# Enumerating permutations sortably by $k$ passes through a pop-stack

Anders Claesson
Bjarki Ágúst Guðmundsson

# Permutations

- *Permutation* of length $n$: ordering of $\{1, 2, \dots, n\}$
  - 1234
  - 1324
  - 4321
- *Identity permutation*: the increasing permutation
  - 123456

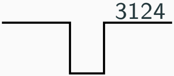
# Stacks

- *Stack*: LIFO data structure with two operations:
  - *Push*: Add an element to the top of the stack
  - *Pop*: Remove the top-most element from the stack

# Sorting with a stack

**Problem (Knuth, 1968)**

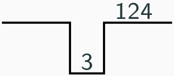
How many permutations of length $n$ can be sorted by a single pass through a stack?



# Sorting with a stack

**Problem (Knuth, 1968)**

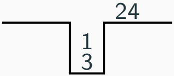
How many permutations of length $n$ can be sorted by a single pass through a stack?



# Sorting with a stack

**Problem (Knuth, 1968)**

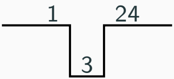
How many permutations of length $n$ can be sorted by a single pass through a stack?



# Sorting with a stack

**Problem (Knuth, 1968)**

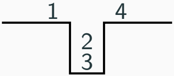
How many permutations of length $n$ can be sorted by a single pass through a stack?



# Sorting with a stack

**Problem (Knuth, 1968)**

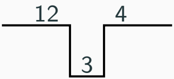
How many permutations of length $n$ can be sorted by a single pass through a stack?



# Sorting with a stack

**Problem (Knuth, 1968)**

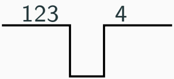
How many permutations of length $n$ can be sorted by a single pass through a stack?



# Sorting with a stack

**Problem (Knuth, 1968)**

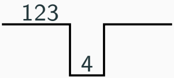
How many permutations of length $n$ can be sorted by a single pass through a stack?



# Sorting with a stack

**Problem (Knuth, 1968)**

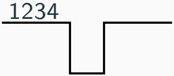
How many permutations of length $n$ can be sorted by a single pass through a stack?



# Sorting with a stack

**Problem (Knuth, 1968)**

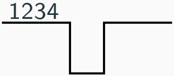
How many permutations of length $n$ can be sorted by a single pass through a stack?



# Sorting with a stack

**Problem (Knuth, 1968)**

How many permutations of length $n$ can be sorted by a single pass through a stack?

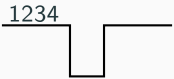



- $3124$ is *stack-sortable*

# Sorting with a stack

**Problem (Knuth, 1968)**

How many permutations of length $n$ can be sorted by a single pass through a stack?

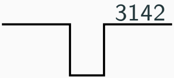



- $3124$ is *stack-sortable*
- Greedy algorithm
  - Keep the stack in increasing order
  - Push when possible
  - Pop when necessary

# Sorting with a stack

**Problem (Knuth, 1968)**

How many permutations of length $n$ can be sorted by a single pass through a stack?

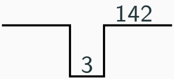



- $3124$ is *stack-sortable*
- Greedy algorithm
  - Keep the stack in increasing order
  - Push when possible
  - Pop when necessary

# Sorting with a stack

**Problem (Knuth, 1968)**

How many permutations of length $n$ can be sorted by a single pass through a stack?

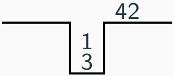



- $3124$ is *stack-sortable*
- Greedy algorithm
  - Keep the stack in increasing order
  - Push when possible
  - Pop when necessary

# Sorting with a stack

**Problem (Knuth, 1968)**

How many permutations of length $n$ can be sorted by a single pass through a stack?

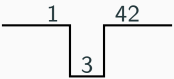



- $3124$ is *stack-sortable*
- Greedy algorithm
  - Keep the stack in increasing order
  - Push when possible
  - Pop when necessary

# Sorting with a stack

**Problem (Knuth, 1968)**

How many permutations of length $n$ can be sorted by a single pass through a stack?

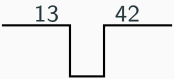



- $3124$ is *stack-sortable*
- Greedy algorithm
  - Keep the stack in increasing order
  - Push when possible
  - Pop when necessary

# Sorting with a stack

**Problem (Knuth, 1968)**

How many permutations of length $n$ can be sorted by a single pass through a stack?

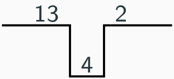



- $3124$ is *stack-sortable*
- Greedy algorithm
  - Keep the stack in increasing order
  - Push when possible
  - Pop when necessary

# Sorting with a stack

**Problem (Knuth, 1968)**

How many permutations of length $n$ can be sorted by a single pass through a stack?

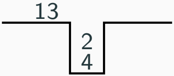



- $3124$ is *stack-sortable*
- Greedy algorithm
  - Keep the stack in increasing order
  - Push when possible
  - Pop when necessary

# Sorting with a stack

**Problem (Knuth, 1968)**

How many permutations of length $n$ can be sorted by a single pass through a stack?

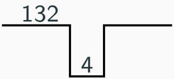



- $3124$ is *stack-sortable*
- Greedy algorithm
  - Keep the stack in increasing order
  - Push when possible
  - Pop when necessary

# Sorting with a stack

**Problem (Knuth, 1968)**

How many permutations of length $n$ can be sorted by a single pass through a stack?

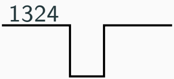



- $3124$ is *stack-sortable*
- Greedy algorithm
  - Keep the stack in increasing order
  - Push when possible
  - Pop when necessary

# Sorting with a stack

**Problem (Knuth, 1968)**

How many permutations of length $n$ can be sorted by a single pass through a stack?

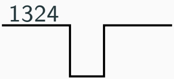



- $3124$ is *stack-sortable*
- Greedy algorithm
  - Keep the stack in increasing order
  - Push when possible
  - Pop when necessary

# Sorting with a stack

**Problem (Knuth, 1968)**

How many permutations of length $n$ can be sorted by a single pass through a stack?

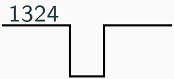



- $3124$ is *stack-sortable*, $3142$ is not
- Greedy algorithm
  - Keep the stack in increasing order
  - Push when possible
  - Pop when necessary

# Sorting with a stack

**Problem (Knuth, 1968)**

How many permutations of length $n$ can be sorted by a single pass through a stack?




- $3124$ is *stack-sortable*, $3142$ is not
- Greedy algorithm
  - Keep the stack in increasing order
  - Push when possible
  - Pop when necessary
- Stack-sortable permutations:
  - Simple description in terms of pattern avoidance
  - Enumerated by the Catalan numbers $C_n$

# Sorting with a stack, multiple passes

**Problem (West, 1990)**

How many permutations of length $n$ can be sorted by **at most two passes** through a stack?

# Sorting with a stack, multiple passes

**Problem (West, 1990)**

How many permutations of length $n$ can be sorted by **at most two passes** through a stack?

- Consider $3142$
  - After one pass: $1324$
  - After two passes: $1234$
  - $3142$ is *2-stack-sortable*

# Sorting with a stack, multiple passes

**Problem (West, 1990)**

How many permutations of length $n$ can be sorted by **at most two passes** through a stack?

- Consider $3142$
  - After one pass: $1324$
  - After two passes: $1234$
  - $3142$ is *2-stack-sortable*
- 2-stack-sortable permutations:
  - Relatively simple description in terms of pattern avoidance
  - Formula for their enumeration proved by (Zeilberger, 1992)

# Sorting with a stack, multiple passes

**Problem (West, 1990)**

How many permutations of length $n$ can be sorted by **at most two passes** through a stack?

- Consider $3142$
  - After one pass: $1324$
  - After two passes: $1234$
  - $3142$ is *2-stack-sortable*
- 2-stack-sortable permutations:
  - Relatively simple description in terms of pattern avoidance
  - Formula for their enumeration proved by (Zeilberger, 1992)
- 3-stack-sortable permutations:
  - Complex description in terms of pattern avoidance (Ulfarsson, 2011)
  - No enumeration results

# Sorting with a stack, multiple passes

**Problem (West, 1990)**

How many permutations of length $n$ can be sorted by **at most two passes** through a stack?

- Consider $3142$
  - After one pass: $1324$
  - After two passes: $1234$
  - $3142$ is *2-stack-sortable*
- 2-stack-sortable permutations:
  - Relatively simple description in terms of pattern avoidance
  - Formula for their enumeration proved by (Zeilberger, 1992)
- 3-stack-sortable permutations:
  - Complex description in terms of pattern avoidance (Ulfarsson, 2011)
  - No enumeration results
- $k$-stack-sortable permutations, $k > 3$:
  - Nothing is known

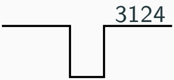
# Pop-stacks

- *Pop-stack*: LIFO data structure with two operations:
  - *Push*: Add an element to the top of the stack
  - *Pop*: Remove all elements from the stack

# Sorting with a pop-stack

**Problem**

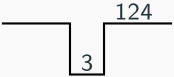
How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?



# Sorting with a pop-stack

**Problem**

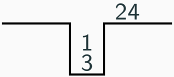
How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?



# Sorting with a pop-stack

**Problem**

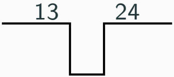
How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?



# Sorting with a pop-stack

**Problem**

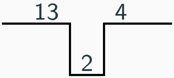
How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?



# Sorting with a pop-stack

**Problem**

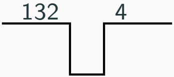
How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?



# Sorting with a pop-stack

**Problem**

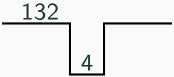
How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?



# Sorting with a pop-stack

**Problem**

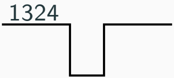
How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?



# Sorting with a pop-stack

**Problem**

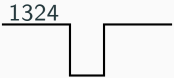
How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?



# Sorting with a pop-stack

**Problem**

How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?

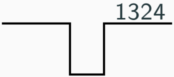



- $3124$ is not pop-stack-sortable

# Sorting with a pop-stack

**Problem**

How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?

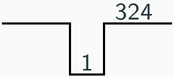



- $3124$ is not pop-stack-sortable

# Sorting with a pop-stack

**Problem**

How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?

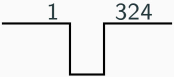



- $3124$ is not pop-stack-sortable

# Sorting with a pop-stack

**Problem**

How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?

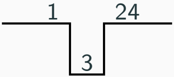



- $3124$ is not pop-stack-sortable

# Sorting with a pop-stack

**Problem**

How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?

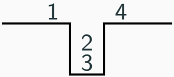



- $3124$ is not pop-stack-sortable

# Sorting with a pop-stack

**Problem**

How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?

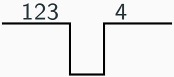



- $3124$ is not pop-stack-sortable

# Sorting with a pop-stack

**Problem**

How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?

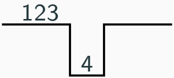



- $3124$ is not pop-stack-sortable

# Sorting with a pop-stack

**Problem**

How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?

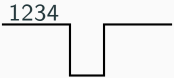



- $3124$ is not pop-stack-sortable

# Sorting with a pop-stack

**Problem**

How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?

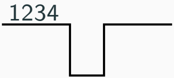



- $3124$ is not pop-stack-sortable

# Sorting with a pop-stack

**Problem**

How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?

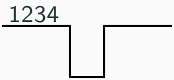



- $3124$ is not pop-stack-sortable, but it is 2-pop-stack-sortable

# Sorting with a pop-stack

**Problem**

How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?

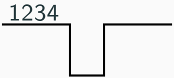



- $3124$ is not pop-stack-sortable, but it is 2-pop-stack-sortable
- pop-stack-sortable permutations:
  - Simple description, and $2^{n-1}$ sortable permutations of length $n$ (Avis and Newborn, 1981)

# Sorting with a pop-stack

**Problem**

How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?

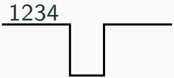



- $3124$ is not pop-stack-sortable, but it is 2-pop-stack-sortable
- pop-stack-sortable permutations:
  - Simple description, and $2^{n-1}$ sortable permutations of length $n$ (Avis and Newborn, 1981)
- 2-pop-stack-sortable permutations:
  - Complex description in terms of pattern avoidance, and formula is known (Pudwell and Smith, 2017)

# Sorting with a pop-stack

**Problem**

How many permutations of length $n$ can be sorted by at most $k$ passes through a **pop-stack**?




- $3124$ is not pop-stack-sortable, but it is 2-pop-stack-sortable
- pop-stack-sortable permutations:
  - Simple description, and $2^{n-1}$ sortable permutations of length $n$ (Avis and Newborn, 1981)
- 2-pop-stack-sortable permutations:
  - Complex description in terms of pattern avoidance, and formula is known (Pudwell and Smith, 2017)
- $k$-pop-stack-sortable permutations, $k > 2$:
  - Open problem—let's try to count them!

# Sorting traces

$$5 \quad 1 \quad 2 \quad 4 \quad 7 \quad 8 \quad 6 \quad 3 \quad 9$$

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |

## Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | | | | | | | |

## Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | | | | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | | | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | | | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |

1 5 2 4 7 3 6 8 9

## Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | | | | | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | | | | | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | | | | | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | | | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | | | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | 6 | | |

## Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | 6 | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | 6 | 8 | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | 6 | 8 | |

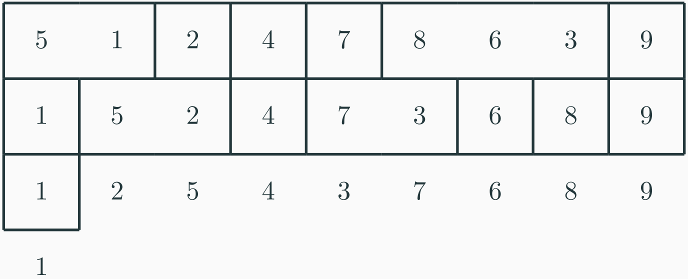
# Sorting traces

<table>
<tr><td colspan="2">5 1</td><td>2</td><td>4</td><td>7</td><td colspan="3">8 6 3</td><td>9</td></tr>
<tr><td>1</td><td colspan="2">5 2</td><td>4</td><td colspan="2">7 3</td><td>6</td><td>8</td><td>9</td></tr>
<tr><td>1</td><td>2</td><td>5</td><td>4</td><td>3</td><td>7</td><td>6</td><td>8</td><td>9</td></tr>
</table>

## Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | 6 | 8 | 9 |

# Sorting traces



# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | 6 | 8 | 9 |
| 1 | | | | | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | 6 | 8 | 9 |
| 1 | 2 | | | | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | 6 | 8 | 9 |
| 1 | 2 | | | | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | 6 | 8 | 9 |
| 1 | 2 | | | | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | 6 | 8 | 9 |
| 1 | 2 | | | | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | 6 | 8 | 9 |
| 1 | 2 | 3 | 4 | 5 | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | 6 | 8 | 9 |
| 1 | 2 | 3 | 4 | 5 | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | 6 | 8 | 9 |
| 1 | 2 | 3 | 4 | 5 | | | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | 6 | 8 | 9 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | 6 | 8 | 9 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | 6 | 8 | 9 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |

# Sorting traces

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | 6 | 8 | 9 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | |

# Sorting traces

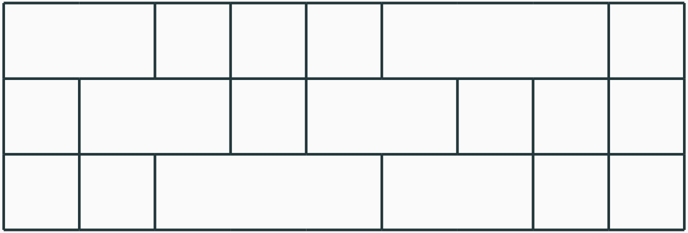
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 5 | 1 | 2 | 4 | 7 | 8 | 6 | 3 | 9 |
| 1 | 5 | 2 | 4 | 7 | 3 | 6 | 8 | 9 |
| 1 | 2 | 5 | 4 | 3 | 7 | 6 | 8 | 9 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

# Sorting traces

<table>
<tr><td colspan="2">5 1</td><td>2</td><td>4</td><td>7</td><td colspan="3">8 6 3</td><td>9</td></tr>
<tr><td>1</td><td colspan="2">5 2</td><td>4</td><td colspan="2">7 3</td><td>6</td><td>8</td><td>9</td></tr>
<tr><td>1</td><td>2</td><td colspan="3">5 4 3</td><td colspan="2">7 6</td><td>8</td><td>9</td></tr>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr>
</table>

- We call this a *sorting trace* of length $9$ and order $3$
  - The numbers within each *block* must be in decreasing order
  - Adjacent numbers in different blocks must form an ascent
  - Each permutation must be the "blockwise reversal" of the permutation above
  - The last permutation is the identity permutation

# Sorting traces



- We call this a *sorting trace* of length $9$ and order $3$
  - The numbers within each *block* must be in decreasing order
  - Adjacent numbers in different blocks must form an ascent
  - Each permutation must be the "blockwise reversal" of the permutation above
  - The last permutation is the identity permutation
- Removing the numbers, the structure that remains we call a *skeleton*
  - A trace of length $n$ and order $k$ has a skeleton with $k$ rows
  - Each row is an integer composition of $n$

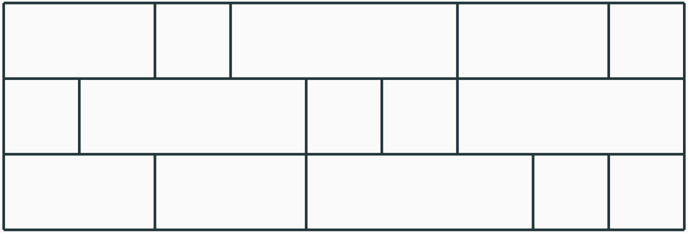
# Validity of skeletons

- Say we have a $k$-pop-stack-sortable permutation of length $n$. We can
  1. generate its trace, and
  2. drop the numbers from the trace.

  This gives us a skeleton of length $n$ and order $k$.
- What about the other direction?

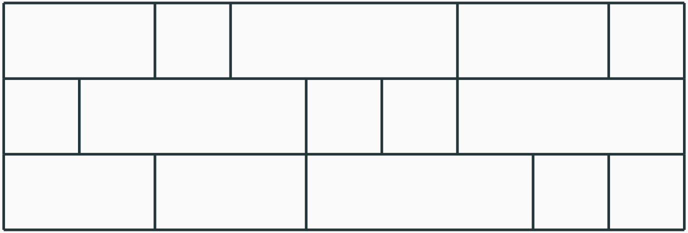
# Validity of skeletons

- Consider the following skeleton of length $9$ and order $3$:



# Validity of skeletons

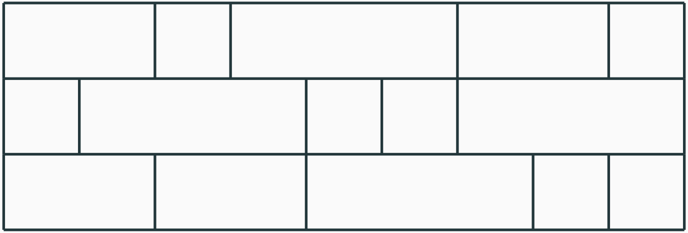
- Consider the following skeleton of length $9$ and order $3$:



- Assume there exists a trace that has this skeleton.

# Validity of skeletons

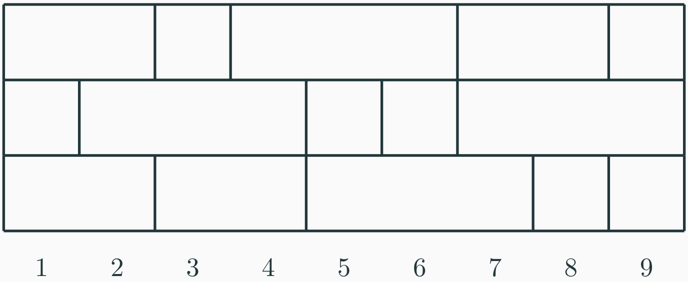
- Consider the following skeleton of length $9$ and order $3$:



- Assume there exists a trace that has this skeleton. Then
  - the last permutation must be the identity

## Validity of skeletons

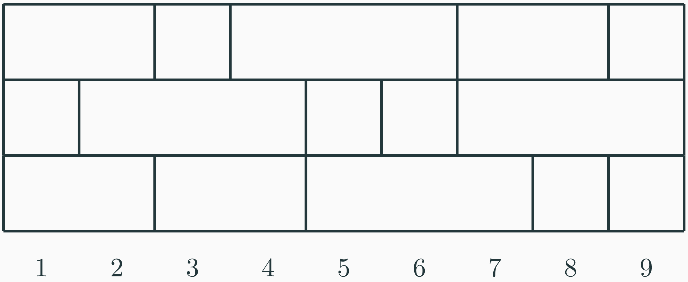
- Consider the following skeleton of length $9$ and order $3$:




- Assume there exists a trace that has this skeleton. Then
  - the last permutation must be the identity

# Validity of skeletons

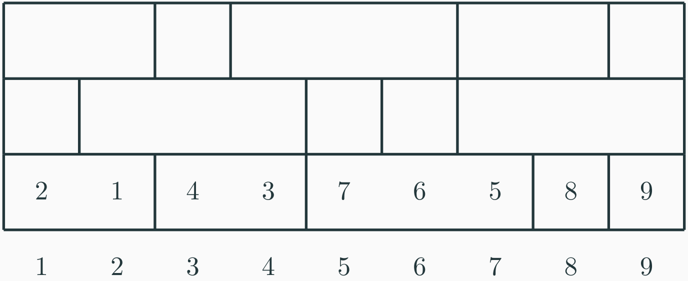
- Consider the following skeleton of length $9$ and order $3$:




- Assume there exists a trace that has this skeleton. Then
  - the last permutation must be the identity, and
  - each permutation is the "blockwise reversal" of the permutation above.

# Validity of skeletons

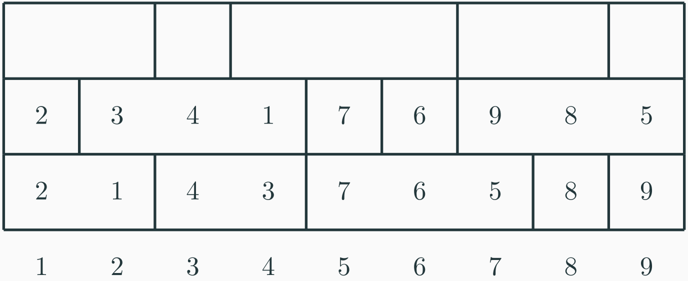
- Consider the following skeleton of length $9$ and order $3$:




- Assume there exists a trace that has this skeleton. Then
  - the last permutation must be the identity, and
  - each permutation is the "blockwise reversal" of the permutation above.

# Validity of skeletons

- Consider the following skeleton of length $9$ and order $3$:




- Assume there exists a trace that has this skeleton. Then
  - the last permutation must be the identity, and
  - each permutation is the "blockwise reversal" of the permutation above.

# Validity of skeletons

- Consider the following skeleton of length $9$ and order $3$:

<table>
<tr><td colspan="2">3 2</td><td>4</td><td colspan="3">6 7 1</td><td colspan="2">8 9</td><td>5</td></tr>
<tr><td>2</td><td colspan="3">3 4 1</td><td>7</td><td>6</td><td colspan="3">9 8 5</td></tr>
<tr><td colspan="2">2 1</td><td colspan="2">4 3</td><td colspan="3">7 6 5</td><td>8</td><td>9</td></tr>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td></tr>
</table>

- Assume there exists a trace that has this skeleton. Then
  - the last permutation must be the identity, and
  - each permutation is the "blockwise reversal" of the permutation above.

# Validity of skeletons

- Consider the following skeleton of length $9$ and order $3$:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 3 | 2 | 4 | 6 | 7 | 1 | 8 | 9 | 5 |
| 2 | 3 | 4 | 1 | 7 | 6 | 9 | 8 | 5 |
| 2 | 1 | 4 | 3 | 7 | 6 | 5 | 8 | 9 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

- Assume there exists a trace that has this skeleton. Then
  - the last permutation must be the identity, and
  - each permutation is the "blockwise reversal" of the permutation above.

# Validity of skeletons

- Consider the following skeleton of length $9$ and order $3$:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 3 | 2 | 4 | 6 | 7 | 1 | 8 | 9 | 5 |
| 2 | 3 | 4 | 1 | 7 | 6 | 9 | 8 | 5 |
| 2 | 1 | 4 | 3 | 7 | 6 | 5 | 8 | 9 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

- Assume there exists a trace that has this skeleton. Then
  - the last permutation must be the identity, and
  - each permutation is the "blockwise reversal" of the permutation above.

# Validity of skeletons

- Consider the following skeleton of length $9$ and order $3$:

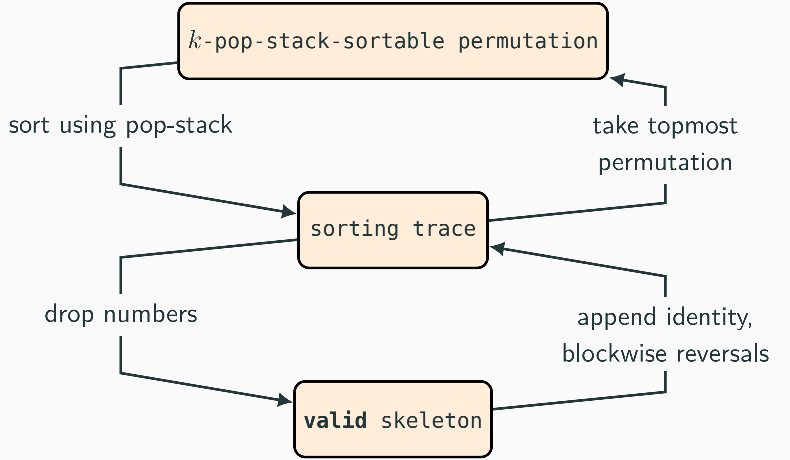
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 3 | 2 | 4 | 6 | 7 | 1 | 8 | 9 | 5 |
| 2 | 3 | 4 | 1 | 7 | 6 | 9 | 8 | 5 |
| 2 | 1 | 4 | 3 | 7 | 6 | 5 | 8 | 9 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

- Assume there exists a trace that has this skeleton. Then
  - the last permutation must be the identity, and
  - each permutation is the "blockwise reversal" of the permutation above.
- This is not a trace, so the skeleton is not *valid*!

# Validity of skeletons

- A skeleton is *valid* if we get a proper sorting trace after filling in the numbers:
  (1) The numbers within each block are in decreasing order
  (2) Adjacent numbers in different blocks form an ascent

# Bijection



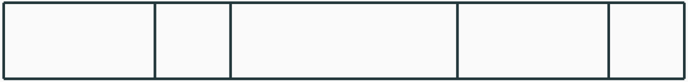

- We have a bijection between $k$-pop-stack-sortable permutations of length $n$ and valid skeletons of length $n$ and order $k$

# Valid skeletons

- To count the $k$-pop-stack-sortable permutations of length $n$ we will count the valid skeletons of length $n$ and order $k$
- How to determine if an arbitrary skeleton is valid?

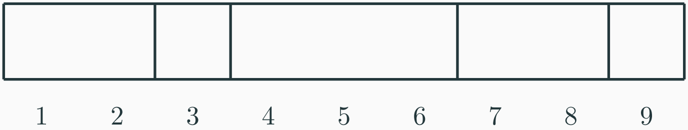
# Valid skeletons for $k = 1$

- Consider the following skeleton of order $1$:



# Valid skeletons for $k = 1$

- Consider the following skeleton of order $1$:



## Valid skeletons for $k = 1$

- Consider the following skeleton of order $1$:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 3 | 6 | 5 | 4 | 8 | 7 | 9 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Blocks: [2 1] [3] [6 5 4] [8 7] [9]

# Valid skeletons for $k = 1$

- Consider the following skeleton of order $1$:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 3 | 6 | 5 | 4 | 8 | 7 | 9 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Blocks: [2 1] [3] [6 5 4] [8 7] [9]

- Recall the two conditions:
  (1) The numbers within each block are in decreasing order

  (2) Adjacent numbers in different blocks form an ascent

# Valid skeletons for $k = 1$

- Consider the following skeleton of order $1$:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $2$ | $1$ | $3$ | $6$ | $5$ | $4$ | $8$ | $7$ | $9$ |
| $1$ | $2$ | $3$ | $4$ | $5$ | $6$ | $7$ | $8$ | $9$ |

- Recall the two conditions:
  (1) The numbers within each block are in decreasing order—this will always be true!
  (2) Adjacent numbers in different blocks form an ascent

# Valid skeletons for $k = 1$

- Consider the following skeleton of order $1$:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 3 | 6 | 5 | 4 | 8 | 7 | 9 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Blocks: [2 1] [3] [6 5 4] [8 7] [9]

- Recall the two conditions:
  (1) The numbers within each block are in decreasing order—this will always be true!
  (2) Adjacent numbers in different blocks form an ascent—this will always be true!

# Valid skeletons for $k = 1$

- Consider the following skeleton of order $1$:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 3 | 6 | 5 | 4 | 8 | 7 | 9 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

Blocks: (2 1) (3) (6 5 4) (8 7) (9)

- Recall the two conditions:
  (1) The numbers within each block are in decreasing order—this will always be true!
  (2) Adjacent numbers in different blocks form an ascent—this will always be true!
- All skeletons of order 1 are valid!

# Valid skeletons for $k = 1$

- Consider the following skeleton of order $1$:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| $2$ | $1$ | $3$ | $6$ | $5$ | $4$ | $8$ | $7$ | $9$ |
| $1$ | $2$ | $3$ | $4$ | $5$ | $6$ | $7$ | $8$ | $9$ |

- Recall the two conditions:
  (1) The numbers within each block are in decreasing order—this will always be true!
  (2) Adjacent numbers in different blocks form an ascent—this will always be true!
- All skeletons of order 1 are valid!
- There are $2^{n-1}$ skeletons of length $n$ and order $1$

# Valid skeletons for $k = 1$

- Consider the following skeleton of order $1$:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 3 | 6 | 5 | 4 | 8 | 7 | 9 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

- Recall the two conditions:
  (1) The numbers within each block are in decreasing order—this will always be true!
  (2) Adjacent numbers in different blocks form an ascent—this will always be true!
- All skeletons of order 1 are valid!
- There are $2^{n-1}$ skeletons of length $n$ and order $1$
- Therefore $2^{n-1}$ pop-stack-sortable permutations of length $n$

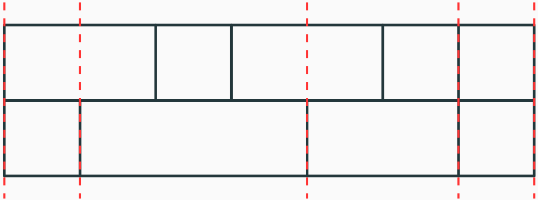
# Valid skeletons for $k = 2$

- Consider an arbitrary **valid** skeleton of order $2$
- Slice it up along the boundaries of the blocks in the second row

- Consider one of the resulting pieces, and let's do case analysis based on the size of the block in the second row

# Valid skeletons for $k = 2$

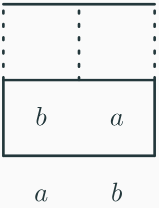
- Consider an arbitrary **valid** skeleton of order $2$
- Slice it up along the boundaries of the blocks in the second row



- Consider one of the resulting pieces, and let's do case analysis based on the size of the block in the second row

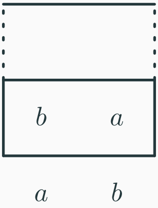
# Valid skeletons for $k = 2$

- Say the lower block is of size $2$
- Then we have two numbers $a, b \in [n]$, with $b = a + 1$



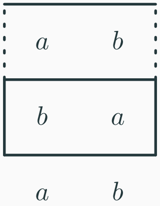
## Valid skeletons for $k = 2$

- Say the lower block is of size $2$
- Then we have two numbers $a, b \in [n]$, with $b = a + 1$



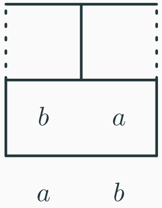
# Valid skeletons for $k = 2$

- Say the lower block is of size $2$
- Then we have two numbers $a, b \in [n]$, with $b = a + 1$



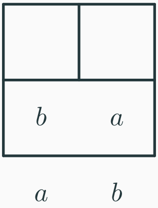
# Valid skeletons for $k = 2$

- Say the lower block is of size $2$
- Then we have two numbers $a, b \in [n]$, with $b = a + 1$



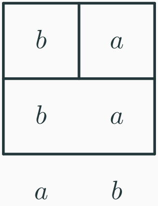
# Valid skeletons for $k = 2$

- Say the lower block is of size $2$
- Then we have two numbers $a, b \in [n]$, with $b = a + 1$



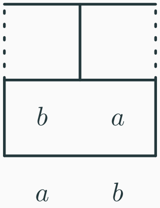
# Valid skeletons for $k = 2$

- Say the lower block is of size $2$
- Then we have two numbers $a, b \in [n]$, with $b = a + 1$



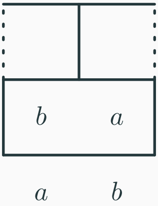
## Valid skeletons for $k = 2$

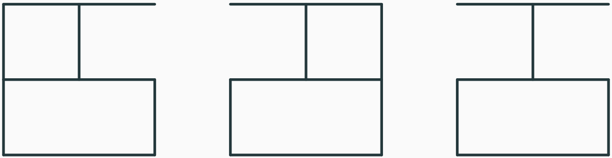
- Say the lower block is of size $2$
- Then we have two numbers $a, b \in [n]$, with $b = a + 1$



# Valid skeletons for $k = 2$

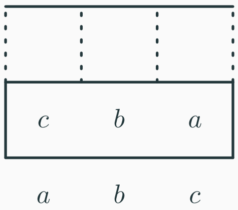
- Say the lower block is of size $2$
- Then we have two numbers $a, b \in [n]$, with $b = a + 1$



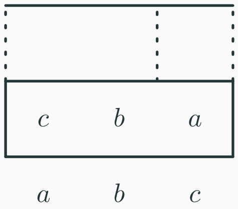
# Valid skeletons for $k = 2$

- Say the lower block is of size $3$
- Then we have three numbers $a, b, c \in [n]$, with $b = a + 1$ and $c = b + 1$



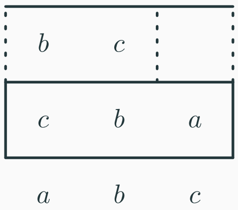
## Valid skeletons for $k = 2$

- Say the lower block is of size $3$
- Then we have three numbers $a, b, c \in [n]$, with $b = a + 1$ and $c = b + 1$



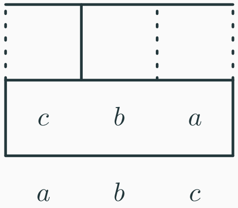
# Valid skeletons for $k = 2$

- Say the lower block is of size $3$
- Then we have three numbers $a, b, c \in [n]$, with $b = a + 1$ and $c = b + 1$



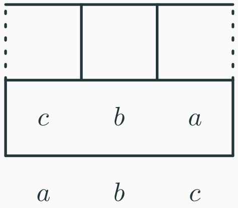
# Valid skeletons for $k = 2$

- Say the lower block is of size $3$
- Then we have three numbers $a, b, c \in [n]$, with $b = a + 1$ and $c = b + 1$



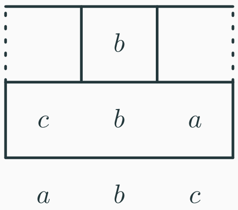
# Valid skeletons for $k = 2$

- Say the lower block is of size $3$
- Then we have three numbers $a, b, c \in [n]$, with $b = a + 1$ and $c = b + 1$



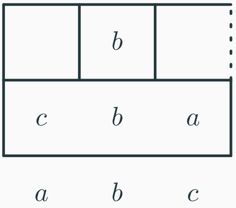
# Valid skeletons for $k = 2$

- Say the lower block is of size $3$
- Then we have three numbers $a, b, c \in [n]$, with $b = a + 1$ and $c = b + 1$



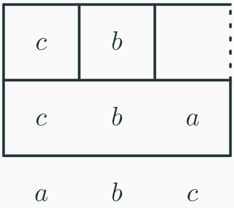
# Valid skeletons for $k = 2$

- Say the lower block is of size $3$
- Then we have three numbers $a, b, c \in [n]$, with $b = a + 1$ and $c = b + 1$



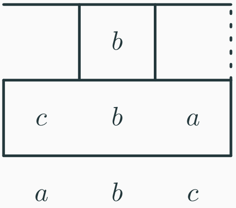
# Valid skeletons for $k = 2$

- Say the lower block is of size $3$
- Then we have three numbers $a, b, c \in [n]$, with $b = a + 1$ and $c = b + 1$



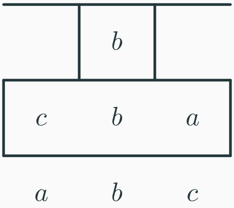
# Valid skeletons for $k = 2$

- Say the lower block is of size $3$
- Then we have three numbers $a, b, c \in [n]$, with $b = a + 1$ and $c = b + 1$



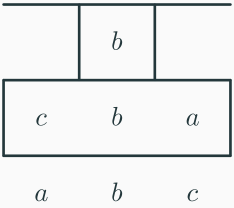
# Valid skeletons for $k = 2$

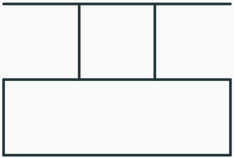
- Say the lower block is of size $3$
- Then we have three numbers $a, b, c \in [n]$, with $b = a + 1$ and $c = b + 1$



# Valid skeletons for $k = 2$

- Say the lower block is of size $3$
- Then we have three numbers $a, b, c \in [n]$, with $b = a + 1$ and $c = b + 1$



## Valid skeletons for $k = 2$

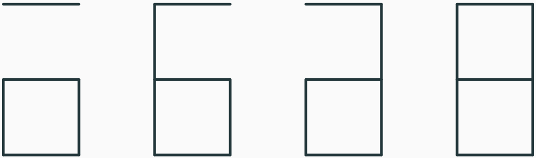
- Say the lower block is of size $1$
- Then we have a number $a \in [n]$


$a$

# Valid skeletons for $k = 2$

- Say the lower block is of size $1$
- Then we have a number $a \in [n]$


$a$



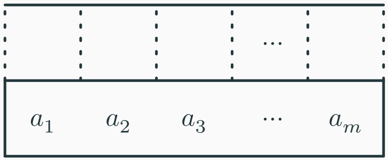
# Detour: Large blocks

- What about blocks of size $4$ or larger?

**Lemma**

In any valid trace, of any order, blocks can only be of size $4$ or greater in the first row

- Assume there is a block, not on the first row, with numbers $a_1, \dots, a_m$, $m \geq 4$
- Valid trace: $a_1 > a_2 > \cdots > a_m$



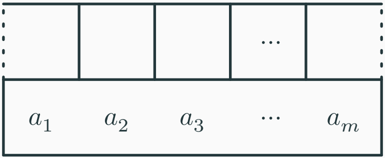
# Detour: Large blocks

- What about blocks of size $4$ or larger?

**Lemma**

In any valid trace, of any order, blocks can only be of size $4$ or greater in the first row

- Assume there is a block, not on the first row, with numbers $a_1, \ldots, a_m$, $m \geq 4$
- Valid trace: $a_1 > a_2 > \cdots > a_m$



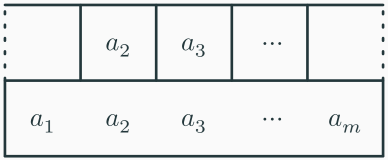
# Detour: Large blocks

- What about blocks of size $4$ or larger?

**Lemma**

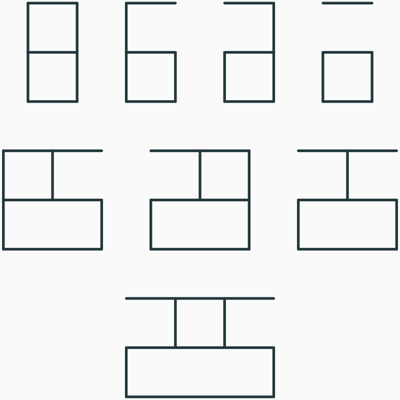
In any valid trace, of any order, blocks can only be of size $4$ or greater in the first row

- Assume there is a block, not on the first row, with numbers $a_1, \dots, a_m$, $m \geq 4$
- Valid trace: $a_1 > a_2 > \cdots > a_m$



# Valid skeletons for $k = 2$

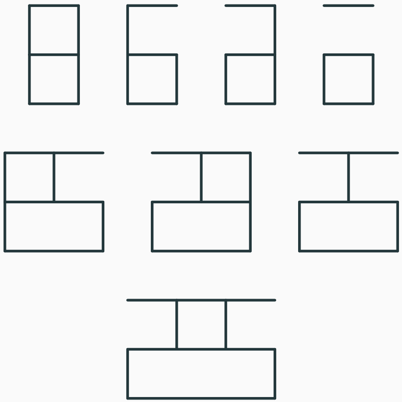
- No pieces of size $4$ or greater
- We have restricted the set of possible pieces to the following:



- These pieces are "necessary"

# Valid skeletons for $k = 2$

- No pieces of size $4$ or greater
- We have restricted the set of possible pieces to the following:

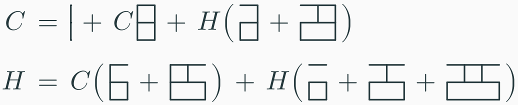


- These pieces are “necessary”
- Turns out they are also “sufficient”!

# Valid skeletons for $k = 2$

- We can now count the valid skeletons. Building them incrementally from left to right, let
  - $C$ be the partial skeletons that end with closed right boundary, and
  - $H$ be the partial skeletons that end with half-closed right boundary.

  Then



- Using the formal variable $x$ to keep track of the length of the partial skeleton:

$$C = 1 + xC + (x + x^2)H$$
$$H = (x + x^2)C + (x + x^2 + x^3)H$$

- Solving for $C$ gives:

$$C = (x^3 + x^2 + x - 1)/(2x^3 + x^2 + 2x - 1)$$

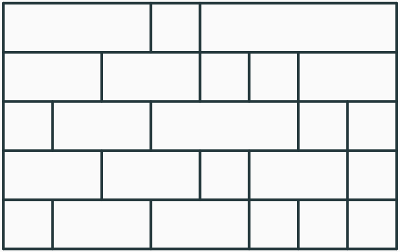
# Valid skeletons in general

- Let's now consider skeletons of some order $k$
- Recall the two conditions that determine if a skeleton is valid:
  1. (1) The numbers within each block are in decreasing order
  2. (2) Adjacent numbers in different blocks form an ascent

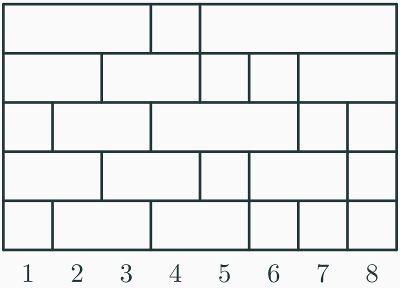
# Valid skeletons in general

- Let's now consider skeletons of some order $k$
- Recall the two conditions that determine if a skeleton is valid:
  (1) The numbers within each block are in decreasing order
  (2) Adjacent numbers in different blocks form an ascent

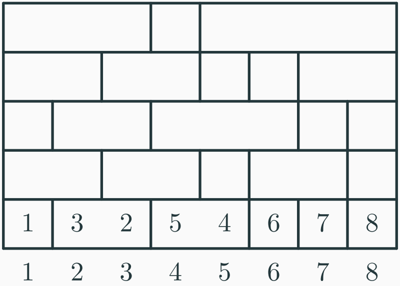
# Decreasing blocks



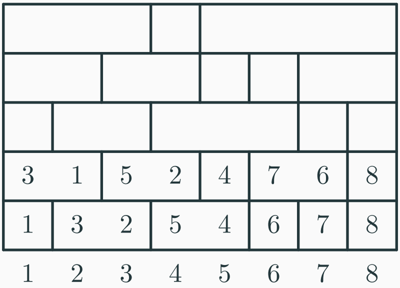
# Decreasing blocks



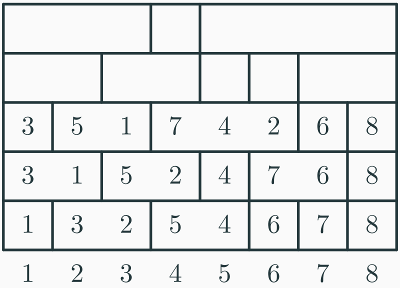
# Decreasing blocks



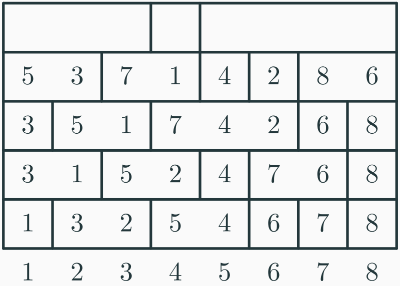
# Decreasing blocks



# Decreasing blocks



# Decreasing blocks



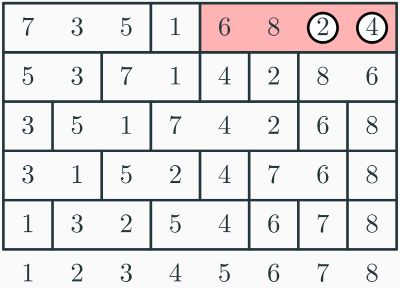
# Decreasing blocks

<table>
<tr><td colspan="3">7 3 5</td><td>1</td><td colspan="4">6 8 2 4</td></tr>
<tr><td colspan="2">5 3</td><td colspan="2">7 1</td><td>4</td><td>2</td><td colspan="2">8 6</td></tr>
<tr><td>3</td><td colspan="2">5 1</td><td colspan="3">7 4 2</td><td>6</td><td>8</td></tr>
<tr><td colspan="2">3 1</td><td colspan="2">5 2</td><td>4</td><td colspan="2">7 6</td><td>8</td></tr>
<tr><td>1</td><td colspan="2">3 2</td><td colspan="2">5 4</td><td>6</td><td>7</td><td>8</td></tr>
<tr><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td></tr>
</table>

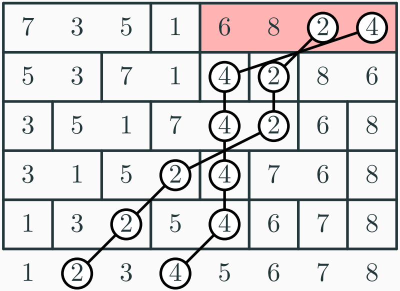
# Decreasing blocks

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | 3 | 5 | 1 | 6 | 8 | 2 | 4 |
| 5 | 3 | 7 | 1 | 4 | 2 | 8 | 6 |
| 3 | 5 | 1 | 7 | 4 | 2 | 6 | 8 |
| 3 | 1 | 5 | 2 | 4 | 7 | 6 | 8 |
| 1 | 3 | 2 | 5 | 4 | 6 | 7 | 8 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

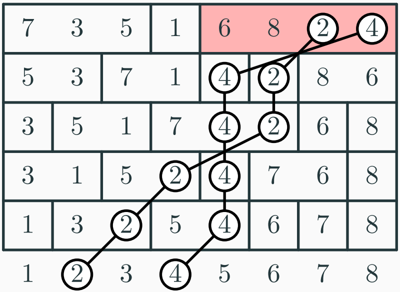
# Decreasing blocks



# Decreasing blocks

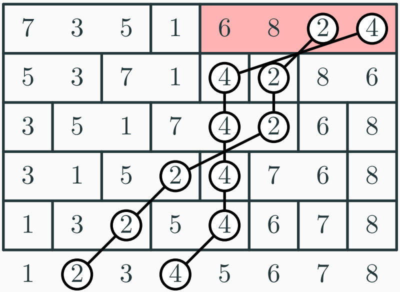


# Decreasing blocks




- They start in increasing order in the bottom permutation
- Every time they appear in a block together, their relative order changes
- In particular, they will be in increasing order the second time they appear together in a block

# Decreasing blocks




- They start in increasing order in the bottom permutation
- Every time they appear in a block together, their relative order changes
- In particular, they will be in increasing order the second time they appear together in a block—a violation of the condition!

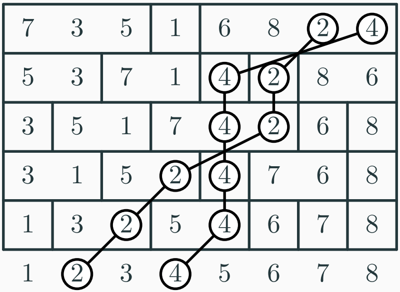
# Decreasing blocks

**Lemma**

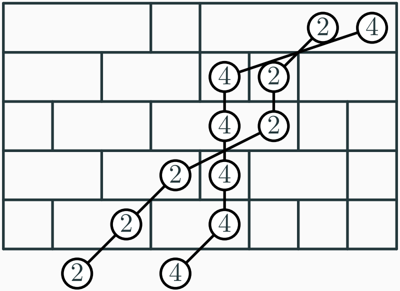
A skeleton satisfies the first condition if and only if, for each pair of numbers $a, b$ in the corresponding trace, the numbers $a, b$ appear at most once together in a block.

- Can we check whether a skeleton satisfies this without looking at the corresponding trace?

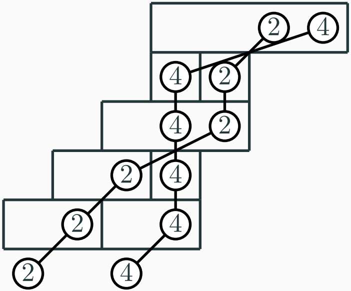
# Decreasing blocks



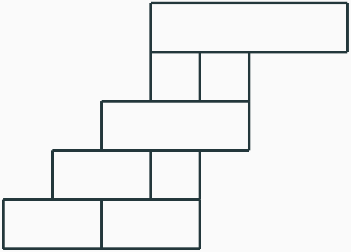
# Decreasing blocks



# Decreasing blocks

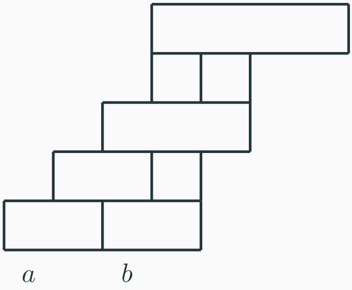


# Decreasing blocks



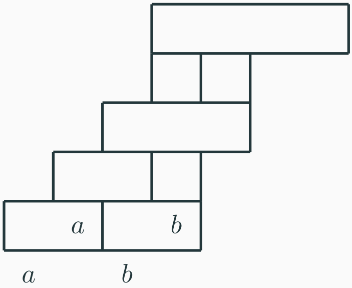
- Any skeleton of order $5$ containing this fragment is invalid
- We call this a forbidden fragment

# Decreasing blocks



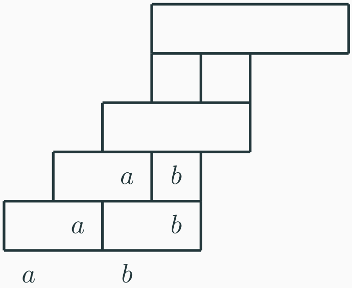

- Any skeleton of order $5$ containing this fragment is invalid
- We call this a forbidden fragment

# Decreasing blocks



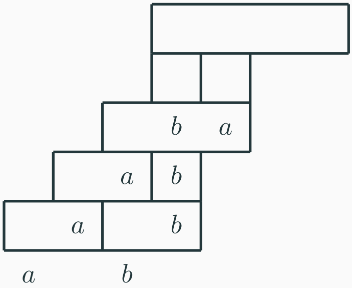

- Any skeleton of order $5$ containing this fragment is invalid
- We call this a forbidden fragment

# Decreasing blocks



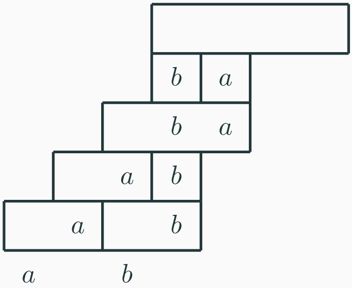

- Any skeleton of order $5$ containing this fragment is invalid
- We call this a forbidden fragment

# Decreasing blocks



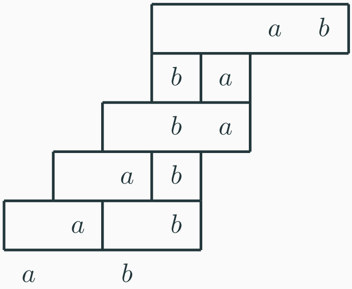

- Any skeleton of order $5$ containing this fragment is invalid
- We call this a forbidden fragment

# Decreasing blocks




- Any skeleton of order $5$ containing this fragment is invalid
- We call this a forbidden fragment

# Decreasing blocks




- Any skeleton of order $5$ containing this fragment is invalid
- We call this a forbidden fragment

# Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:

1
2
3
4
5
6
7
8

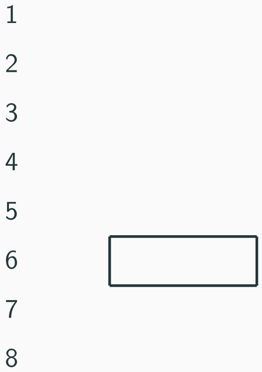
# Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:

1
2
3
4
5
6
7
8

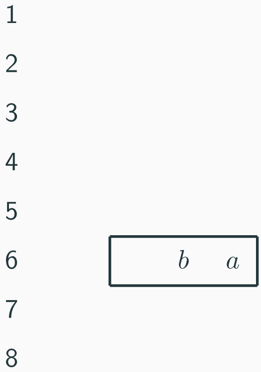
# Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:



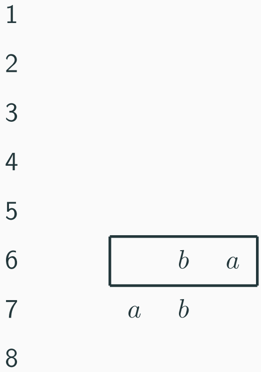
# Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:



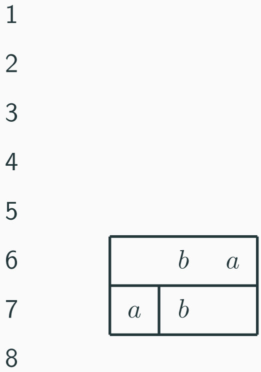
# Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:



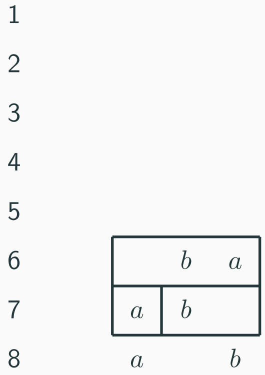
# Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:



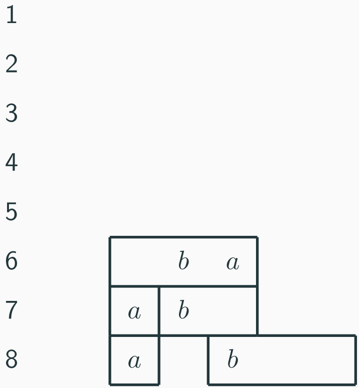
# Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:



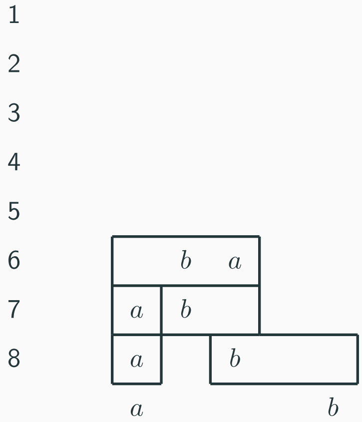
# Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:



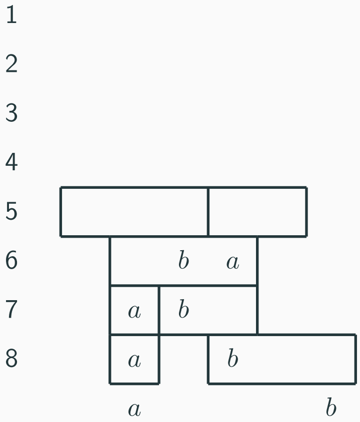
# Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:



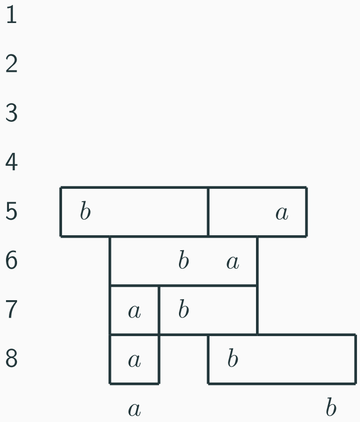
# Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:



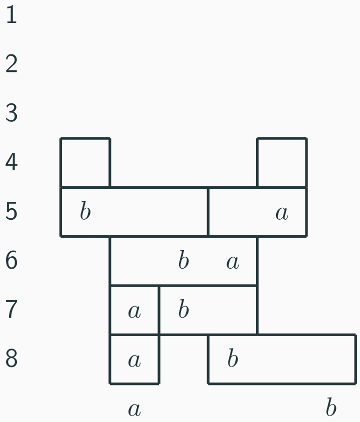
## Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:



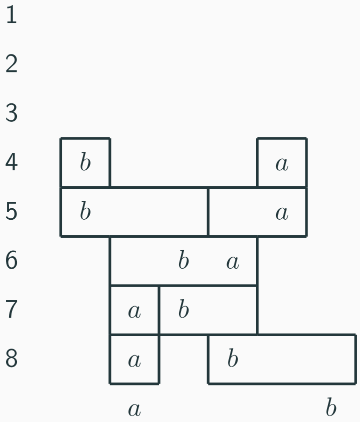
# Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:



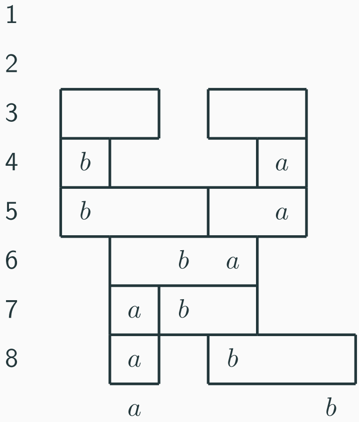
## Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:



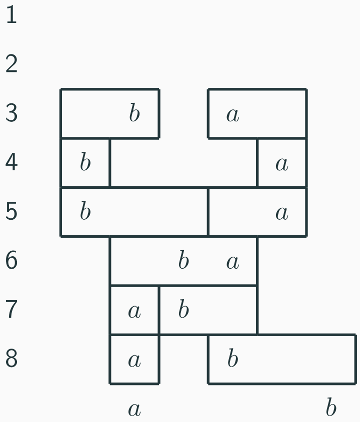
# Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:



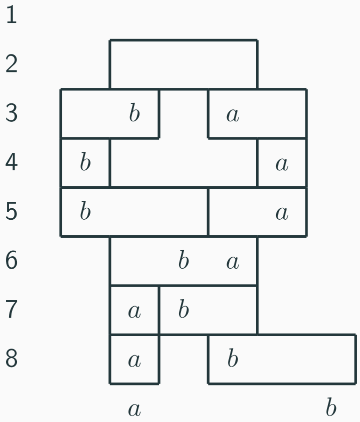
# Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:



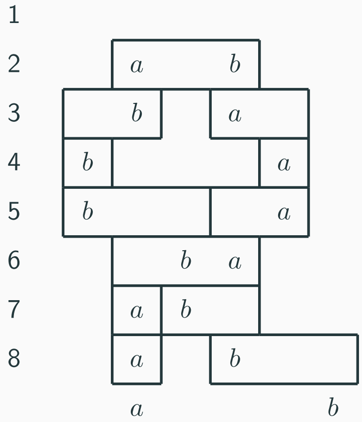
# Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:



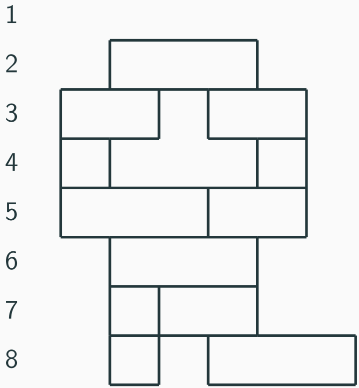
# Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:



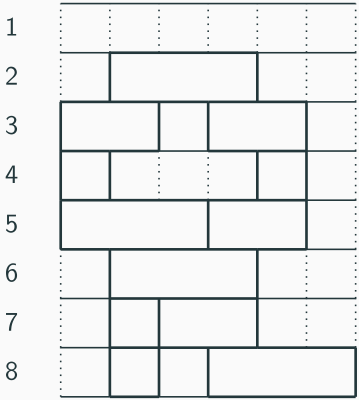
# Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:



# Decreasing blocks — Forbidden fragments

- We can list all the minimal forbidden fragments that cause two elements to appear at least twice together in a block:

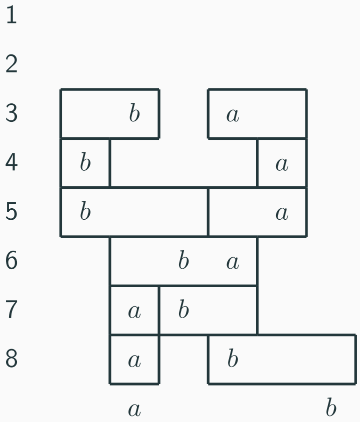


# Decreasing blocks — Forbidden fragments

- Special case: The second time the two numbers appear together in a block, they are in a large block on the first row

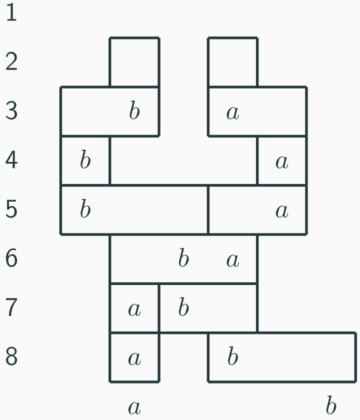
# Decreasing blocks — Forbidden fragments

- Special case: The second time the two numbers appear together in a block, they are in a large block on the first row



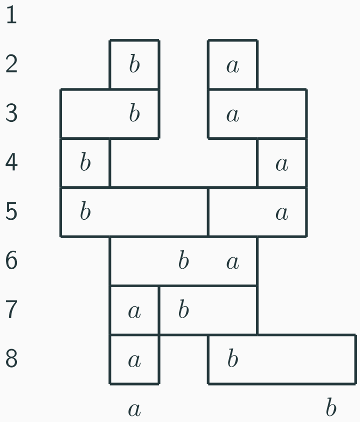
# Decreasing blocks — Forbidden fragments

- Special case: The second time the two numbers appear together in a block, they are in a large block on the first row



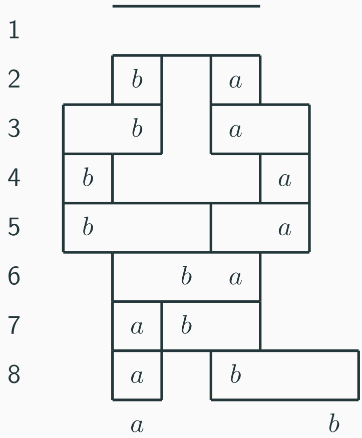
# Decreasing blocks — Forbidden fragments

- Special case: The second time the two numbers appear together in a block, they are in a large block on the first row



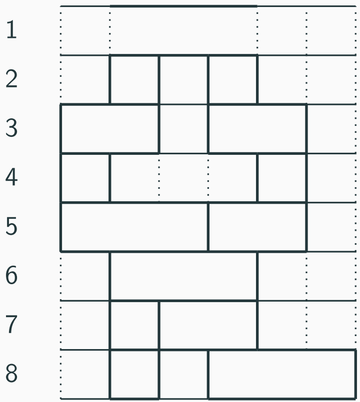
# Decreasing blocks — Forbidden fragments

- Special case: The second time the two numbers appear together in a block, they are in a large block on the first row



# Decreasing blocks — Forbidden fragments

- Special case: The second time the two numbers appear together in a block, they are in a large block on the first row



# Decreasing blocks — Forbidden fragments

- When generating these forbidden fragments for a fixed $k$:
  - There are $k-1$ choices for the row where the numbers first occur together in a block
  - There are $2$ choices for the size of this block
  - There are at most $2$ choices for how they are placed inside this block
  - Since each block is of size at most $3$, the distance the two numbers can travel away from this first block is bounded by $2k$
- There are finitely many forbidden fragments for the first condition
- We can list all of them, and (somehow) remove the skeletons that contain at least one of them

# Valid skeletons in general

- Recall the two conditions that determine if a skeleton is valid:
  (1) The numbers within each block are in decreasing order
  (2) Adjacent numbers in different blocks form an ascent

# Valid skeletons in general

- Recall the two conditions that determine if a skeleton is valid:
  (1) The numbers within each block are in decreasing order
  (2) Adjacent numbers in different blocks form an ascent
- Now assume the skeletons satisfy the first condition

# Valid skeletons in general

- Recall the two conditions that determine if a skeleton is valid:
  (1) The numbers within each block are in decreasing order
  (2) Adjacent numbers in different blocks form an ascent
- Now assume the skeletons satisfy the first condition

# Ascent across boundary

- Let's take another look at the invalid skeleton from before:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | 3 | 5 | 1 | 6 | 8 | 2 | 4 |
| 5 | 3 | 7 | 1 | 4 | 2 | 8 | 6 |
| 3 | 5 | 1 | 7 | 4 | 2 | 6 | 8 |
| 3 | 1 | 5 | 2 | 4 | 7 | 6 | 8 |
| 1 | 3 | 2 | 5 | 4 | 6 | 7 | 8 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

# Ascent across boundary

- Let's take another look at the invalid skeleton from before:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | 3 | 5 | 1 | 6 | 8 | 2 | 4 |
| 5 | 3 | 7 | 1 | 4 | 2 | 8 | 6 |
| 3 | 5 | 1 | 7 | 4 | 2 | 6 | 8 |
| 3 | 1 | 5 | 2 | 4 | 7 | 6 | 8 |
| 1 | 3 | 2 | 5 | 4 | 6 | 7 | 8 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

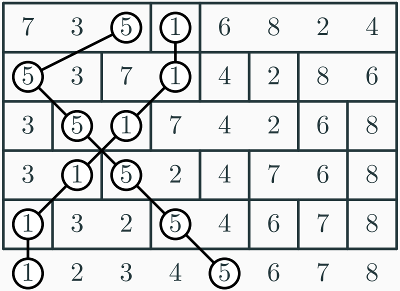
# Ascent across boundary

- Let's take another look at the invalid skeleton from before:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 7 | 3 | 5 | 1 | 6 | 8 | 2 | 4 |
| 5 | 3 | 7 | 1 | 4 | 2 | 8 | 6 |
| 3 | 5 | 1 | 7 | 4 | 2 | 6 | 8 |
| 3 | 1 | 5 | 2 | 4 | 7 | 6 | 8 |
| 1 | 3 | 2 | 5 | 4 | 6 | 7 | 8 |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

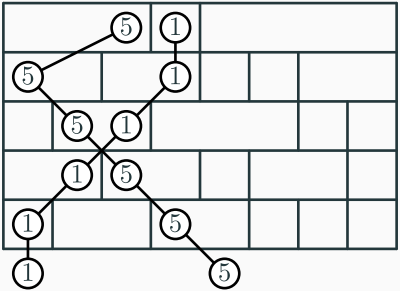
# Ascent across boundary

- Let's take another look at the invalid skeleton from before:



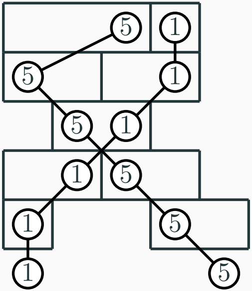
# Ascent across boundary

- Let's take another look at the invalid skeleton from before:



# Ascent across boundary

- Let's take another look at the invalid skeleton from before:



# Ascent across boundary

**Lemma**

A skeleton satisfies the second condition if and only if, for each pair of numbers $a, b$ in the corresponding trace, the numbers $a, b$ are never adjacent, separated by a block boundary, and appearing an odd number of times together in a block on the rows below.

# Ascent across boundary

**Lemma**

A skeleton satisfies the second condition if and only if, for each pair of numbers $a, b$ in the corresponding trace, the numbers $a, b$ are never adjacent, separated by a block boundary, and appearing together in a block on the rows below.

## Ascent across boundary — Forbidden fragments

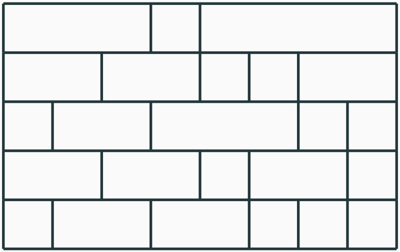
- We can generate the minimal forbidden fragments for the second condition in the same manner as for the first condition
- Again there will be finitely many of them

# Forbidden fragments

- We now have a finite set of these forbidden fragments
  - Finite fragments of a skeleton, that may contain block boundaries that are “wildcards”
- A skeleton is valid if and only if it avoids these forbidden fragments
- Can we use this characterization to enumerate the valid skeletons?

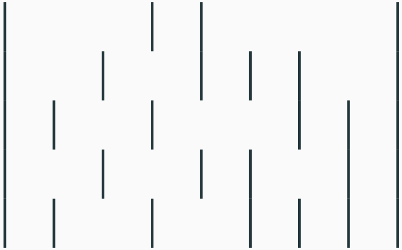
# Formal language

- Let's encode skeletons as a formal language



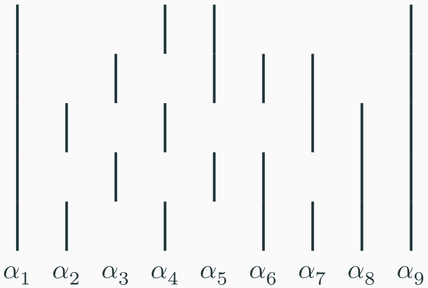
# Formal language

- Let's encode skeletons as a formal language



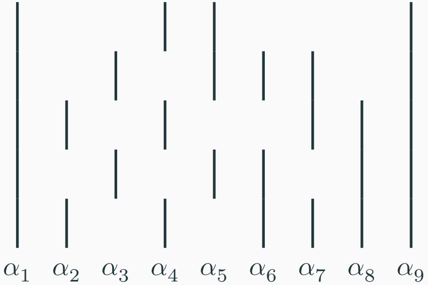
# Formal language

- Let's encode skeletons as a formal language



# Formal language

- Let's encode skeletons as a formal language




- The alphabet $\Sigma$ consists of the $2^k$ possible columns in a skeleton
- Let $S$ be the language that consists of skeletons:
  - First and last columns are solid
  - No block is of size greater than 3, except in the first row
- We can make a DFA that accepts $S$, so it is regular

# Avoiding forbidden fragments

- We want the set of skeletons that avoid all the forbidden fragments
- For a given forbidden fragment, we can create a DFA that accepts the set of skeletons that contain that particular fragment
- Taking the complement of the DFA gives us the set of skeletons that avoid it
- Doing this for all the forbidden fragments, and then taking the intersection of all the resulting DFA, we get a DFA for the set of valid skeletons!

# Generating function from DFA

- We want to count the valid skeletons of length $n$
  - In terms of the language, the skeletons that have $n+1$ columns
- Want to count how many strings of length $n+1$ our DFA accepts
- A system of linear equations gives us the generating function
  - accepted strings of length $n+1$
  - valid skeletons of length $n$
  - $k$-pop-stack-sortable permutations of length $n$

# Rational generating function

- The generating function for the set of accepted strings of a DFA is rational

**Theorem**

For any fixed $k$, the generating function for the $k$-pop-stack-sortable permutations is rational.

# Deriving the generating functions

- Theoretical result is nice, but can we actually derive the generating functions?
- Carrying out the calculations by hand is impractical
- Instead we implemented the whole procedure so that it could be carried out by a computer
- Used the Garpur cluster to crunch out the generating functions for $k \leq 6$

# Results

| $k$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Forbidden fragments | 0 | 8 | 85 | 2451 | 686 485 | 3 581 406 |
| Vertices in DFA | 2 | 4 | 11 | 31 | 99 | 339 |
| Edges in DFA | 4 | 10 | 33 | 119 | 477 | 2010 |
| Degree of GF | 1 | 3 | 10 | 25 | 71 | 213 |

# Generating functions

| $k$ | Generating function |
|---|---|
| 1 | $(x-1)/(2x-1)$ |
| 2 | $(x^3+x^2+x-1)/(2x^3+x^2+2x-1)$ |
| 3 | $(2x^{10}+4x^9+2x^8+5x^7+11x^6+8x^5+6x^4+6x^3+2x^2+x-1)/(4x^{10}+8x^9+4x^8+10x^7+22x^6+16x^5+8x^4+6x^3+2x^2+2x-1)$ |
| 4 | $(64x^{25}+448x^{24}+1184x^{23}+1784x^{22}+2028x^{21}+1948x^{20}+1080x^{19}+104x^{18}-180x^{17}+540x^{16}+1156x^{15}+696x^{14}+252x^{13}+238x^{12}+188x^{11}+502x^{10}+806x^9+544x^8+263x^7+185x^6+99x^5+33x^4+13x^3+3x^2+x-1)/(128x^{25}+896x^{24}+2368x^{23}+3568x^{22}+3928x^{21}+3064x^{20}+176x^{19}-2304x^{18}-2664x^{17}-1580x^{16}-352x^{15}-576x^{14}-1104x^{13}-760x^{12}-138x^{11}+686x^{10}+1238x^9+869x^8+382x^7+210x^6+102x^5+27x^4+12x^3+3x^2+2x-1)$ |