1-5 Data Structures

魏恒峰

hfwei@nju.edu.cn

2019年11月21日



Pseudocode

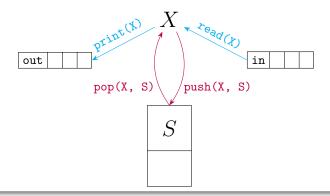


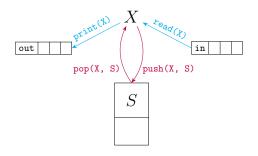
"Executable" at an abstract level.

Stackable Permutations

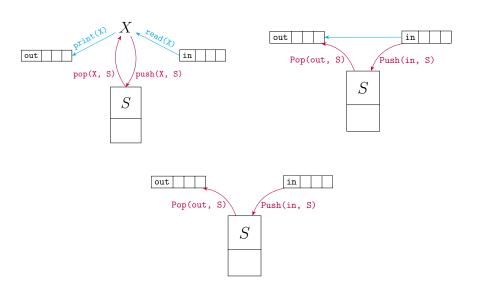
Definition (Stackable Permutations)

$$\boxed{\mathtt{out} = (a_1, \cdots, a_n) \overset{S = \emptyset}{\longleftarrow} \mathtt{in} = (1, \cdots, n)}$$

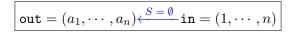


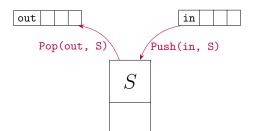


We can assume that X is always blank.



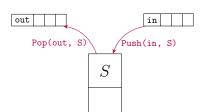
Definition (Stackable Permutations)





DH 2.12: Stackable Permutations

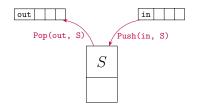
- (a) Show that the following permutations *are* stackable:
 - (i) (3,2,1)
 - (ii) (3,4,2,1)
 - (iii) (3, 5, 7, 6, 8, 4, 9, 2, 10, 1)





DH 2.13: Stackable Permutations Checking Algorithm

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



```
1: procedure STACKABLE(out)
```

2: **for all** $a_i \in out \mathbf{do}$

3: while $top(S) \neq a_j do$

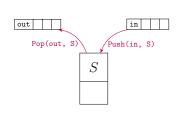
4: Push(in, S)

5: $\mathsf{Pop}(out, S)$

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.



```
1: procedure STACKABLE(out)
       for all a_i \in out do
            while top(S) \neq a_i \land in \neq \emptyset do
3:
                Push(in, S)
4:
            if top(S) = a_i then
5:
                Pop(out, S)
6:
7:
            else \triangleright \mathsf{top}(S) \neq a_i \land in = \emptyset
                return F
8:
       return T
9:
```

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)

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

312-Pattern

Theorem (Stackable Permutations)

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

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

Proof.

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

$$312$$
-Pattern \Longrightarrow non-stackable

Theorem (Stackable Permutations)

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

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

312-Pattern \Longrightarrow non-stackable.

$$i < j \land a_j < a_i$$
: 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, \dots, a_n) is stackable \iff it is not the case that

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

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

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

```
1: procedure STACKABLE(out)
2: for all a_j \in out do
3: while top(S) \neq a_j \land in \neq \emptyset do
4: Push(in, S)
5: if top(S) = a_j then
6: Pop(out, S)
7: else \triangleright top(S) \neq a_j \land in = \emptyset
```

return F

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

 a_j is covered by some a_k in k

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

Why is a_k in S?

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

 ${f return}\ T$

Hengfeng Wei (hfwei@nju.edu.cn)

8:

DH 2.12: Stackable Permutations

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

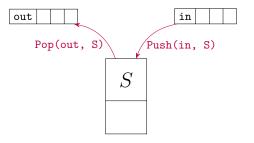
$$(1, 4, 2, 3), (2, 4, 1, 3), (3, 1, 2, 4), (3, 1, 4, 2), (3, 4, 1, 2)$$

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

 $Q: What about A_n?$

DH 2.12: Stackable Permutations

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



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

- (i) # of "Push" = n # of "Pop" = n
- (ii) \forall prefix: (# of "Pop") \leq (# of "Push")

of admissible operation sequences = # of stackable perms

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

 $f(s) \triangleq Execute$ this admissible operation sequence s

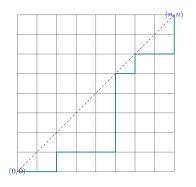
Why is f bijective (1-1)?

Theorem

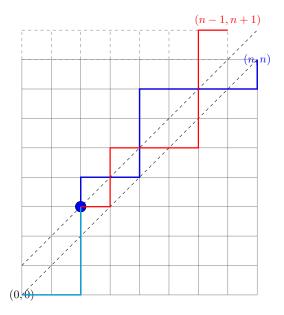
The number of admissible operation sequences of "Push" and "Pop" is $\binom{2n}{n} - \binom{2n}{n-1}$.

Proof: The Reflection Method.

$$\mathtt{Push}: \to \qquad \mathtt{Pop}: \uparrow$$



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



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

Catalan Number

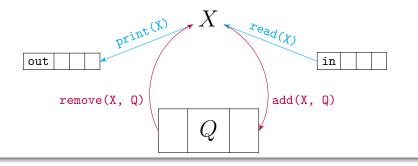
$$(3,2,1):((()))$$
 $(1,2,3):()()()$

Queueable Permutations



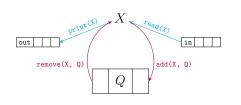
DH 2.14: Queueable Permutations

$$\mathsf{out} = (a_1, \cdots, a_n) \overset{Q = \emptyset}{\underset{X = \bot}{\longleftarrow}} \mathtt{in} = (1, \cdots, n)$$



DH 2.14: Queueable Permutations

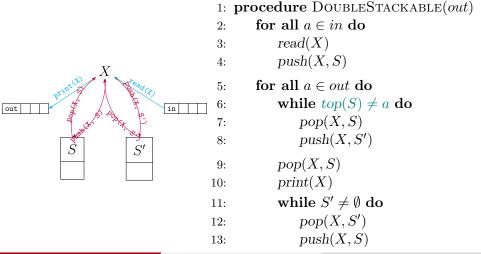
(b) Prove that every permutation are queueable.

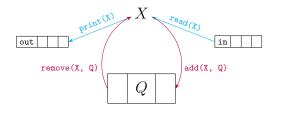


```
1: procedure QUEUEABLE(out)
       for all a \in in do
2:
          read(X)
3:
          add(X,Q)
4:
       for all a \in out do
5:
          while Head(Q) \neq a do
6:
             remove(X,Q)
7:
             add(X,Q)
8:
          remove(X,Q)
9:
          print(X)
10:
```

DH 2.14: Queueable Permutations

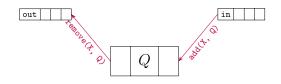
(c) Prove that every permutation can be obtained by two stacks.





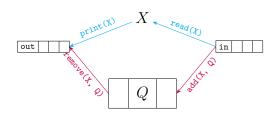
NOT VERY INTERESTING

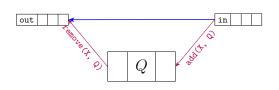
 $All \ are \ queueable.$



Only one is queueable.









$3\ 2\ 1$ is not queueable

Theorem (Queueable Permutations)

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

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

Proof.

Now, it's your turn.



Theorem (# of Queueable Permutations)

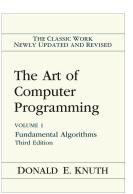
The number of queueable permutations of $[1 \cdots n]$ is $\binom{2n}{n} - \binom{2n}{n-1}$.

Proof.

Now, it's your turn.



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





Chapter 2—Information Structures	232
2.1. Introduction	232
2.2. Linear Lists	238
2.2.1. Stacks, Queues, and Deques	238
2.2.2. Sequential Allocation	244
2.2.3. Linked Allocation	254
2.2.4. Circular Lists	273
2.2.5. Doubly Linked Lists	280
2.2.6. Arrays and Orthogonal Lists	298
2.3. Trees	308
2.3.1. Traversing Binary Trees	318
ŭ i	
2.3.2. Binary Tree Representation of Trees	334
2.3.3. Other Representations of Trees	348
2.3.4. Basic Mathematical Properties of Trees	362
2.3.4.1. Free trees	363
2.3.4.2. Oriented trees	372
*2.3.4.3. The "infinity lemma"	382
*2.3.4.4. Enumeration of trees	386
2.3.4.5. Path length	399
*2.3.4.6. History and bibliography	406
2.3.5. Lists and Garbage Collection	408
2.4. Multilinked Structures	424
2.5. Dynamic Storage Allocation	435
2.6 History and Ribliography	457

THE CLASSIC WORK NEWLY UPDATED AND REVISED

The Art of Computer Programming

VOLUME 1

Fundamental Algorithms Third Edition

DONALD E. KNUTH







Thank You!