Pop Stack Sorting with a Bypass

Lapo Cioni [1] Luca Ferrari [1] Rebecca Smith [2]

[1]University of Firenze

[2]SUNY Brockport

July 7, 2025

The Plan:

Introduction

The Plan:

Introduction

Sorting with a pop stack with a bypass

The Plan:

Introduction

Sorting with a pop stack with a bypass

The pop stack with a bypass map

The Plan:

Introduction

Sorting with a pop stack with a bypass

The pop stack with a bypass map

Two pop stacks in parallel with a bypass

The Plan:

Introduction

Sorting with a pop stack with a bypass

The pop stack with a bypass map

Two pop stacks in parallel with a bypass

More pop stacks in parallel with a bypass

Introduction

Definition

A *stack* is a LIFO (last-in, first-out) sorting device with push and pop operations.

Definition

A *stack* is a LIFO (last-in, first-out) sorting device with push and pop operations.

The permutation $\pi = \pi_1\pi_2\dots\pi_n$ can be sorted (that is, by applying push and pop operations one can output the identity $123\dots n$) exactly if π avoids 231 (Knuth).

A restricted version of a stack that has received more attention recently is a pop stack, first introduced by Avis and Newborn in 1981.

A restricted version of a stack that has received more attention recently is a pop stack, first introduced by Avis and Newborn in 1981.

Definition

A *pop stack* has the same push and pop operations as a stack, but whenever a pop operation occurs, every entry in the pop stack is popped immediately.

A restricted version of a stack that has received more attention recently is a pop stack, first introduced by Avis and Newborn in 1981.

Definition
A *pop stack* has the same push and pop operations as a stack, but whenever a pop operation occurs, every entry in the pop stack is popped immediately.

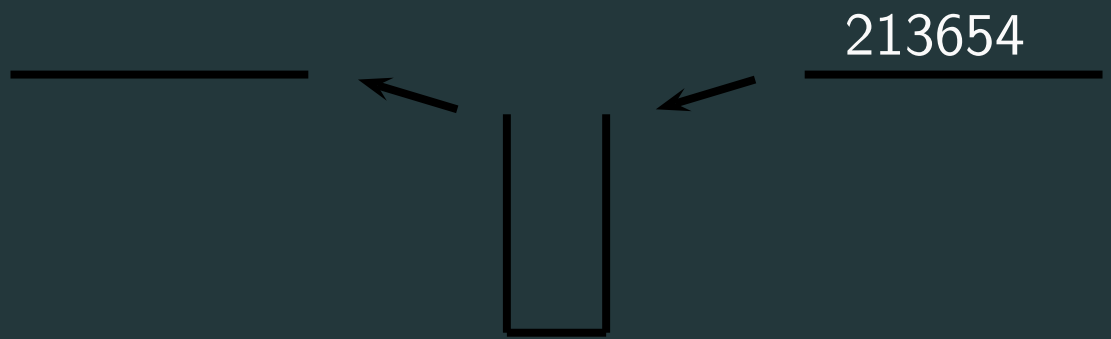


Figure 1: Sorting 213654 with a pop stack

A restricted version of a stack that has received more attention recently is a pop stack, first introduced by Avis and Newborn in 1981.

Definition
A *pop stack* has the same push and pop operations as a stack, but whenever a pop operation occurs, every entry in the pop stack is popped immediately.

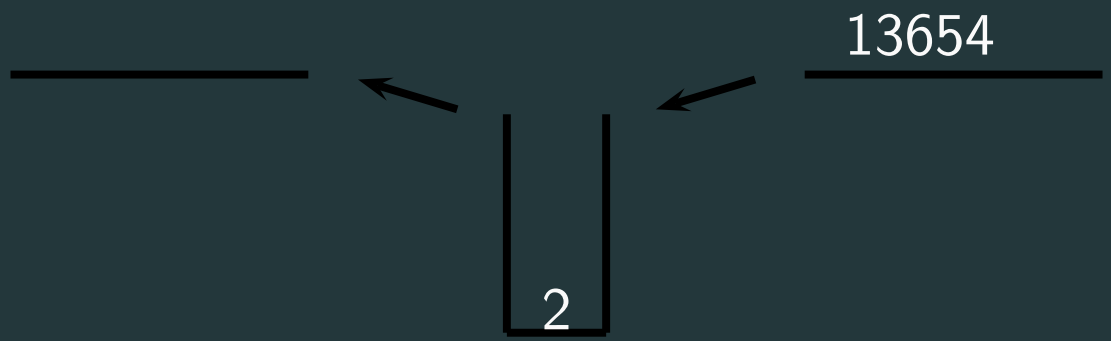


Figure 1: Sorting 213654 with a pop stack

A restricted version of a stack that has received more attention recently is a pop stack, first introduced by Avis and Newborn in 1981.

Definition
A *pop stack* has the same push and pop operations as a stack, but whenever a pop operation occurs, every entry in the pop stack is popped immediately.

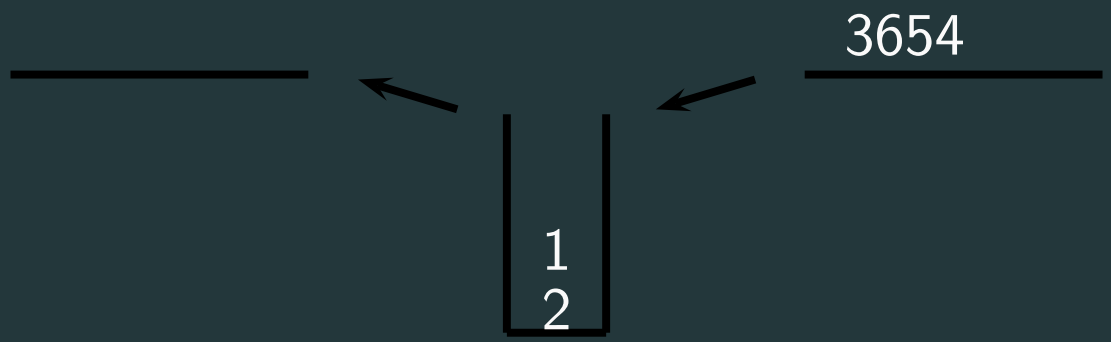


Figure 1: Sorting 213654 with a pop stack

A restricted version of a stack that has received more attention recently is a pop stack, first introduced by Avis and Newborn in 1981.

Definition
A *pop stack* has the same push and pop operations as a stack, but whenever a pop operation occurs, every entry in the pop stack is popped immediately.

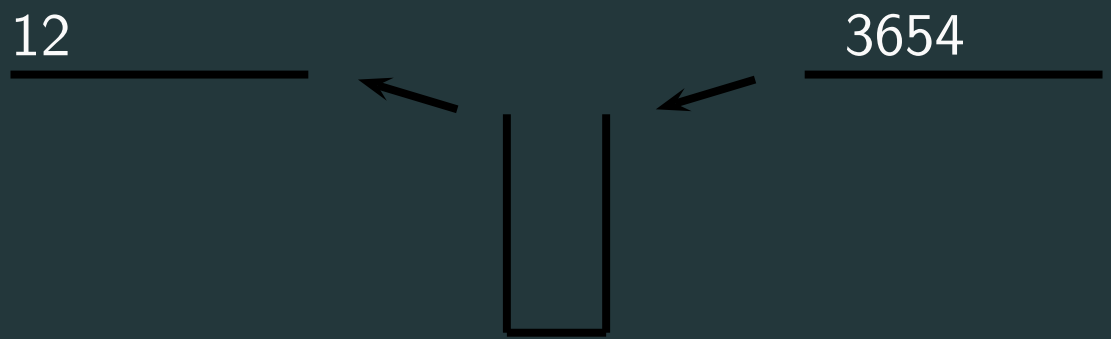


Figure 1: Sorting 213654 with a pop stack

A restricted version of a stack that has received more attention recently is a pop stack, first introduced by Avis and Newborn in 1981.

Definition
A *pop stack* has the same push and pop operations as a stack, but whenever a pop operation occurs, every entry in the pop stack is popped immediately.

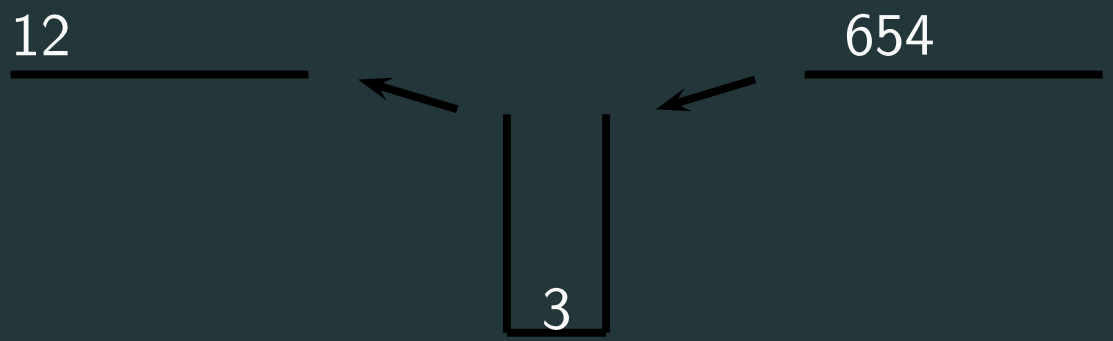


Figure 1: Sorting 213654 with a pop stack

A restricted version of a stack that has received more attention recently is a pop stack, first introduced by Avis and Newborn in 1981.

Definition
A *pop stack* has the same push and pop operations as a stack, but whenever a pop operation occurs, every entry in the pop stack is popped immediately.

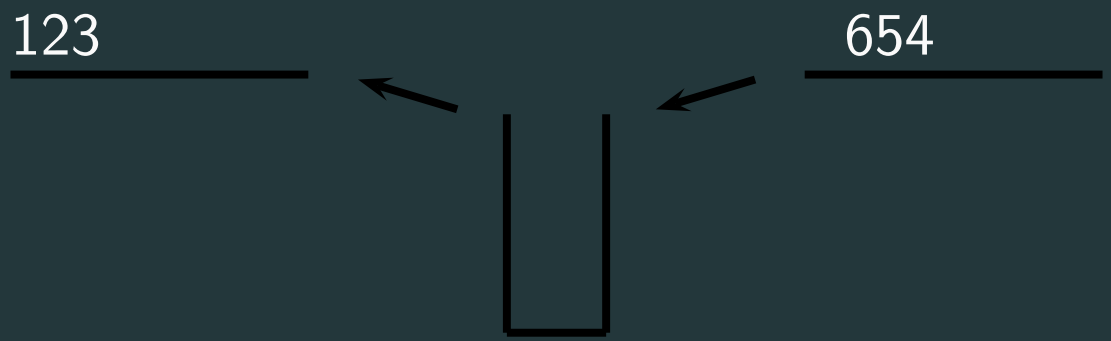


Figure 1: Sorting 213654 with a pop stack

A restricted version of a stack that has received more attention recently is a pop stack, first introduced by Avis and Newborn in 1981.

Definition
A *pop stack* has the same push and pop operations as a stack, but whenever a pop operation occurs, every entry in the pop stack is popped immediately.

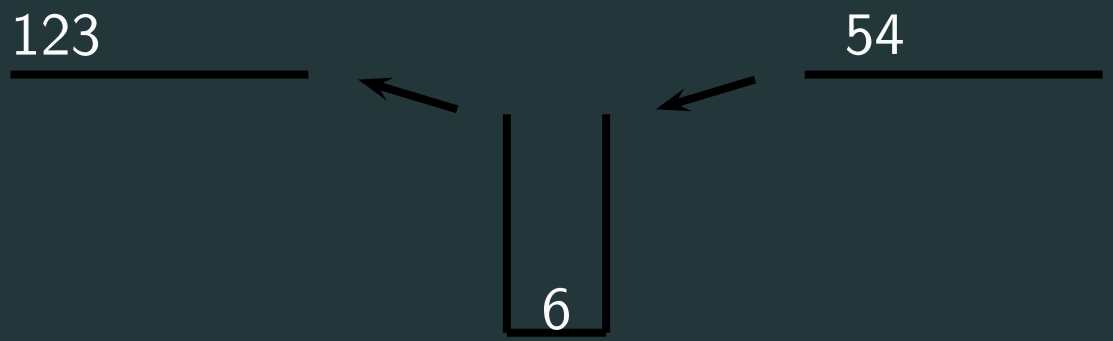


Figure 1: Sorting 213654 with a pop stack

A restricted version of a stack that has received more attention recently is a pop stack, first introduced by Avis and Newborn in 1981.

Definition
A *pop stack* has the same push and pop operations as a stack, but whenever a pop operation occurs, every entry in the pop stack is popped immediately.

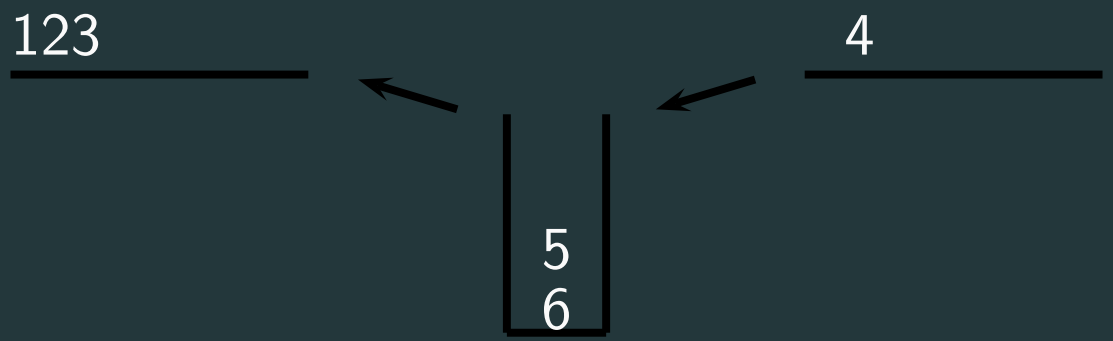


Figure 1: Sorting 213654 with a pop stack

A restricted version of a stack that has received more attention recently is a pop stack, first introduced by Avis and Newborn in 1981.

Definition
A *pop stack* has the same push and pop operations as a stack, but whenever a pop operation occurs, every entry in the pop stack is popped immediately.

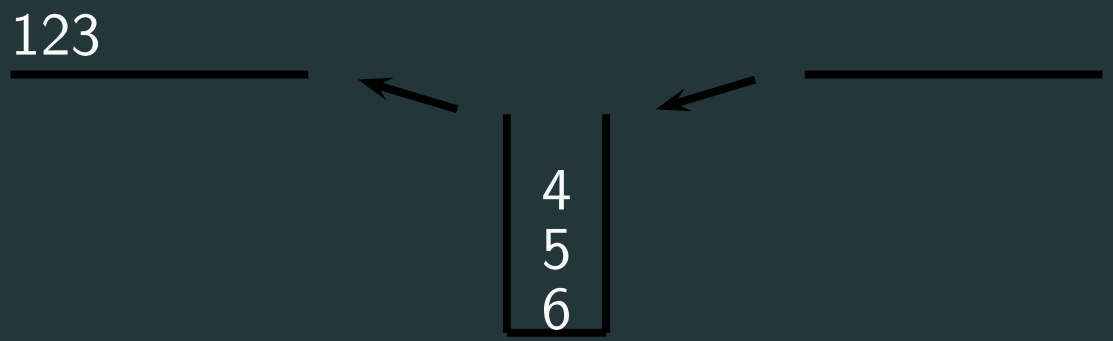


Figure 1: Sorting 213654 with a pop stack

A restricted version of a stack that has received more attention recently is a pop stack, first introduced by Avis and Newborn in 1981.

Definition
A *pop stack* has the same push and pop operations as a stack, but whenever a pop operation occurs, every entry in the pop stack is popped immediately.

Figure 1: Sorting 213654 with a pop stack

Permutations sortable by a single pop stack are exactly those that avoid 231 and 312, also known as the *layered* permutations.

Permutations sortable by a single pop stack are exactly those that avoid 231 and 312, also known as the *layered* permutations.

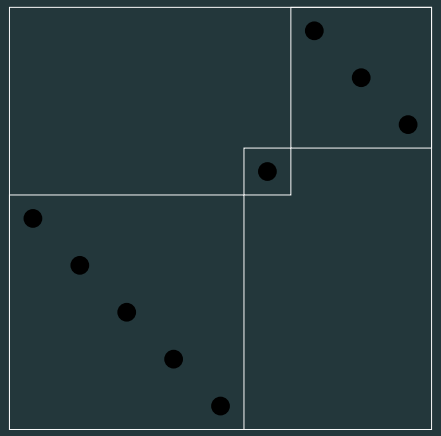


Figure 2: The layered permutation $\pi = 543216987$

Sorting with a pop stack with a bypass

We introduce a natural additional operation to the pop stack called a bypass allowing entries to skip the pop stack.

We introduce a natural additional operation to the pop stack called a bypass allowing entries to skip the pop stack.

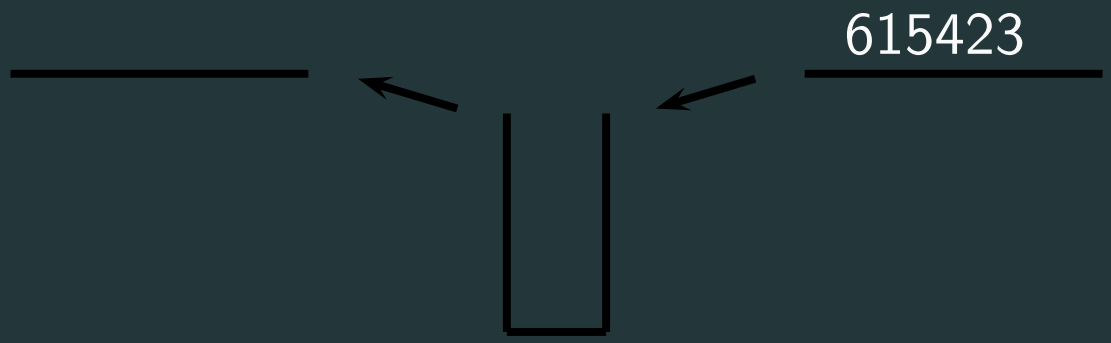


Figure 3: Sorting 615423 with a pop stack with a bypass

We introduce a natural additional operation to the pop stack called a bypass allowing entries to skip the pop stack.

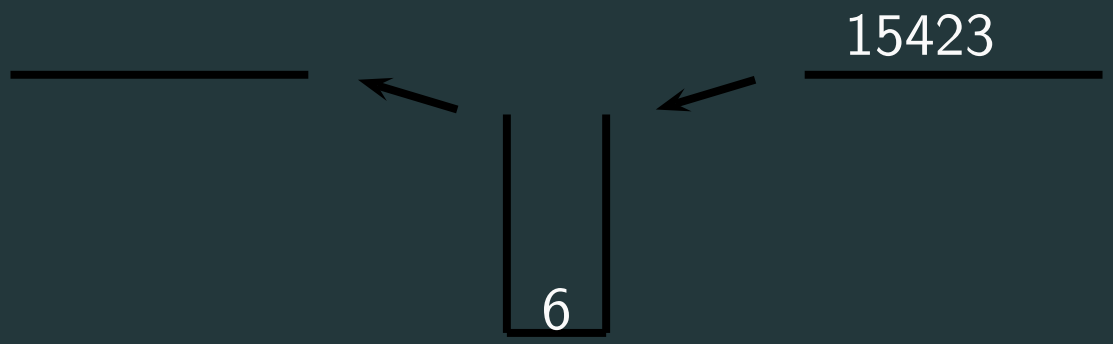


Figure 3: Sorting 615423 with a pop stack with a bypass

We introduce a natural additional operation to the pop stack called a bypass allowing entries to skip the pop stack.

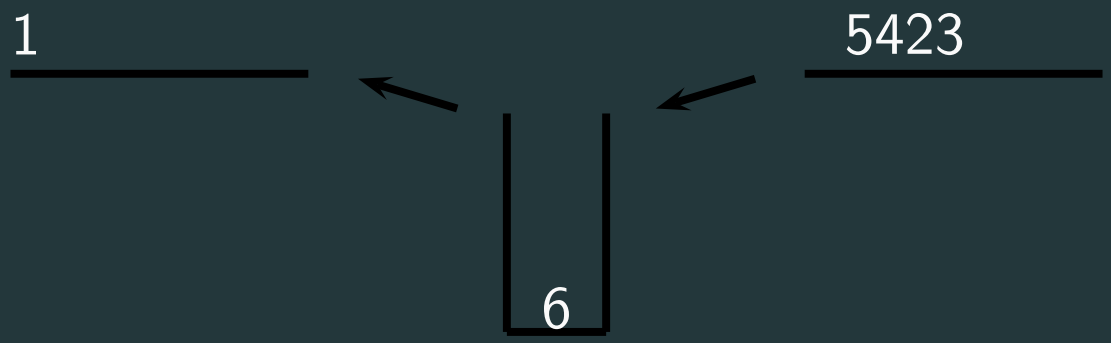


Figure 3: Sorting 615423 with a pop stack with a bypass

We introduce a natural additional operation to the pop stack called a bypass allowing entries to skip the pop stack.

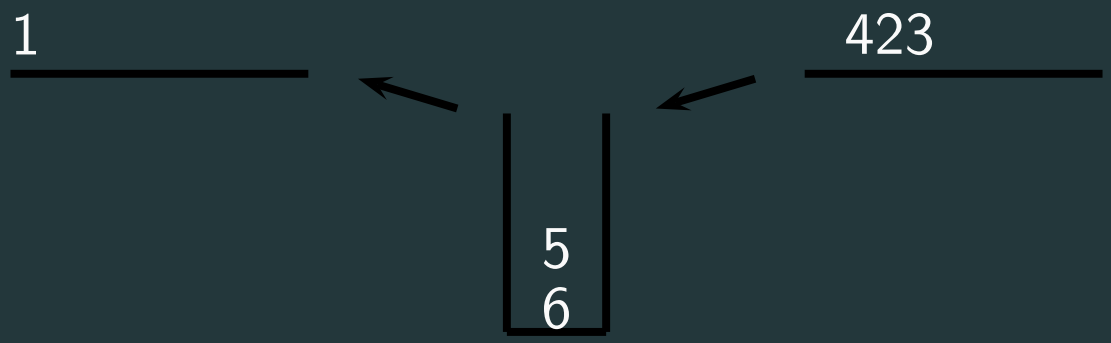


Figure 3: Sorting 615423 with a pop stack with a bypass

We introduce a natural additional operation to the pop stack called a bypass allowing entries to skip the pop stack.

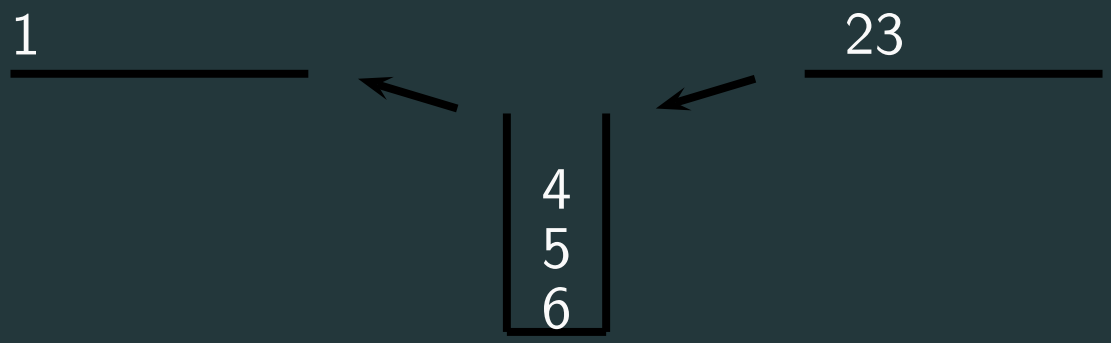


Figure 3: Sorting 615423 with a pop stack with a bypass

We introduce a natural additional operation to the pop stack called a bypass allowing entries to skip the pop stack.

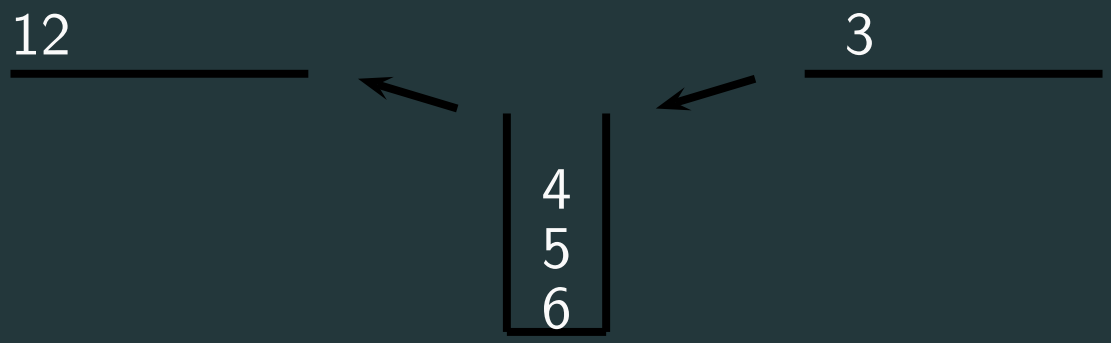


Figure 3: Sorting 615423 with a pop stack with a bypass

We introduce a natural additional operation to the pop stack called a bypass allowing entries to skip the pop stack.

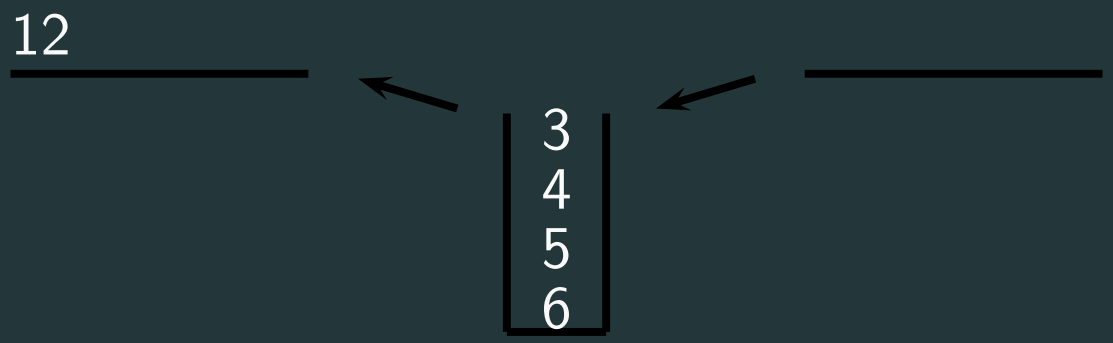


Figure 3: Sorting 615423 with a pop stack with a bypass

We introduce a natural additional operation to the pop stack called a bypass allowing entries to skip the pop stack.

Figure 3: Sorting 615423 with a pop stack with a bypass

Theorem

The permutations sortable by pop stack with a bypass form a class with basis $\{231, 4213\}$.

Theorem
The permutations sortable by pop stack with a bypass form a class with basis $\{231, 4213\}$.

The enumeration of these sortable permutations is shown by Atkinson to be the odd indexed Fibonacci numbers, namely $|\operatorname{Av}_n(231, 4213)| = F_{2n-1}$ where we take the liberty of defining $F_{-1} = 1$.

Theorem
The permutations sortable by pop stack with a bypass form a class with basis $\{231, 4213\}$.

The enumeration of these sortable permutations is shown by Atkinson to be the odd indexed Fibonacci numbers, namely $|\operatorname{Av}_n(231, 4213)| = F_{2n-1}$ where we take the liberty of defining $F_{-1} = 1$.

However, we can give an alternate proof of this enumeration and in the process construct a bijection between these sortable permutations and a restricted set of Motzkin paths.

```
1  S := ∅;
2  i := 1;
3  while i ≤ n do
4      if S = ∅ or π_i = TOP(S) − 1 then
5          PUSH;
6      else if π_i < TOP(S) − 1 then
7          BYPASS;
8      else
9          POP;
10         PUSH;
11     i := i + 1;
12 POP;
```

Algorithm 1: PSB (S is the pop stack; $\texttt{TOP}(S)$ is the current top element of the pop stack; $\pi = \pi_1 \cdots \pi_n$ is the input).

By this algorithm, each element of the input causes one option in the loop to occur which translate respectively to the following components of a Motzkin Path:

Push: Up/Northeast step

Bypass: Horizontal/East step

Pop, then push: Down/Southeast steps until reaching the horizontal axis followed by a single Up/Northeast step

By this algorithm, each element of the input causes one option in the loop to occur which translate respectively to the following components of a Motzkin Path:

Push: Up/Northeast step

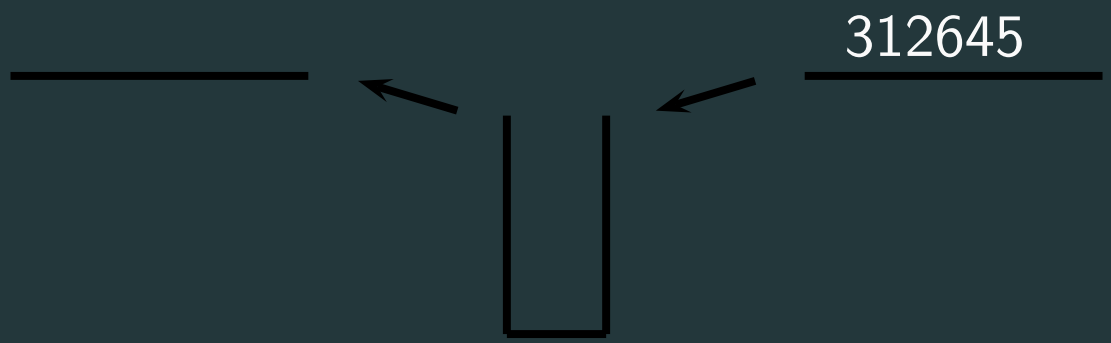
Bypass: Horizontal/East step

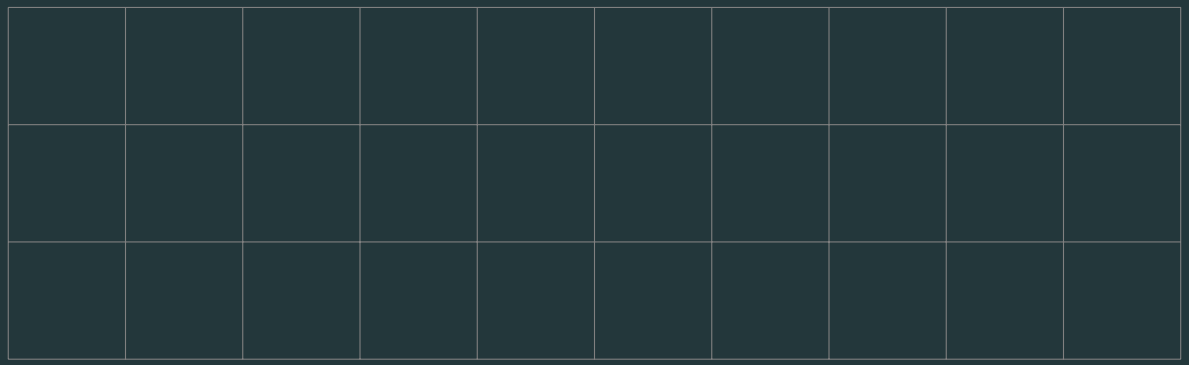
Pop, then push: Down/Southeast steps until reaching the horizontal axis followed by a single Up/Northeast step

Note the final pop of the popstack will also induce Down/Southeast steps until reaching the horizontal axis.

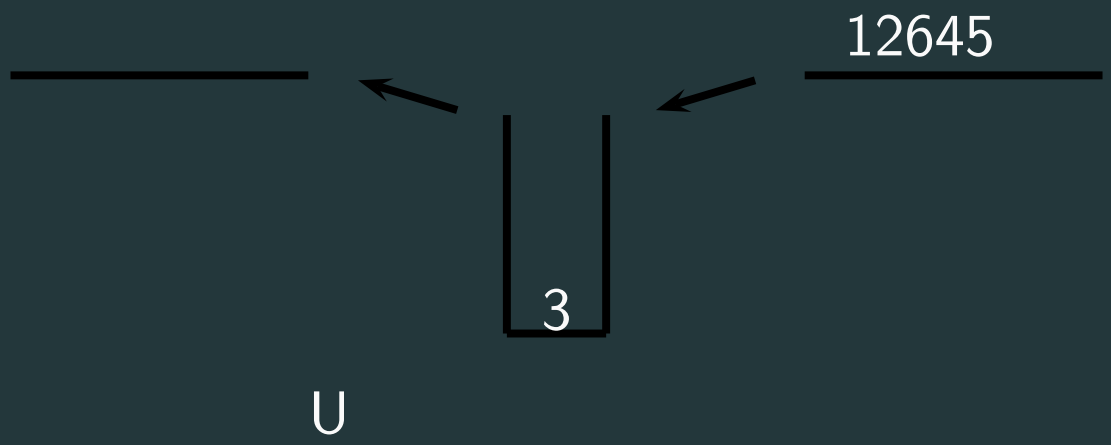
For example, we get the following Motzkin word by sorting 312645:

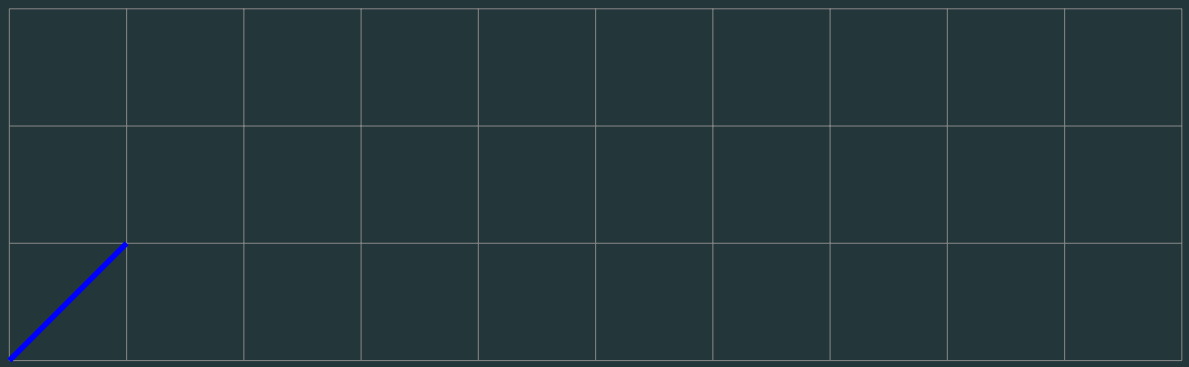
For example, we get the following Motzkin word by sorting 312645:



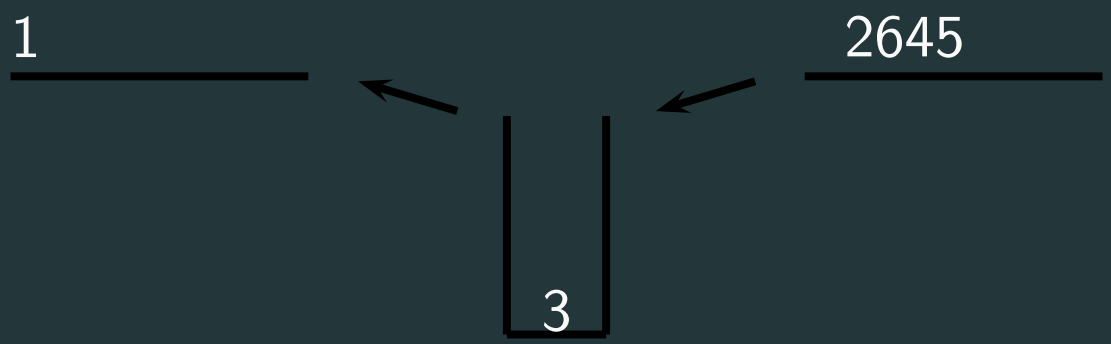


For example, we get the following Motzkin word by sorting 312645:

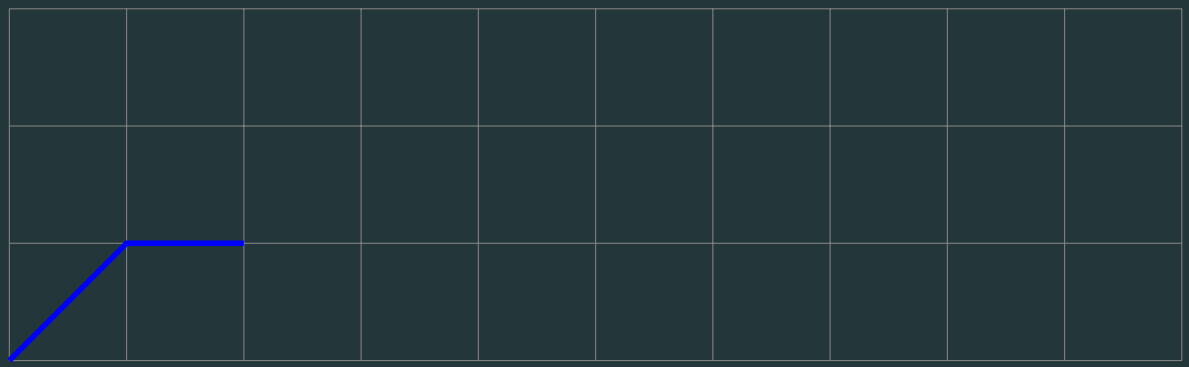




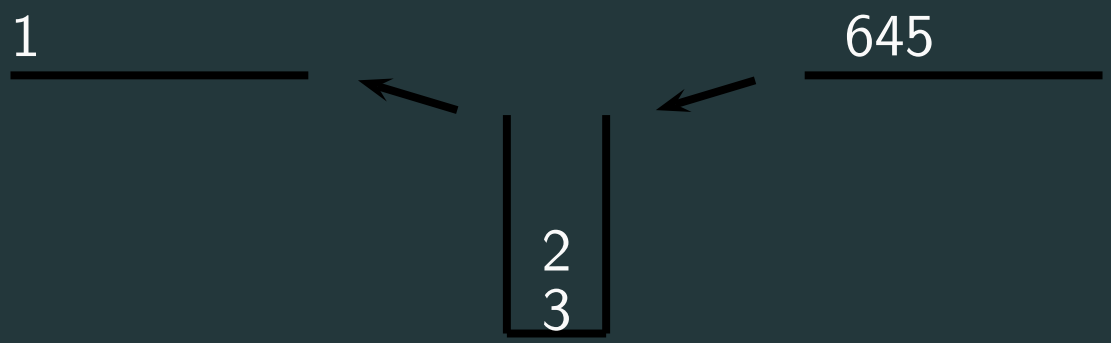
For example, we get the following Motzkin word by sorting 312645:



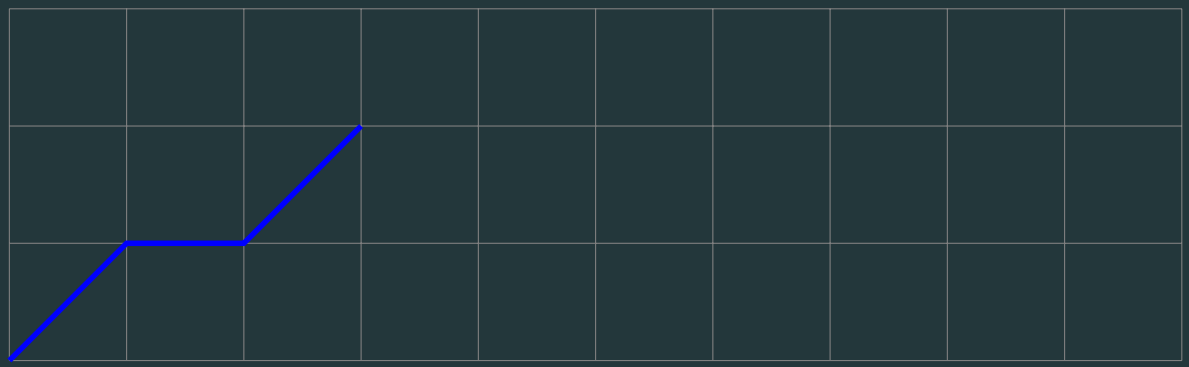
UH



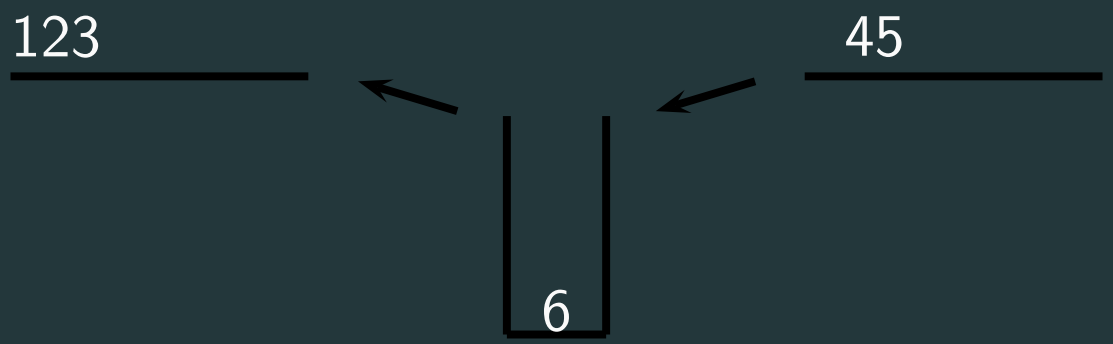
For example, we get the following Motzkin word by sorting 312645:



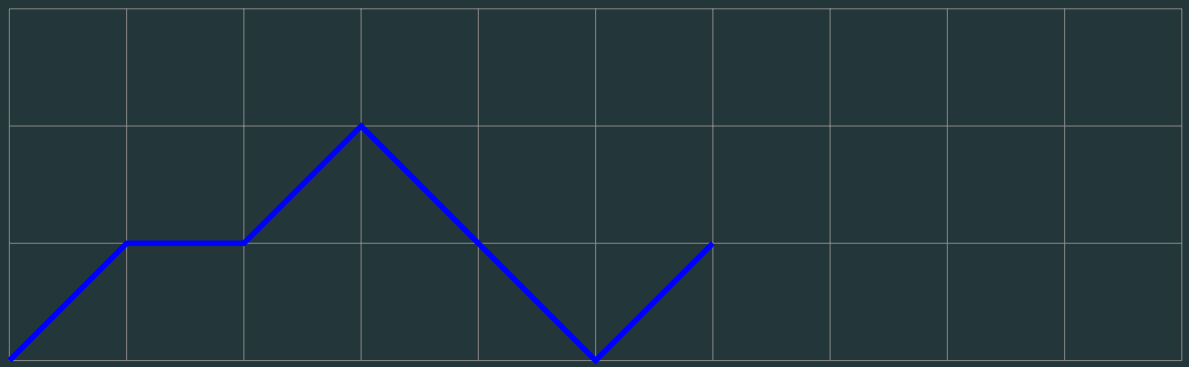
UHU



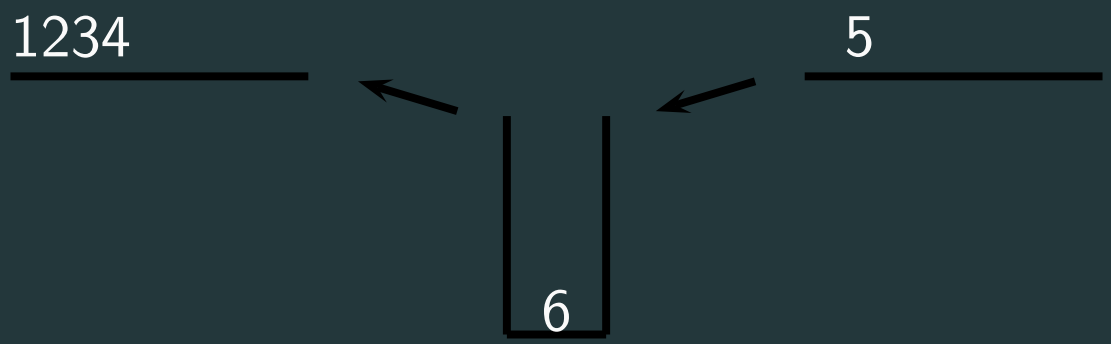
For example, we get the following Motzkin word by sorting 312645:



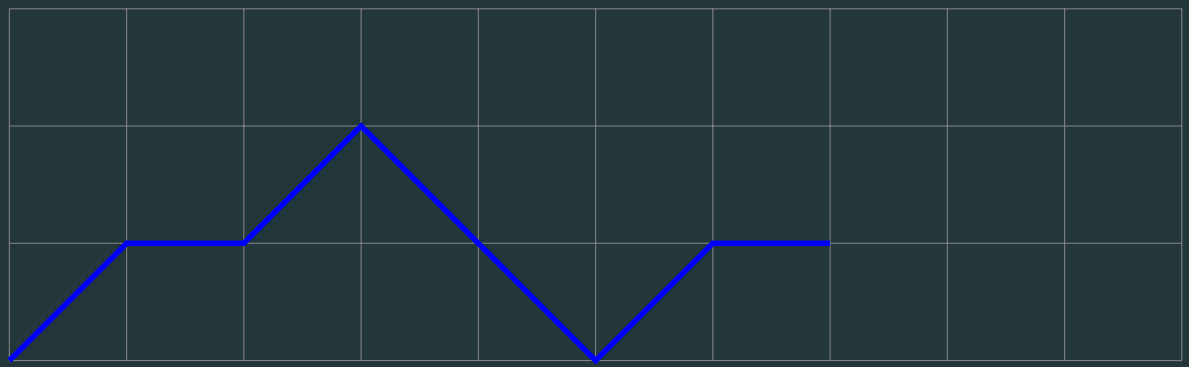
UHUDDU



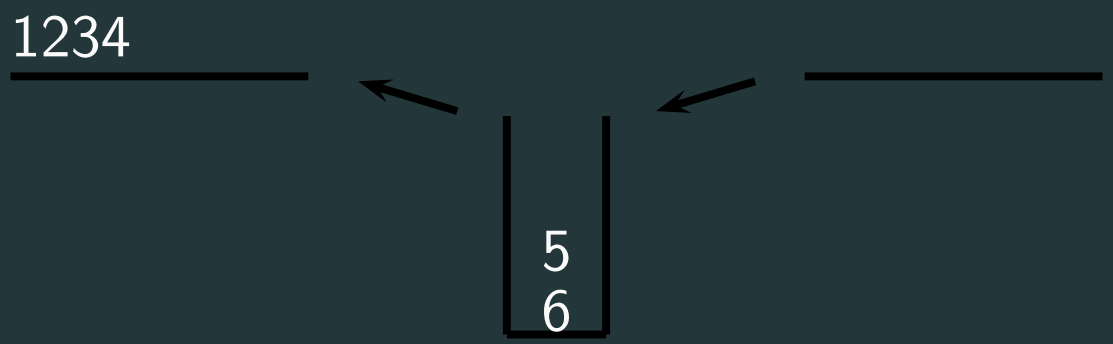
For example, we get the following Motzkin word by sorting 312645:



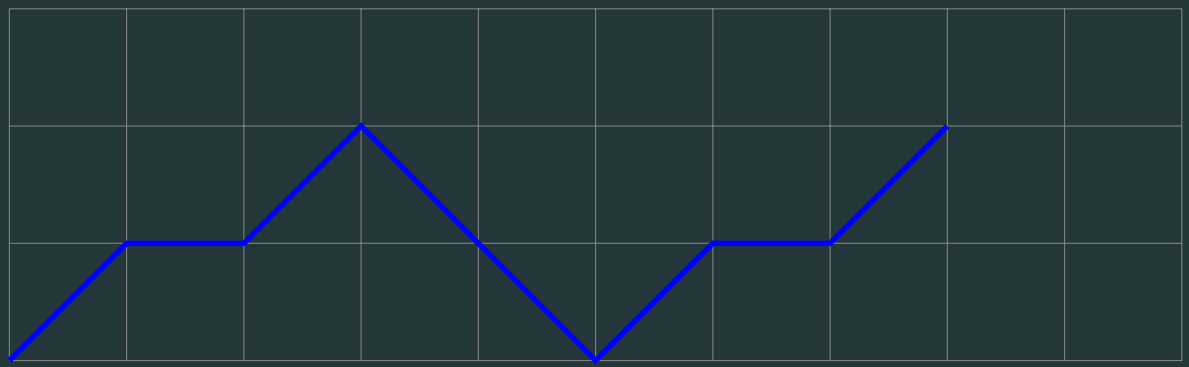
UHUDDUH



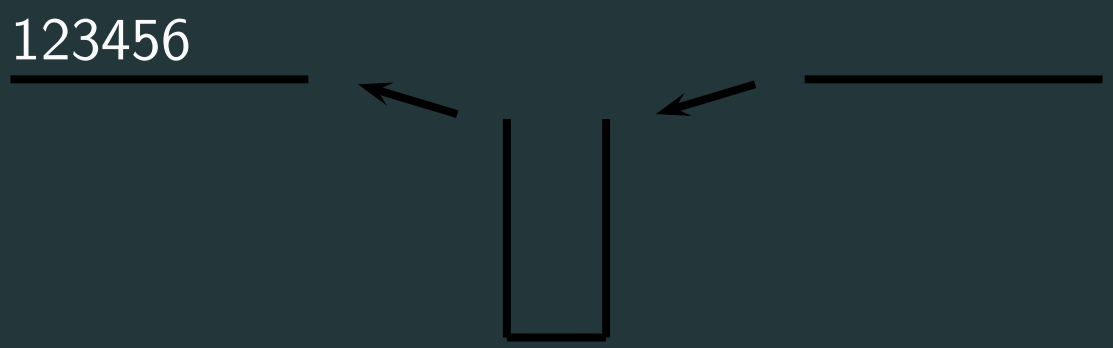
For example, we get the following Motzkin word by sorting 312645:



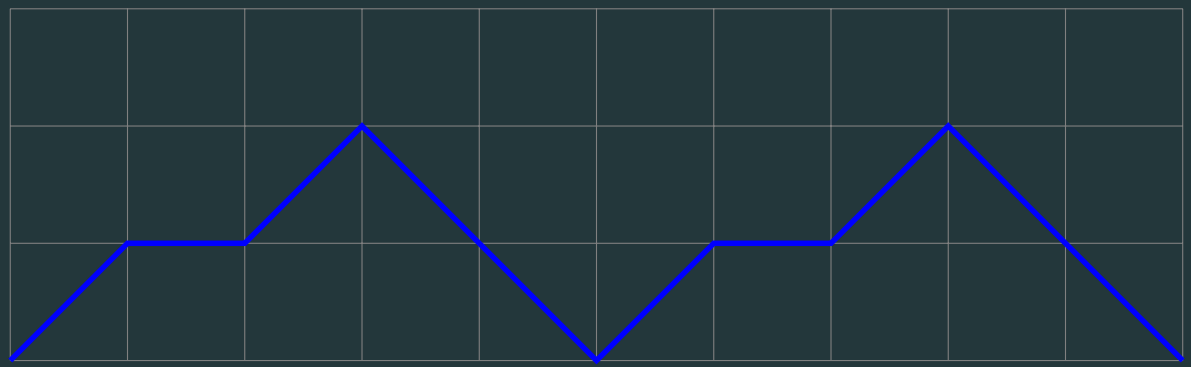
UHUDDUHU



For example, we get the following Motzkin word by sorting 312645:



UHUDDUHUDD



The Motzkin paths that sortable permutations of size n correspond to are those with exactly n total Up and Horizontal steps which:

1. Begin with an up step.
2. End with a down step.
3. Are such that any down step must be followed by another down step until the horizontal axis is reached.
4. Down steps are never adjacent to horizontal steps.

The algorithm also can be used to describe the sortable permutations uniquely as ternary words where

- Push: 0
- Bypass: 1
- Pop, then push: 2

The algorithm also can be used to describe the sortable permutations uniquely as ternary words where

- Push: 0
- Bypass: 1
- Pop, then push: 2

So,

- Sorting words begin with 0

The algorithm also can be used to describe the sortable permutations uniquely as ternary words where

- Push: 0
- Bypass: 1
- Pop, then push: 2

So,

- Sorting words begin with 0
- Sorting words end with 0 or 2

The algorithm also can be used to describe the sortable permutations uniquely as ternary words where

- Push: 0
- Bypass: 1
- Pop, then push: 2

So,

- Sorting words begin with 0
- Sorting words end with 0 or 2
- Sorting words avoid the consecutive word 12

One can describe the allowable words using the regular expression $0(\ 0 \mid 2 \mid 1^{+}\ 0\ )^{*}$.

One can describe the allowable words using the regular expression $0(\ 0 \mid 2 \mid 1^{+}\ 0\ )^{*}$.

Letting M_n be the number of sortable permutations of length n, these words satisfy the recurrence

$$M_0 = 1$$
$$M_1 = 1$$
$$M_n = 2M_{n-1} + \sum_{i=1}^{n-2} M_i \qquad \text{for } n \geq 2$$

which makes M_n match the recurrence for F_{2n-1} with F_{-1} defined as 1.

The pop stack with a bypass map

Define the map on permutations by pop stack with a bypass algorithm as PSB.

Define the map on permutations by pop stack with a bypass algorithm as PSB.

Proposition

Let π, σ be permutations such that $\mathbf{PSB}(\pi) = \sigma$. If m is a left-to-right maximum of π, then m is a left-to-right maximum of σ as well.

Define the map on permutations by pop stack with a bypass algorithm as PSB.

Proposition

Let π, σ be permutations such that $\mathbf{PSB}(\pi) = \sigma$. If m is a left-to-right maximum of π, then m is a left-to-right maximum of σ as well.

Why?

Define the map on permutations by pop stack with a bypass algorithm as PSB.

Proposition

Let π, σ be permutations such that $\mathbf{PSB}(\pi) = \sigma$. If m is a left-to-right maximum of π, then m is a left-to-right maximum of σ as well.

Why? Because PSB does not create new inversions.

Figure 4: Applying PSB to 426513 to get PSB(426513) = 241356.

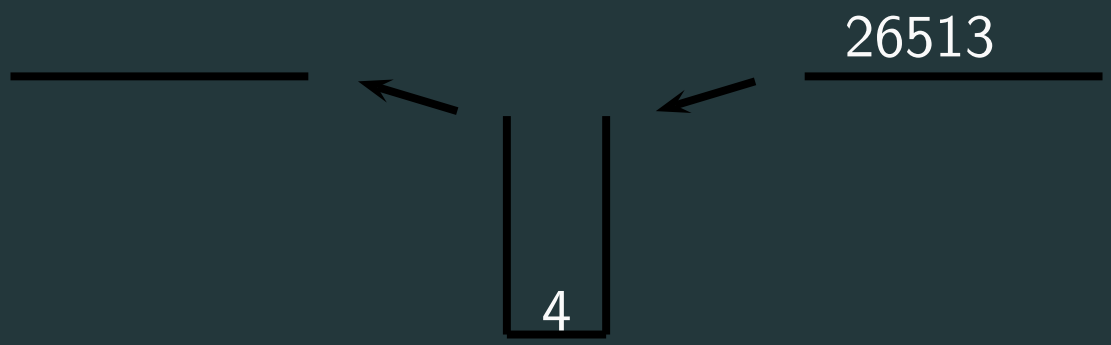


Figure 4: Applying PSB to 426513 to get PSB(426513) = 241356.

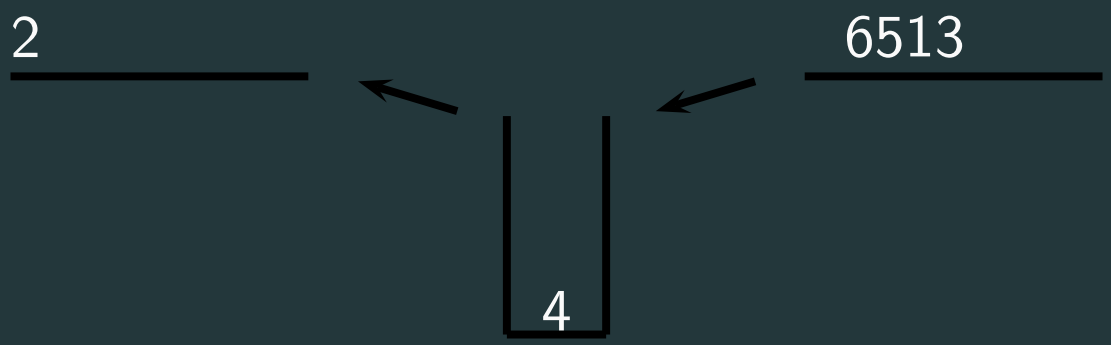


Figure 4: Applying PSB to 426513 to get PSB(426513) = 241356.

Figure 4: Applying PSB to 426513 to get PSB(426513) = 241356.

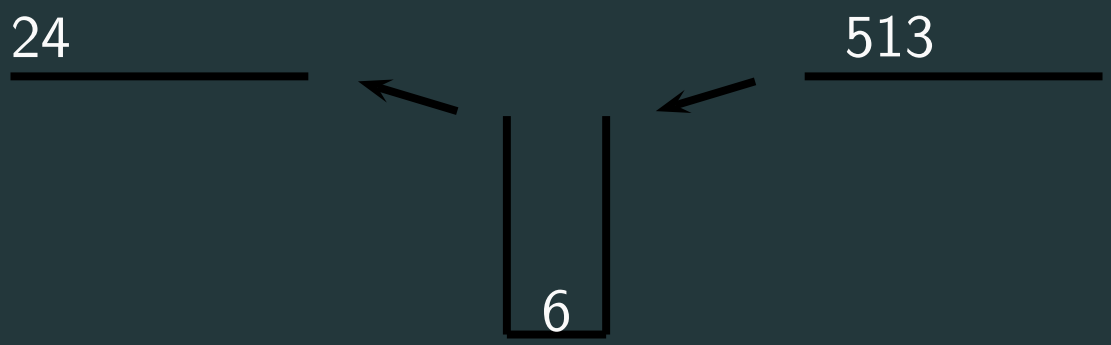


Figure 4: Applying PSB to 426513 to get PSB(426513) = 241356.

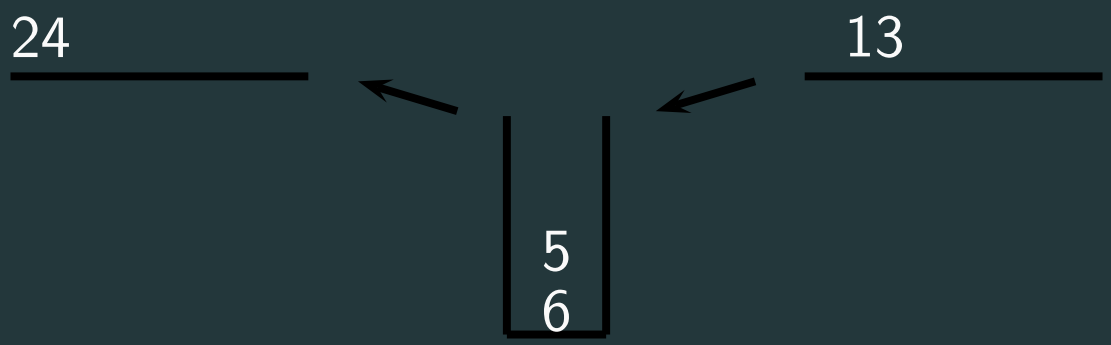


Figure 4: Applying PSB to 426513 to get PSB(426513) = 241356.

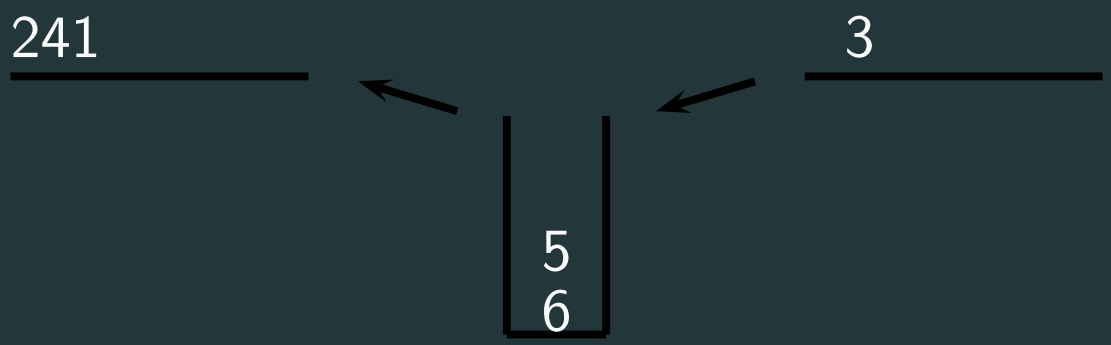


Figure 4: Applying PSB to 426513 to get PSB(426513) = 241356.

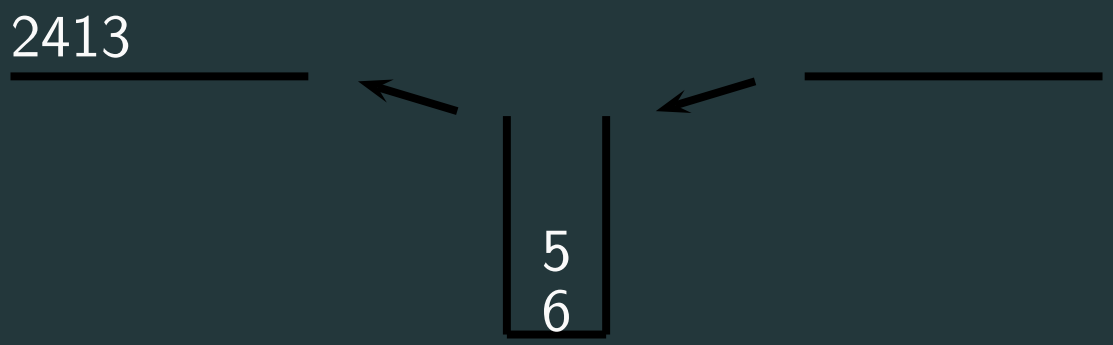


Figure 4: Applying PSB to 426513 to get PSB(426513) = 241356.

Figure 4: Applying PSB to 426513 to get PSB(426513) = 241356.

Let π be a permutation of size n. Notice if $\sigma = \mathrm{PSB}(\pi)$, then $\sigma_n = n$.

Let π be a permutation of size n. Notice if $\sigma = \text{PSB}(\pi)$, then $\sigma_n = n$.

Suppose that $\sigma = \sigma_1\sigma_2 \cdots \sigma_n = \alpha\mu_k$, where μ_k is the maximum suffix of consecutive left-to-right maxima of σ (and α is the remaining prefix). Considering each entry m in μ_k as the start of the suffix and continuing recursively, we can construct all preimages of σ under PSB.

Consider the example where $\mathrm{PSB}(426513) = 241356 = \sigma$.

Consider the example where PSB(426513) $= 241356 = \sigma$.

First, remove the suffix μ_k of left-to-right maxima starting with m.

Consider the example where PSB(426513) $= 241356 = \sigma$.

First, remove the suffix μ_k of left-to-right maxima starting with m. When $\sigma = 241356$, $\mu_2 = 56$. Consider $m = 5$.

Consider the example where PSB(426513) $= 241356 = \sigma$.

First, remove the suffix μ_k of left-to-right maxima starting with m. When $\sigma = 241356$, $\mu_2 = 56$. Consider $m = 5$.

Then reinsert the removed elements back into the remaining permutation in all possible ways, according to the following rules:

- the elements are reinserted in decreasing order;
- the maximum (i.e. n) is inserted to the immediate right of one of the remaining left-to-right maxima of σ or at the beginning of σ;
- the minimum (i.e. m) is inserted to the right of $m - 1$.

Consider the example where PSB(426513) $= 241356 = \sigma$.

First, remove the suffix μ_k of left-to-right maxima starting with m. When $\sigma = 241356$, $\mu_2 = 56$. Consider $m = 5$.

Then reinsert the removed elements back into the remaining permutation in all possible ways, according to the following rules:

- the elements are reinserted in decreasing order;
- the maximum (i.e. n) is inserted to the immediate right of one of the remaining left-to-right maxima of σ or at the beginning of σ;
- the minimum (i.e. m) is inserted to the right of $m-1$.

When $\sigma = 241356$, we insert $n = 6$ immediately after 2 or 4 or at the beginning and insert $m = 5$ after $m-1 = 4$ to get $246135, 246153, 246513, 264135, 264153, 264513, 624135, 624153, 624513$.

Consider the example where $\mathrm{PSB}(426513) = 241356 = \sigma$.

So far we have:

$246135, 246153, 246513, 264135, 264153, 264513, 624135, 624153, 624513$

Consider the example where $\mathrm{PSB}(426513) = 241356 = \sigma$.

So far we have:
$246135, 246153, 246513, 264135, 264153, 264513, 624135, 624153, 624513$

For each of these permutations, consider the prefix of all elements strictly before n and (recursively) continue to compute all possible preimages.

Consider the example where $\mathrm{PSB}(426513) = 241356 = \sigma$.

So far we have:
$246135, 246153, 246513, 264135, 264153, 264513, 624135, 624153, 624513$

For each of these permutations, consider the prefix of all elements strictly before n and (recursively) continue to compute all possible preimages.

We get: $246135, 426135, 246153, 426153, 246513, 426513,$
$264135, 264153, 264513, 624135, 624153, 624513.$

Consider the example where $\mathrm{PSB}(426513) = 241356 = \sigma$.

So far we have:
$246135, 246153, 246513, 264135, 264153, 264513, 624135, 624153, 624513$

For each of these permutations, consider the prefix of all elements strictly before n and (recursively) continue to compute all possible preimages.

We get: $246135, 426135, 246153, 426153, 246513, 426513,$
$264135, 264153, 264513, 624135, 624153, 624513.$

Using the same process with $m = n = 6$, we get the last preimages as $245136, 425136$.

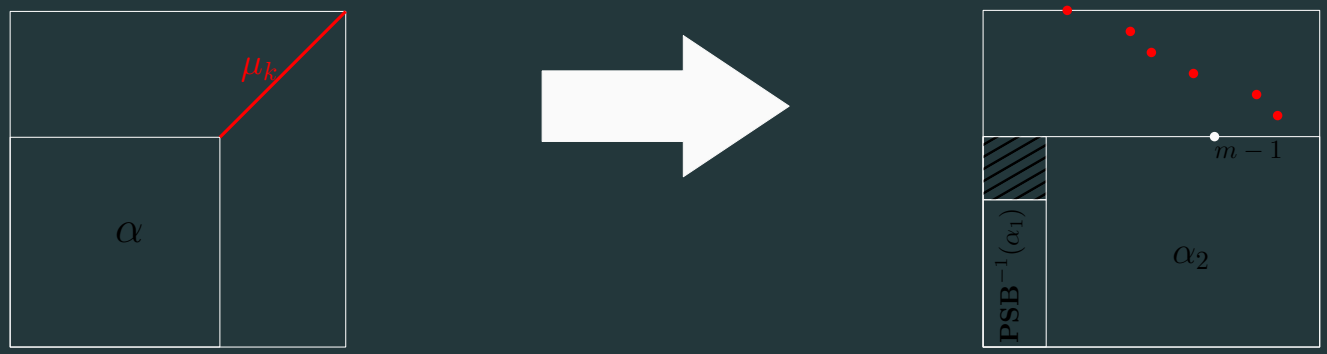


Figure 5: Computing one preimage of $\sigma = \alpha\mu_k$ when m is the first (i.e. smallest) element of μ_k.

Two pop stacks in parallel with a bypass

For this question, we first considered an optimal algorithm for sorting permutations that could be sorted by a machine consisting of two pop stacks in parallel with a bypass.

```
1  S_1, S_2, O := ∅;
2  i := 1;
3  while i ≤ n do
4      if π_i = TOP(S_1) − 1 then
5          PUSH_1;
6      else if π_i = TOP(S_2) − 1 then
7          PUSH_2;
8      else if S_1 = ∅ then
9          PUSH_1;
10     else if S_2 = ∅ then
11         PUSH_2;
12     else if π_i < max(TOP(S_1) − 1, TOP(S_2) − 1) then
13         BYPASS;
14     else
15         if TOP(S_1) < TOP(S_2) then
16             POP_1;
17             if π_i = TOP(S_2) − 1 then
18                 PUSH_2;
19             else
20                 PUSH_1;
21         else
22             POP_2;
23             if π_i = TOP(S_1) − 1 then
24                 PUSH_1;
25             else
26                 PUSH_2;
27 if S_1 = ∅ then
28     POP_2;
29 else if S_2 = ∅ then
30     POP_1;
31 else if TOP(S_1) < TOP(S_2) then
32     POP_1;
33     POP_2;
34 else
35     POP_2;
36     POP_1;
```

Proposition

The class of permutations sortable by two pop stacks in parallel with a bypass is

$$\mathrm{Av}(2341, 25314, 42513, 42531, 45213, 45231, 52314, 642135, 642153).$$

Proposition

The class of permutations sortable by two pop stacks in parallel with a bypass is

$$\mathrm{Av}(2341, 25314, 42513, 42531, 45213, 45231, 52314, 642135, 642153).$$

The inverse of the above class (i.e. the class whose basis is the set of inverses of the above permutations) has a regular insertion encoding (Vatter). Thus it is possible to deduce its rational generating function using the Combinatorial Exploration framework (Bean, Eliasson, Magnusson, Nadeau, Pantone, Ulfarsson) and (Albert, Bean, Claesson, Nadeau, Pantone, and Ulfarsson):

$$\frac{(1-x)(1-2x)(1-4x)}{1-8x+20x^2-18x^3+3x^4}.$$

Proposition
The class of permutations sortable by two pop stacks in parallel with a bypass is

$$\mathrm{Av}(2341, 25314, 42513, 42531, 45213, 45231, 52314, 642135, 642153).$$

The inverse of the above class (i.e. the class whose basis is the set of inverses of the above permutations) has a regular insertion encoding (Vatter). Thus it is possible to deduce its rational generating function using the Combinatorial Exploration framework (Bean, Eliasson, Magnusson, Nadeau, Pantone, Ulfarsson) and (Albert, Bean, Claesson, Nadeau, Pantone, and Ulfarsson):

$$\frac{(1-x)(1-2x)(1-4x)}{1-8x+20x^2-18x^3+3x^4}.$$

The sequence corresponding to the sortable permutations of size n begins $1, 2, 6, 23, 97, 418, 1800, 7717, 32969, 140558, \ldots$

More pop stacks in parallel with a bypass

Following the method of Atkinson and Sack, we are able to prove:

Theorem

There is a finite set of permutations B_k such that a permutation π is sortable by k pop stacks in parallel with a bypass if and only if $\pi \in \mathrm{Av}(B_k)$.

Following the method of Atkinson and Sack, we are able to prove:

Theorem
There is a finite set of permutations B_k such that a permutation π is sortable by k pop stacks in parallel with a bypass if and only if $\pi \in \mathrm{Av}(B_k)$.

Also, using the same insertion encoding method (with one extra slot) for pop stacks without a bypass (S and Vatter), we have:

Theorem
For any positive integer k, the set of permutations sortable by k pop stacks in parallel with a bypass has a rational generating function.

Conjecture

Let a_n be the number of simple permutations of length n which are sortable by a sorting machine consisting of two pop stacks in parallel where entries are allowed to bypass the pop stacks. Then,

$$a_n = \begin{cases} 1 & \text{if } n = 0 \\ 1 & \text{if } n = 1 \\ 2 & \text{if } n = 2 \\ F_{2n-5} - 1 & \text{if } n \geq 3 \text{ is odd} \\ F_{2n-5} & \text{if } n > 3 \text{ is even} \end{cases}$$

where F_n is the n-th Fibonacci number.

References:

Albert, M. H., Bean, C., Claesson, A. Nadeau, E., Pantone, J., and Ulfarsson, H.

Combinatorial Exploration: An algorithmic framework for enumeration.

Memoirs of the American Mathematical Society (to appear).

Atkinson, M. D.

Generalized stack permutations.

Combin. Probab. Comput. 7 (1998), 239–246.

Atkinson, M. D. and Sack, J.-R.

Pop-stacks in parallel.

Inform. Process. Lett. 70 (1999), 63–67.


Avis, D., and Newborn, M.

On pop-stacks in series.

Utilitas Math. 19 (1981), 129–140.


Bean, C., Eliasson, J. S., Magnusson, T. K., Nadeau, E., Pantone, J., and Ulfarsson, H.

Tilings: combinatorial exploration for permutation classes.

at https://github.com/PermutaTriangle/Tilings/.


Knuth, D. E.

The art of computer programming. Volume 1.

Addison-Wesley Publishing Co., Reading, Mass., 1969.

Fundamental Algorithms.

Smith, R. and Vatter, V.

The enumeration of permutations sortable by pop stacks in parallel.

Inform. Process. Lett. 109 (2009), 626–629.

Vatter, V.

Finding regular insertion encodings for permutation classes.

J. Symbolic Comput. 47 (2012), 259–265.

Thank you!