

1-5 数据与数据结构 (I)

魏恒峰

hfwei@nju.edu.cn

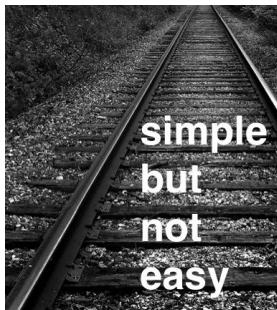
2017 年 11 月 13 日



Permutations

Generating All Permutations
Stackable/Queueable Permutations

Generating All Permutations



DH 2.9: # of Permutations

Prove that the number of permutations of n (distinct) elements is $n!$.

“坊间”证明.

For a_1 : We have n choices.

For a_2 : We have $n - 1$ choices.

For \dots : \dots

Then, # of perms is

$$n \times (n - 1) \times \dots \times 1 = n!$$



Prove by mathematical induction on n .

DH 2.9: # of Permutations

Prove that the number of permutations of n (distinct) elements is $n!$.

Prove by mathematical induction on n .

$P(n)$: # of perms of n (distinct) element is $n!$

B.S. $P(1)$

I.H. $P(n)$

I.S. $P(n) \rightarrow P(n+1)$

$$\underbrace{(n+1)}_{\text{1st choice}} \times \underbrace{n!}_{\text{I.H.}} = (n+1)!$$



DH 2.11: Generate All Permutations

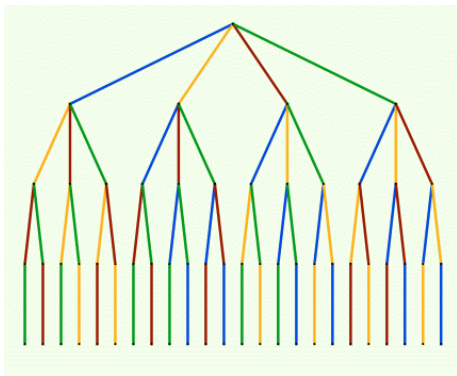
Design an algorithm which, given a positive integer n , generates/prints all the permutations of $[0 \cdots n)$.

```
void perms (A[], n) {  
    if (n == 1)  
        print 'A[0] '  
    else  
        for (int i = 0; i < n; ++i)  
            print 'A[i] '  
            perms(A ← A \ A[i], n - 1)  
            print '\n'  
}
```

generate-perms.c



$$A = [0, 1, 2, 3] \quad n = 4$$




```

void perms (prefix, A[], n) {
    if (n == 1)
        print ' 'prefix ++ A[0] ' '
    else
        for (int i = 0; i < n; ++i)
            perms(prefix ← prefix ++ A[i],
                  A ← A \ A[i], n - 1) // Space???
        print ' '\n'
}

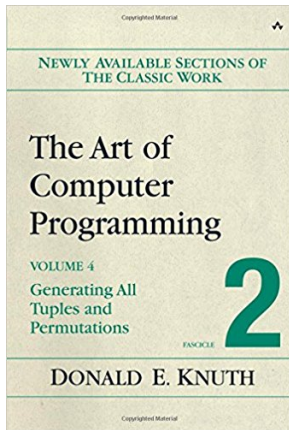
```

```

perms(' ', A, n);

```

For more about “Generating All Permutations”:



DH 2.10: Permutation Checking

- ▶ An integer n
- ▶ An array of integers P of length n

To check whether P is a permutation of $1 \cdots n$?

1. Boolean array $[1 \cdots n]$

$O(n)$
time

$O(n)$
space

2. Sort and then scan

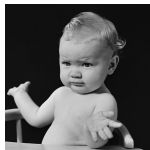
$O(n \log n)$
time

$O(1)$
space



$O(n)$
time

$O(1)$
space

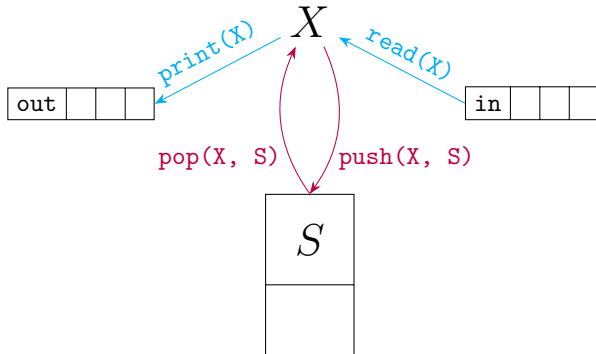


Stackable Permutations

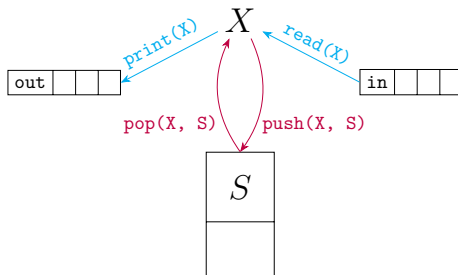


Definition (Stackable Permutations)

$$\text{out} = (a_1, \dots, a_n) \xleftarrow[X=0]{S=\emptyset} \text{in} = (1, \dots, n)$$



Definition (Stackable Permutations)



Q_1 : Meaning of “read, print, push, pop”?

Q_2 : Using **only** “read, print, push, pop”?

$$a == X \quad a > X \ (a < X) \quad \text{top}(S)$$

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.

read print push pop is-empty

$X = 0$ $S = \emptyset$ in \neq EOF

```
foreach 'a' in out:
    if (! is-empty(S)
        && 'a' == top(S))
        pop(S, X)
        print(X)
        continue
    else ... // T.B.C
```

```
else // T.B.C
    while (in  $\neq$  EOF)
        read(X)
        if (X == 'a')
            print(X)
            continue
        else
            push(X, S)
ERR // How???
```


DH 2.12: Stackable Permutations

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

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

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

$(3, 1, 2)$

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

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

312-Pattern

Theorem (Stackable Permutations)

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

312-Pattern : $out = \dots a_i \dots a_j \dots a_k : i < j < k \wedge a_j < a_k < a_i$

Proof.



NO PROOF WARRANTY

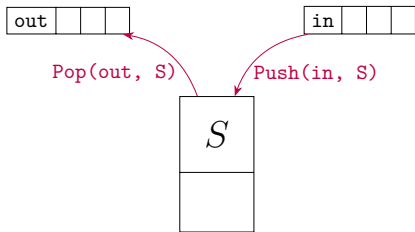
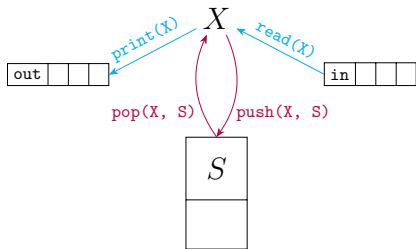


DH 2.12: Stackable Permutations

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

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

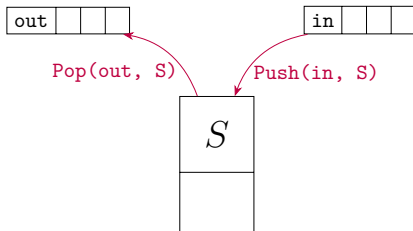
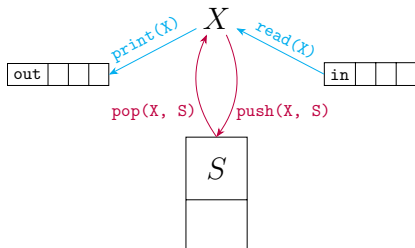
Q : What about A_n ?



Q : Are $S + X$ and S are **equivalent**?

Producing the same set of permutations.

Accepting the same set of *admissible* operation sequences.



By simulations.

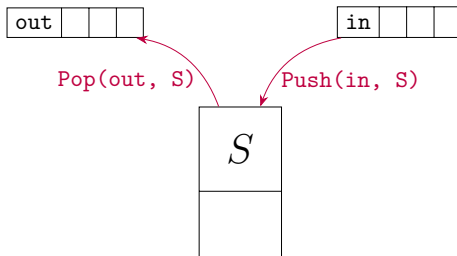
Simulate S by $S + X$:

- Push
- Pop

Simulate $S + X$ by S :

By iterative transformations.



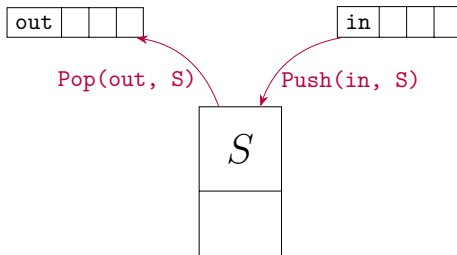


$(1, 2, 3) : \text{Push Pop Push Pop Push Pop}$

$(3, 2, 1) : \text{Push Push Push Pop Pop Pop}$

DH 2.12: Stackable Permutations

How many permutations of $\{1 \cdots n\}$ are stackable on the model S ?



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 stackable perms = # of admissible operation sequences



Theorem

Different admissible operation sequences correspond to different permutations.

Proof.

Push Push Push Pop Pop **Push**...

Push Push Push Pop Pop **Pop**...

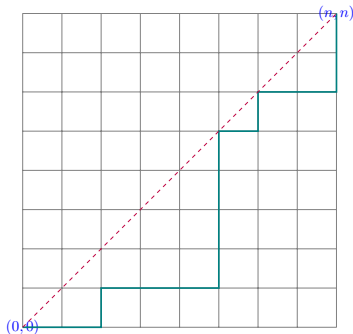


Theorem

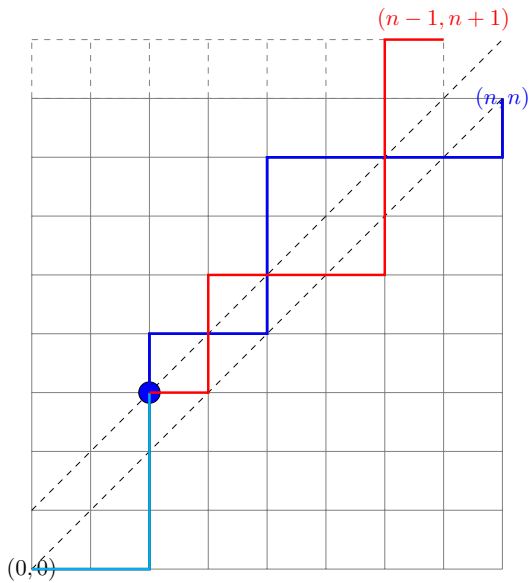
The number of admissible operation sequences of “Push” and “Pop” is $\binom{2n}{n} - \binom{2n}{n-1}$.

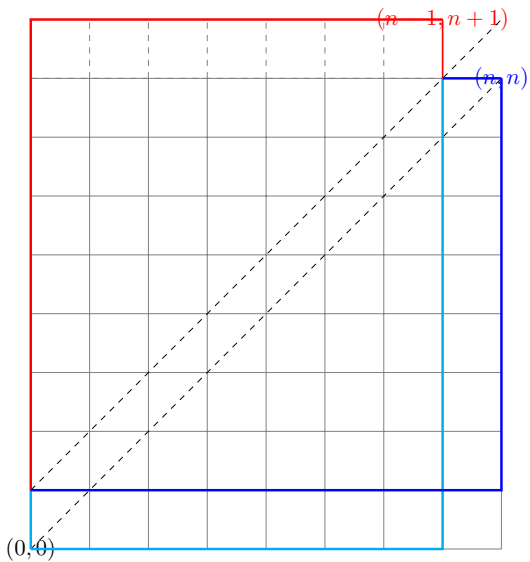
Proof: The Reflection Method.

Push : \rightarrow Pop : \uparrow



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

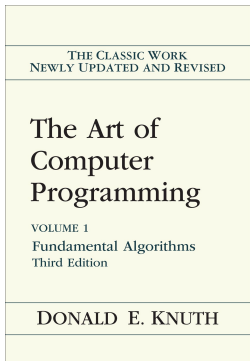




Catalan Number

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

For more about “Stackable Permutations” (Section 2.2.1):



Thank
You!