

## 4. 栈与队列

### (c3) 栈应用：栈混洗

邓俊辉

[deng@tsinghua.edu.cn](mailto:deng@tsinghua.edu.cn)

## 栈混洗

❖ 考查栈  $A = \langle a_1, a_2, \dots, a_n \rangle$ 、 $B = S = \emptyset$

//左端为栈顶

❖ 只允许 将A的顶元素弹出并压入S，或  
将S的顶元素弹出并压入B

//S.push(A.pop())

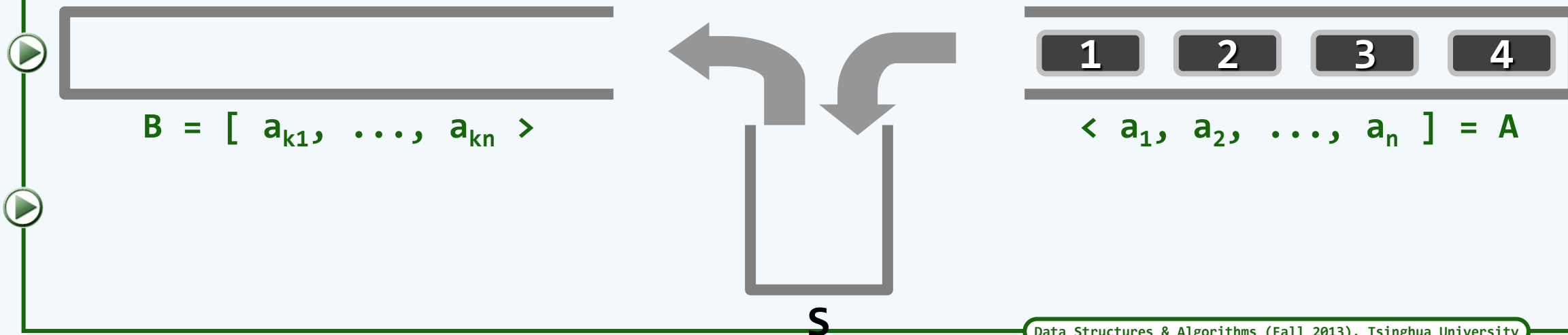
//B.push(S.pop())

❖ 若经过一系列以上操作后，A中元素全部转入B中

$B = \langle a_{k1}, \dots, a_{kn} \rangle$

//右端为栈顶

则称之为A的一个栈混洗 (stack permutation)



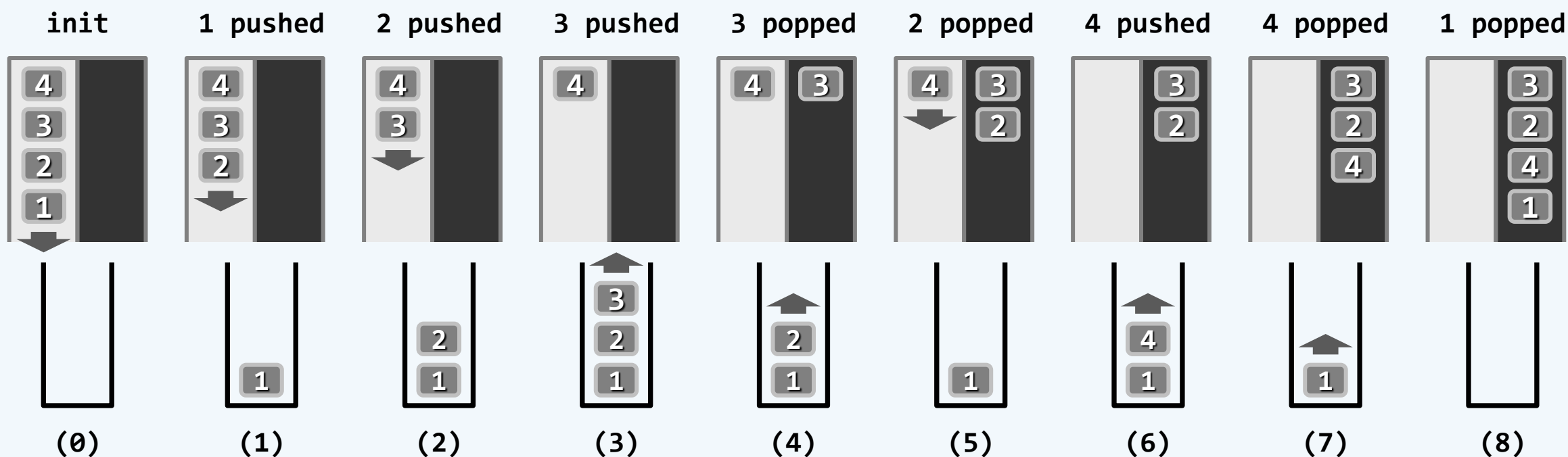
## 计数

❖ 同一输入序列，可有多种栈混洗

[ 1, 2, 3, 4 > , [ 4, 3, 2, 1 > , [ 3, 2, 4, 1 > , ...

❖ 长度为n的序列，可能的混洗总数 $SP(n) = ?$

//显然， $SP(n) \leq n!$



## 计数

❖  $SP(1) = 1$

❖ 设栈S在第k次pop()之后重新变空，则k无非n种情况：

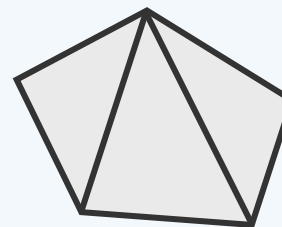
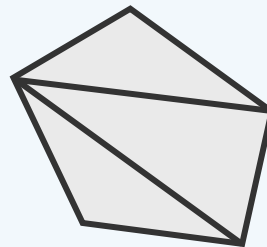
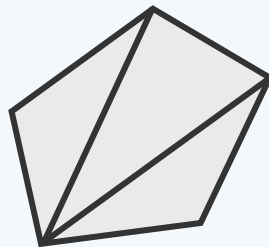
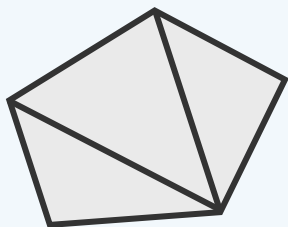
$$SP(n) = \sum_{k=1}^n SP(k-1) \cdot SP(n-k) = \text{Catalan}(n) = (2n)! / (n+1)! / n!$$

❖  $SP(2) = 4! / 3! / 2! = 2$

$SP(3) = 6! / 4! / 3! = 5$

... ..

$SP(6) = 12! / 7! / 6! = 132$



## 甄别

❖ 输入序列  $\langle 1, 2, 3, \dots, n \rangle$  的  
任一排列  $[p_1, p_2, p_3, \dots, p_n]$   
是否为栈混洗？

❖ 简单情况： $\langle 1, 2, 3 \rangle, n = 3$

栈混洗共  $6! / 4! / 3! = 5$  种

全排列共  $3! = 6$  种

//少了一种...

❖  $[3, 1, 2]$

//为什么是它？

❖ 观察：任意三个元素能否按某相对次序出现于混洗中，与其它元素无关 //故可推而广之...

❖ 对于任何  $1 \leq i < j < k \leq n$ ,  $[ \dots, \boxed{k}, \dots, \boxed{i}, \dots, \boxed{j}, \dots ]$  必非栈混洗

❖ 反过来，不存在“312”模式的序列，一定是栈混洗吗？

## 甄别

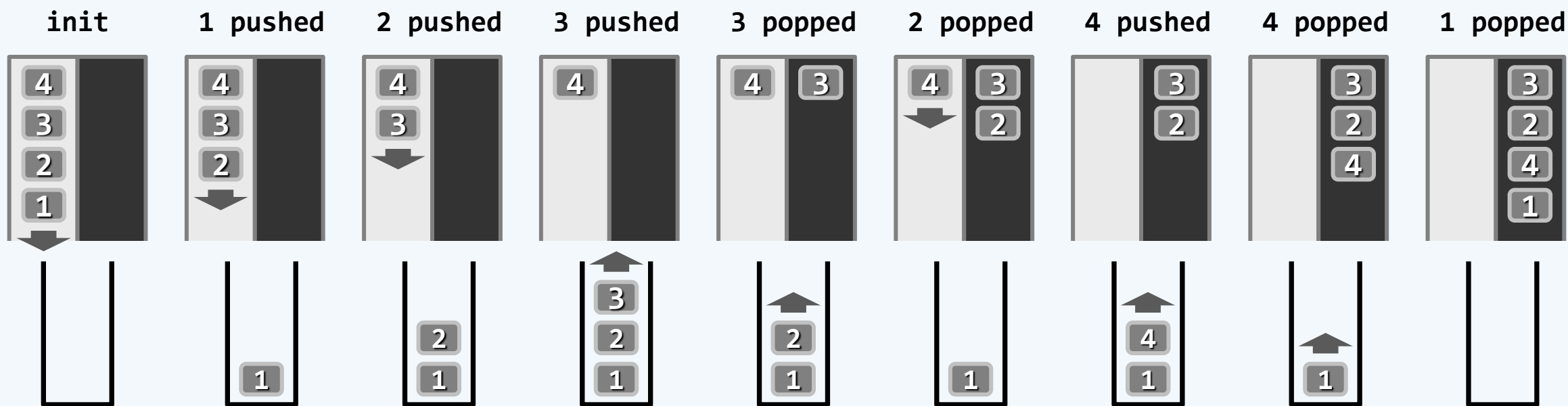
- ❖ 充要性： A permutation is a stack permutation **iff**  
(Knuth, 1968) it does **NOT** involve the permutation **312** //习题[4-3]
- ❖ 如此，可得一个 $O(n^3)$ 的甄别算法 //进一步地...
- ❖  $[p_1, p_2, p_3, \dots, p_n]$ 是 $[1, 2, 3, \dots, n]$ 的栈混洗，**当且仅当**  
对于任意 $i < j$ ，不含模式 $[\dots, \boxed{j+1}, \dots, \boxed{i}, \dots, \boxed{j}, \dots]$
- ❖ 如此，可得一个 $O(n^2)$ 的甄别算法 //再进一步地...
- ❖  $O(n)$ 算法：直接借助栈A、B和S，模拟混洗过程 //为何可行？  
每次S.pop()之前，检测S是否已空；或需弹出的元素在S中，却非顶元素

## 括号匹配

❖ 观察：每一栈混洗，都对应于栈S的  $n$  次 push 与  $n$  次 pop 操作构成的序列

( ( ( ) ) ( ) )

push(1) push(2) push(3) pop(3) pop(2) push(4) pop(4) pop(1)



❖  $n$  个元素的栈混洗，等价于  $n$  对括号的匹配