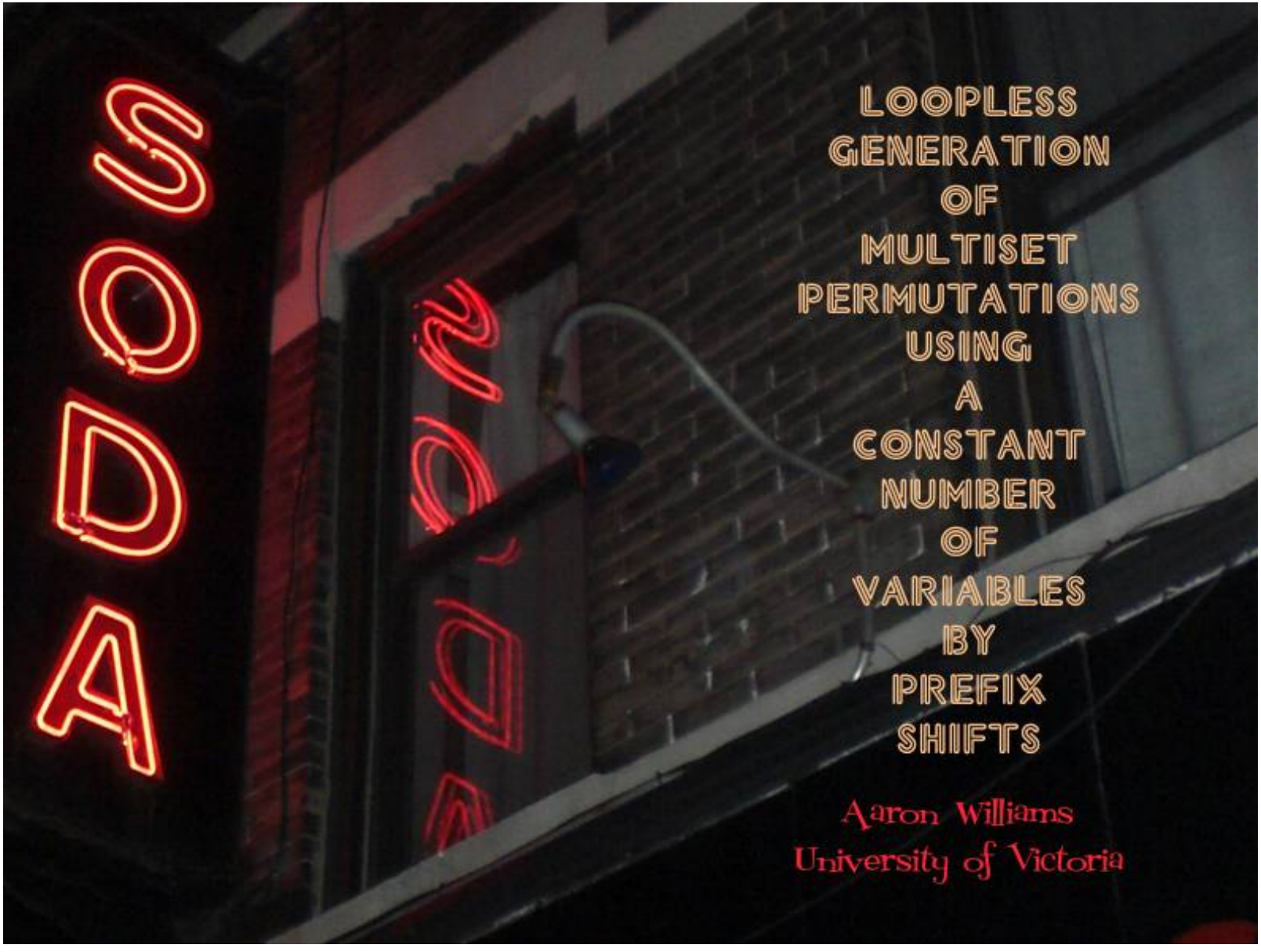

SODA
LOOPLESS
GENERATION
OF
MULTISET
PERMUTATIONS
USING
A
CONSTANT
NUMBER
OF
VARIABLES
BY
PREFIX
SHIFTS
Aaron Williams
University of Victoria

Basic Definitions

Multisets are sets that allow repetitions

$$\mathbb{E} = \{1, 1, 2, 3, 4, 4\}.$$

Assume $\mathbb{E}$ has m totally-ordered distinct symbols and n total symbols.

The set $\Pi(\mathbb{E})$ includes every permutation of the n symbols in $\mathbb{E}$

$$\Pi(\mathbb{E}) = \{112344, 112434, 112443, 113244, \ldots, 443211\}.$$

Given a permutation a *prefix-shift* moves the symbol in a given position into the first position

$$\sigma_5(412341) = 441231.$$

Basic Definitions

Multisets are sets that allow repetitions

$$\mathbb{E} = \{1, 1, 2, 3, 4, 4\}.$$

Assume $\mathbb{E}$ has m totally-ordered distinct symbols and n total symbols.

The set $\Pi(\mathbb{E})$ includes every permutation of the n symbols in $\mathbb{E}$

$$\Pi(\mathbb{E}) = \{112344, 112434, 112443, 113244, \ldots, 443211\}.$$

Given a permutation a *prefix-shift* moves the symbol in a given position into the first position

$$\sigma_5(412341) = 441231.$$

Basic Definitions

Multisets are sets that allow repetitions

$$\mathbb{E} = \{1, 1, 2, 3, 4, 4\}.$$

Assume $\mathbb{E}$ has m totally-ordered distinct symbols and n total symbols.

The set $\Pi(\mathbb{E})$ includes every permutation of the n symbols in $\mathbb{E}$

$$\Pi(\mathbb{E}) = \{112344, 112434, 112443, 113244, \ldots, 443211\}.$$

Given a permutation a *prefix-shift* moves the symbol in a given position into the first position

$$\sigma_5(412341) = 441231.$$

Basic Definitions

Multisets are sets that allow repetitions

$$\mathbb{E} = \{1, 1, 2, 3, 4, 4\}.$$

Assume $\mathbb{E}$ has m totally-ordered distinct symbols and n total symbols.

The set $\Pi(\mathbb{E})$ includes every permutation of the n symbols in $\mathbb{E}$

$$\Pi(\mathbb{E}) = \{112344, 112434, 112443, 113244, \ldots, 443211\}.$$

Given a permutation a *prefix-shift* moves the symbol in a given position into the first position

$$\sigma_5(412341) = 441231.$$

Discrete Question

Can multiset permutations be ordered by prefix shifts?

- each permutation in $\Pi(\mathbb{E})$ appears exactly once in the order
- every permutation $\mathbf{s} \in \Pi(\mathbb{E})$ is followed by some $\sigma_j(\mathbf{s})$

Discrete Question

Can multiset permutations be ordered by prefix shifts?

- each permutation in $\Pi(\mathbb{E})$ appears exactly once in the order
- every permutation $\mathbf{s} \in \Pi(\mathbb{E})$ is followed by some $\sigma_j(\mathbf{s})$

Discrete Question

Can multiset permutations be ordered by prefix shifts?

- each permutation in $\Pi(\mathbb{E})$ appears exactly once in the order
- every permutation $\mathbf{s} \in \Pi(\mathbb{E})$ is followed by some $\sigma_j(\mathbf{s})$

Discrete Question

Can multiset permutations be ordered by prefix shifts?

- each permutation in $\Pi(\mathbb{E})$ appears exactly once in the order
- every permutation $\mathbf{s} \in \Pi(\mathbb{E})$ is followed by some $\sigma_j(\mathbf{s})$

For example,

321

231

123

213

is ordered by prefix shifts, but it does not contain every permutation of $\{1, 2, 3\}$ and it cannot be extended using prefix-shifts.

Discrete Question

Can multiset permutations be ordered by prefix shifts?

- each permutation in $\Pi(\mathbb{E})$ appears exactly once in the order
- every permutation $\mathbf{s} \in \Pi(\mathbb{E})$ is followed by some $\sigma_j(\mathbf{s})$

On the other hand,

321

231

123

312

132

213

is ordered by prefix shifts, and contains every permutation of $\{1, 2, 3\}$ exactly once.

Historical Context

$\Pi(\mathbb{E})$ are *permutations* when $n = m$, and *combinations* when $m = 2$.

Operation \ Object	multiset permutations	permutations	combinations
prefix-shifts	open	yes	yes
σ_{n-1},σ_n	open	yes	open
shifts	yes	yes	yes

Historical Context

$\Pi(\mathbb{E})$ are *permutations* when $n = m$, and *combinations* when $m = 2$.

Operation \ Object	multiset permutations	permutations	combinations
prefix-shifts	open	yes	yes
σ_{n-1},σ_n	open	yes	open
shifts	yes	yes	yes

Historical Context

$\Pi(\mathbb{E})$ are *permutations* when $n = m$, and *combinations* when $m = 2$.

Operation \ Object	multiset permutations	permutations	combinations
prefix-shifts	open	yes	yes
σ_{n-1},σ_n	open	yes	open
shifts	yes	yes	yes

- **W**. SODA 2009.

Historical Context

$\Pi(\mathbb{E})$ are *permutations* when $n = m$, and *combinations* when $m = 2$.

Operation \ Object	multiset permutations	permutations	combinations
prefix-shifts	open	yes	yes
σ_{n-1},σ_n	open	yes	open
shifts	yes	yes	yes

- **F. Ruskey, W.** *The coolest way to generate combinations*. Discrete Mathematics, (in press), 2008.

Historical Context

$\Pi(\mathbb{E})$ are *permutations* when $n = m$, and *combinations* when $m = 2$.

Operation \ Object	multiset permutations	permutations	combinations
prefix-shifts	open	yes	yes
σ_{n-1},σ_n	open	yes	open
shifts	yes	yes	yes

- **P. F. Corbett**. *Rotator graphs: An efficient topology for point-to-point multiprocessor networks*. IEEE Transactions on Parallel and Distributed Systems, 3:622–626, 1992.
- **M. Jiang, F. Ruskey**. *Determining the Hamilton- connectedness of certain vertex-transitive graphs*. Discrete Mathematics, 133:159–170, 1994.
- Hamiltonian cycle in directed Cayley graph with generators $(1\ 2), (1\ 2\ 3), \ldots, (1\ 2\ \cdots\ n)$

Historical Context

$\Pi(\mathbb{E})$ are *permutations* when $n = m$, and *combinations* when $m = 2$.

Operation \ Object	multiset permutations	permutations	combinations
prefix-shifts	open	yes	yes
σ_{n-1},σ_n	open	yes	open
shifts	yes	yes	yes

- **F. Ruskey, W.** *An explicit universal cycle for the $(n-1)$-permutations of an n-set.* ACM Transactions on Algorithms, (in press) 2008.
- Hamiltonian cycle in directed Cayley graph with generators $(1\ 2\ \cdots\ n-1), (1\ 2\ \cdots\ n)$

Historical Context

$\Pi(\mathbb{E})$ are *permutations* when $n = m$, and *combinations* when $m = 2$.

Operation \ Object	multiset permutations	permutations	combinations
prefix-shifts	open	yes	yes
σ_{n-1},σ_n	open	yes	open
shifts	yes	yes	yes

- **J. F. Korsh, S. Lipschutz**. *Generating multiset permutations in constant time*. Journal of Algorithms, 25:321–335, 1997.

Historical Context

Transpositions swap the symbols in two positions.

Operation \ Object	multiset permutations	permutations	combinations
adjacent-transposition	no	yes	no
transposition	yes	yes	yes

Historical Context

Transpositions swap the symbols in two positions.

Operation \ Object	multiset permutations	permutations	combinations
adjacent-transposition	no	yes	no
transposition	yes	yes	yes

- **S. M. Johnson**. *Generation of permutations by adjacent transpositions*. Mathematics of Computation, 17:282–285, 1963.
- **H. Steinhaus**. *One Hundred Problems in Elementary Mathematics*. Pergamon Press, 1963.
- **H. Trotter**. *Algorithm 115: Perm*. Comm. ACM, 434–435, 1963.
- Hamiltonian cycle in directed Cayley graph with generators $(1\ 2), (2\ 3), \ldots, (n-1\ n)$

Historical Context

Transpositions swap the symbols in two positions.

Operation \ Object	multiset permutations	permutations	combinations
adjacent-transposition	no	yes	no
transposition	yes	yes	yes

- **F. Ruskey**. *Generating linear extensions of posets by transpositions*. Journal of Combinatorial Theory (B), 54:77–101, 1992.
- **M. Buck, D. Wiedemann**. *Gray codes with restricted density*. Discrete Math., 48:163–171, 1984.
- **P. Eades, M. Hickey, R. Read**. *Some Hamilton Paths and a Minimal Change Algorithm*, Journal of the ACM, 31:19–29, 1984.

Historical Context

Transpositions swap the symbols in two positions.

Operation \ Object	multiset permutations	permutations	combinations
adjacent-transposition	no	yes	no
transposition	yes	yes	yes

- **D. T. Tang, C.N. Liu**. *Distance-2 Cycle Chaining of Constant Weight Codes*, IEEE Transactions, C–22:176–180, 1973.
- **P. Eades, B. McKay**. *An Algorithm for Generating Subsets of Fixed Size with a Strong Minimal Change Property*, Information Processing Letters, 19:131–133, 1984.
- **P. J. Chase**. *Combination Generation and Graylex Ordering*, Congressus Numerantium, 69:215–242, 1989.

Historical Context

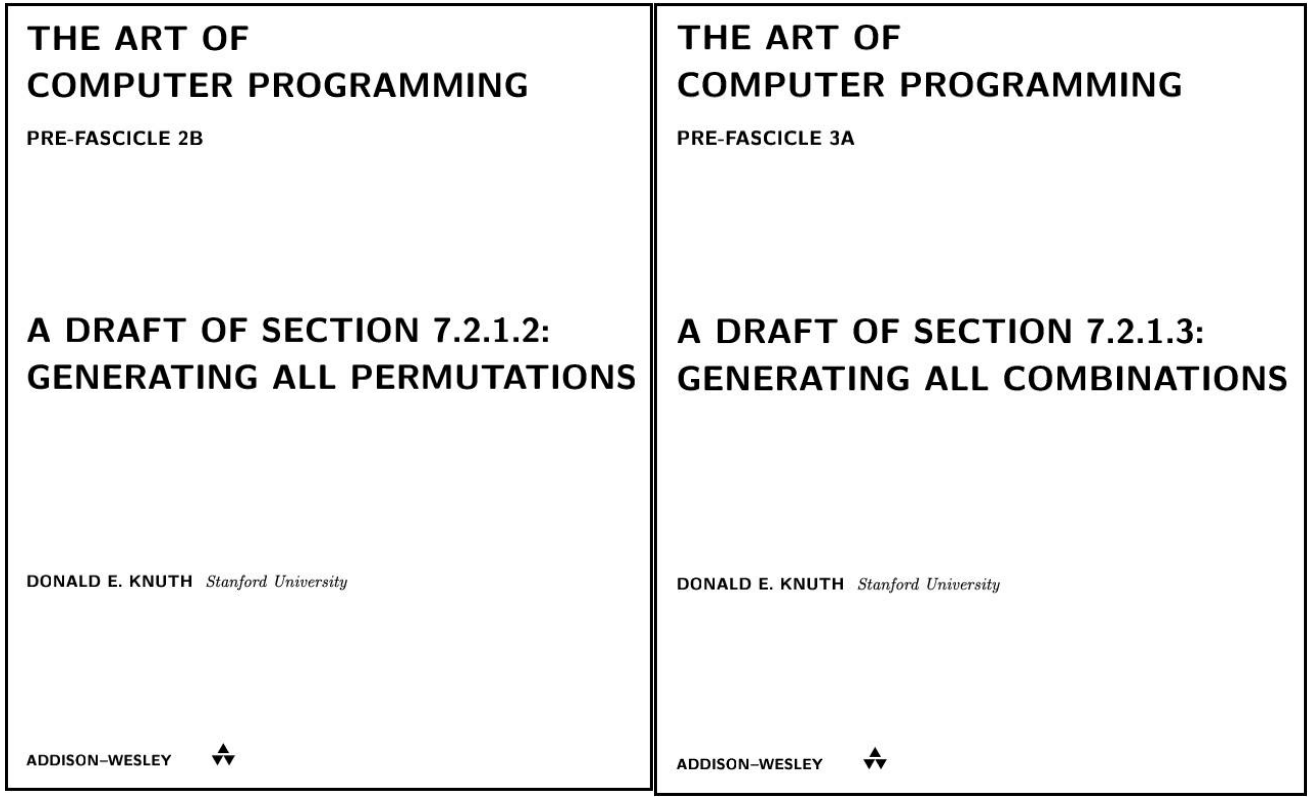
Transpositions swap the symbols in two positions.

Operation \ Object	multiset permutations	permutations	combinations
adjacent-transposition	no	yes	no
transposition	yes	yes	yes

- **C. W. Ko, F. Ruskey**. Generating permutations of a bag by interchanges. Information Processing Letters, 41:263–269, 1992.
- **T. Takaoka**. *An O(1) time algorithm for generating multiset permutations*. Lecture Notes in Computer Science (ISAAC 99), 1741:237–246, 1999.
- **V. Vajnovszki**. *A loopless algorithm for generating the permutations of a multiset*. Theoretical Computer Science, 2(307):415–431, 2003.
- **J. F. Korsh, P. S. LaFollette**. *Loopless array generation of multiset permutations*. The Computer Journal, 47(5):612–621, 2004.

Historical Context

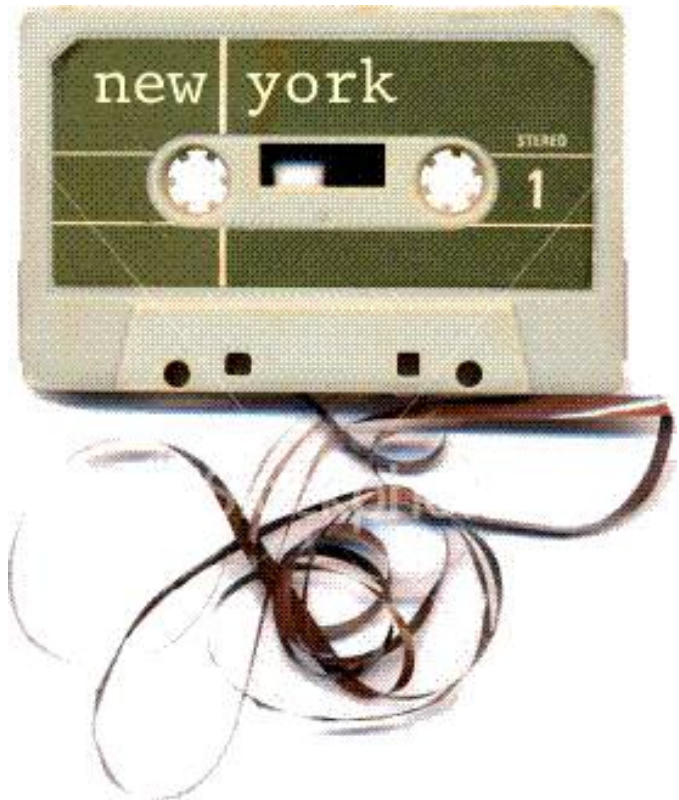
Combinatorial generation is featured in over 400 pages of the next volume of *The Art of Computer Programming*.



Modern history of combinatorial generation dates back to the *Binary Reflected Gray Code* and *de Bruijn cycles* circa 1940.

Historical Context

Combinatorial generation is featured in over 400 pages of the next volume of *The Art of Computer Programming*.



Modern history of combinatorial generation dates back to the *Binary Reflected Gray Code* and *de Bruijn cycles* circa 1940.

Algorithmic Question

Can multiset permutations be ordered in $O(1)$-time using $O(1)$-variables?

- single shared object modified "in-place"
  - multiset permutation stored in array or linked list requiring n variables
- $O(1)$-*time* per modification in the worst case
  - "loopless algorithm" must use minimal-change order
- $O(1)$-*variables* means $O(1)$-*additional variables*
  - variables storing the single shared object are not counted
  - stack variables are counted

The last condition forbids storing the symbol frequencies, the total number of permutations, or from using recursion. (The best current algorithms that use $O(1)$-time use at least $O(n)$-variables.)

Algorithmic Question

Can multiset permutations be ordered in $O(1)$-time using $O(1)$-variables?

- single shared object modified "in-place"
  - multiset permutation stored in array or linked list requiring n variables
- $O(1)$-*time* per modification in the worst case
  - "loopless algorithm" must use minimal-change order
- $O(1)$-*variables* means $O(1)$-*additional variables*
  - variables storing the single shared object are not counted
  - stack variables are counted

The last condition forbids storing the symbol frequencies, the total number of permutations, or from using recursion. (The best current algorithms that use $O(1)$-time use at least $O(n)$-variables.)

Algorithmic Question

Can multiset permutations be ordered in $O(1)$-time using $O(1)$-variables?

- single shared object modified "in-place"
  - multiset permutation stored in array or linked list requiring n variables
- $O(1)$-*time* per modification in the worst case
  - "loopless algorithm" must use minimal-change order
- $O(1)$-*variables* means $O(1)$-*additional variables*
  - variables storing the single shared object are not counted
  - stack variables are counted

The last condition forbids storing the symbol frequencies, the total number of permutations, or from using recursion. (The best current algorithms that use $O(1)$-time use at least $O(n)$-variables.)

Algorithmic Question

Can multiset permutations be ordered in $O(1)$-time using $O(1)$-variables?

- single shared object modified "in-place"
  - multiset permutation stored in array or linked list requiring n variables
- $O(1)$-*time* per modification in the worst case
  - "loopless algorithm" must use minimal-change order
- $O(1)$-*variables* means $O(1)$-*additional variables*
  - variables storing the single shared object are not counted
  - stack variables are counted

The last condition forbids storing the symbol frequencies, the total number of permutations, or from using recursion. (The best current algorithms that use $O(1)$-time use at least $O(n)$-variables.)

Algorithmic Question

Can multiset permutations be ordered in $O(1)$-time using $O(1)$-variables?

- single shared object modified "in-place"
  - multiset permutation stored in array or linked list requiring n variables
- $O(1)$-*time* per modification in the worst case
  - "loopless algorithm" must use minimal-change order
- $O(1)$-*variables* means $O(1)$-*additional variables*
  - variables storing the single shared object are not counted
  - stack variables are counted

The last condition forbids storing the symbol frequencies, the total number of permutations, or from using recursion. (The best current algorithms that use $O(1)$-time use at least $O(n)$-variables.)

Algorithmic Question

Can multiset permutations be ordered in $O(1)$-time using $O(1)$-variables?

- single shared object modified "in-place"
  - multiset permutation stored in array or linked list requiring n variables
- $O(1)$-*time* per modification in the worst case
  - "loopless algorithm" must use minimal-change order
- $O(1)$-*variables* means $O(1)$-*additional variables*
  - variables storing the single shared object are not counted
  - stack variables are counted

The last condition forbids storing the symbol frequencies, the total number of permutations, or from using recursion. (The best current algorithms that use $O(1)$-time use at least $O(n)$-variables.)

Algorithmic Question

Can multiset permutations be ordered in $O(1)$-time using $O(1)$-variables?

- single shared object modified "in-place"
  - multiset permutation stored in array or linked list requiring n variables
- $O(1)$-*time* per modification in the worst case
  - "loopless algorithm" must use minimal-change order
- $O(1)$-*variables* means $O(1)$-*additional variables*
  - variables storing the single shared object are not counted
  - stack variables are counted

The last condition forbids storing the symbol frequencies, the total number of permutations, or from using recursion. (The best current algorithms that use $O(1)$-time use at least $O(n)$-variables.)

Algorithmic Question

Can multiset permutations be ordered in $O(1)$-time using $O(1)$-variables?

- single shared object modified "in-place"
  - multiset permutation stored in array or linked list requiring n variables
- $O(1)$-*time* per modification in the worst case
  - "loopless algorithm" must use minimal-change order
- $O(1)$-*variables* means $O(1)$-*additional variables*
  - variables storing the single shared object are not counted
  - stack variables are counted

The last condition forbids storing the symbol frequencies, the total number of permutations, or from using recursion. (The best current algorithms that use $O(1)$-time use at least $O(n)$-variables.)

Algorithmic Question

Can multiset permutations be ordered in $O(1)$-time using $O(1)$-variables?

- single shared object modified "in-place"
  - multiset permutation stored in array or linked list requiring n variables
- $O(1)$-*time* per modification in the worst case
  - "loopless algorithm" must use minimal-change order
- $O(1)$-*variables* means $O(1)$-*additional variables*
  - variables storing the single shared object are not counted
  - stack variables are counted

The last condition forbids storing the symbol frequencies, the total number of permutations, or from using recursion. (The best current algorithms that use $O(1)$-time use at least $O(n)$-variables.)

Non-increasing Prefix

The *non-increasing prefix* of a permutation $\mathbf{s} = s_1 s_2 \cdots s_n \in \Pi(\mathbb{E})$

$$\searrow(\mathbf{s}) = s_1 s_2 \cdots s_k$$

such that $s_i \geq s_{i+1}$ for i within $1 \leq i < k$, and either $k = n$ or $s_k < s_{k+1}$.
For example,

$$\searrow(443211) = 443211 \qquad \searrow(431421) = 431$$
$$\searrow(344211) = 3 \qquad \searrow(443121) = 4431$$

Non-increasing Prefix

The *non-increasing prefix* of a permutation $\mathbf{s} = s_1 s_2 \cdots s_n \in \Pi(\mathbb{E})$

$$\searrow(\mathbf{s}) = s_1 s_2 \cdots s_k$$

such that $s_i \geq s_{i+1}$ for i within $1 \leq i < k$, and either $k = n$ or $s_k < s_{k+1}$. For example,

$$\searrow(443211) = 443211 \qquad \searrow(431421) = 431$$
$$\searrow(344211) = 3 \qquad \searrow(443121) = 4431$$

Non-increasing Prefix

Lemma

Prefix-shifts often causes the non-increasing prefix to grow by one symbol or to be reset to a single symbol. More precisely, if $\sigma_j(\mathbf{s}) \neq \mathbf{s}$ *then*

$$|\searrow(\sigma_j(\mathbf{s}))| = \begin{cases} 1 & \text{if } s_j < s_1 \\ |\searrow(\mathbf{s})| + 1 & \text{if } s_j \geq s_1 \text{ and } s_{j-1} < s_{j+1} \end{cases}$$

The lemma avoids the problem case when $s_j \geq s_1$ and $s_{j-1} \geq s_{j+1}$. In this case, the non-increasing prefix can change dramatically. For example,

$$\searrow(43432132) = 43 \text{ and } \searrow(\sigma_5(43432132)) = \searrow(44332132) = 443321.$$

If this problem case is avoided then the length of the non-increasing prefix can be updated $O(1)$-time using $O(1)$-variables after a prefix-shift.

Non-increasing Prefix

Lemma

Prefix-shifts often causes the non-increasing prefix to grow by one symbol or to be reset to a single symbol. More precisely, if $\sigma_j(\mathbf{s}) \neq \mathbf{s}$ *then*

$$|\searrow(\sigma_j(\mathbf{s}))| = \begin{cases} 1 & \text{if } s_j < s_1 \\ |\searrow(\mathbf{s})| + 1 & \text{if } s_j \geq s_1 \text{ and } s_{j-1} < s_{j+1} \end{cases}$$

The lemma avoids the problem case when $s_j \geq s_1$ and $s_{j-1} \geq s_{j+1}$. In this case, the non-increasing prefix can change dramatically. For example,

$$\searrow(43432132) = 43 \text{ and } \searrow(\sigma_5(43432132)) = \searrow(44332132) = 443321.$$

If this problem case is avoided then the length of the non-increasing prefix can be updated $O(1)$-time using $O(1)$-variables after a prefix-shift.

Non-increasing Prefix

Lemma

Prefix-shifts often causes the non-increasing prefix to grow by one symbol or to be reset to a single symbol. More precisely, if $\sigma_j(\mathbf{s}) \neq \mathbf{s}$ *then*

$$|\searrow(\sigma_j(\mathbf{s}))| = \begin{cases} 1 & \text{if } s_j < s_1 \\ |\searrow(\mathbf{s})| + 1 & \text{if } s_j \geq s_1 \text{ and } s_{j-1} < s_{j+1} \end{cases}$$

The lemma avoids the problem case when $s_j \geq s_1$ and $s_{j-1} \geq s_{j+1}$. In this case, the non-increasing prefix can change dramatically. For example,

$$\searrow(43432132) = 43 \text{ and } \searrow(\sigma_5(43432132)) = \searrow(44332132) = 443321.$$

If this problem case is avoided then the length of the non-increasing prefix can be updated $O(1)$-time using $O(1)$-variables after a prefix-shift.

Cool Prefix-Shift

The *cool prefix-shift* is defined below for any $\mathbf{s} \in \Pi(\mathbb{E})$ where $k = |\succ(\mathbf{s})|$

$$\lhd(\mathbf{s}) = \begin{cases} \sigma_n(\mathbf{s}) & \text{if } k = n \\ \sigma_{k+1}(\mathbf{s}) & \text{if } k = n-1 \text{ or } s_k < s_{k+2} \\ \sigma_{k+2}(\mathbf{s}) & \text{otherwise } (k \leq n-2 \text{ and } s_k \geq s_{k+2}) \end{cases}$$

For example,

In the case of combinations,

$$\lhd(\mathbf{s}) = \sigma_i(\mathbf{s}) \text{ where } i = \min(k+2, n)$$

since $\sigma_{k+1}(1^*0^*011\cdots) = \sigma_{k+2}(1^*0^*011\cdots)$.

Cool Prefix-Shift

The *cool prefix-shift* is defined below for any $\mathbf{s} \in \Pi(\mathbb{E})$ where $k = |\succeq(\mathbf{s})|$

$$\lhd(\mathbf{s}) = \begin{cases} \sigma_n(\mathbf{s}) & \text{if } k = n \\ \sigma_{k+1}(\mathbf{s}) & \text{if } k = n-1 \text{ or } s_k < s_{k+2} \\ \sigma_{k+2}(\mathbf{s}) & \text{otherwise } (k \leq n-2 \text{ and } s_k \geq s_{k+2}) \end{cases}$$

For example,

$$\lhd(\underline{443211}) = 1443211.$$

In the case of combinations,

$$\lhd(\mathbf{s}) = \sigma_i(\mathbf{s}) \text{ where } i = \min(k+2, n)$$

since $\sigma_{k+1}(1^*0^*011\cdots) = \sigma_{k+2}(1^*0^*011\cdots)$.

Cool Prefix-Shift

The *cool prefix-shift* is defined below for any $\mathbf{s} \in \Pi(\mathbb{E})$ where $k = |\succ(\mathbf{s})|$

$$\lhd(\mathbf{s}) = \begin{cases} \sigma_n(\mathbf{s}) & \text{if } k = n \\ \sigma_{k+1}(\mathbf{s}) & \text{if } k = n-1 \text{ or } s_k < s_{k+2} \\ \sigma_{k+2}(\mathbf{s}) & \text{otherwise } (k \leq n-2 \text{ and } s_k \geq s_{k+2}) \end{cases}$$

For example,

$$\lhd(\underline{442113}) = 344211.$$

In the case of combinations,

$$\lhd(\mathbf{s}) = \sigma_i(\mathbf{s}) \text{ where } i = \min(k+2, n)$$

since $\sigma_{k+1}(1^*0^*011\cdots) = \sigma_{k+2}(1^*0^*011\cdots)$.

Cool Prefix-Shift

The *cool prefix-shift* is defined below for any $\mathbf{s} \in \Pi(\mathbb{E})$ where $k = |\succ(\mathbf{s})|$

$$\lhd(\mathbf{s}) = \begin{cases} \sigma_n(\mathbf{s}) & \text{if } k = n \\ \sigma_{k+1}(\mathbf{s}) & \text{if } k = n-1 \text{ or } s_k < s_{k+2} \\ \sigma_{k+2}(\mathbf{s}) & \text{otherwise } (k \leq n-2 \text{ and } s_k \geq s_{k+2}) \end{cases}$$

For example,

$$\lhd(\underline{421}431) = 442131.$$

In the case of combinations,

$$\lhd(\mathbf{s}) = \sigma_i(\mathbf{s}) \text{ where } i = \min(k+2, n)$$

since $\sigma_{k+1}(1^*0^*011\cdots) = \sigma_{k+2}(1^*0^*011\cdots)$.

Cool Prefix-Shift

The *cool prefix-shift* is defined below for any $\mathbf{s} \in \Pi(\mathbb{E})$ where $k = |\succ(\mathbf{s})|$

$$\lhd(\mathbf{s}) = \begin{cases} \sigma_n(\mathbf{s}) & \text{if } k = n \\ \sigma_{k+1}(\mathbf{s}) & \text{if } k = n-1 \text{ or } s_k < s_{k+2} \\ \sigma_{k+2}(\mathbf{s}) & \text{otherwise } (k \leq n-2 \text{ and } s_k \geq s_{k+2}) \end{cases}$$

For example,

$$\lhd(\underline{421}413) = 142143.$$

In the case of combinations,

$$\lhd(\mathbf{s}) = \sigma_i(\mathbf{s}) \text{ where } i = \min(k+2, n)$$

since $\sigma_{k+1}(1^*0^*011\cdots) = \sigma_{k+2}(1^*0^*011\cdots)$.

Cool Prefix-Shift

The *cool prefix-shift* is defined below for any $\mathbf{s} \in \Pi(\mathbb{E})$ where $k = |\succeq(\mathbf{s})|$

$$\lhd(\mathbf{s}) = \begin{cases} \sigma_n(\mathbf{s}) & \text{if } k = n \\ \sigma_{k+1}(\mathbf{s}) & \text{if } k = n-1 \text{ or } s_k < s_{k+2} \\ \sigma_{k+2}(\mathbf{s}) & \text{otherwise } (k \leq n-2 \text{ and } s_k \geq s_{k+2}) \end{cases}$$

For example,

In the case of combinations,

$$\lhd(\mathbf{s}) = \sigma_i(\mathbf{s}) \text{ where } i = \min(k+2, n)$$

since $\sigma_{k+1}(1^*0^*011\cdots) = \sigma_{k+2}(1^*0^*011\cdots)$.

Successive Iterations

When the multiset permutation is stored in a singly linked list then successive iterations can be applied using two comparisons and two pointers.

```
if inc.next ∧ min.val ≥ inc.next.val then
    temp = inc
else
    temp = min
end if
newhead = temp.next
temp.next = newhead.next
newhead.next = head
if newhead.val < head.val then
    min = newhead
end if
inc = min.next
head = newhead
visit(head)
```

Successive Iterations

When the multiset permutation is stored in a singly linked list then successive iterations can be applied using two comparisons and two pointers.

```
if inc.next ∧ min.val ≥ inc.next.val then
    temp = inc
else
    temp = min
end if
newhead = temp.next
temp.next = newhead.next
newhead.next = head
if newhead.val < head.val then
    min = newhead
end if
inc = min.next
head = newhead
visit(head)
```

Successive Iterations

When the multiset permutation is stored in a singly linked list then successive iterations can be applied using two comparisons and two pointers.

```
if inc.next ∧ min.val ≥ inc.next.val then
    temp = inc
else
    temp = min
end if
newhead = temp.next
temp.next = newhead.next
newhead.next = head
if newhead.val < head.val then
    min = newhead
end if
inc = min.next
head = newhead
visit(head)
```

Main Theorem

Theorem

Suppose $\mathbf{s}$ *is any permutation of the multiset* $\mathbb{E}$*. Then,*

$$\mathbf{s}, \lhd(\mathbf{s}), \lhd^2(\mathbf{s}), \lhd^3(\mathbf{s}), \ldots, \lhd^x(\mathbf{s})$$

contains every permutation of the multiset $\mathbb{E}$ *exactly once, where* $x = |\Pi(\mathbb{E})| - 1$ *and* $\lhd^i(\mathbf{s}) = \lhd\left(\lhd^{i-1}(\mathbf{s})\right)$.

Corollary

Multiset permutations can be ordered by prefix-shifts and can be ordered in $O(1)$*-time and* $O(1)$*-space.*

The theorem is illustrated and then its proof is sketched.

Main Theorem

Theorem

Suppose $\mathbf{s}$ *is any permutation of the multiset* $\mathbb{E}$*. Then,*

$$\mathbf{s}, \lhd(\mathbf{s}), \lhd^2(\mathbf{s}), \lhd^3(\mathbf{s}), \ldots, \lhd^x(\mathbf{s})$$

contains every permutation of the multiset $\mathbb{E}$ *exactly once, where* $x = |\Pi(\mathbb{E})| - 1$ *and* $\lhd^i(\mathbf{s}) = \lhd\left(\lhd^{i-1}(\mathbf{s})\right)$.

Corollary

Multiset permutations can be ordered by prefix-shifts and can be ordered in $O(1)$*-time and* $O(1)$*-space.*

The theorem is illustrated and then its proof is sketched.

Main Theorem

Theorem

Suppose $\mathbf{s}$ *is any permutation of the multiset* $\mathbb{E}$*. Then,*

$$\mathbf{s}, \lhd(\mathbf{s}), \lhd^2(\mathbf{s}), \lhd^3(\mathbf{s}), \ldots, \lhd^x(\mathbf{s})$$

contains every permutation of the multiset $\mathbb{E}$ *exactly once, where* $x = |\Pi(\mathbb{E})| - 1$ *and* $\lhd^i(\mathbf{s}) = \lhd\left(\lhd^{i-1}(\mathbf{s})\right)$.

Corollary

Multiset permutations can be ordered by prefix-shifts and can be ordered in $O(1)$*-time and* $O(1)$*-space.*

The theorem is illustrated and then its proof is sketched.

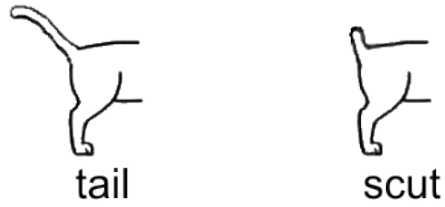
Cool-lex Order

The *cool-lex order* of $\Pi(\mathbb{E})$ for $\{1, 1, 2, 3, 4, 4\}$ is now animated. [Click to begin the animation and press up/down arrow keys to change the speed.]

Notice that the rightmost symbols change infrequently.

Scuts

scut [n]: Stubby erect tail, as that of a hare, rabbit, or antelope.



A *scut* is a shortest string over $\mathbb{E}$ that is not a suffix of the non-increasing permutation of $\mathbb{E}$. For example,

$$\text{scut}(244311) = 311$$

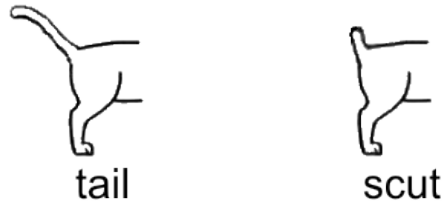
since 311 is the shortest suffix that is not also a suffix of 443211. Every permutation of $\mathbb{E} = \{1, 1, 2, 3, 4, 4\}$ has one of the following scuts

$$4, 41, 411, 4211, 3, 31, 311, 2, 21.$$

Scuts are determined by their first symbol and their length.

Scuts

scut [n]: Stubby erect tail, as that of a hare, rabbit, or antelope.



A *scut* is a shortest string over $\mathbb{E}$ that is not a suffix of the non-increasing permutation of $\mathbb{E}$. For example,

$$\text{scut}(244311) = 311$$

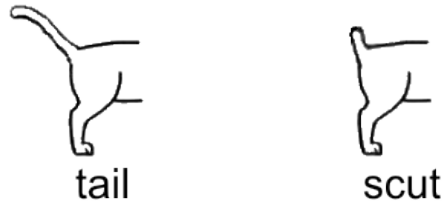
since 311 is the shortest suffix that is not also a suffix of 443211. Every permutation of $\mathbb{E} = \{1, 1, 2, 3, 4, 4\}$ has one of the following scuts

$$4, 41, 411, 4211, 3, 31, 311, 2, 21.$$

Scuts are determined by their first symbol and their length.

Scuts

scut [n]: Stubby erect tail, as that of a hare, rabbit, or antelope.



A *scut* is a shortest string over $\mathbb{E}$ that is not a suffix of the non-increasing permutation of $\mathbb{E}$. For example,

$$\text{scut}(244311) = 311$$

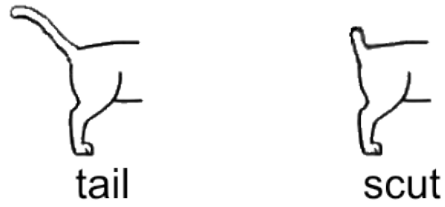
since 311 is the shortest suffix that is not also a suffix of 443211. Every permutation of $\mathbb{E} = \{1, 1, 2, 3, 4, 4\}$ has one of the following scuts

$$4, 41, 411, 4211, 3, 31, 311, 2, 21.$$

Scuts are determined by their first symbol and their length.

Scuts

scut [n]: Stubby erect tail, as that of a hare, rabbit, or antelope.



A *scut* is a shortest string over $\mathbb{E}$ that is not a suffix of the non-increasing permutation of $\mathbb{E}$. For example,

$$\text{scut}(244311) = 311$$

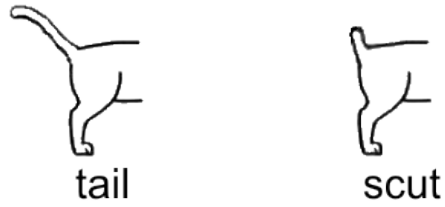
since 311 is the shortest suffix that is not also a suffix of 443211. Every permutation of $\mathbb{E} = \{1, 1, 2, 3, 4, 4\}$ has one of the following scuts

$$4, 41, 411, 4211, 3, 31, 311, 2, 21.$$

Scuts are determined by their first symbol and their length.

Scuts

scut [n]: Stubby erect tail, as that of a hare, rabbit, or antelope.



A *scut* is a shortest string over $\mathbb{E}$ that is not a suffix of the non-increasing permutation of $\mathbb{E}$. For example,

$$\text{scut}(244311) = 311$$

since 311 is the shortest suffix that is not also a suffix of 443211. Every permutation of $\mathbb{E} = \{1, 1, 2, 3, 4, 4\}$ has one of the following scuts

$$4, 41, 411, 4211, 3, 31, 311, 2, 21.$$

Scuts are determined by their first symbol and their length.

Co-lex vs Cool-lex

Co-lex recursively orders its strings by increasing rightmost distinct symbol. In the example above the rightmost distinct symbols appear in the following order

$$1, 2, 3, 4.$$

Co-lex vs Cool-lex

Alternatively, co-lex order recursively orders its strings by decreasing scut length then by increasing first scut symbol. In the example above the scuts appear in the following order

$$4211, 311, 411, 21, 31, 41, 2, 3, 4$$

and the non-increasing string is first.

Co-lex vs Cool-lex

Cool-lex recursively orders its strings by increasing first scut symbol then by decreasing scut length. In the example above the scuts appear in the following order

$$21, 2, 311, 31, 3, 4211, 411, 41, 4$$

and the non-increasing string is last.

Additional Applications of Cool-lex Order

Publications

- Combinations
  - appears in *The Art of Computer Programming*
  - $O(1)$-time and $O(1)$-space in array, linked list, or computer word
  - Hewlett-Packard Media Lab
- Balanced parentheses and binary trees
  - $O(1)$-time and $O(1)$-space in array, linked list, or binary tree
  - first simultaneous $O(1)$-time algorithm for balanced parentheses and binary trees
  - "Well, I am very happy to be able to include one of the most beautiful combinatorial algorithm I am aware of!" - Joerg Arndt

Additional Applications of Cool-lex Order

Thesis

- Multiset necklaces and Lyndon words
  - first Gray codes of any kind
- Degree sequences for ordered trees
  - first $O(1)$-time and $O(1)$-variables algorithm
- Linear-extensions of B-posets
  - first $O(1)$-time and $O(1)$-variables algorithm
- Shorthand universal cycles for multiset permutations
  - allows only $\sigma_{n-1}(\mathbf{s})$ and $\sigma_n(\mathbf{s})$
  - first proof of existence and construction

Open Problems

- Does cool-lex minimize the total prefix-shifting distance?
- Natural extension to languages that do not have fixed-content?
- Applications: genetics, manufacturing, ...?

Thank you!

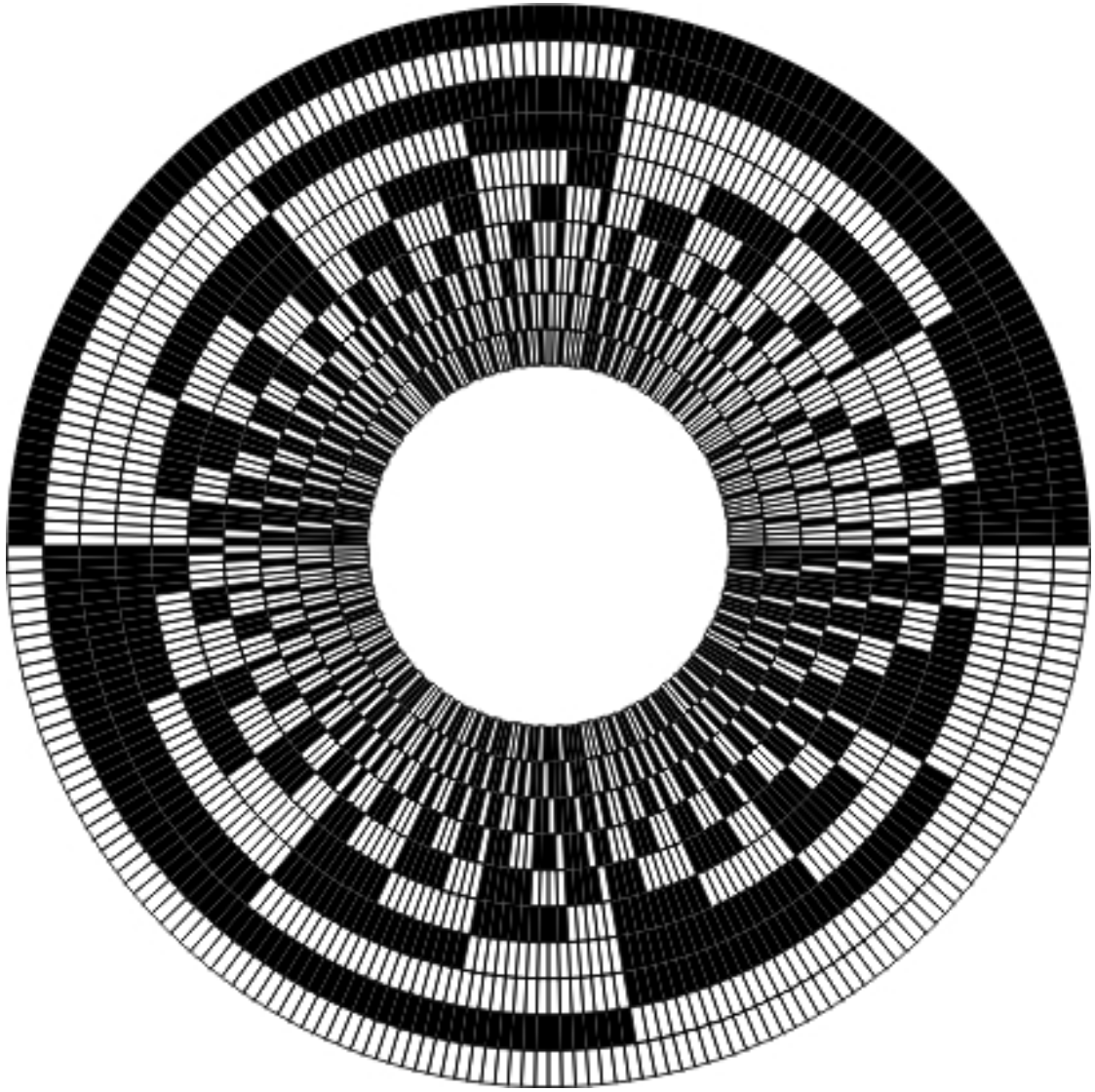


Figure: Co-lex Order for permutations of $\{0, 0, 0, 0, 0, 1, 1, 1, 1, 1\}$.

Thank you!

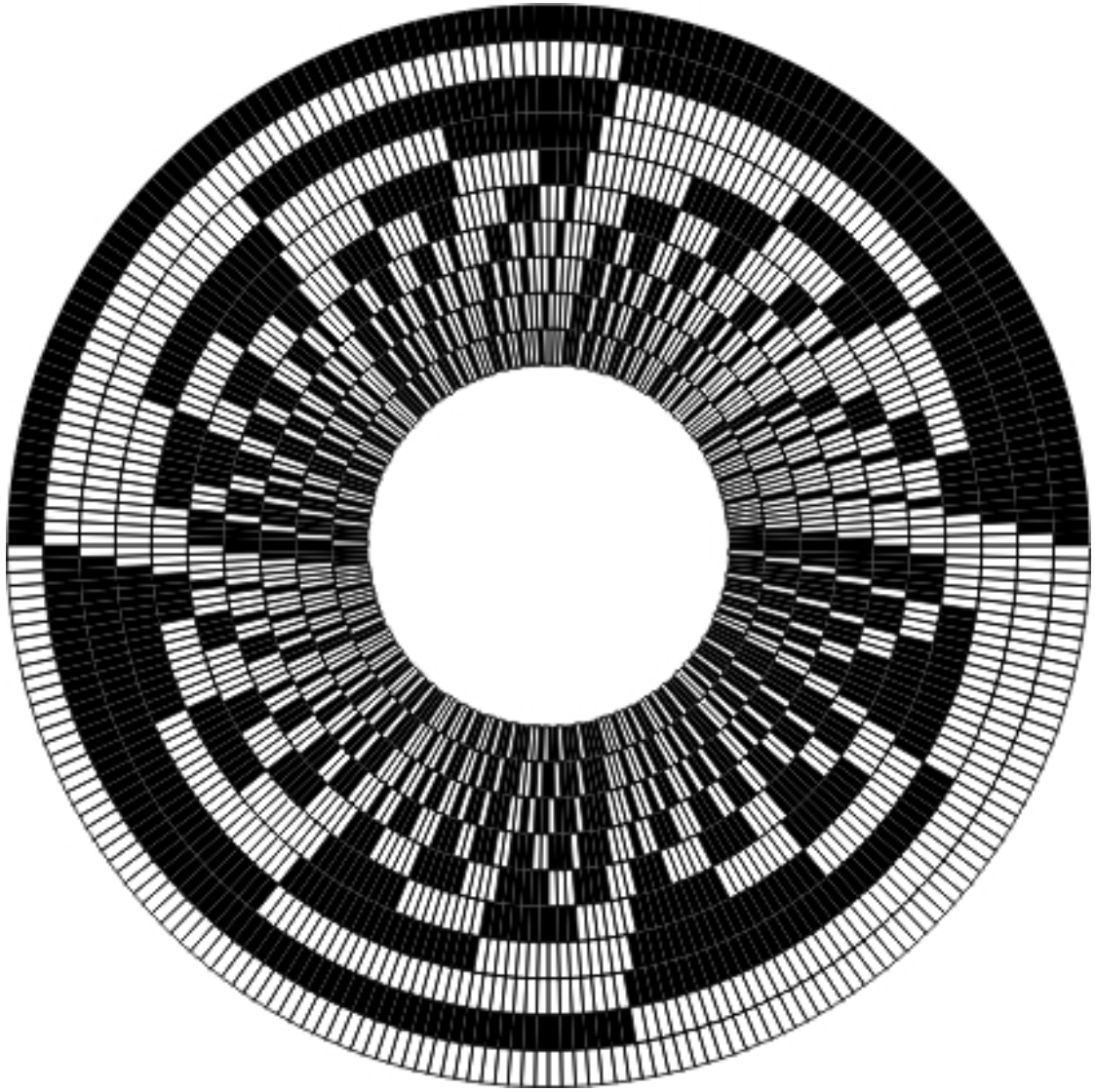


Figure: Cool-lex Order for permutations of $\{0, 0, 0, 0, 0, 1, 1, 1, 1, 1\}$.