1-5 数据与数据结构

魏恒峰

hfwei@nju.edu.cn

2017 年 11 月 06 日

Permutations

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 $n!$.

DH 2.9: # of Permutations

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

.

For a_1: We have n choices.

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

For $\cdots$: $\cdots$

Then, # of perms is

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

$\square$

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 $\cdots$: $\cdots$

Then, # of perms is

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

□

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.

$$P(n): \#\text{ of perms of } n \text{ (distinct) element is } n!$$

DH 2.9: # of Permutations

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

Prove by mathematical induction on n.

$$P(n) : \#\text{ of perms of } n \text{ (distinct) element is } n!$$

B.S. $P(1)$

I.H. $P(n)$

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

DH 2.9: # of Permutations

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

Prove by mathematical induction on n.

$$P(n) : \# \text{ of perms of } n \text{ (distinct) element is } n!$$

B.S. $P(1)$

I.H. $P(n)$

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

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

$\square$

DH 2.11: Generate All Permutations

Design an algorithm which, given a positive integer n, generates/prints all the permutations of $[1 \cdots n]$.

DH 2.11: Generate All Permutations

Design an algorithm which, given a positive integer n, generates/prints all the permutations of $[1 \cdots n]$.

```
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''
}
```

DH 2.11: Generate All Permutations

Design an algorithm which, given a positive integer n, generates/prints all the permutations of $[1 \cdots n]$.

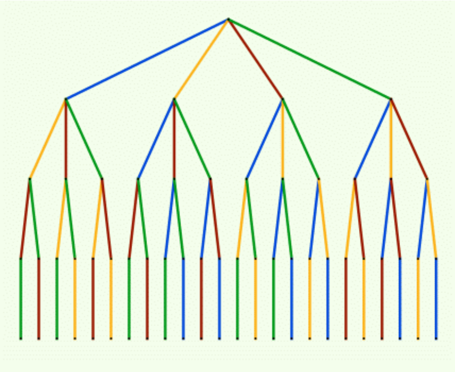
```
void perms (A[], n) {
  if (n == 1)
    print ''A[0]''
  else
    for (int i = 0; i < n; ++i)
      print ''A[i]''
      perms(A ← A \ A[i], n - 1)
      print ''\n''
}
```

generate-perms.c


ERROR


ERROR

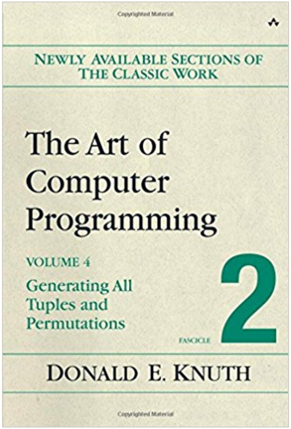
$$A = [0, 1, 2, 3] \qquad n = 4$$



```
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)
      print ''\n''
}
```

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

```
perms('''', A, n);
```


Copyrighted Material
NEWLY AVAILABLE SECTIONS OF THE CLASSIC WORK
The Art of Computer Programming
VOLUME 4
Generating All Tuples and Permutations
FASCICLE
2
DONALD E. KNUTH
Copyrighted Material


Don Knuth

DH 2.10: Permutation Checking

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

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

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 $[1 \cdots n]$

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 $[1 \cdots n]$

$$\underbrace{O(n)}_{\texttt{time}} \quad \underbrace{O(n)}_{\text{space}}$$

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 $[1 \cdots n]$

   $\underbrace{O(n)}_{\texttt{time}} \quad \underbrace{O(n)}_{\text{space}}$

2. Sort and then scan

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 $[1 \cdots n]$

   $\underbrace{O(n)}_{\texttt{time}} \quad \underbrace{O(n)}_{\texttt{space}}$

2. Sort and then scan

   $\underbrace{O(n \log n)}_{\texttt{time}} \quad \underbrace{O(1)}_{\texttt{space}}$

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 $[1 \cdots n]$

   $\underbrace{O(n)}_{\texttt{time}}$ $\underbrace{O(n)}_{\text{space}}$

2. Sort and then scan

   $\underbrace{O(n \log n)}_{\texttt{time}}$ $\underbrace{O(1)}_{\text{space}}$

$\underbrace{O(n)}_{\texttt{time}}$ $\underbrace{O(1)}_{\text{space}}$

DH 2.10: Permutation Checking

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

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

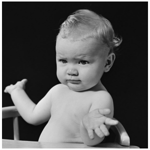
1. Boolean array $[1 \cdots n]$

$\underbrace{O(n)}_{\texttt{time}}$ $\underbrace{O(n)}_{\texttt{space}}$

2. Sort and then scan

$\underbrace{O(n \log n)}_{\texttt{time}}$ $\underbrace{O(1)}_{\texttt{space}}$

$\underbrace{O(n)}_{\texttt{time}}$ $\underbrace{O(1)}_{\texttt{space}}$



Stackable Permutations

Definition (Stackable Permutations)

`read(X)`: `in >>` X

`print(X)`: `out <<` X

`push(X, S)`: $S \Leftarrow X$

`pop(X, S)`: $X \Leftarrow S$

Q_1 : What are X and out after `print(X)`?

A_1 : Elements move around.

Q_2 : $'a' >=< X$, `top`(S)?

A_2 : Yes.

Definition (Stackable Permutations)

`read(X)`: `in >>` X

`print(X)`: `out <<` X

`push(X, S)`: $S \Leftarrow X$

`pop(X, S)`: $X \Leftarrow S$

Q_1 : What are X and out after `print(X)`?

A_1 : Elements move around.

Q_2 : $'a' >=< X$, top(S)?

A_2 : Yes.

$$\texttt{in} = (1, \cdots, n) \xrightarrow[X=0]{S=\emptyset} \texttt{out} = (a_1, \cdots, a_n)$$

fig here.

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.

`read` `print` `push` `pop` `is-empty`

```
X = 0
S = ∅
```

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

```
else // T.B.C
  while (in != ∅)
    read(X)
    if (X == 'a')
      print(X)
      break
    else if (X < 'a')
      push(X, S)
    else // (X > 'a')
      ERR
  ERR
```

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

$$(3, 1, 2)$$

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

$$a_i \cdots a_j \cdots a_k : i < j < k \wedge a_j < a_k < a_i$$

DH 2.12: Stackable Permutations

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

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

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

$$(3, 1, 2)$$

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

$$a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i$$

312-Pattern

Theorem (Stackable Permutations)

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

$$a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i$$

Theorem (Stackable Permutations)

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

$$a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i$$

Proof.

$$\iff$$

Theorem (Stackable Permutations)

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

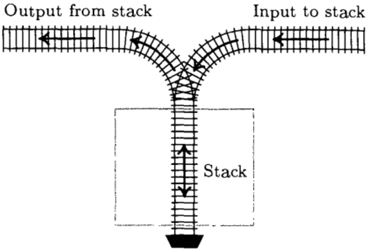
$$a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i$$

Proof.

$$\iff$$

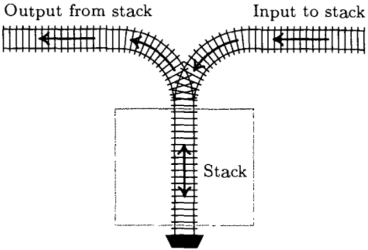
$$\Rightarrow \qquad\qquad \Leftarrow$$

$\square$



Theorem (Equivalence)

These two models ($S + X$ and S) are equivalent.



Theorem (Equivalence)

These two models ($S + X$ and S) are equivalent.

Proof.

By simulations.

Simulate S by $S + X$:

- `Push`
- `Pop`

Simulate $S + X$ by S:
By iterative transformations.

□

Theorem (Stackable Permutations)

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

$$a_i \cdots a_j \cdots a_k : i < j < k \wedge a_j < a_k < a_i$$

Proof.

$$\implies$$

By contradiction.

$a_j < a_k < a_i$: When a_i is poped, a_j and a_k are on the stack.

$j < k$: a_j is above a_k on the stack.

$a_j < a_k$: Contradiction.

□

Theorem (Stackable Permutations)

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

$$a_i \cdots a_j \cdots a_k : i < j < k \land a_j < a_k < a_i$$

Proof.

$$\Longleftarrow$$

According to our algorithms and by contradiction.

$$a_j \notin \texttt{in} \land a_j \,!= \texttt{top}(S) \implies \exists k > j : a_k > a_j$$

$$a_j, a_k \implies \exists i < j (< k) : a_j < a_k < a_i$$

□

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

How many permutations of A_n *cannot* be obtained by a stack?

DH 2.12: Stackable Permutations

How many permutations of A_n *cannot* be obtained by a stack?

$$\texttt{Push} : + \qquad \texttt{Pop} : -$$

$$(3, 2, 5, 6, 1, 4) : + + + - - + + - + - - -$$

Definition (Admissible Sequences)

A sequence of "$+$" and "$-$" is admissible if and only if

Definition (Admissible Sequences)

A sequence of "$+$" and "$-$" is admissible if and only if

1. # of "$+$" $= n$ # of "$-$" $= n$
2. $\forall$ prefix : (# of "$-$") $\leq$ (# of "$+$")

Definition (Admissible Sequences)

A sequence of "$+$" and "$-$" is admissible if and only if

1. # of "$+$" $= n$ # of "$-$" $= n$
2. $\forall$ prefix : (# of "$-$") $\leq$ (# of "$+$")

Theorem

Different admissible sequences correspond to different permutations.

Definition (Admissible Sequences)

A sequence of "$+$" and "$-$" is admissible if and only if

1. $\#$ of "$+$" $= n$ $\qquad$ $\#$ of "$-$" $= n$
2. $\forall$ prefix : ($\#$ of "$-$") $\leq$ ($\#$ of "$+$")

Theorem

Different admissible sequences correspond to different permutations.

$$+ + + - - + \cdots$$

$$+ + + - - - \cdots$$

Theorem (Reflection Method)

The number of stackable permutations is $\binom{2n}{n} - \binom{2n}{n-1}$.


Thank You!