Permutation Patterns and Object moving environments

Michael Albert

Department of Computer Science, University of Otago

malbert@cs.otago.ac.nz

Rearranging with a stack

A sequence of input items are processed through a stack onto an output queue. In what orders can/can't they emerge?

An obvious problem comes in trying to convert input items

$$\cdots a \cdots b \cdots c \cdots$$

to output

$$\cdots c \cdots a \cdots b \cdots \ .$$

312-avoidance

Definition: *A permutation*

$$\pi = \pi_1 \pi_2 \cdots \pi_n$$

contains the pattern 312, if, for some $i < j < k$, $\pi_j < \pi_k < \pi_i$.

Proposition: *(Knuth, ∼1970) The permutations of an input sequence which can be generated by a single stack are exactly those that avoid the pattern 312.*

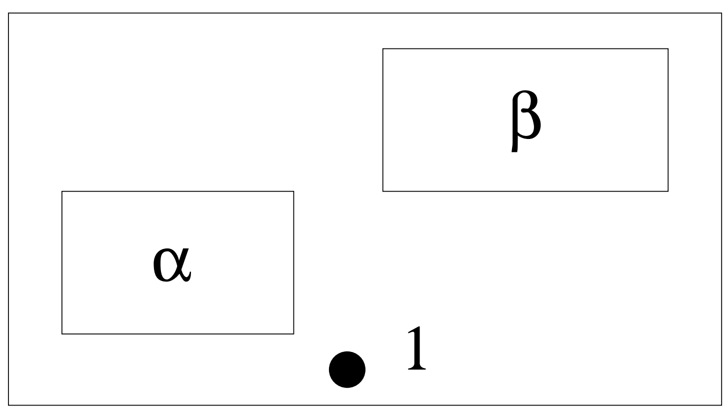
Enumeration of 312 avoiders

Consider the push-pop operation sequence of a stack in producing a 312-avoider. This provides a bijection between 312-avoiders of length n and balanced bracket sequences with n pairs of brackets. Therefore the number of such is given by the Catalan numbers:

$$\frac{1}{n+1}\binom{2n}{n}.$$

Alternatively, a bijection with binary trees by considering:

$\pi =$



α, β avoid 312

The research frontier

- Note that Knuth’s result also gives a linear time algorithm for recognizing a 312-avoider. Just run the stack and see if it works.

The research frontier

- Note that Knuth's result also gives a linear time algorithm for recognizing a 312-avoider. Just run the stack and see if it works.
- Given a permutation, determine whether it can be generated by two stacks in series.

The research frontier

- Note that Knuth's result also gives a linear time algorithm for recognizing a 312-avoider. Just run the stack and see if it works.
- Given a permutation, determine whether it can be generated by two stacks in series.
- Essentially nothing is known about this.

The research frontier

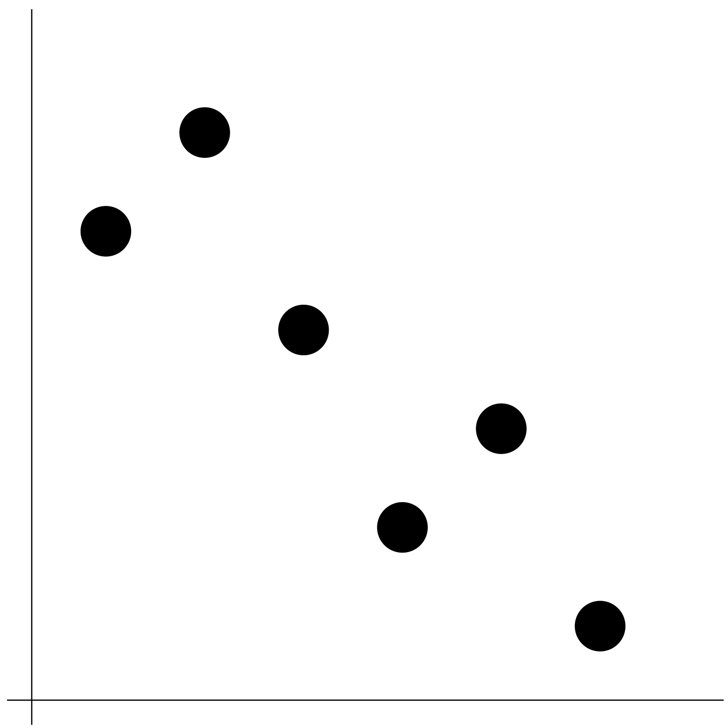
- Note that Knuth's result also gives a linear time algorithm for recognizing a 312-avoider. Just run the stack and see if it works.
- Given a permutation, determine whether it can be generated by two stacks in series.
- Essentially nothing is known about this.
- It is known, that there are infinitely many permutations which cannot be generated by two stacks in series, but which have the property that the deletion of any single element produces a permutation which can be generated.

Involvement

Definition: *A permutation σ is involved in a permutation π ($\sigma \preceq \pi$) if some subsequence of π has the same relative ordering as all of σ.*

Involvement

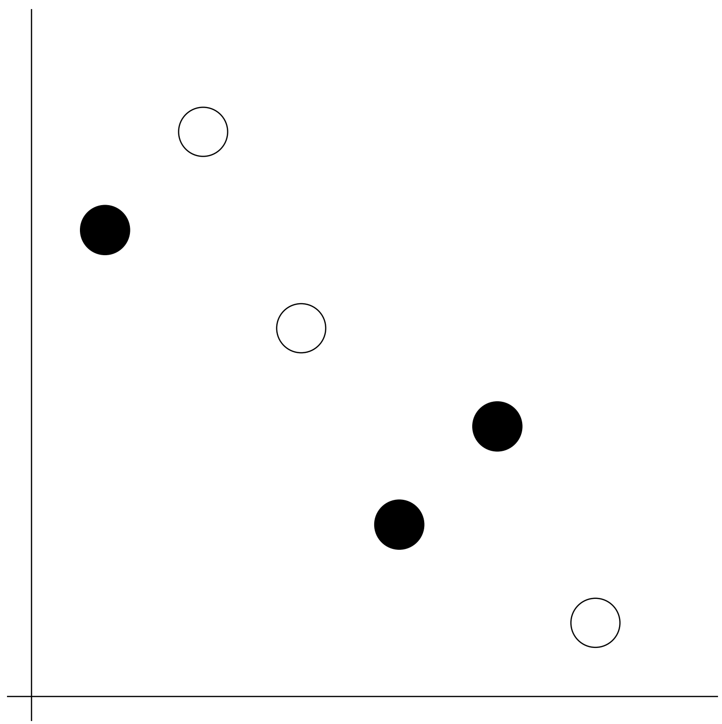
Definition: *A permutation σ is* *involved* *in a permutation π ($\sigma \preceq \pi$) if some subsequence of π has the same relative ordering as all of σ.*



5 6 4 2 3 1

Involvement

Definition: *A permutation σ is involved in a permutation π ($\sigma \preceq \pi$) if some subsequence of π has the same relative ordering as all of σ.*



5 6 4 2 3 1

involves

3 1 2

Pattern Classes

- A pattern class, $\mathcal{C}$, is a collection of permutations closed downwards under the involvement relation.
- The minimal permutations (if any) not belonging to $\mathcal{C}$ are called its basis.

Note that the basis of a pattern class is an antichain with respect to the involvement ordering. Conversely, given any such antichain, $\mathcal{A}$, we can define the pattern class of which this is the basis. It consists of all those permutations that do not involve any member of $\mathcal{A}$.

Examples

- The permutations which we can generate from $12 \cdots n$ by a stack (basis $\{312\})$
- The permutations which we can generate from $12 \cdots n$ by two parallel queues (basis $\{321\}$).
- The permutations which we can generate from $12 \cdots n$ by a “riffle shuffle” (basis $\{321, 2143, 2413\}$).
- The permutations whose graphs can be decomposed (recursively) into high-low, or low-high blocks (basis $\{2413, 3142\}$).

Questions

Basis Problem Given a pattern class $\mathcal{C}$ determine its basis. Is it finite? How many elements of size n does it contain?

Membership problem Is there an algorithm for deciding membership in a given pattern class? Is there an *efficient* algorithm?

Enumeration Problem Given a pattern class, determine how many permutations of length n it contains.

Reflection

- The questions are interesting but reflect a certain *ad hoc* approach.
- Perhaps of greater interest would be results pertaining to what the answers to those questions could be.
- For example, what sort of growth rates/generating functions can proper pattern classes have?

Wilf-Stanley

Conjecture: If $\mathcal{C}$ is a proper pattern class, then for some constant q:

$$\lim_{n\to\infty} |\mathcal{C} \cap S_n|^{1/n} = q.$$

Wilf-Stanley

Conjecture: If $\mathcal{C}$ is a proper pattern class, then for some constant q:

$$\lim_{n\to\infty} |\mathcal{C} \cap S_n|^{1/n} = q.$$

Theorem: *(Alon, Friedgut 2000) If $\mathcal{C}$ is a proper pattern class, then there exists a constant q such that for all n*

$$|\mathcal{C} \cap S_n| \leq q^{n\gamma(n)}$$

where γ is a very slowly growing function.

Wilf-Stanley

Conjecture: If $\mathcal{C}$ is a proper pattern class, then for some constant q:

$$\lim_{n\to\infty} |\mathcal{C} \cap S_n|^{1/n} = q.$$

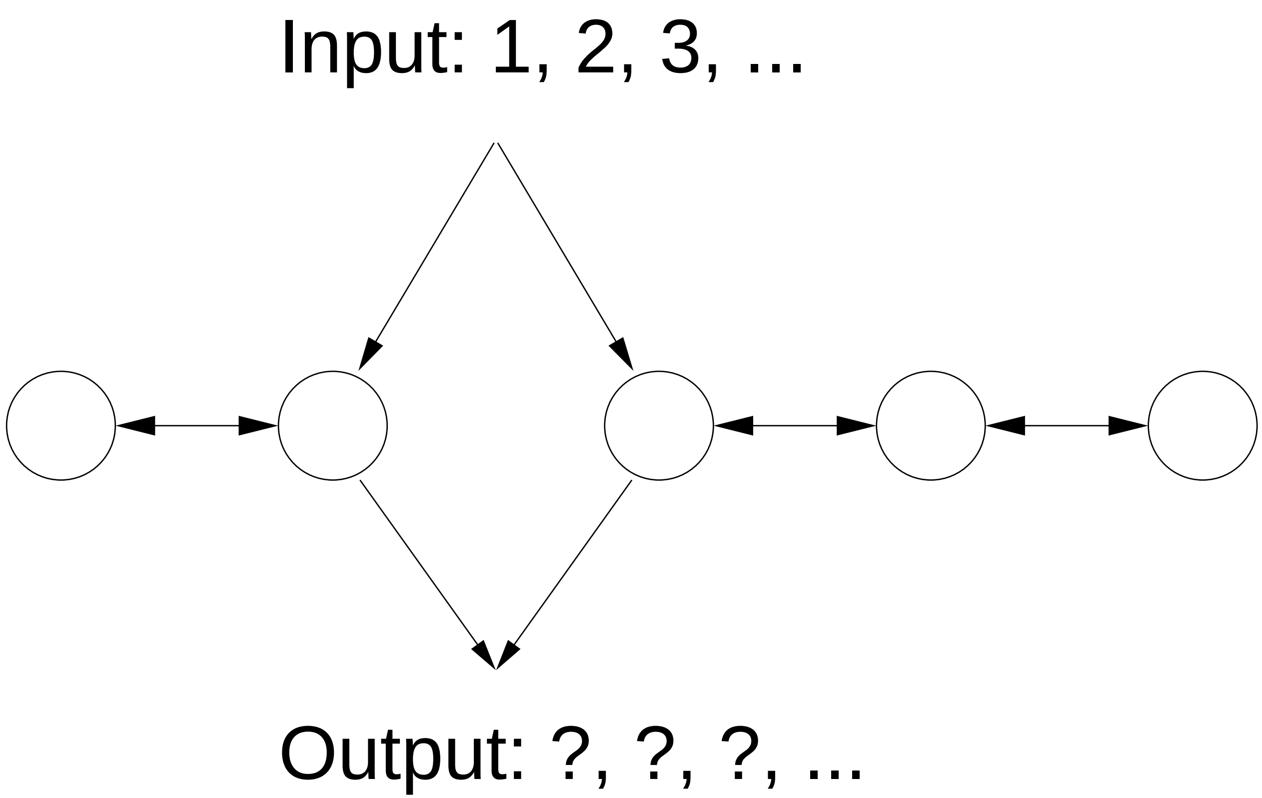
Theorem: *(Marcus, Tardos 2003) If $\mathcal{C}$ is a proper pattern class, then there exists a constant q such that for all n*

$$|\mathcal{C} \cap S_n| \leq q^n$$

A slightly embarassing proof.

Bounded memory machines

- Consider machines for generating permutations whose memory is only capable of holding say M items of input at one time. Each symbol in the permutation is among the first M by rank of the remaining symbols.



M-bounded permutations

- The collection of M-bounded permutations is a pattern class.
- Its basis consists of all the permutations of length $M+1$ which begin with $M+1$.
- It is generated by the "machine" which consists of a desk large enough to hold M pieces of paper.
- M-bounded permutations can be represented by their rank-encoding. This gives a representation over a finite alphabet:

$$341526 \quad \longrightarrow \quad 331211.$$

Regular classes

- A regular language is one which is recognized by a finite automaton.
- A regular permutation class (A, Atkinson, Ruškuc) is one whose rank encoding gives a regular language.
- For instance, the classes provided by bounded memory machines are regular if the machine has only finitely many internal states.
- Bounded memory machines are highly non-deterministic.

Theorems about regular classes

- A bounded class is regular if and only if its basis is regular.
- Given (an automaton for) the class, we can construct (an automaton for) the basis (and vice versa).
- A regular class has a rational generating function. That is, the number of permutations of length n satisfies a linear recurrence.
- There are linear time algorithms for recognizing and generating the permutations belonging to a regular class.

And yet . . .

Regular classes can still be very complicated.

- Consider the basis of the class generated by the machine consisting of two stacks, one of capacity 2, the other of capacity 3, operating in parallel. This is the first explicit example of an antichain in the involvement ordering whose size grows as a function of length.
- The procedure for passing from a class to its basis and vice versa involves determinization and complement (in several iterations). Techniques for reducing the size of intermediate automata are necessary for effective computation.

Finite networks

A finite network has a capacity, which is the largest rank of an item it is capable of delivering to output.

Theorem: *(A, Linton, Ruškuc) For any fixed capacity c there are only finitely many permutation classes generated by networks of that capacity.*

This includes an explicit catalog for $c \leq 3$ (maybe 4).

Context free classes

- One view of the generation of a permutation is by maximum insertion.
- View each insertion event as possibly creating, or filling holes in which further events will happen.

$$\circ \to \circ 1 \circ \to \circ 21 \circ \to 3 \circ 21 \circ \to 3421 \circ \to 34215.$$

- If, as here, we only ever operate on the first slot, then the result is a 312 avoiding permutation.
- Use of context free languages to represent permutation classes in this way, unifies (and extends) many of the known enumeration results.

Where to from here?

- A “structure theory” for pattern classes.
- General principles for manipulating and analysing bounded memory machines.
- Good algorithms for more realistic object moving environments (eg. directed networks of queues).

Where to from here?

- A “structure theory” for pattern classes.
- General principles for manipulating and analysing bounded memory machines.
- Good algorithms for more realistic object moving environments (eg. directed networks of queues).

Thank you!