1-5 Data Structures

魏恒峰

hfwei@nju.edu.cn

2019 年 11 月 21 日

Pseudocode

Pseudocode

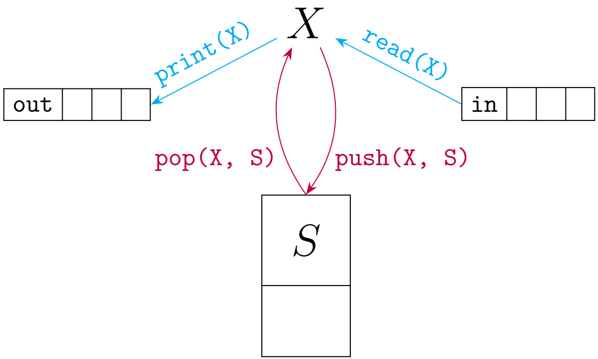

Make everything as
SIMPLE
as possible,
but not simpler.
- ALBERT EINSTEIN (1879-1955)
QUOTESEVERLASTING.COM

Pseudocode

"Executable" at an abstract level.

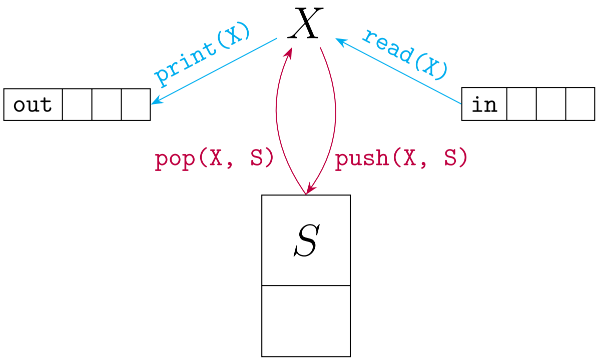
Stackable Permutations

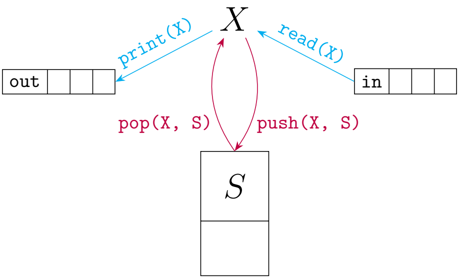
Definition (Stackable Permutations)



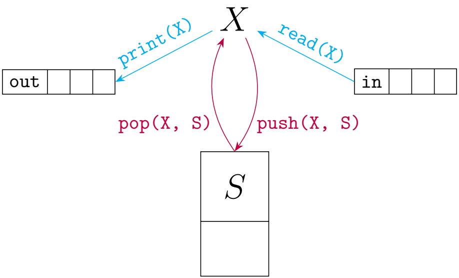
Definition (Stackable Permutations)

$$\texttt{out} = (a_1, \cdots, a_n) \xleftarrow[X = \perp]{S = \emptyset} \texttt{in} = (1, \cdots, n)$$





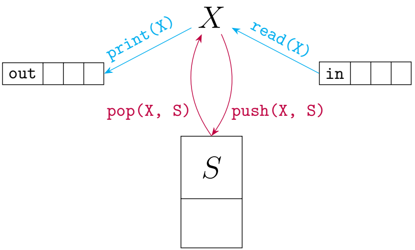
We can assume that X is always blank.

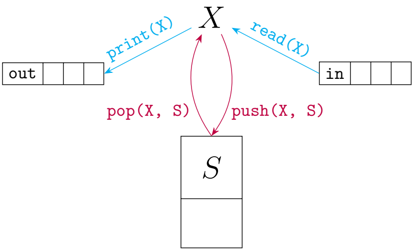


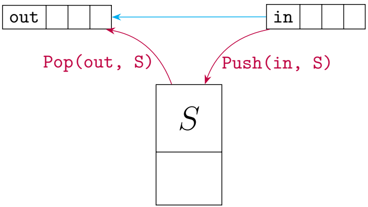
We can assume that X is always blank.

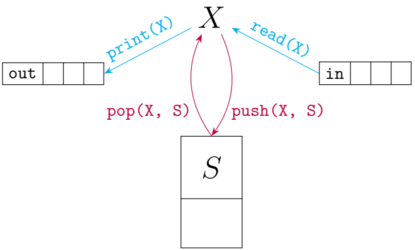
`read` + `push`   `read` + `print`

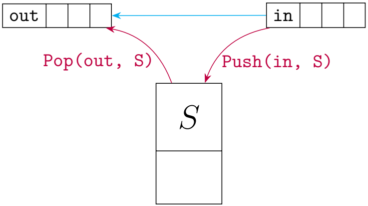
`pop` + `print`   `pop` + `push`

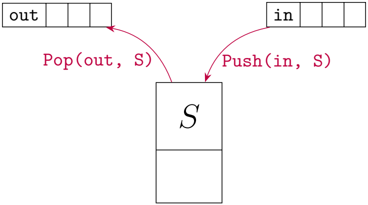

X
print(X)
read(X)
out
in
pop(X, S)
push(X, S)
S


X
print(X)
read(X)
out
in
pop(X, S)
push(X, S)
S


out
in
Pop(out, S)
Push(in, S)
S

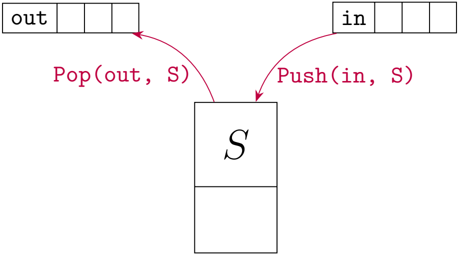

X
print(X)
read(X)
out
in
pop(X, S)
push(X, S)
S


out
in
Pop(out, S)
Push(in, S)
S


out
in
Pop(out, S)
Push(in, S)
S

Definition (Stackable Permutations)

$$\texttt{out} = (a_1, \cdots, a_n) \xleftarrow{S = \emptyset} \texttt{in} = (1, \cdots, n)$$



DH 2.12: Stackable Permutations

(a) Show that the following permutations *are* stackable:

  (i) $(3, 2, 1)$

  (ii) $(3, 4, 2, 1)$

  (iii) $(3, 5, 7, 6, 8, 4, 9, 2, 10, 1)$

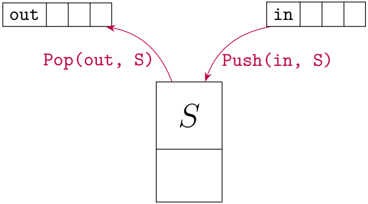
DH 2.12: Stackable Permutations

(a) Show that the following permutations *are* stackable:

(i) $(3, 2, 1)$

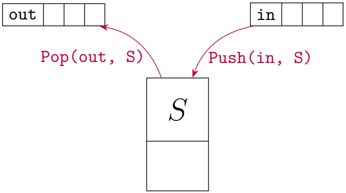
(ii) $(3, 4, 2, 1)$

(iii) $(3, 5, 7, 6, 8, 4, 9, 2, 10, 1)$



DH 2.13: Stackable Permutations Checking Algorithm

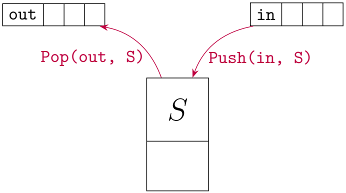
To check whether a given permutation can be obtained by a stack.




```
procedure STACKABLE(out)
    for all a_j ∈ out do
        while top(S) ≠ a_j do
            Push(in, S)
        Pop(out, S)
```

DH 2.13: Stackable Permutations Checking Algorithm

To check whether a given permutation can be obtained by a stack.

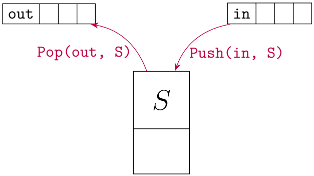


procedure STACKABLE(out)
  for all $a_j \in out$ **do**
    while top(S) $\neq a_j$ **do**
      Push(in, S)
    Pop(out, S)

Q : What is wrong with STACKABLE*?*

DH 2.13: Stackable Permutations Checking Algorithm

To check whether a given permutation can be obtained by a stack.



procedure STACKABLE(out)
  for all $a_j \in out$ **do**
    while $\text{top}(S) \neq a_j \land in \neq \emptyset$ **do**
      $\text{Push}(in, S)$
    if $\text{top}(S) = a_j$ **then**
      $\text{Pop}(out, S)$
    else ▷ $\text{top}(S) \neq a_j \land in = \emptyset$
      return F
  return T

DH 2.12: Stackable Permutations

(b) **Prove** that the following permutations are *not* stackable:

(i) $(3, 1, 2)$

(ii) $(4, 5, 3, 7, 2, 1, 6)$

DH 2.12: Stackable Permutations

(b) **Prove** that the following permutations are *not* stackable:

(i) $(3, 1, 2)$

(ii) $(4, 5, 3, 7, 2, 1, 6)$

$$(3, 1, 2)$$

$$(4, 5, 3, 7, 2, 1, 6)$$

DH 2.12: Stackable Permutations

(b) **Prove** that the following permutations are *not* stackable:

(i) $(3, 1, 2)$

(ii) $(4, 5, 3, 7, 2, 1, 6)$

$$(3, 1, 2)$$

$$(4, 5, 3, 7, 2, 1, 6)$$

$$\texttt{out} = \cdots a_i \cdots a_j \cdots a_k : i < j < k \wedge a_j < a_k < a_i$$

DH 2.12: Stackable Permutations

(b) **Prove** that the following permutations are *not* stackable:

(i) $(3, 1, 2)$

(ii) $(4, 5, 3, 7, 2, 1, 6)$

$$(3, 1, 2)$$

$$(4, 5, 3, 7, 2, 1, 6)$$

$$\texttt{out} = \cdots a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i$$

312-Pattern

Theorem (Stackable Permutations)

A permutation $(a_1, \cdots, a_n)$ *is stackable* $\iff$ *it is not the case that*

$$312\text{-}Pattern: \boxed{out = \cdots a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i}$$

Theorem (Stackable Permutations)

A permutation $(a_1, \cdots, a_n)$ *is stackable* $\iff$ *it is not the case that*

$$312\text{-}Pattern: \boxed{out = \cdots a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i}$$

Proof.

stackable $\Longrightarrow \nexists$ 312-Pattern

$\nexists$ 312-Pattern $\Longrightarrow$ stackable

□

Theorem (Stackable Permutations)

A permutation $(a_1, \cdots, a_n)$ is stackable $\iff$ it is not the case that

$$312\text{-}Pattern : \boxed{out = \cdots a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i}$$

Proof.

stackable $\implies \nexists$ 312-Pattern

$\nexists$ 312-Pattern $\implies$ stackable

312-Pattern $\implies$ non-stackable

□

Theorem (Stackable Permutations)

A permutation $(a_1, \cdots, a_n)$ *is stackable* $\iff$ *it is not the case that*

$$312\text{-}Pattern : \boxed{out = \cdots a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i}$$

Proof.

stackable $\implies \nexists$ 312-Pattern

$\nexists$ 312-Pattern $\implies$ stackable

312-Pattern $\implies$ non-stackable

$\exists$ algorithm

□

Theorem (Stackable Permutations)

A permutation $(a_1, \cdots, a_n)$ *is stackable* $\iff$ *it is not the case that*

$$312\text{-}Pattern: \boxed{out = \cdots a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i}$$

312-Pattern $\Longrightarrow$ non-stackable.

Theorem (Stackable Permutations)

A permutation $(a_1, \cdots, a_n)$ *is stackable* $\iff$ *it is not the case that*

$$312\text{-}Pattern : \boxed{out = \cdots a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i}$$

312-Pattern $\implies$ non-stackable.

$i < j \land a_j < a_i$: Push_j Push_i Pop_i Pop_j

$j < k \land a_j < a_k$: Push_j Pop_j Push_k Pop_k

$i < k \land a_k < a_i$: Push_k Push_i Pop_i Pop_k

□

Theorem (Stackable Permutations)

A permutation $(a_1, \cdots, a_n)$ *is stackable* $\iff$ *it is not the case that*

$$312\text{-}Pattern : \boxed{out = \cdots a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i}$$

$\nexists$ 312-Pattern $\Longrightarrow$ Obtainable by STACKABLE.

```
procedure STACKABLE(out)
    for all a_j ∈ out do
        while top(S) ≠ a_j ∧ in ≠ ∅ do
            Push(in, S)

        if top(S) = a_j then
            Pop(out, S)
        else        ▷ top(S) ≠ a_j ∧ in = ∅
            return F

    return T
```

Theorem (Stackable Permutations)

A permutation $(a_1, \cdots, a_n)$ is stackable $\iff$ it is not the case that

$$312\text{-}Pattern : \boxed{out = \cdots a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i}$$

$\nexists$ 312-Pattern $\implies$ Obtainable by STACKABLE.

STACKABLE fails $\implies$ $\exists$ 312-Pattern.

```
procedure STACKABLE(out)
    for all a_j ∈ out do
        while top(S) ≠ a_j ∧ in ≠ ∅ do
            Push(in, S)

        if top(S) = a_j then
            Pop(out, S)
        else        ▷ top(S) ≠ a_j ∧ in = ∅
            return F
    return T
```

Theorem (Stackable Permutations)

A permutation $(a_1, \cdots, a_n)$ *is stackable* $\iff$ *it is not the case that*

$$312\text{-}Pattern: \boxed{out = \cdots a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i}$$

$\nexists$ 312-Pattern $\implies$ Obtainable by STACKABLE.

STACKABLE fails $\implies$ $\exists$ 312-Pattern.

```
procedure STACKABLE(out)
    for all a_j ∈ out do
        while top(S) ≠ a_j ∧ in ≠ ∅ do
            Push(in, S)

        if top(S) = a_j then
            Pop(out, S)
        else        ▷ top(S) ≠ a_j ∧ in = ∅
            return F

    return T
```

$a_j \neq \mathsf{top}(S) \land in = \emptyset$

Theorem (Stackable Permutations)

A permutation $(a_1, \cdots, a_n)$ is stackable $\iff$ it is not the case that

$$312\text{-}Pattern: \boxed{out = \cdots a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i}$$

$\nexists$ 312-Pattern $\implies$ Obtainable by STACKABLE.

STACKABLE fails $\implies$ $\exists$ 312-Pattern.

```
procedure STACKABLE(out)
    for all a_j ∈ out do
        while top(S) ≠ a_j ∧ in ≠ ∅ do
            Push(in, S)

        if top(S) = a_j then
            Pop(out, S)
        else        ▷ top(S) ≠ a_j ∧ in = ∅
            return F
    return T
```

$a_j \neq \mathsf{top}(S) \land in = \emptyset$

a_j is covered by some a_k in S

Theorem (Stackable Permutations)

A permutation $(a_1, \cdots, a_n)$ *is stackable* $\iff$ *it is not the case that*

$$312\text{-}Pattern : \boxed{out = \cdots a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i}$$

$\nexists$ 312-Pattern $\Longrightarrow$ Obtainable by STACKABLE.

STACKABLE fails $\Longrightarrow$ $\exists$ 312-Pattern.

```
procedure STACKABLE(out)
    for all a_j ∈ out do
        while top(S) ≠ a_j ∧ in ≠ ∅ do
            Push(in, S)

        if top(S) = a_j then
            Pop(out, S)
        else                ▷ top(S) ≠ a_j ∧ in = ∅
            return F
    return T
```

$a_j \neq \text{top}(S) \land in = \emptyset$

a_j is covered by some a_k in S

$\exists k : j < k \land a_j < a_k$

Theorem (Stackable Permutations)

A permutation $(a_1, \cdots, a_n)$ *is stackable* $\iff$ *it is not the case that*

$$312\text{-}Pattern: \boxed{out = \cdots a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i}$$

$\nexists$ 312-Pattern $\implies$ Obtainable by STACKABLE.

STACKABLE fails $\implies$ $\exists$ 312-Pattern.

procedure STACKABLE(out)
  for all $a_j \in out$ **do**
    while $\text{top}(S) \neq a_j \land in \neq \emptyset$ **do**
      $\text{Push}(in, S)$
    if $\text{top}(S) = a_j$ **then**
      $\text{Pop}(out, S)$
    else ▷ $\text{top}(S) \neq a_j \land in = \emptyset$
      return F
  return T

$a_j \neq \text{top}(S) \land in = \emptyset$

a_j is covered by some a_k in S

$\exists k : j < k \land a_j < a_k$

Why is a_k in S?

Theorem (Stackable Permutations)

A permutation $(a_1, \cdots, a_n)$ *is stackable* $\iff$ *it is not the case that*

$$312\text{-}Pattern : \boxed{out = \cdots a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i}$$

$\nexists$ 312-Pattern $\implies$ Obtainable by STACKABLE.

STACKABLE fails $\implies$ $\exists$ 312-Pattern.

```
procedure STACKABLE(out)
    for all a_j ∈ out do
        while top(S) ≠ a_j ∧ in ≠ ∅ do
            Push(in, S)

        if top(S) = a_j then
            Pop(out, S)
        else        ▷ top(S) ≠ a_j ∧ in = ∅
            return F

    return T
```

$a_j \neq \mathsf{top}(S) \land in = \emptyset$

a_j is covered by some a_k in S

$\exists k : j < k \land a_j < a_k$

Why is a_k in S?

$\exists i : i < j \land a_k < a_i$

DH 2.12: Stackable Permutations

(c) How many permutations of A_4 *cannot* be obtained by a stack?

$$(1,4,2,3),(2,4,1,3),(3,1,2,4),(3,1,4,2),(3,4,1,2)$$
$$(4,1,2,3),(4,1,3,2),(4,2,1,3),(4,2,3,1),(4,3,1,2)$$

DH 2.12: Stackable Permutations

(c) How many permutations of A_4 *cannot* be obtained by a stack?

$$(1, 4, 2, 3), (2, 4, 1, 3), (3, 1, 2, 4), (3, 1, 4, 2), (3, 4, 1, 2)$$
$$(4, 1, 2, 3), (4, 1, 3, 2), (4, 2, 1, 3), (4, 2, 3, 1), (4, 3, 1, 2)$$

DH 2.12: Stackable Permutations

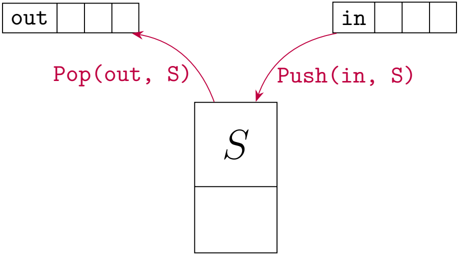
(c) How many permutations of A_4 *cannot* be obtained by a stack?

$$(1, 4, 2, 3), (2, 4, 1, 3), (3, 1, 2, 4), (3, 1, 4, 2), (3, 4, 1, 2)$$
$$(4, 1, 2, 3), (4, 1, 3, 2), (4, 2, 1, 3), (4, 2, 3, 1), (4, 3, 1, 2)$$

Q : What about A_n?

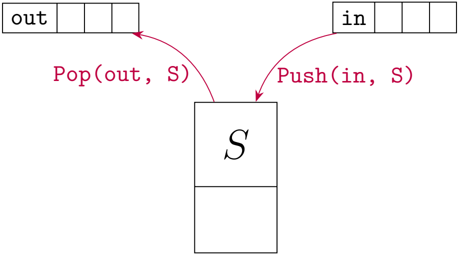
DH 2.12: Stackable Permutations

How many permutations of $\{1 \cdots n\}$ are stackable?



DH 2.12: Stackable Permutations

How many permutations of $\{1 \cdots n\}$ are stackable?



Q : How many *admissible* operation sequences of "Push" and "Pop"?

Definition (Admissible Operation Sequences)

An operation sequence of "`Push`" and "`Pop`" is *admissible* if and only if

Definition (Admissible Operation Sequences)

An operation sequence of "`Push`" and "`Pop`" is *admissible* if and only if

(i) # of "`Push`" $= n$ # of "`Pop`" $= n$

Definition (Admissible Operation Sequences)

An operation sequence of "`Push`" and "`Pop`" is *admissible* if and only if

(i) # of "`Push`" $= n$ # of "`Pop`" $= n$

(ii) $\forall$ prefix : (# of "`Pop`") $\leq$ (# of "`Push`")

Definition (Admissible Operation Sequences)

An operation sequence of "`Push`" and "`Pop`" is *admissible* if and only if

(i) # of "`Push`" $= n$ # of "`Pop`" $= n$

(ii) $\forall$ prefix : (# of "`Pop`") $\leq$ (# of "`Push`")

# of admissible operation sequences = # of stackable perms

Definition (Admissible Operation Sequences)

An operation sequence of "Push" and "Pop" is *admissible* if and only if

(i) # of "Push" $= n$ # of "Pop" $= n$

(ii) $\forall$ prefix : (# of "Pop") $\le$ (# of "Push")

# of admissible operation sequences = # of stackable perms

$$\{\text{admissible operation sequences}\} \xrightarrow{\exists f:1-1} \{\text{stackable perms}\}$$

Definition (Admissible Operation Sequences)

An operation sequence of "Push" and "Pop" is *admissible* if and only if

(i) # of "Push" $= n$ # of "Pop" $= n$

(ii) $\forall$ prefix : (# of "Pop") $\leq$ (# of "Push")

# of admissible operation sequences = # of stackable perms

$$\{\text{admissible operation sequences}\} \xrightarrow{\exists f:1-1} \{\text{stackable perms}\}$$

$$f(s) \triangleq \textit{Execute} \text{ this admissible operation sequence } s$$

Definition (Admissible Operation Sequences)

An operation sequence of "`Push`" and "`Pop`" is *admissible* if and only if

(i) # of "`Push`" $= n$ # of "`Pop`" $= n$

(ii) $\forall$ prefix : (# of "`Pop`") $\leq$ (# of "`Push`")

# of admissible operation sequences = # of stackable perms

$$\{\text{admissible operation sequences}\} \xrightarrow{\exists f:1-1} \{\text{stackable perms}\}$$

$$f(s) \triangleq \textit{Execute} \text{ this admissible operation sequence } s$$

Why is f bijective (1-1)?

Theorem

The number of admissible operation sequences of "Push" and "Pop" is $\binom{2n}{n} - \binom{2n}{n-1}$.

Theorem

The number of admissible operation sequences of "Push" and "Pop" is $\binom{2n}{n} - \binom{2n}{n-1}$.

Proof: The Reflection Method.

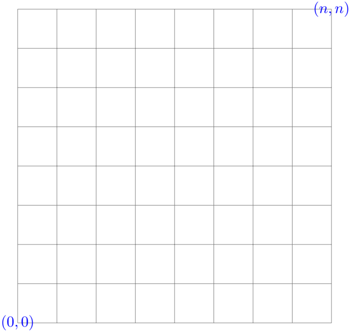
$$\texttt{Push} : \rightarrow \qquad \texttt{Pop} : \uparrow$$

Theorem

The number of admissible operation sequences of “Push” and “Pop” is $\binom{2n}{n} - \binom{2n}{n-1}$.

Proof: The Reflection Method.

$$\texttt{Push}: \rightarrow \qquad \texttt{Pop}: \uparrow$$



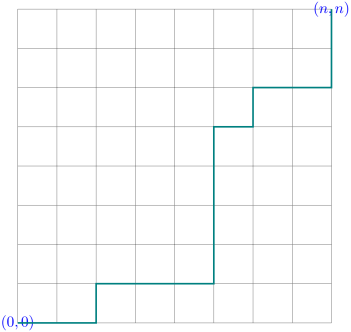
□

Theorem

The number of admissible operation sequences of "Push" and "Pop" is $\binom{2n}{n} - \binom{2n}{n-1}$.

Proof: The Reflection Method.

$$\texttt{Push} : \rightarrow \qquad \texttt{Pop} : \uparrow$$



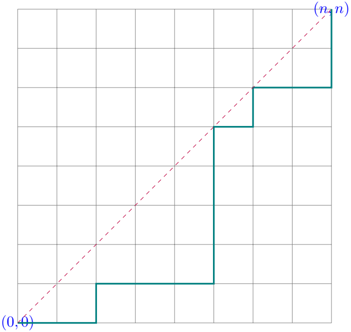
□

Theorem

The number of admissible operation sequences of "Push" and "Pop" is $\binom{2n}{n} - \binom{2n}{n-1}$.

Proof: The Reflection Method.

Push : $\rightarrow$ Pop : $\uparrow$

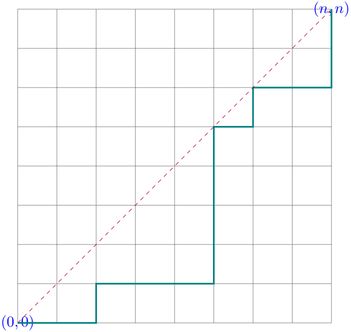


Theorem

The number of admissible operation sequences of "Push" and "Pop" is $\binom{2n}{n} - \binom{2n}{n-1}$.

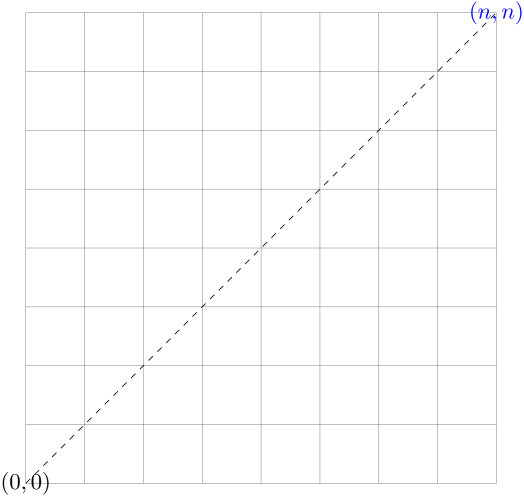
Proof: The Reflection Method.

Push : $\rightarrow$ Pop : $\uparrow$

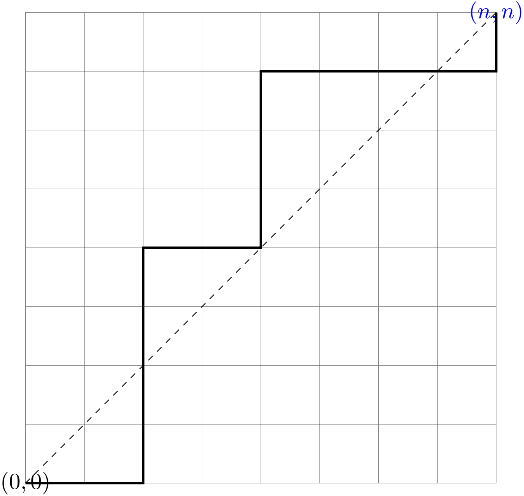


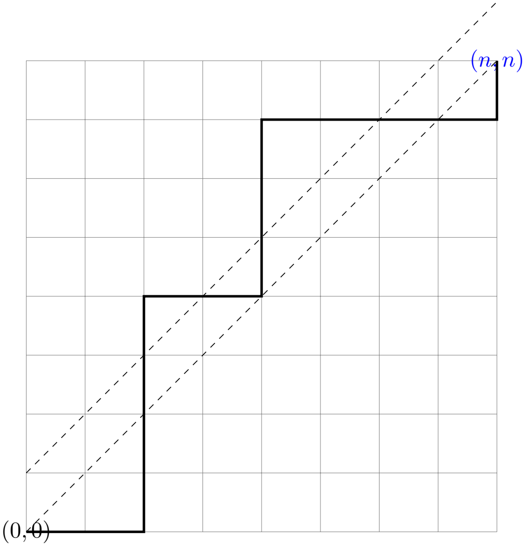
$$\underbrace{\binom{2n}{n}}_{\text{all}} - \underbrace{\binom{2n}{n-1}}_{\text{inadmissible}}$$

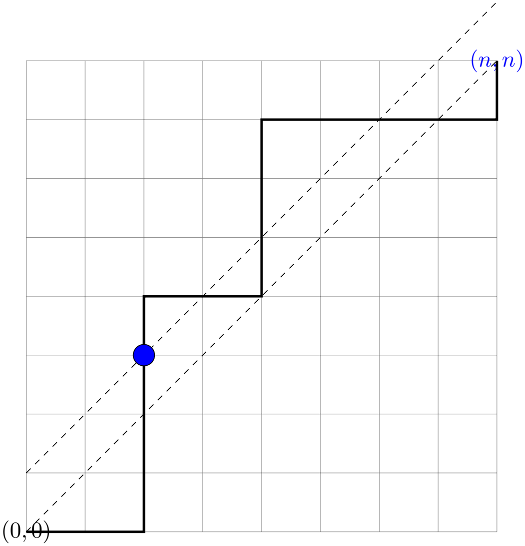
□



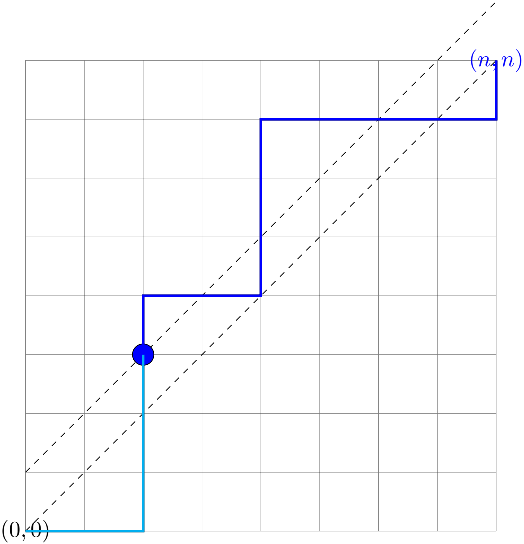
(n, n)
(0, 0)

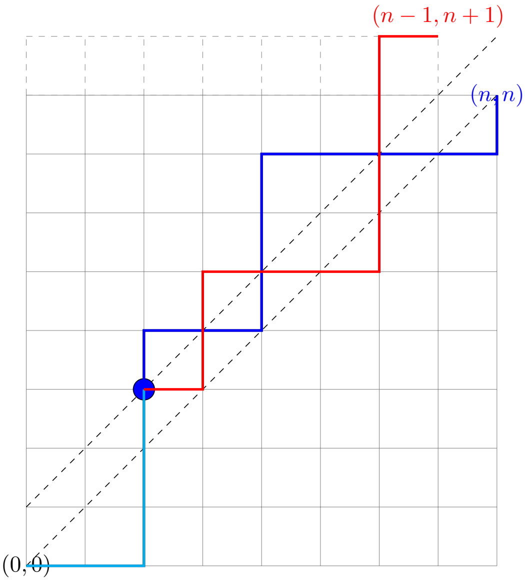

(n, n)
(0, 0)


(n, n)
(0, 0)



(n,n)
$(0,0)$


(n, n)
(0, 0)


$(n-1, n+1)$
(n, n)
$(0, 0)$

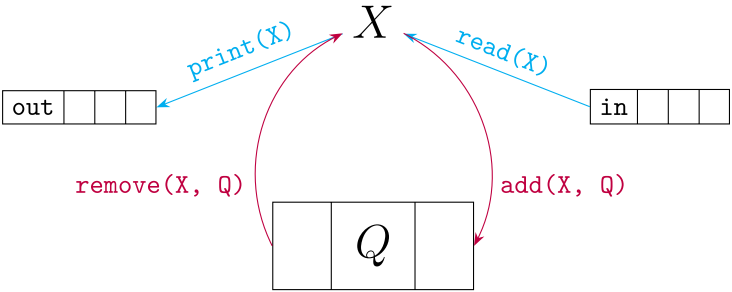
$$\binom{2n}{n} - \binom{2n}{n-1}$$

Catalan Number

$$(3,2,1) : ((())) \qquad (1,2,3) : ()()()$$

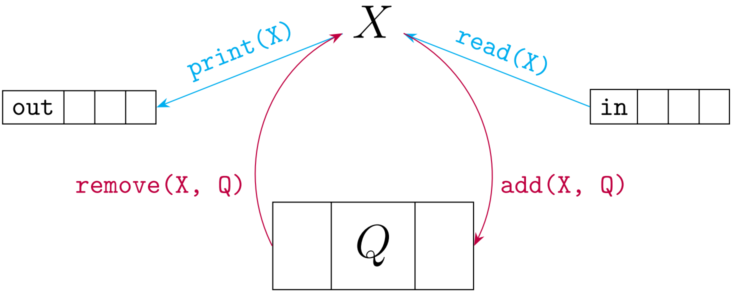
Queueable Permutations

DH 2.14: Queueable Permutations



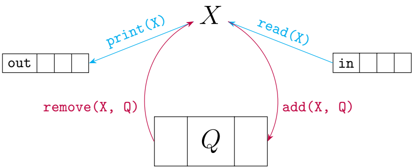
DH 2.14: Queueable Permutations

$$\texttt{out} = (a_1, \cdots, a_n) \xleftarrow[X = \perp]{Q = \emptyset} \texttt{in} = (1, \cdots, n)$$



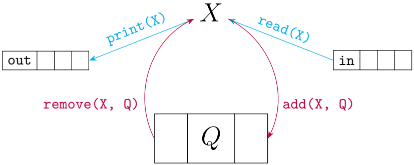
DH 2.14: Queueable Permutations

(b) Prove that every permutation are queueable.



DH 2.14: Queueable Permutations

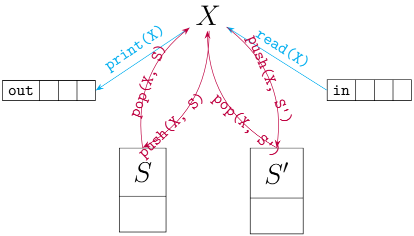
(b) Prove that every permutation are queueable.




```
procedure QUEUEABLE(out)
    for all a ∈ in do
        read(X)
        add(X, Q)
    for all a ∈ out do
        while Head(Q) ≠ a do
            remove(X, Q)
            add(X, Q)
        remove(X, Q)
        print(X)
```

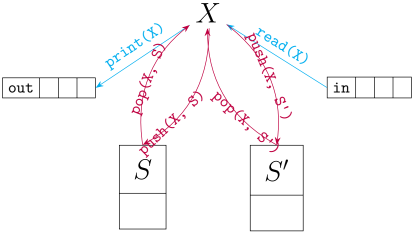
DH 2.14: Queueable Permutations

(c) Prove that every permutation can be obtained by two stacks.

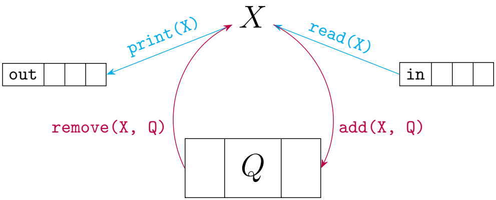


DH 2.14: Queueable Permutations

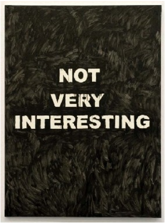
(c) Prove that every permutation can be obtained by two stacks.

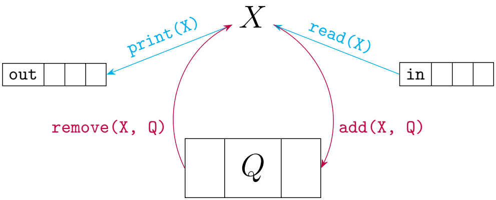



```
procedure DoubleStackable(out)
    for all a ∈ in do
        read(X)
        push(X, S)
    for all a ∈ out do
        while top(S) ≠ a do
            pop(X, S)
            push(X, S')
        pop(X, S)
        print(X)
        while S' ≠ ∅ do
            pop(X, S')
            push(X, S)
```

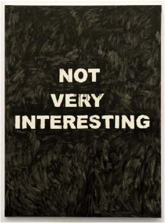


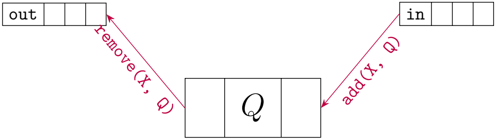
All are queueable.



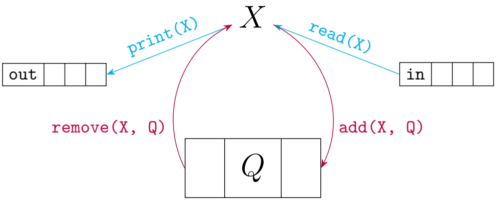


All are queueable.

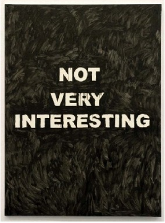


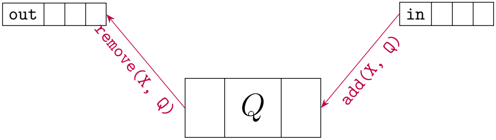


Only one is queueable.

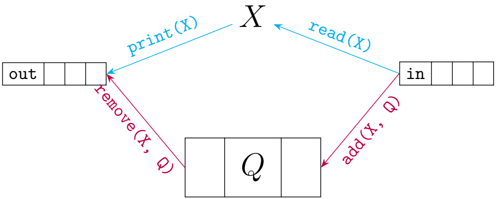


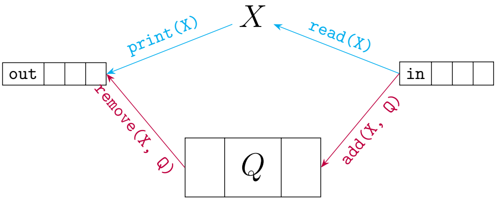
All are queueable.

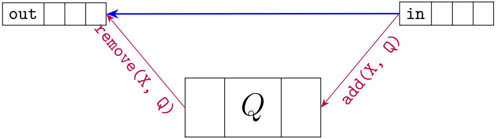


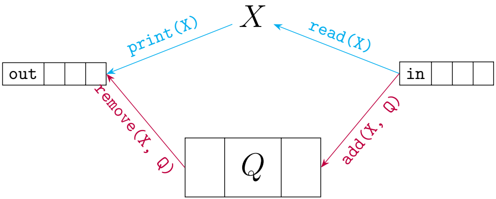


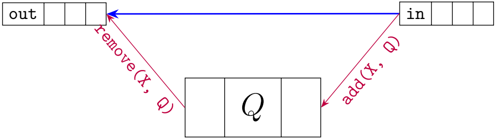
Only one is queueable.


X
print(X)
read(X)
out
in
remove(X, Q)
add(X, Q)
Q

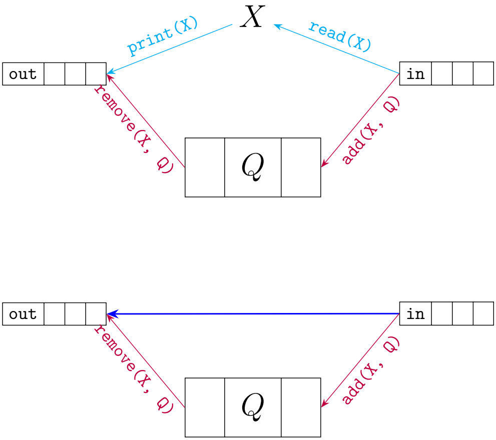

X
print(X)
read(X)
out
in
remove(X, Q)
add(X, Q)
Q


out
in
remove(X, Q)
add(X, Q)
Q





3 2 1 *is not queueable*



3 2 1 *is not queueable*

Theorem (Queueable Permutations)

A permutation $(a_1, \cdots, a_n)$ *is* *queueable* $\iff$ *it is not the case that*

$$321\text{-}Pattern: \boxed{out = \cdots a_i \cdots a_j \cdots a_k : i < j < k \land a_i > a_j > a_k}$$

Theorem (Queueable Permutations)

A permutation $(a_1, \cdots, a_n)$ is queueable $\iff$ it is not the case that

$$321\text{-}Pattern: \boxed{out = \cdots a_i \cdots a_j \cdots a_k : i < j < k \land a_i > a_j > a_k}$$

Proof.

□

Theorem (# of Queueable Permutations)

The number of queueable permutations of $[1 \cdots n]$ *is* $\binom{2n}{n} - \binom{2n}{n-1}$.

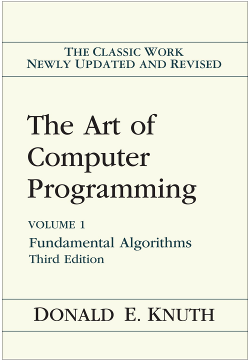
Theorem (# of Queueable Permutations)

The number of queueable permutations of $[1 \cdots n]$ *is* $\binom{2n}{n} - \binom{2n}{n-1}$.

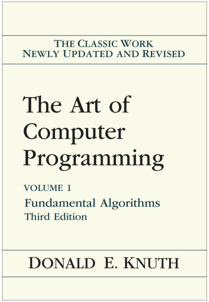
Proof.

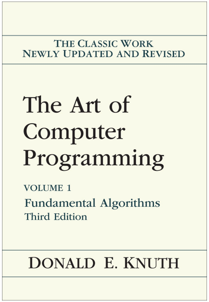
□

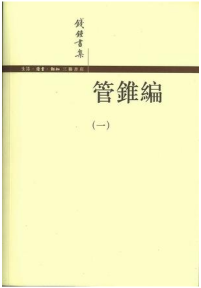
For more about "Stackable/Queueable Permutations"
(Section 2.2.1)



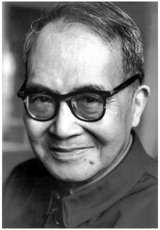
Chapter 2 — Information Structures 232


THE CLASSIC WORK
NEWLY UPDATED AND REVISED
The Art of Computer Programming
VOLUME 1
Fundamental Algorithms
Third Edition
DONALD E. KNUTH


THE CLASSIC WORK
NEWLY UPDATED AND REVISED
The Art of Computer Programming
VOLUME 1
Fundamental Algorithms
Third Edition
DONALD E. KNUTH


錢鍾書集
管錐編
(一)


Don Knuth




Thank You!