

# 04-E

栈与队列

栈混洗

邓俊辉

deng@tsinghua.edu.cn

# Stack Permutation

❖ 考查栈  $\mathcal{A} = \langle a_1, a_2, a_3, \dots, a_n \rangle$

$$\mathcal{B} = \mathcal{S} = \emptyset$$

❖ 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]

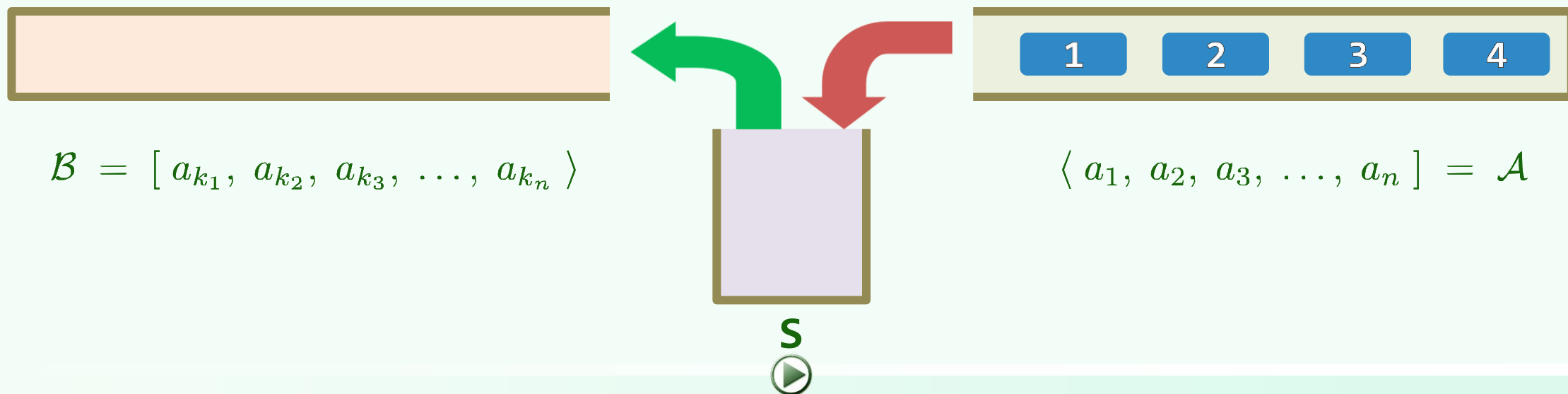
❖ 亦即  $\mathcal{S}.push(\mathcal{A}.pop())$

$$\mathcal{B}.push(\mathcal{S}.pop())$$

❖ 若经一系列以上操作后, A中元素全部转入B中

$$\mathcal{B} = \langle a_{k_1}, a_{k_2}, a_{k_3}, \dots, a_{k_n} \rangle$$

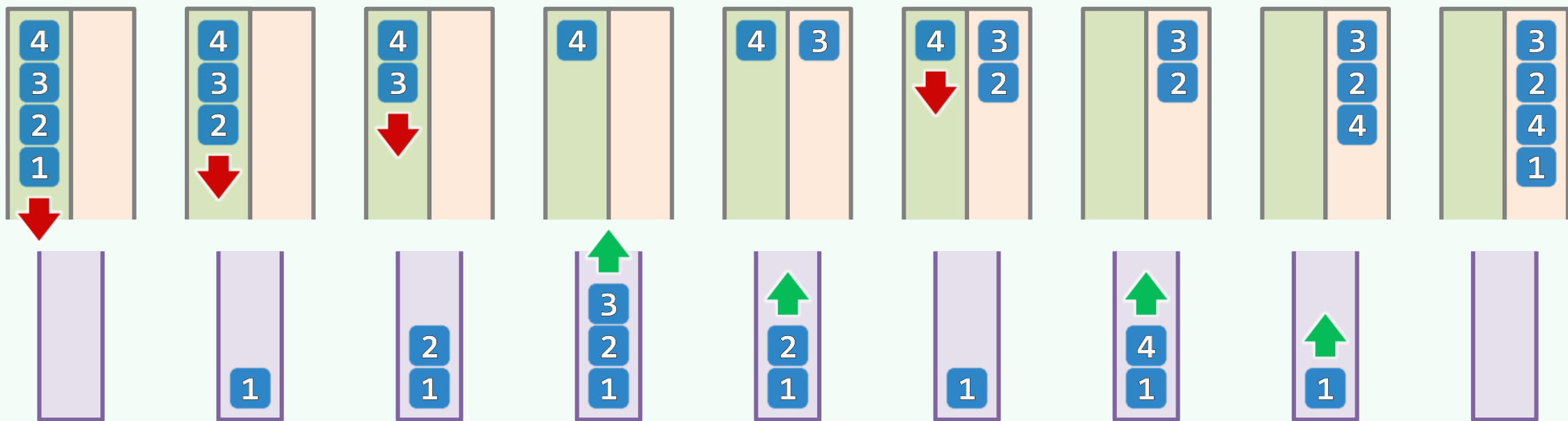
则称为A的一个**栈混洗**



## 计数: $SP(n)$

❖ 同一输入序列, 可有多种栈混洗:  $[1, 2, 3, 4 >$ ,  $[4, 3, 2, 1 >$ ,  $[3, 2, 4, 1 > \dots$

❖ 一般地, 对于长度为 $n$ 的序列, 混洗总数 $SP(n) = ?$



❖ 显然,  $SP(n) \leq n!$ ; 更准确地呢?

# 计数: catalan(n)

❖  $SP(1) = 1$

❖ 考查s再度变空 (A首元素从s中弹出) 的时刻, 无非n种情况:

$$SP(n) = \sum_{k=1}^n SP(k-1) \cdot SP(n-k)$$

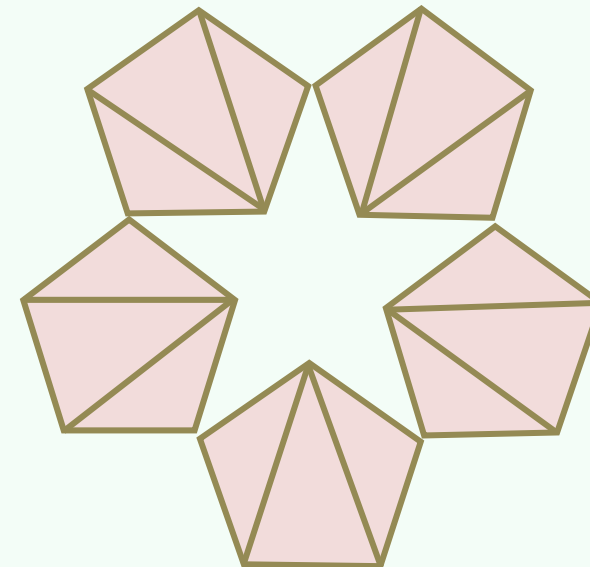
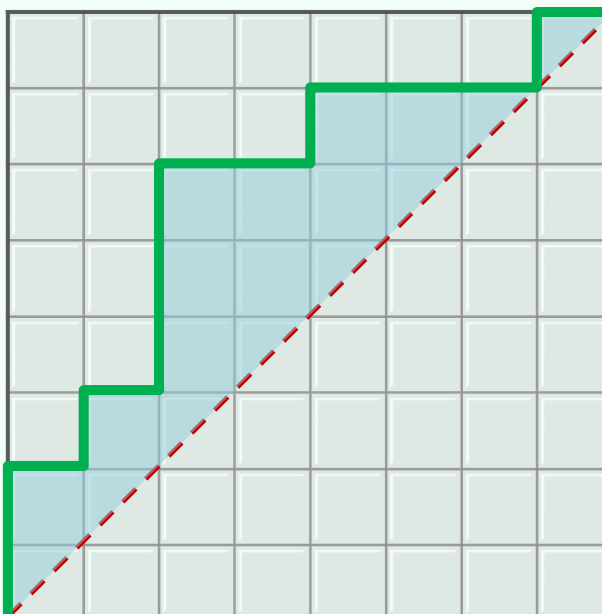
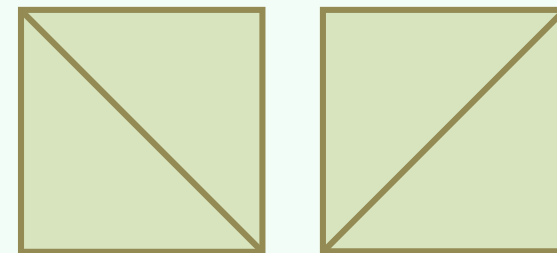
$$= catalan(n) = \frac{(2n)!}{(n+1)! \cdot n!}$$

$$SP(2) = 4!/3!/2! = 2$$

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

...                      ...                      ...

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



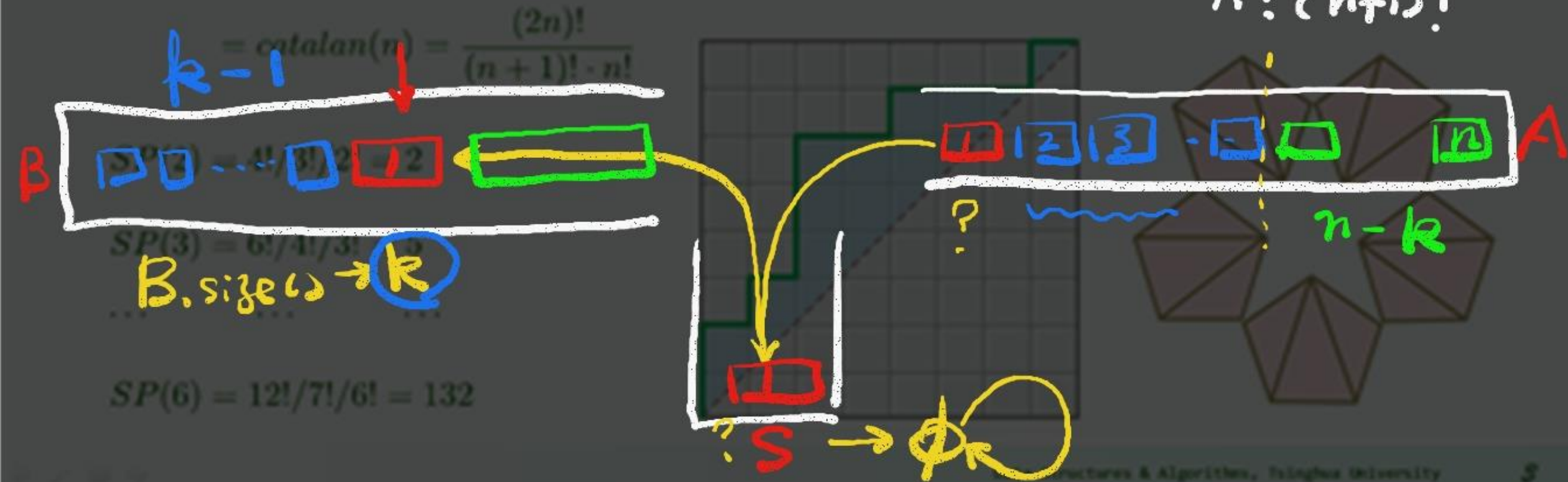
计数:  $catalan(n)$

$$SP(n) = \sum_{k=1}^n SP(k-1) \cdot SP(n-k)$$

✧ 考查S再度变空 (A首元素从S中弹出) 的时刻, 无非n种情况:

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

$$k-1 = \text{catalan}(n) = \frac{(2n)!}{(n+1)! \cdot n!}$$



## 甄别：检测禁形

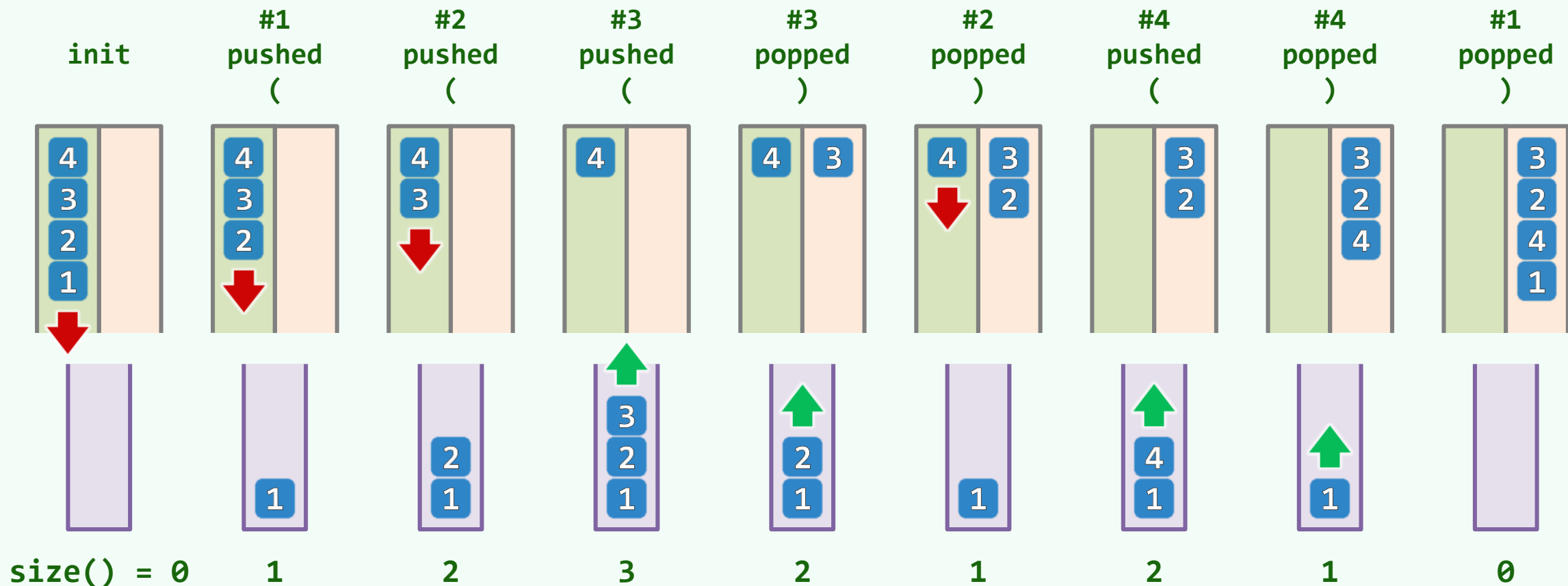
- ❖ 输入序列  $\langle 1, 2, 3, \dots, n \rangle$  的任一排列  $[p_1, p_2, p_3, \dots, p_n]$  是否为栈混洗？
- ❖ 先考查简单情况：  $n = 3, A = \langle 1, 2, 3 \rangle$ 
  - 栈混洗共  $6! / 4! / 3! = 5$  种；全排列共  $3! = 6$  种 //少了一种...
- ❖  $[3, 1, 2]$  //为什么是它？
- ❖ 观察：任意三个元素能否按某相对次序出现于混洗中，与其它元素无关 //故可推而广之...
- ❖ 禁形：对任何  $1 \leq i < j < k \leq n$ ,  $[\dots, \boxed{k}, \dots, \boxed{i}, \dots, \boxed{j}, \dots]$  **必非**栈混洗
- ❖ 反过来，不存在 “ $\boxed{312}$ ”模式的序列，一定是栈混洗吗？

## 甄别：直接模拟

- ❖ 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 $[\dots, j+1, \dots, i, \dots, j, \dots]$
- ❖ 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操作构成的某一序列；反之亦然



❖ n个元素的栈混洗，等价于n对括号的匹配；二者的组合数，也自然相等