Combinatorial Exploration

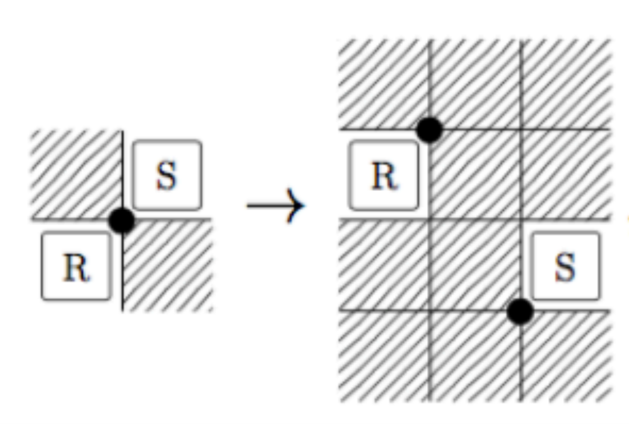
Henning Ulfarsson
ICE-TCS Theory Day 2019

A collaborative research project with the Permuta Triangle

permutatriangle.github.io

Permuta Triangle   About Blog People Papers Programs Research Projects

The study of permutation patterns is a very active area of research and has connections to many other fields of mathematics as well as to computer science and physics. One of the main questions in the field is the enumeration problem: Given a particular set of permutations, how many permutations does the set have of each length? The main goal of this research group is to develop a novel algorithm which will aid researchers in finding structures in sets of permutations and use those structures to find generating functions to enumerate the set. Our research interests lead also into various topics in discrete mathematics and computer science.



Members

- Michael Albert, Professor, Otago University
- Christian Bean, Postdoctoral Researcher, Reykjavik University
- Anders Claesson, Professor, University of Iceland
- Jay Pantone, Assistant Professor, Marquette University
- Henning Ulfarsson, Assistant Professor, Reykjavik University

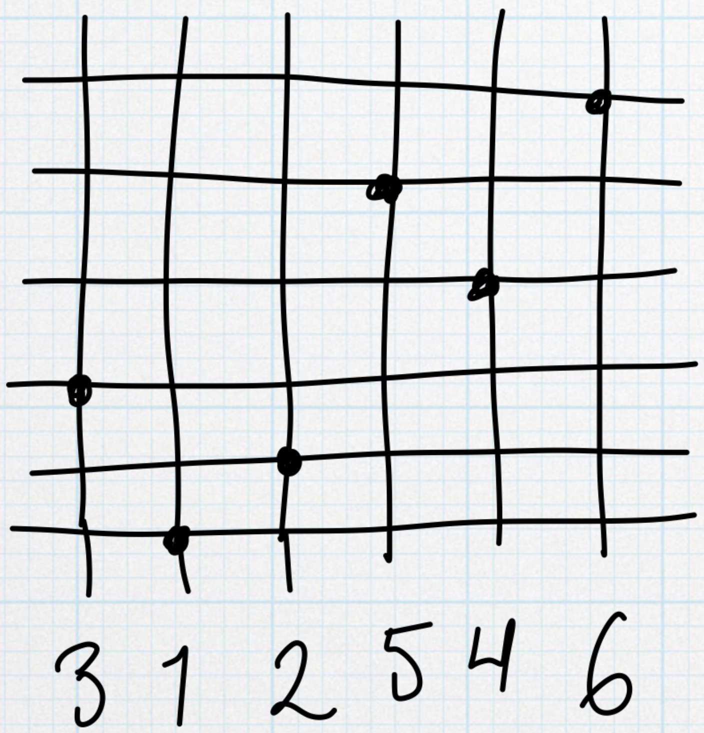
Current students

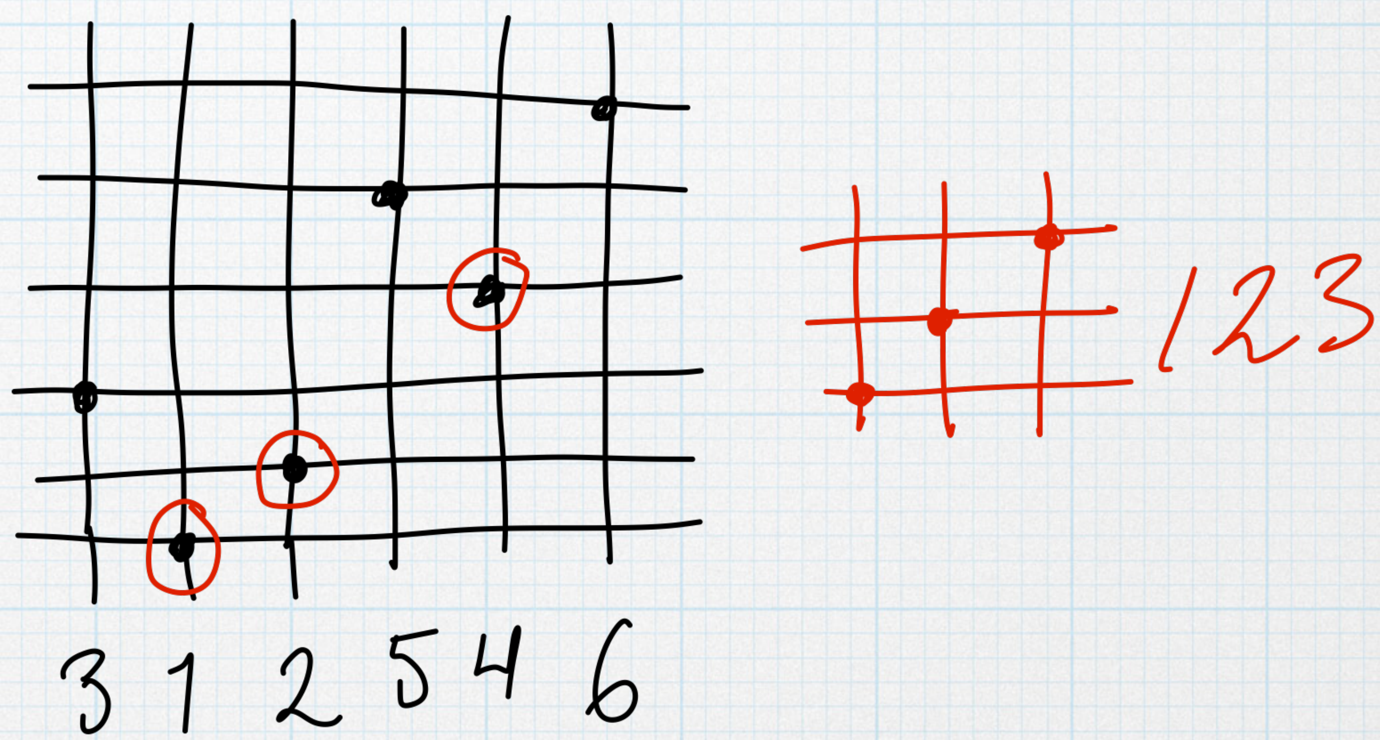
- Ragnar Pall Ardal, MSc student at Reykjavik University
- Arnar Bjarni Arnarson, MSc student at Reykjavik University
- Unnar Freyr Erlendsson, MSc student at Reykjavik University

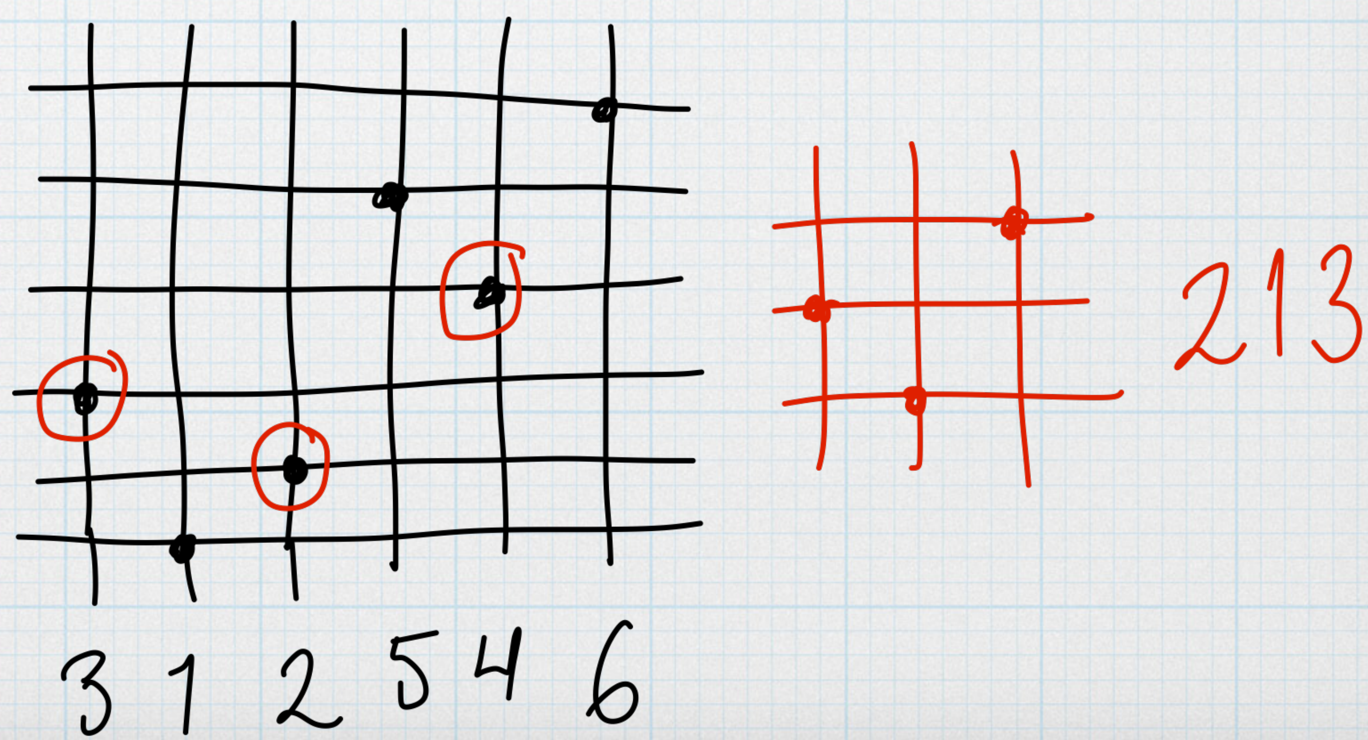
Permutations

* ε
* 1
* 12, 21
* 123, 132, 213, 231, 312, 321
* 1234, 1243, 1324, 1342, ...

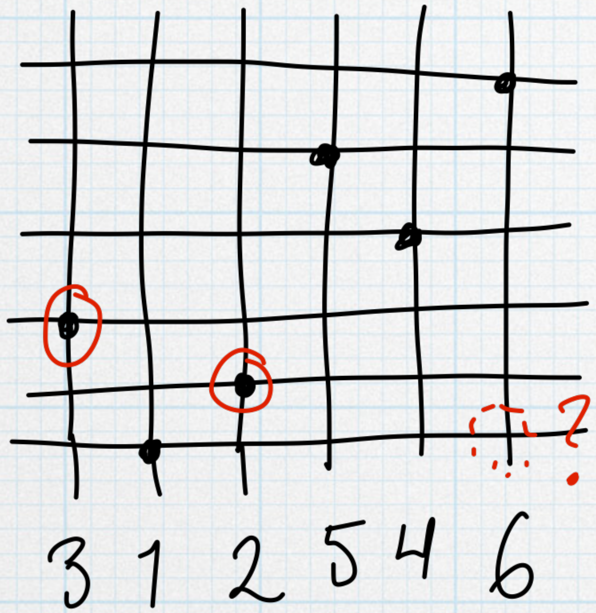
Patterns



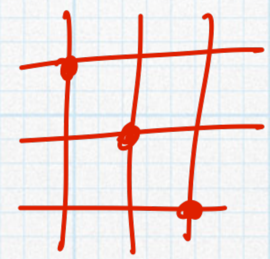




Avoidance



No



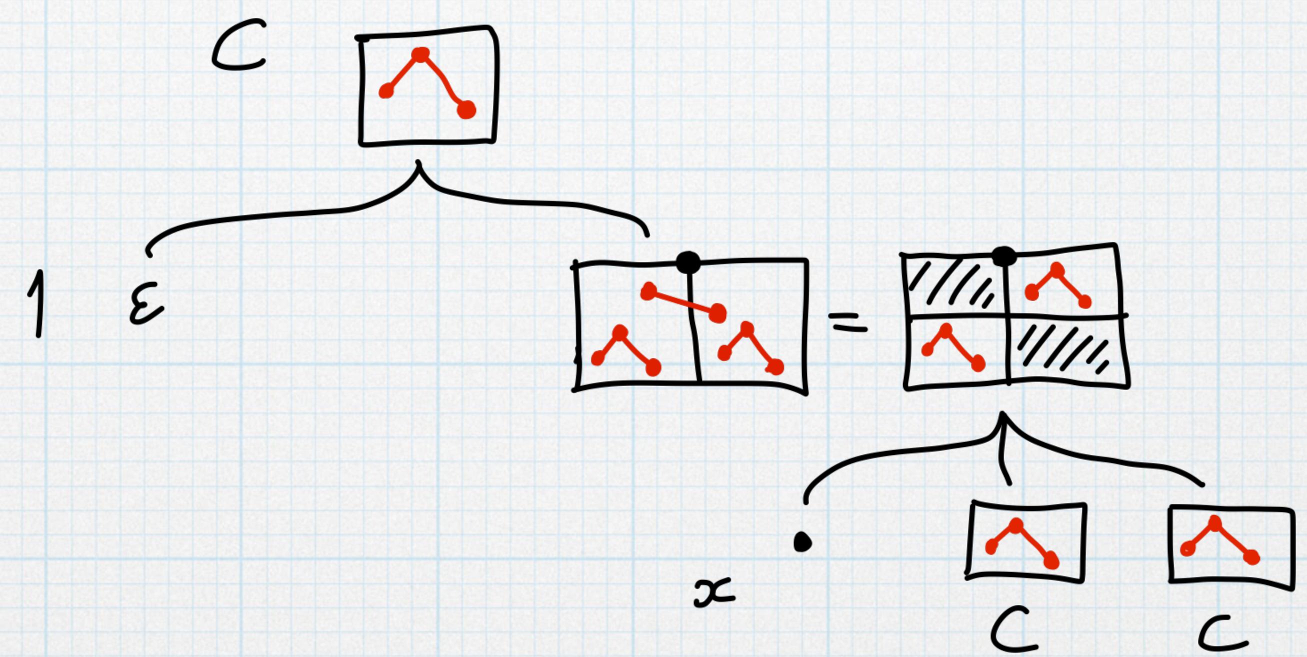
321

So 312546 <u>avoids</u> 321

$312546 \in Av(321)$

"Original problem"

How many permutations of length n are in $A = Av(231)$?



We now understand the structure.

How many? C = generating function

$$C = 1 + x \cdot C \cdot C$$

$$C = \frac{1 - \sqrt{1 - 4x}}{2x} = 1 + x + 2x^2 + 5x^3 + 14x^4 + 42x^5 + \cdots$$

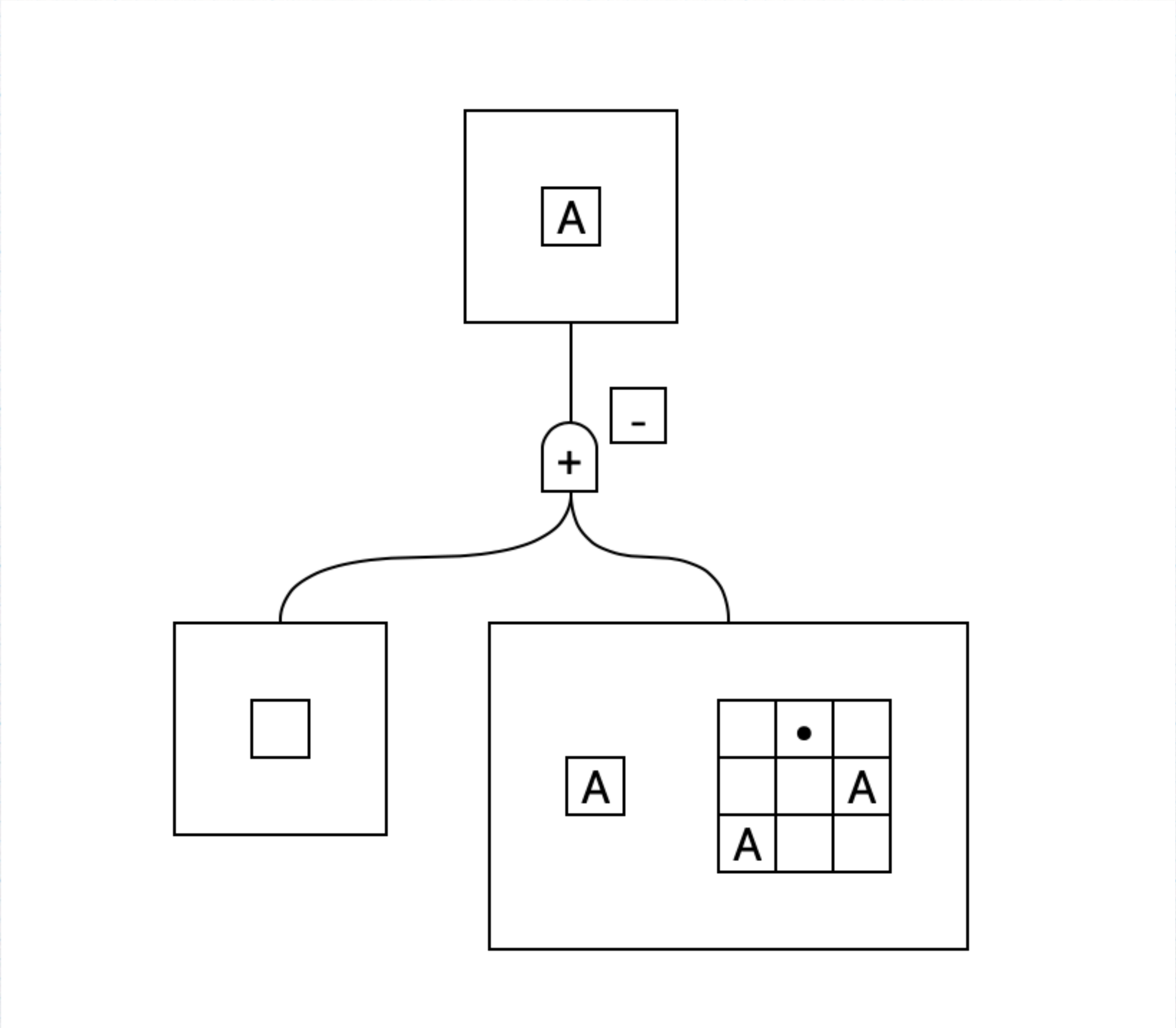
number of perms in Av(231) of length 4

Running the algorithm

```
combrunner 231 point_placements
```


{"root": {"label": 0, "eqv_path_labels": [0], "eqv_path_objects": [{"obstructions": [{"patt": [1, 2, 0], "pos": [[0, 0], [0, 0], [0, 0]]}], "requirements": []}], "eqv_explanations": [], "children": [{"label": 1, "eqv_path_labels": [1], "eqv_path_objects": [{"obstructions": [{"patt": [0], "pos": [[0, 0]]}], "requirements": []}], "eqv_explanations": [], "children": [], "strategy_verified": true, "decomposition": false, "disjoint_union": false, "recursion": false, "formal_step": "The tiling is a subset of the class."}, {"label": 3, "eqv_path_labels": [2, 4, 9], "eqv_path_objects": [{"obstructions": [{"patt": [1, 2, 0], "pos": [[0, 0], [0, 0], [0, 0]]}], "requirements": [[{"patt": [0], "pos": [[0, 0]]}]]}, {"obstructions": [{"patt": [0], "pos": [[0, 1]]}, {"patt": [0], "pos": [[1, 0]]}, {"patt": [0], "pos": [[2, 1]]}, {"patt": [0, 1], "pos": [[1, 1], [1, 1]]}, {"patt": [1, 0], "pos": [[0, 0], [2, 0]]}, {"patt": [1, 0], "pos": [[1, 1], [1, 1]]}, {"patt": [1, 2, 0], "pos": [[0, 0], [0, 0], [0, 0]]}, {"patt": [1, 2, 0], "pos": [[2, 0], [2, 0], [2, 0]]}], "requirements": [[{"patt": [0], "pos": [[1, 1]]}]]}, {"obstructions": [{"patt": [0], "pos": [[0, 1]]}, {"patt": [0], "pos": [[0, 2]]}, {"patt": [0], "pos": [[1, 0]]}, {"patt": [0], "pos": [[1, 1]]}, {"patt": [0], "pos": [[2, 0]]}, {"patt": [0], "pos": [[2, 2]]}, {"patt": [0, 1], "pos": [[1, 2], [1, 2]]}, {"patt": [1, 0], "pos": [[1, 2], [1, 2]]}, {"patt": [1, 2, 0], "pos": [[0, 0], [0, 0], [0, 0]]}, {"patt": [1, 2, 0], "pos": [[2, 1], [2, 1], [2, 1]]}], "requirements": [[{"patt": [0], "pos": [[1, 2]]}]]}], "eqv_explanations": ["Placing point (0, 0) of requirement 0: (0, 0) with force 1~[0:]~", "Separated rows [0] and columns []~[0:]~"], "children": [], "strategy_verified": true, "decomposition": false, "disjoint_union": false, "recursion": false, "formal_step": "The tiling is a subset of the class."}], "strategy_verified": false, "decomposition": false, "disjoint_union": true, "recursion": false,

Use combopal.ru.is to draw



Mechanical Mathematician

WILF CLASSIFICATION OF SUBSETS OF EIGHT AND NINE FOUR-LETTER PATTERNS

***Toufik Mansour*[1,*] *and Matthias Schork*[2,†]**
[1]Department of Mathematics, University of Haifa, Haifa, Israel
[2]Im Haindell, Sulzbach, Germany

Received: September 5, 2016; Accepted: December 13, 2016

Takes about two minutes on this laptop

Pushing the boundary

Contents lists available at ScienceDirect

Journal of Combinatorial Theory, Series A

www.elsevier.com/locate/jcta

Generating permutations with restricted containers

Michael H. Albert[a], Cheyne Homberger[b], Jay Pantone[c,1], Nathaniel Shar[d], Vincent Vatter[e,1]

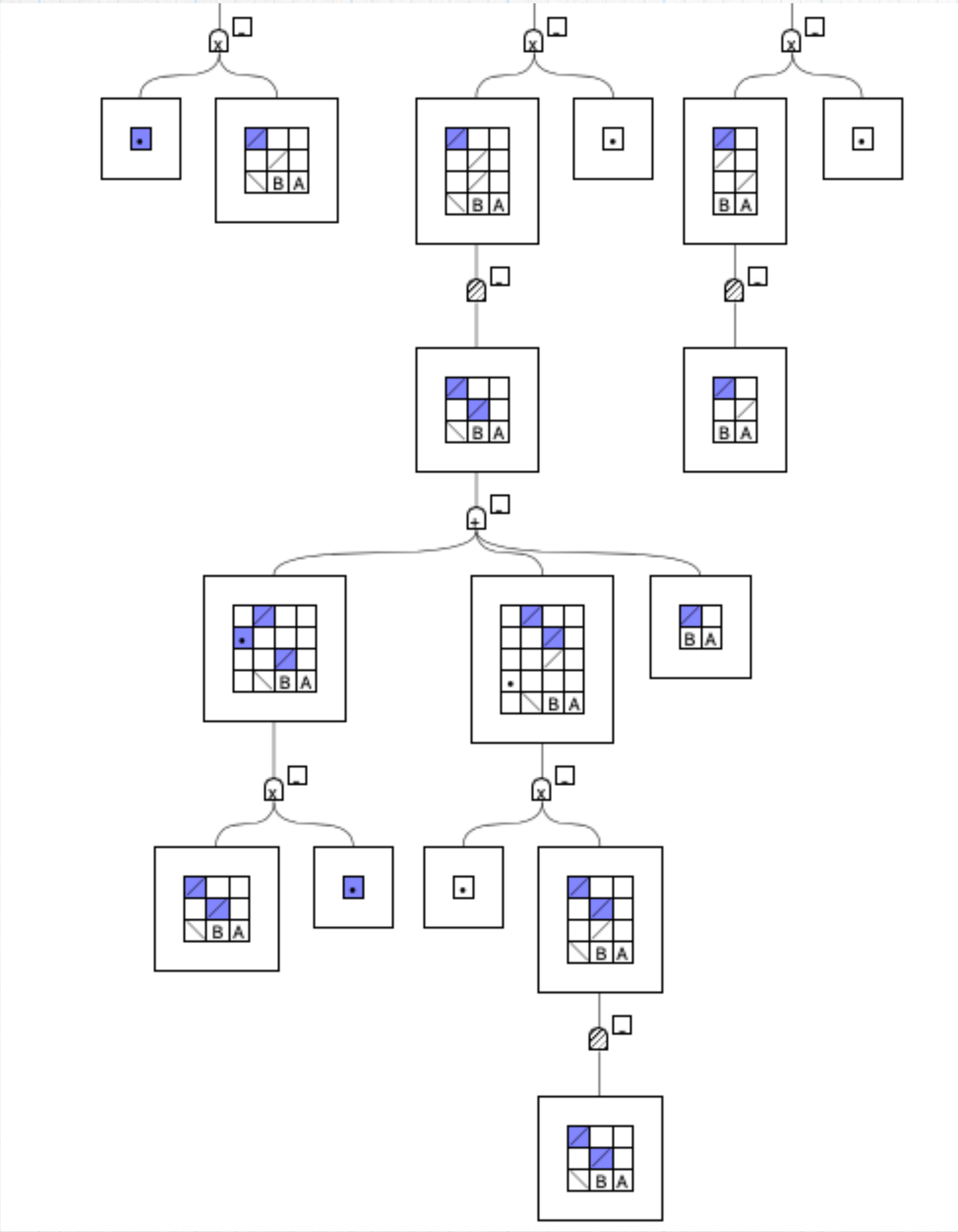
One of the problems: Av(0132, 0213, 0321)

Best known answer: Polynomial time algorithm

```
pypy3 guided_search.py 0132_0213_0321
```

(20 seconds)

Pushing the boundary




```python
def F_25(n):
    if n < 0:
        return 0
    if n in mem['F_25']:
        return mem['F_25'][n]
    ans = F_66(n) + F_0(n) + F_65(n)
    mem['F_25'][n] = ans
    return ans


def F_37(n):
    if n < 0:
        return 0
    if n in mem['F_37']:
        return mem['F_37'][n]
    ans = 0
    ans += F_51(n-1)
    mem['F_37'][n] = ans
    return ans


def F_66(n):
    if n < 0:
        return 0
    if n in mem['F_66']:
        return mem['F_66'][n]
    ans = 0
    ans += F_25(n-1)
    mem['F_66'][n] = ans
    return ans


def F_65(n):
    if n < 0:
        return 0
    if n in mem['F_65']:
```

Pushing the boundary

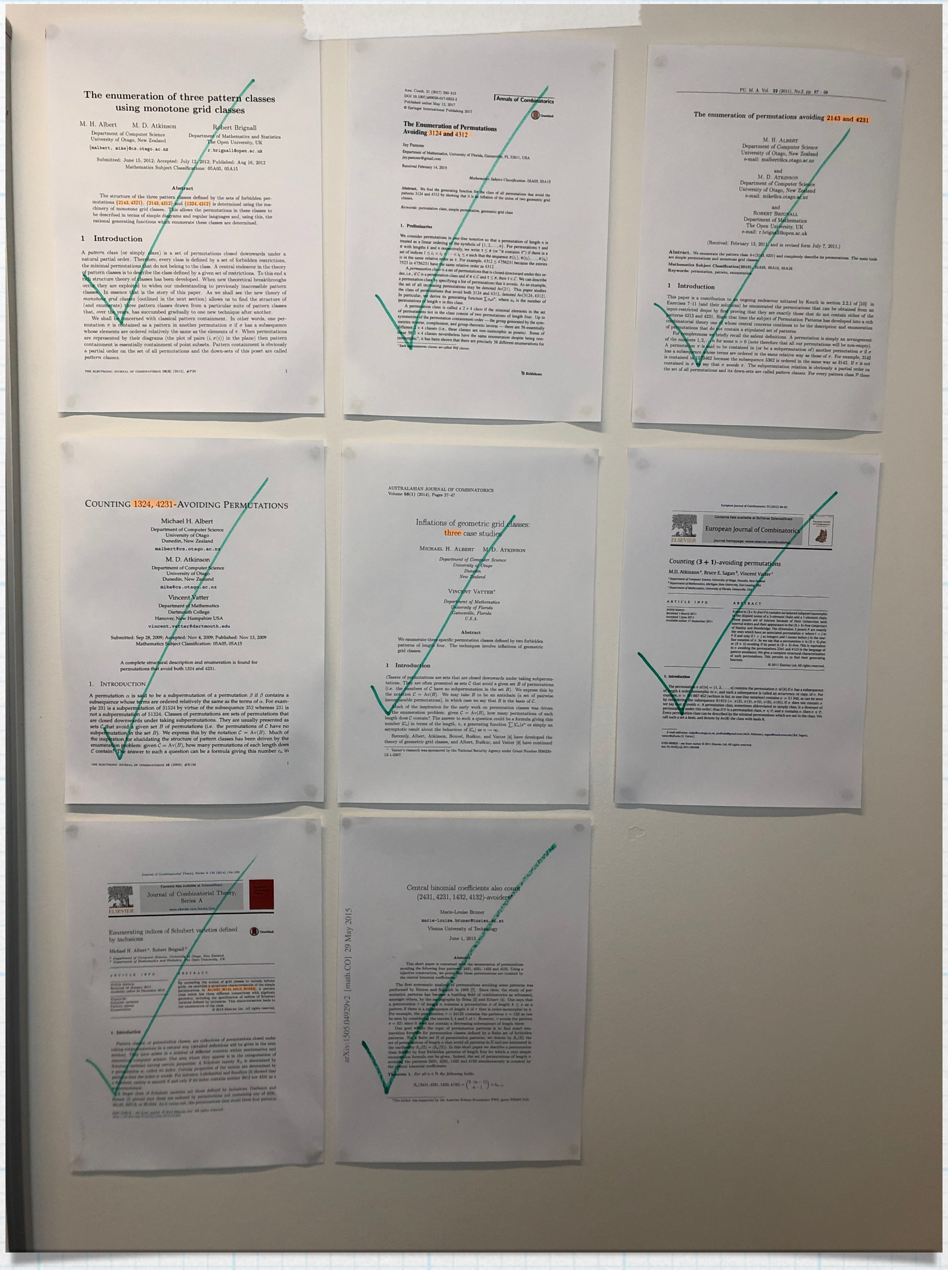
```
0 1
1 1
2 2
3 6
4 21
5 79
6 310
7 1251
8 5150
9 21517
10 90921
11 387595
12 1663936
13 7183750
14 31158310
15 135661904
16 592558096
17 2595232344
18 11392504426
19 50109205789
20 220777103354
21 974162444028
22 4303957562319
23 19036842605855
24 84285643628790
25 373502845338552
26 1656428550764640
27 7351106011540209
```

```
28 32643855249507805
29 145040974005303590
30 644756480385363800
31 2867442403207032074
32 12757585143032068182
33 56780610004782571243
34 252798432449723547544
35 1125843555685097572217
36 5015325404095559202548
37 22347395143382248367695
38 99598284248651168753584
39 443982013127877605506147
40 1979519308346532905984901
41 8827302227974854711515881
42 39369858650227527845883378
43 175615012892661627796131637
44 783458875391672965505811358
45 3495609284854653450577518243
46 15598306151970705636710473030
47 69610702701262572930242668697
48 310681295389380234925043627878
49 1386728712301161960573947374434
50 6190166029373744732373971697006
51 27634067043313137021192439931503
52 123371996584352526991153923424297
53 550827253340411737104334312407403
54 2459457014511099806791429799028545
55 10982119437623379765887688782143966
```

```
56 49040452750976218070516617613158297
57 218999223271877657741353100032153959
58 978022876699353645477012485745401680
59 4367897556652838639626552729913318200
60 19507949775469229494935893182644904656
61 87129541176056788381026188909976316233
62 389164114678586336259516607579816690990
63 1738251994459855933913907704023744449462
64 7764337044905207069031902151127909529130
65 34682219900748563032070821593895685351992
66 154924282707720826777065139321457056145322
67 692056401804988484028609491549160101806613
68 3091521139767007058307640919370924191478 72
69 1381055325863899058736890397337552711395 23
70 6169606948462754143462753856221698919159 00
71 2756201462781226936213511594757643654477 88
72 1231320128199939879625283771254250636864 72
73 5500942536106913026600246721347365010192 37
74 2457587166711566554096749956308989940785 59
75 1097958838157818686987685475067047331383 56
76 4905328928569131457536016073012756648313 41
77 2191567358023519366222677755179924713119 98
78 9791423051112751052815613506561543987618 76
79 4374624913302405687960899610033584112705 10
80 1954517553081682911153300613254269885581 42
81 8732564964122722436849307889331187343791 06
82 3901641301200802794721636911793272203523 21
83 1743234863700087362191852308086423355590 44
```

Successes

- To run heavier computations we use the Garpur cluster
- Have automated about two dozen research articles
- Subsumed several previous algorithmic methods



Future

* As part of his PhD thesis, Christian Bean developed a general framework to apply combinatorial exploration to mathematical objects

* Set partitions, polyominoes, trees, graphs, ...