 13 17 19 7 9),$$
$$(12 1 18 11 2 3 14 6 8 16 5 4 15 10 13 17 19 7 9).$$

Interestingly, all these decks terminate at a same non-sorted position (1 10 9 8 7 6 5 4 3 2 11 12 13 14 15 16 17 18 19). The largest initial deck that terminates at the sorted position is known to take 209 steps (see A000376 of OEIS [5]), which is twelve less than the value of $f(19)$.

### Acknowledgements

This work was partially supported by JSPS Kakenhi Grant Numbers 18K11152 and 18H04090.
Table 1: The number of traversed nodes for $n = 19$.

| Level | # of traversed nodes | Level | # of traversed nodes |
|-------|-----------------------|-------|-----------------------|
| 0     | 1                     | 10    | 46335514956           |
| 1     | 17                    | 11    | 304773283939          |
| 2     | 272                   | 12    | 1716889839183         |
| 3     | 3952                  | 13    | 8059154346527         |
| 4     | 52861                 | 14    | 30428256670076        |
| 5     | 653126                | 15    | 89242470628183        |
| 6     | 7419100               | 16    | 200111553921243       |
| 7     | 77075852              | 17    | 326581145735086       |
| 8     | 726678384             | 18    | 276853558861087       |
| 9     | 6158057798            |       |                       |
| 10    | 46335514956           |       |                       |

References

[1] D. Berman, M. S. Klamkin and D. E. Knuth, Problem 76-17. A reverse card shuffle, SIAM Review 19, pp. 739–741 (1977)

[2] D.E. Knuth, The Art of Computer Programming Volume 4 Fascicle 2, Addison-Wesley Prof., pp. 119 (2005)

[3] D.E. Knuth, “topswops-fwd.w” (the source code of a “better” algorithm), https://www-cs-faculty.stanford.edu/~knuth/programs/topswops-fwd.w (accessed Mar. 3, 2021)

[4] L. Morales, H. Sudborough, A quadratic lower bound for Topswops, Theoretical Computer Science, Vol. 411, pp. 3965–3970 (2010)

[5] OEIS Foundation Inc., The On-Line Encyclopedia of Integer Sequences, http://oeis.org (2021)