k-Pop Stack Sortable Permutations

Yoong Kuan Goh (Andrew)

University of Technology Sydney

Permutation Patterns 2020

Pop stack

Pop stack

A pop stack is a sorting device which operates with two operations

1. **Push**: Move a token from the input stream to the top of the stack

Pop stack

Pop stack

A pop stack is a sorting device which operates with two operations

1. **Push**: Move a token from the input stream to the top of the stack
2. **Pop**: Remove the entire stack contents from top to bottom to the output stream.

A *deterministic* pop stack always performs the push move unless the token on the top of the stack is smaller in value than the token to be pushed from the input stream.

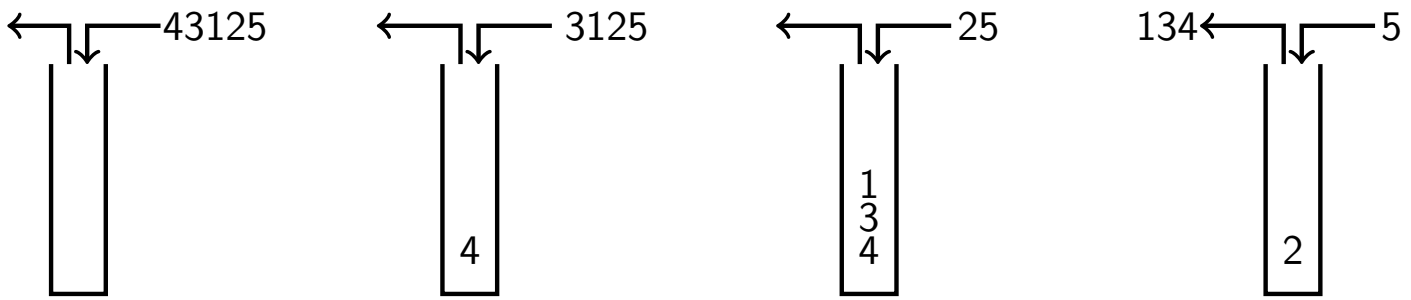
Pop stack

Pop stack

A pop stack is a sorting device which operates with two operations

1. **Push**: Move a token from the input stream to the top of the stack
2. **Pop**: Remove the entire stack contents from top to bottom to the output stream.

A *deterministic* pop stack always performs the push move unless the token on the top of the stack is smaller in value than the token to be pushed from the input stream.
Eg: input 43125



Pop stack

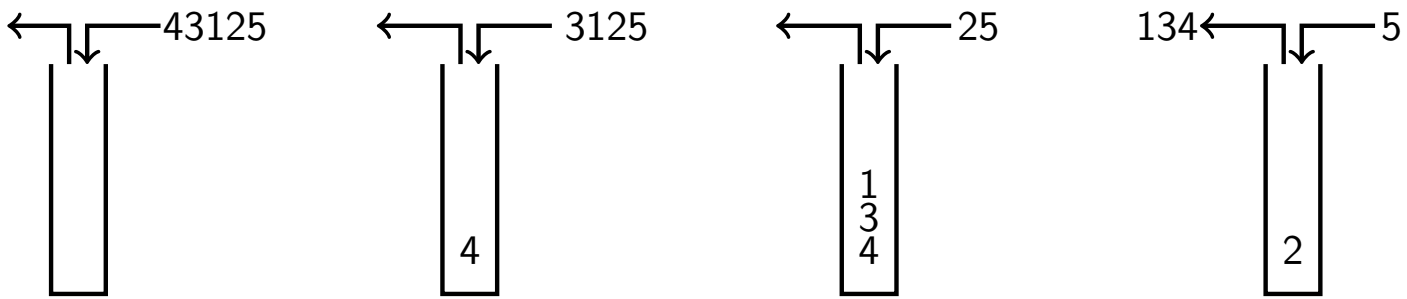
Pop stack

A pop stack is a sorting device which operates with two operations

1. **Push**: Move a token from the input stream to the top of the stack
2. **Pop**: Remove the entire stack contents from top to bottom to the output stream.

A *deterministic* pop stack always performs the push move unless the token on the top of the stack is smaller in value than the token to be pushed from the input stream.

Eg: input 43125



output: 13425

k-pop stack sortable permutation

A permutation is *k-pop stack sortable* if it can be sorted by passing it through a deterministic pop stack k times.

k-pop stack sortable permutation

A permutation is *k-pop stack sortable* if it can be sorted by passing it through a deterministic pop stack k times.

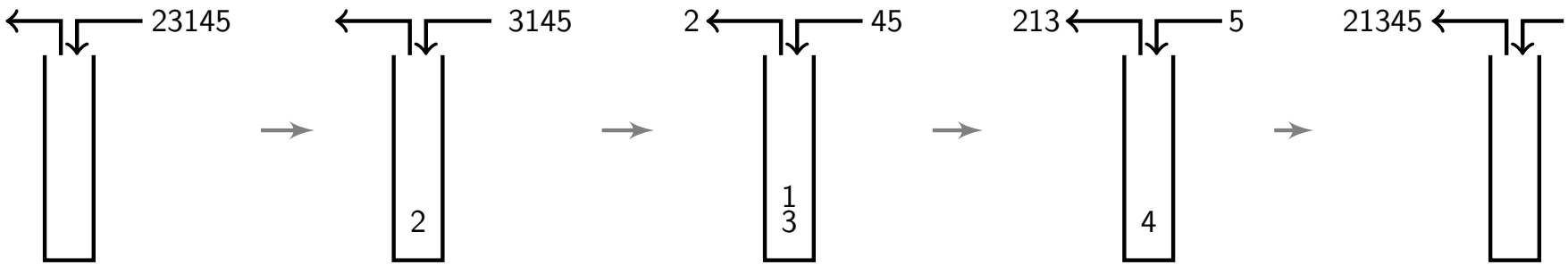
Example: Given 23145, it is 2-pop stack sortable?

k-pop stack sortable permutation

A permutation is *k-pop stack sortable* if it can be sorted by passing it through a deterministic pop stack k times.

Example: Given 23145, it is 2-pop stack sortable?

1-pass:

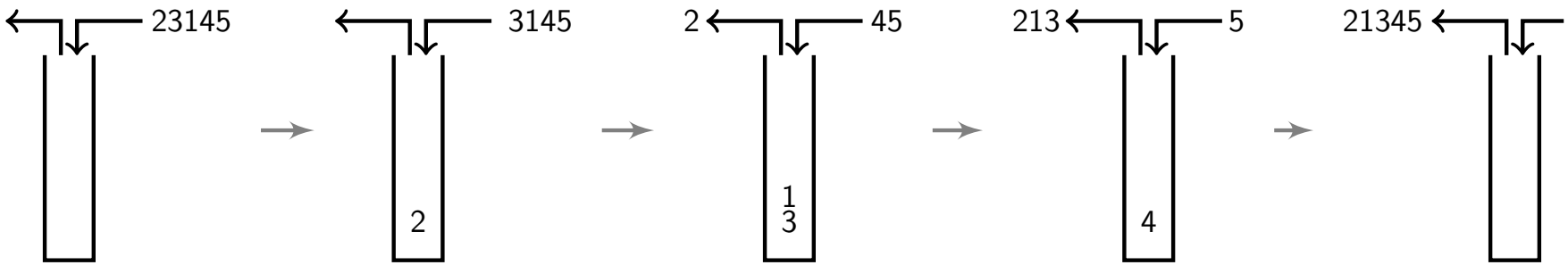


k-pop stack sortable permutation

A permutation is *k-pop stack sortable* if it can be sorted by passing it through a deterministic pop stack k times.

Example: Given 23145, it is 2-pop stack sortable?

1-pass:



2-pass:

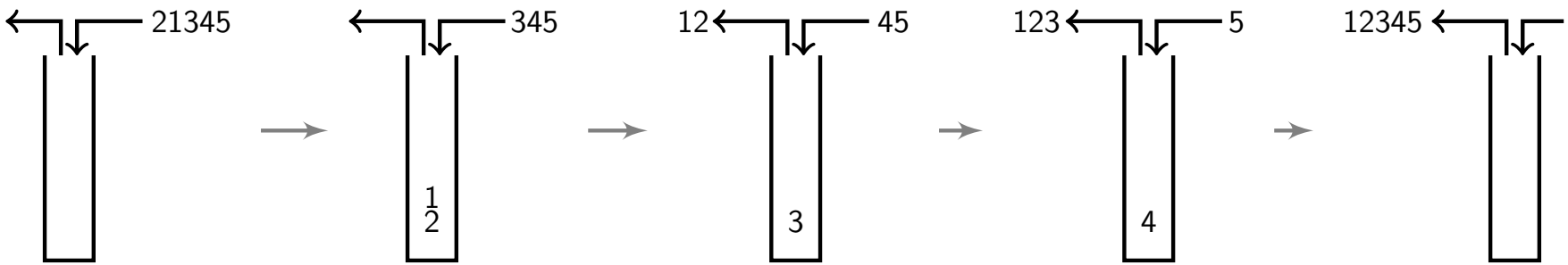


Figure: Sorting 23145 with a 2-pass pop stack

Background

In 1981, Avis and Newborn characterised permutations sortable by 1-pop stack, initiating the study of pop stack sorting.

Background

In 1981, Avis and Newborn characterised permutations sortable by 1-pop stack, initiating the study of pop stack sorting.

Later in 2018, Pudwell and Smith [4] characterised permutations sorted by 2-pop stack and computed a rational generating function for the number of such permutations.

Background

In 1981, Avis and Newborn characterised permutations sortable by 1-pop stack, initiating the study of pop stack sorting.

Later in 2018, Pudwell and Smith [4] characterised permutations sorted by 2-pop stack and computed a rational generating function for the number of such permutations.

Claesson and Guðmundsson [2] then computed a rational generating function for permutations sorted by any finite number of passes and asked the following question:

Background

In 1981, Avis and Newborn characterised permutations sortable by 1-pop stack, initiating the study of pop stack sorting.

Later in 2018, Pudwell and Smith [4] characterised permutations sorted by 2-pop stack and computed a rational generating function for the number of such permutations.

Claesson and Guðmundsson [2] then computed a rational generating function for permutations sorted by any finite number of passes and asked the following question:

Question (Claesson and Guðmundsson [2])

Is there a "useful permutation pattern characterization of the k-pop stack-sortable permutations?"

Some notation

1. $\gamma <_{\text{subperm}} \alpha$: γ is a subpermutation of α.
2. red(γ) : permutation obtained by replacing the tth smallest entry of γ by the integer i.
3. Let σ be a permutation. Call a factor $B_i = a_{i,1}a_{i,2}\ldots a_{i,n_i}$ of σ a **block** if $n_i > 0$ and $a_{i,j} > a_{i,j+1}$ for all $1 \leqslant j < n_i$. (a factor means the entries are contiguous in σ.)

Result: 2-avoidance

In order to provide such a characterization, the current notions of bar avoidance would not suffice. Therefore, we introduce a new notion called 2-*avoidance* which was defined and explained in Murray Elder's talk.

Result: 2-avoidance

In order to provide such a characterization, the current notions of bar avoidance would not suffice. Therefore, we introduce a new notion called 2-*avoidance* which was defined and explained in Murray Elder's talk.

2-avoidance

Let σ be a permutation and $F, G \subseteq S^\infty$ be two sets of reduced permutations. We say that σ 2-*avoids* (F, G) if for all $\gamma <_{\text{subperm}} \sigma$, if $\text{red}(\gamma) \in F$ then there exists $\delta <_{\text{subperm}} \sigma$ such that $\gamma <_{\text{subperm}} \delta$ and $\text{red}(\delta) \in G$.

We denote the set of all permutations in S^∞ which 2-avoid (F, G) by $\text{Av}_2(F, G)$.

Result: k-pass pop stack sortable

With this notion, proved that k-pop stack-sortable permutations are characterised by a **finite** list of forbidden patterns for all $k \in \mathbb{N}$.

Result: k-pass pop stack sortable

With this notion, proved that k-pop stack-sortable permutations are characterised by a **finite** list of forbidden patterns for all $k \in \mathbb{N}$.

Theorem (E, Goh)

Let $k \in \mathbb{N}_+$. There exists a pair of finite sets (F_k, G_k) such that the set of all k-pass pop stack sortable permutations is equal to $\mathrm{Av}_2(F_k, G_k)$. Moreover, the sets F_k, G_k can be algorithmically constructed.

Let S_k denote the set of all k-pass pop stack sortable permutations. We proceed by induction, with the base case $k = 1$ established by Avis and Newborn [1] (specifically, $F_1 = \{231, 312\}, G_1 = \emptyset$).

Let S_k denote the set of all k-pass pop stack sortable permutations. We proceed by induction, with the base case $k = 1$ established by Avis and Newborn [1] (specifically, $F_1 = \{231, 312\}, G_1 = \emptyset$).

Assume F_{k-1}, G_{k-1} have been constructed, are finite, and $S_{k-1} = \mathrm{Av}_2(F_{k-1}, G_{k-1})$.

Let S_k denote the set of all k-pass pop stack sortable permutations. We proceed by induction, with the base case $k = 1$ established by Avis and Newborn [1] (specifically, $F_1 = \{231, 312\}, G_1 = \emptyset$).

Assume F_{k-1}, G_{k-1} have been constructed, are finite, and $S_{k-1} = \mathrm{Av}_2(F_{k-1}, G_{k-1})$.

Let $f_{\max} = \max\{|\beta| \mid \beta \in F_{k-1}\}$ and $C = 3^{k+2} f_{\max}$. Then define

$$\begin{aligned} \Omega_1 &= \{\tau \in S^\infty \mid |\tau| \leqslant 3f_{\max}, \tau \notin S_k\}, \\ \Omega_2 &= \{\kappa \in S^\infty \mid |\kappa| \leqslant C, \kappa \in S_k, \exists \tau \in \Omega_1[\tau \leqslant \kappa]\}. \end{aligned}$$

1. Claim 1: Ω_i are both finite: both are subsets of the set of all permutations of length at most C.
2. Claim 2: Ω_i are algorithmically constructible: There is only finitely many τ, κ of length at most $3f_{\max}, C$ respectively.
3. Claim 3: $\sigma \notin S_k$ if and only if σ 2-contains (Ω_1, Ω_2): this is to check the closure properties such that everything not sortable by k-pop stack must 2-contains (Ω_1, Ω_2).
    1. We proved that $p_1(\sigma)$ 2-contains (F_{k-1}, G_{k-1}) which implies $p_1(\sigma) \notin S_{k-1}$ which implies $\sigma \notin S_k$.
    2. For contradiction, we assume $\sigma \in S_k$. Then, we proved that $\kappa <_{\text{subperm}} \sigma$ such that $\kappa \in S_k, \gamma <_{\text{subperm}} \kappa$ and $|\kappa| \leqslant C$ can be constructed. After that, can construct $\alpha \in \Omega_2$ with $\alpha \sim \kappa$ and $\text{red}(\gamma) \leqslant \alpha$, which means α saves[1] $\gamma <_{\text{subperm}} \sigma$, and this gives a contradiction that σ 2-contains (Ω_1, Ω_2) because of γ.

[1] refer to Murray' talk

Construction of κ

1. Starting with $\kappa{=}\sigma$, mark the tokens corresponding to γ **bold**.
2. Delete non-bold tokens given that no blocks in k are merged.
3. At the end of this process, each block in k contains at most two non-bold entries. See Figure 2.

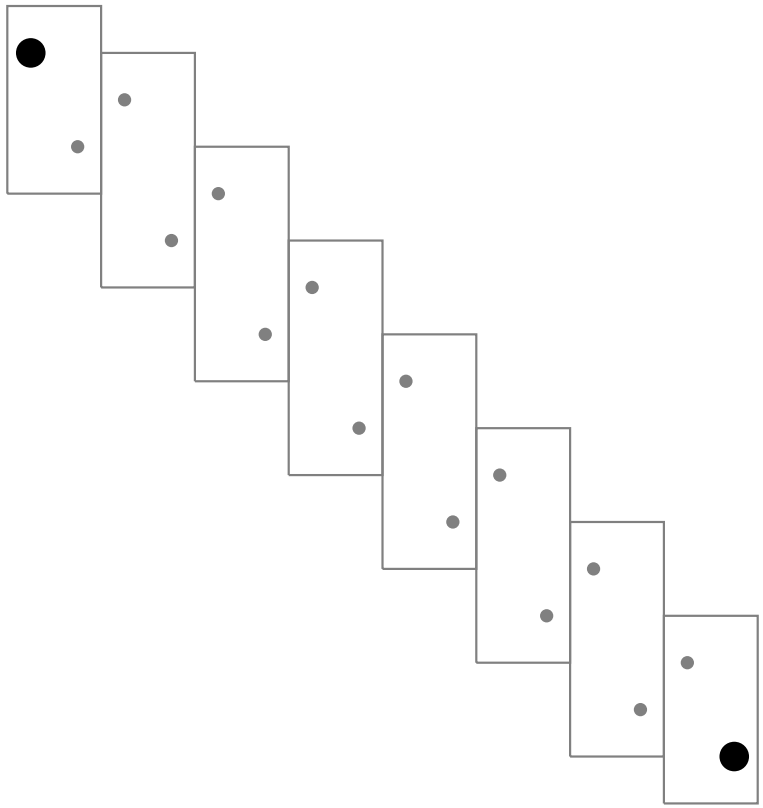


Figure: $\kappa = \mathbf{16}, 14, 15, 12, 13, 10, 11, 896745231$

Lemma

Let σ be a permutation with block decomposition $B_1 B_2 B_3 \dots B_m$, $a \in B_{i+1}, b \in B_{i+n}$ two entries of σ with $a > b$, $n \geqslant 1$. (See for example Figure 2). If $n \geqslant 3^k$ then σ is not k-pass pop stack sortable.

With this lemma, κ cannot be sorted by k-pop stacks when the number of blocks between the first and last block is $\geqslant k$

Lemma

Let σ be a permutation with block decomposition $B_1 B_2 B_3 \dots B_m$, $a \in B_{i+1}, b \in B_{i+n}$ two entries of σ with $a > b$, $n \geqslant 1$. (See for example Figure 2). If $n \geqslant 3^k$ then σ is not k-pass pop stack sortable.

With this lemma, κ cannot be sorted by k-pop stacks when the number of blocks between the first and last block is $\geqslant k$

If κ cannot be sorted, then $p_1(\kappa) \notin S_{k-1}$ because of some subpermutation τ with $\mathrm{red}(\tau) \in F_{k-1}$, but since no block has merged in obtaining $p_1(\kappa)$, $p_1(\sigma)$ also contains τ which cannot be saved since blocks containing the tokens forming τ are fixed. Thus $p_1(\sigma) \notin S_{k-1}$, so $\sigma \notin S_k$, contradiction.

Lemma

Let σ be a permutation with block decomposition $B_1 B_2 B_3 \dots B_m$, $a \in B_{i+1}, b \in B_{i+n}$ two entries of σ with $a > b$, $n \geqslant 1$. (See for example Figure 2). If $n \geqslant 3^k$ then σ is not k-pass pop stack sortable.

With this lemma, κ cannot be sorted by k-pop stacks when the number of blocks between the first and last block is $\geqslant k$

If κ cannot be sorted, then $p_1(\kappa) \notin S_{k-1}$ because of some subpermutation τ with $\mathrm{red}(\tau) \in F_{k-1}$, but since no block has merged in obtaining $p_1(\kappa)$, $p_1(\sigma)$ also contains τ which cannot be saved since blocks containing the tokens forming τ are fixed. Thus $p_1(\sigma) \notin S_{k-1}$, so $\sigma \notin S_k$, contradiction.

Thus we have $|\kappa|$ is at most $3|\gamma|$ (the bold blocks with a non-bold entry first and last) plus $2.3^k|\gamma|$ (3^k factors each containing 2 tokens, as in Figure 2).

Lemma

Let σ be a permutation with block decomposition $B_1 B_2 B_3 \dots B_m$, $a \in B_{i+1}, b \in B_{i+n}$ two entries of σ with $a > b$, $n \geqslant 1$. (See for example Figure 2). If $n \geqslant 3^k$ then σ is not k-pass pop stack sortable.

With this lemma, κ cannot be sorted by k-pop stacks when the number of blocks between the first and last block is $\geqslant k$

If κ cannot be sorted, then $p_1(\kappa) \notin S_{k-1}$ because of some subpermutation τ with $\mathrm{red}(\tau) \in F_{k-1}$, but since no block has merged in obtaining $p_1(\kappa)$, $p_1(\sigma)$ also contains τ which cannot be saved since blocks containing the tokens forming τ are fixed. Thus $p_1(\sigma) \notin S_{k-1}$, so $\sigma \notin S_k$, contradiction.

Thus we have $|\kappa|$ is at most $3|\gamma|$ (the bold blocks with a non-bold entry first and last) plus $2.3^k|\gamma|$ (3^k factors each containing 2 tokens, as in Figure 2).

Worst case:

$$|\kappa| \leqslant (3 + 2.3^k)|\gamma| \leqslant (3 + 2.3^k)3f_{\max} \leqslant 3^{k+2} f_{\max} = C$$

Conclusion

Our proof shows that pattern characterization of k-pop stack sortable permutation is finite, but does not give the smallest sets of patterns. This is because the number of elements to be checked is $C!$ (upper bound). Recall that $C = 3^{k+2} f_{\max}$.

Conclusion

Our proof shows that pattern characterization of k-pop stack sortable permutation is finite, but does not give the smallest sets of patterns. This is because the number of elements to be checked is $C!$ (upper bound). Recall that $C = 3^{k+2} f_{\max}$.

For example, assume $f_{\max} = 5$ and $k = 3$, then $C = 1215$. This means the number of elements to be checked is 1215!

Conclusion

Our proof shows that pattern characterization of k-pop stack sortable permutation is finite, but does not give the smallest sets of patterns. This is because the number of elements to be checked is $C!$ (upper bound). Recall that $C = 3^{k+2} f_{\max}$.

For example, assume $f_{\max} = 5$ and $k = 3$, then $C = 1215$. This means the number of elements to be checked is 1215!

However, this upper bound can certainly be lowered with some lemmas. The detail of the lemmas can be seen in this paper [3].

Thank you!

References

D. Avis and M. Newborn.

On pop-stacks in series.

Utilitas Math., 19:129–140, 1981.

A. Claesson and B. Ágúst Guðmundsson.

Enumerating permutations sortable by k passes through a pop-stack.

Sém. Lothar. Combin., 80B:Art. 43, 12, 2018.

M. Elder and Y. K. Goh.

k-pop stack sortable permutations and 2-avoidance.

arXiv e-prints, Nov. 2019.

L. Pudwell and R. Smith.

Two-stack-sorting with pop stacks.

Australasian Journal of Combinatorics, 74(1):179 – 195, 2019.