State machine bounds on pattern avoiding permutations

Atli Fannar Franklín

University of Iceland

June 18, 2025

What is a permutation pattern?

- We say that $a_1, \ldots, a_m$ and $b_1, \ldots, b_m$ are order-isomorphic if $a_i < a_j$ if and only if $b_i < b_j$ for all i, j.
- For permutations $\pi = \pi_1 \ldots \pi_n$ and $\tau = \tau_1 \ldots \tau_m$ we say that π contains the pattern τ if there exists a set of indices $i_1 < \ldots < i_m$ such that $\pi_{i_1}, \ldots, \pi_{i_m}$ is order-isomorphic to τ.

Motivation

- Denote the set of all permutations of size n avoiding the pattern τ by $\mathrm{Av}_n(\tau)$.
- The Stanley-Wilf conjecture says for any pattern τ the Stanley-Wilf limit $\lim_{n\to\infty} \sqrt[n]{|\mathrm{Av}_n(\tau)|}$ exists for any τ.
- This limit is 4 for all patterns of length 3, but even just for length four it is not known in all cases. The known ones have limits 8 and 9, but the limit for 1324 is not known.
- Determining much of anything about 1324-avoiding permutations seems hard.

Motivation

- Tightening bounds on the growth rate of them would be good, so perhaps a different way to enumerate could help.
- Currently the best known lower bound is 10.271 and best known upper bound 13.5.
- We will look at one possible avenue that could lead to improvements on the lower bound.

Framework

- We could imagine constructing a pattern avoiding permutation one element at a time by always inserting a new maximum element at some point in the sequence.
- Then each permutation corresponds to a sequence of moves made.
- This is very similar to the situation when enumerating words that a finite state machine accepts, so let us define finite state machines.

State machines

- Informally we have some alphabet and a set of states. For each state we have to define what state we move to upon reading each symbol of the alphabet. Some state is designated as the initial state, and some subset of them is accepting.
- Then a word, a sequence of symbols, is accepted iff the last state visited is accepting.
- We will allow ourselves to omit transitions and immediately reject a word if it needs to use an omitted transition.

Usefulness

- The number of words accepted by such a machine can be read out from its adjacency matrix.
- These matrices have entries ≥ 0, so to analyse the growth rate of the number of words of length n we can often use the Perron-Frobenius theorem.
- If all states can reach all other states, the theorem applies.

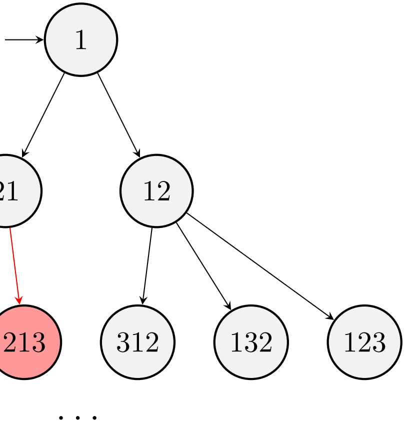
Perron-Frobenius theorem

- The Perron-Frobenius theorem says our adjacency matrix A will have a unique eigenvalue λ of largest magnitude and $\lambda \in \mathbb{R}$.
- By considering determinants it can be shown that the number of accepted words will then grow like $\sim \lambda^n$.
- Furthermore the Collatz-Wielandt formula states that if we have a vector $v \geq 0$ then $\lambda \geq \min_i (Av)_i / v_i$.
- So if we can represent our problem as a state machine like this, then find a vector v such that $Av \geq \lambda v$ then λ is a lower bound for the Stanley-Wilf limit.

Example

- To highlight how this all functions, let us work through finding a lower bound for the Stanley-Wilf limit of 213-avoiding permutations.
- We have an initial state representing a permutation on one element. Our transitions are inserting a new maximum element.

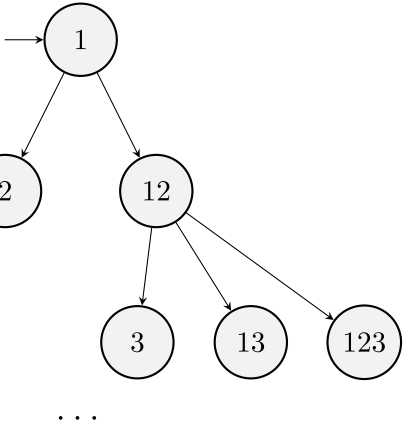
Bad example



Improving the example

- This machine will be much too large. What we can note however is that inserting a maximum element after an inversion will produce an instance of the pattern 213.
- Thus for the purposes of this machine, only the initial increasing run has to be preserved, and the rest can be thrown out.

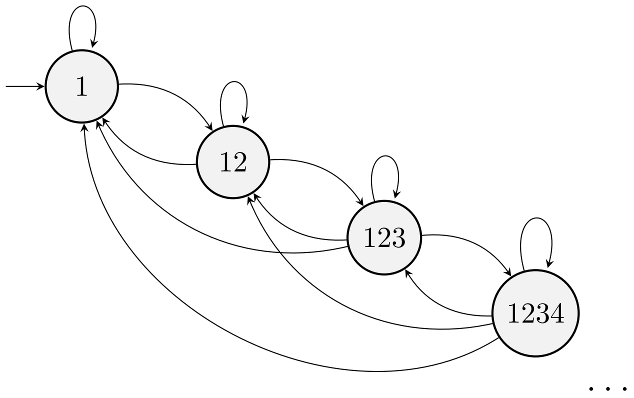
Still bad example



Improving the example

- We can merge identical nodes, but furthermore we only care about relative order. The state 23 behaves exactly the same as 12, as the newly inserted maximum will be larger than all previous values anyway.
- This means we can canonicalise the states we end up in to remove even more states.

Better example



Bounds from example

- We have to cut off at some size n, which still gives a lower bound on the true growth rate. But if we prove the bound for an arbitrary n, we can take the limit after.
- Assign weights $1, 1, 2/3, (2/3)^2, \ldots$ to the states $1, 12, 123, 1234, \ldots$.
- The sum of weights pointing into any given state will then be exactly four times its weight minus $2^{n-1}/3^n$. As $n \to \infty$ this means the bound tends to 4, so the Stanley-Wilf limit of 213 avoiding permutations is ≥ 4.

Finding the vector?

- The weights $1, 1, 2/3, (2/3)^2, \ldots$ may seem pulled from thin air, but they are easy to approximate.
- They will be similar to the dominant eigenvector of our adjacency matrix for a given n, getting closer for larger n. The Perron-Frobenius theorem assures us that if we start with any $v \geq 0$ and iteratively replace v by $Av/\left\|Av\right\|$, this will tend to the eigenvector.
- Thus we generate weights for successive n and guess, e.g. for $n = 8$ we get

```
0.276397, 0.276397, 0.200001, 0.123606,

0.068326, 0.0341619, 0.0152771, 0.00583506
```

How does this generalise?

- This same idea works for any pattern $\tau_1\tau_2\ldots\tau_n$ where $\tau_n = n$.
- Our states will be $\bigcup_{k=1}^{n} \mathrm{Av}_k(\tau_1\tau_2\ldots\tau_{n-1})$ for some n (up to minor modifications possibly, see later).
- From the state π we insert $|\pi|+1$ into π in all possible positions. We remove the final element of those permutations until they avoid $\tau_1\tau_2\ldots\tau_{n-1}$. Lastly we put them in canonical form and remove any permutations of size $> n$. This set of permutations are then the permutations π can transition to.

Attempting a length four pattern

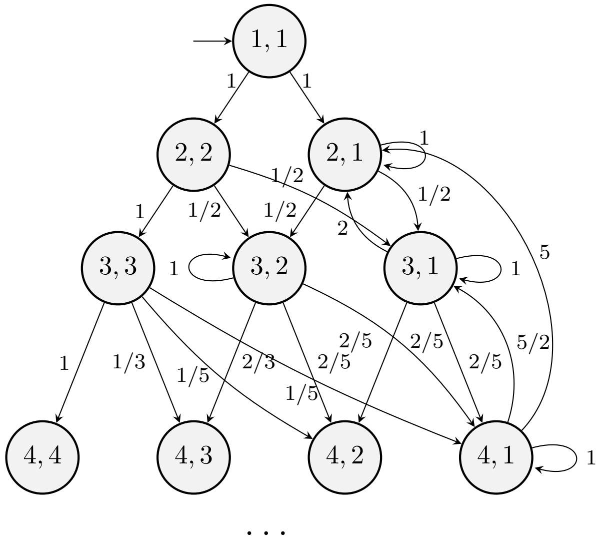
- So why would this be better than other computational lower bounds attempted before?
- It seems more amenable to mathematical simplification to improve runtime. Just like we were able to solve the 213 case fully mathematically, we will now solve the 2134 case well enough to get within 1.1% of the true value computationally.
- Our machine will have states in $\bigcup_{k=1}^{n} \mathrm{Av}_k(213)$.

Transitions on $\mathrm{Av}(213)$

- What happens when we insert a maximum into a 213-avoiding permutation?
- That insertion can happen in an initial increasing run, increasing the length of that run and the permutation by 1.
- Otherwise the insertion happens after a descent so we will have to delete everything after the inserted element, and then the inserted element as well. This decreases the size of the permutation but leaves the initial increasing run unchanged.

Transitions on $\mathrm{Av}(213)$

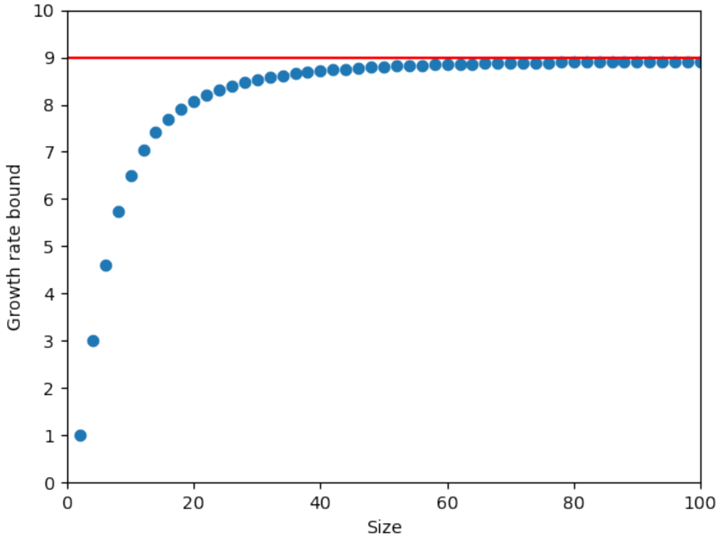
- Denote the subset of $\mathrm{Av}_n(213)$ with initial increasing run of length r by $A(n,r)$. One can show $|A(n,r)| = \frac{n-r}{n}\binom{n+r-1}{r}$.
- For each $\pi \in A(n,r)$ we have one transition to $A(n+1,x)$ for each $1 \leq x \leq r+1$ and one transition to $A(x,r)$ for each $r < x \leq n$.
- In sum this means we can group each set $A(n,r)$ together computationally to speed things, only simulating quadratically many states.


1, 1
2, 2
2, 1
3, 3
3, 2
3, 1
4, 4
4, 3
4, 2
4, 1
1
1
1
1/2
1/2
1/2
1/2
2
5
1
1
1
1
1/3
1/5
2/3
2/5
1/5
2/5
2/5
2/5
5/2
1
. . .

Minor fixes

- You may have noticed $2, 1$ can not reach the state x, x for any x. Every state can reach $2, 1$ however. The states x, x correspond to increasing permutations, so it does not affect the growth rate if we simply delete them.
- The following plot shows the growth rate calculated from the earlier state machine for different n, omitting increasing permutations.

Growth



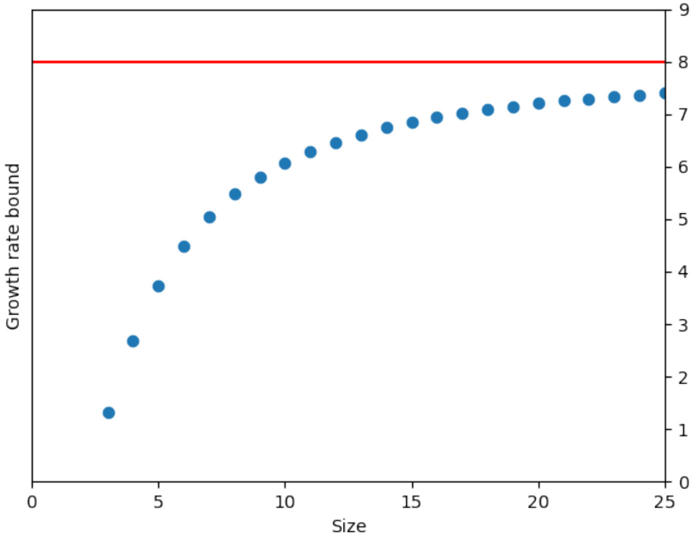
2134 bound

- This gives a bound of 8.90845 for 2134. One could go quite a bit further than $n = 100$, it simply starts to introduce numerical precision issues. In terms of raw speed, my laptop can handle $n = 500$ in under a minute.
- Let's look at another example.

3124 bound

- A similar, but not as good, method can be applied to get bounds for 3124.
- Instead of grouping by size and initial increasing run, we group by the descent set.
- This has 2^n states, which is a large improvement over 4^n, but still bad. For $n = 25$, which takes a few hours on a personal computer, we get the bound 7.40341.

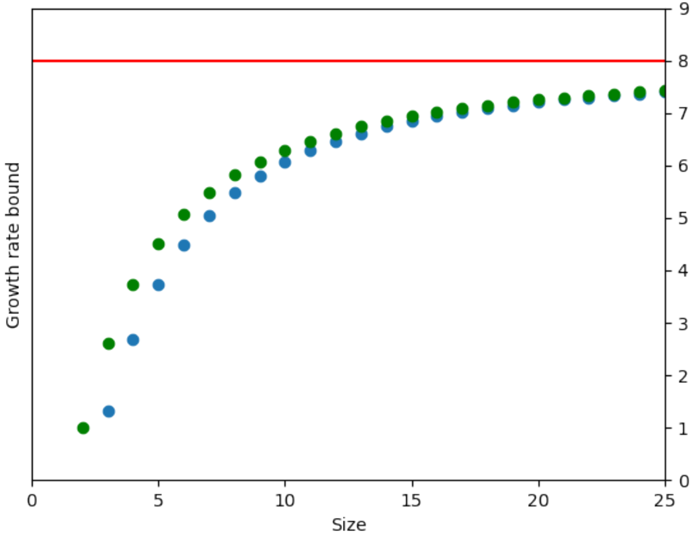
3124 bound



Minor improvements

- One can also implement various minor tricks, for example injecting paths that would lead to larger elements back into the state machine.
- These kinds of tricks (seem to) usually only really improve the bounds much for lower n and the gain quickly decays to nothing.

Minor improvements (7.43254)



1324 bound

- It is tricky to find statistics for the 132 state machine that allow us to recover the original path.
- Without grouping, the bound is not useful. The $n = 18$ case gives a bound of 8.326, which required a machine with $\sim 650M$ states.
- Audience suggestions are most welcome.
- Determining a good enough eigenvector directly like in the small case is also possible, in theory.

Thanks for listening!