UNIVERSITY OF LIVERPOOL

LEVERHULME TRUST

The Leverhulme Research Centre for Functional Materials Design

# Ranking Bracelets in Polynomial time

Duncan Adamson
University of Liverpool, Department of Computer Science

June 8, 2021

diamond

Hartree Centre

Imperial College London

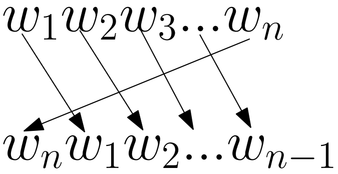
ICNS

King Abdullah University of Science and Technology

University of Southampton

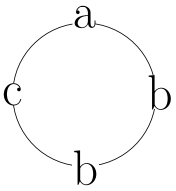
# What are Necklaces?

- A **necklace** is an equivalence class of words under the **cyclic shift** operation.
- The canonical representative of a necklace is the **lexicographically smallest word** in the equivalence class.



$w_1w_2w_3\ldots w_n$

$w_nw_1w_2\ldots w_{n-1}$



**abbc**
bbca
bcab
cabb

# What are Bracelets?

- A **bracelet** is an equivalence class of words under the **cyclic shift** and **reflection** operations.
- The canonical representation of a bracelet is the **lexicographically smallest word** in the equivalence class.

| unreflected | reflected |
|---|---|
| **abbc** | cbba |
| bbca | *acbb* |
| bcab | bacb |
| cabb | bbac |

# What is ranking?

- The ranking problem asks, given an object $o$ in a strictly ordered set $S$, how many members of S are smaller than $o$.
- For the set of bracelets of a given length $n$ over an alphabet of size $k$ ($\mathrm{B}(n, k)$), the ordering is defined over the canonical representations.

1. aaaa
2. aaab
3. aaac
4. aabb
5. aabc
6. aacc
7. abab
8. abac
9. abbb
**10. abbc**
11. abcb
12. abcc
13. acac
14. acbc
15. acccc
16. bbbb
17. bbbc
18. bbcc
19. bcbc
20. bccc
21. cccc

- Note that there are approximately $O(k^n)$ bracelets, making explicit generation of the set unfeasible for large values of $n$ and $k$.

# What is Unranking?

- The unranking problem can be thought of as the inverse of the ranking problem.
- The unranking problem asks, for a strictly ordered set $S$ what is the element at the $i^{th}$ position.
- For bracelets (and other classes of cyclic words) the exponential size of these sets makes the naive approach impractical.

# Ranking Cyclic Words

- The problem of ranking classes of cyclic words originates from the problem of ranking de Bruijn Sequences [2].
- The first class of cyclic words to be ranked was **Lyndon words** (aperiodic necklaces).
- This was generalised to ranking necklaces [3, 4] in quadratic time.
- More recently, an algorithm to rank the class of **Fixed density** necklaces in cubic time has been presented [1].

| Class | Solved by | Best Run time |
|---|---|---|
| Lyndon words | Kociumaka et. al. [2] | $O(n^2)$ ([4]) |
| Necklaces | Kopparty et. al. [3] | $O(n^2)$ ([4]) |
| Fixed Density Necklaces | Hartman and Sawada | $O(n^3)$ ([1]) |

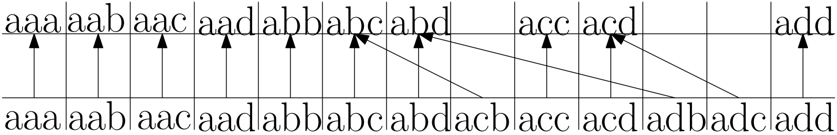
Table 1: Unranking algorithms for all sets have been implemented with an additional factor of $O(n \log k)$ [1, 4]

# Our Results

- We provide an $O(n^4 k^2)$ time algorithm to rank a word $w$ among the set of all bracelets of length $n$ over the alphabet of size $k$.
- Further, we provide an $O(n^5 \cdot k^2 \cdot \log(k))$ time algorithm to unrank bracelets.

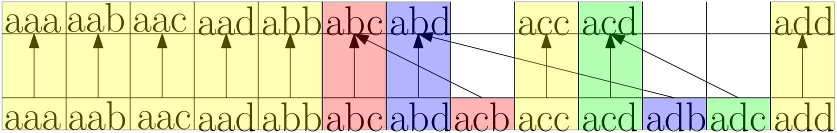
# Bracelets and Necklaces

- Observe that **every** bracelets corresponds to either one or two necklace classes.
- A bracelet is **Palindromic** if it only corresponds to a single necklace class.
- A bracelet is **Apalindromic** if it corresponds to two necklace classes.
- An apalindromic bracelet $b = n_1 \cup n_2$ **encloses** a word $w$ if $n_1 < w < n_2$.



# Bracelets and Necklaces

- Observe that **every** bracelets corresponds to either one or two necklace classes.
- A bracelet is **Palindromic** if it only corresponds to a single necklace class.
- A bracelet is **Apalindromic** if it corresponds to two necklace classes.
- An apalindromic bracelet $b = n_1 \cup n_2$ **encloses** a word $w$ if $n_1 < w < n_2$.

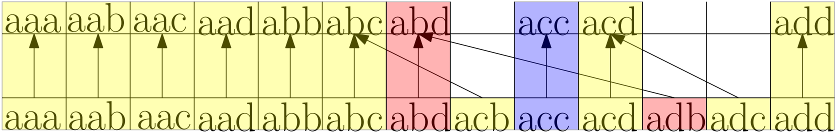


aaa aab aac aad abb abc abd acc acd add

aaa aab aac aad abb abc abd acb acc acd adb adc add

# Bracelets and Necklaces

- Observe that **every** bracelets corresponds to either one or two necklace classes.
- A bracelet is **Palindromic** if it only corresponds to a single necklace class.
- A bracelet is **Apalindromic** if it corresponds to two necklace classes.
- An apalindromic bracelet $b = n_1 \cup n_2$ **encloses** a word $w$ if $n_1 < w < n_2$.



# High Level Idea

- The number of bracelets smaller than $w$ can be split into three categories:
  - The number of palindromic bracelets smaller than $w$ ($RP(w)$).
  - The number of apalindromic bracelets smaller than $w$ that do not enclose $w$ ($RA(w)$).
  - The number of bracelets enclosing $w$ ($RE(w)$).

# High Level Idea

- The number of bracelets smaller than $w$ can be split into three categories:
  - The number of palindromic bracelets smaller than $w$ ($RP(w)$).
  - The number of apalindromic bracelets smaller than $w$ that do not enclose $w$ ($RA(w)$).
  - The number of bracelets enclosing $w$ ($RE(w)$).
- **Idea:** count each of these sets separately and add them together.

# Counting the number of apalindromic bracelets

- Rather than computing the number of apalindromic bracelets that do not enclose $w$ directly, we can use the number of necklaces smaller than $w$ ($RN(w)$), along with the other two sets ($RP(w)$ and $RE(w)$) to count the size of $RA(w)$.
- Note that the number of necklaces smaller than $w$ is equals the sum of:
    - 2 times the number of apalindromic bracelets not enclosing $w$.
    - The number of palindromic bracelets smaller than $w$.
    - The number of enclosing bracelets smaller than $w$.
- $RN(w) = 2RA(w) + RP(w) + RE(w)$.
- This summation can be rearranged to give: $RA(w) = \frac{1}{2}\left(RN(w) - (RP(w) + RE(w))\right)$.

# Counting $RP(w)$

- In order to count the number of palindromic bracelets smaller than $w$, it is useful to have a characterisation of palindromic bracelets.
- Recall that a word $w$ is palindromic if and only if $w = w^R$.
    - $aba = aba^R$.
- Any bracelet containing a palindromic word must be palindromic.
- However not all palindromic bracelets contain palindromic words
    - The bracelet $\{abab, baba\}$ is palindromic, however $abab \neq abab^R$ and $baba \neq baba^R$.

# Odd length palindromic bracelets

Lemma

*A bracelet b of odd length n is palindromic if and only if it contains* exactly *one unique word of the form $\phi x \phi^R$ for some word $\phi$ and symbol x.*

- Note that any word $w = \phi x \phi^r = w_1 w_2 \dots w_{(n-1)/2} x w_{(n-1)/2} \dots w_2 w_1$ is palindromic.

# Odd length palindromic bracelets

Lemma

*A bracelet $b$ of odd length $n$ is palindromic if and only if it contains* exactly *one unique word of the form $\phi x \phi^R$ for some word $\phi$ and symbol $x$.*

- Note that any word $w = \phi x \phi^r = w_1 w_2 \dots w_{(n-1)/2} x w_{(n-1)/2} \dots w_2 w_1$ is palindromic.
- To show that $b$ must contain at least 1 palindromic word, let $|b|$ be the number of words in $b$.
    - Note that $|b|$ must be odd.

# Odd length palindromic bracelets

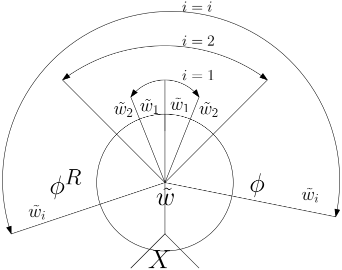
## Lemma

*A bracelet $b$ of odd length $n$ is palindromic if and only if it contains* exactly *one unique word of the form $\phi x \phi^R$ for some word $\phi$ and symbol $x$.*

- Note that any word $w = \phi x \phi^r = w_1 w_2 \dots w_{(n-1)/2} x w_{(n-1)/2} \dots w_2 w_1$ is palindromic.
- To show that $b$ must contain at least 1 palindromic word, let $|b|$ be the number of words in $b$.
    - Note that $|b|$ must be odd.
- Let $a$ be the set of apalindromic words in $b$.
- Note that if $v \in a$ then $v^R \in a$ and $v \neq v^R$.
- As $|a|$ must be even, $|b| - |a| \geq 1$.

# Counting the number of odd length palindromic bracelets

- **Main Idea:** The main idea is to use the uniqueness of the palindromic representations to count the number of bracelets smaller than $w$.
- This is done by counting the number of unique prefixes of the form $\phi x \phi^R$ iteratively.



# Counting the number of odd length palindromic bracelets

- Explicitly generating the whole prefix tree is impractical.
- Instead, we look at sets $\mathrm{PO}(w, i, j, s)$ containing every word $u$ where:
    - $|u| = i$
    - $j$ is the longest suffix of $u$ that is a prefix of $w$.
    - $s$ is the lexicographically greatest subword of $w$ that is less than or equal to $u$.

| | |
|---|---|
| $\mathrm{PO}(abbcabbc, 3, 0, cab)$ | $ccc, ccb, cbc, cac$ |
| $\mathrm{PO}(abbcabbc, 3, 1, cab)$ | $cca, cba$ |
| $\mathrm{PO}(abbcabbc, 3, 2, cab)$ | $cab$ |

# Counting the number of odd length palindromic bracelets

- Explicitly generating the whole prefix tree is impractical.
- Instead, we look at sets $\mathrm{PO}(w, i, j, s)$ containing every word $u$ where:
    - $|u| = i$
    - **j** is the longest suffix of $u$ that is a prefix of $w$.
    - $s$ is the lexicographically greatest subword of $w$ that is less than or equal to $u$.

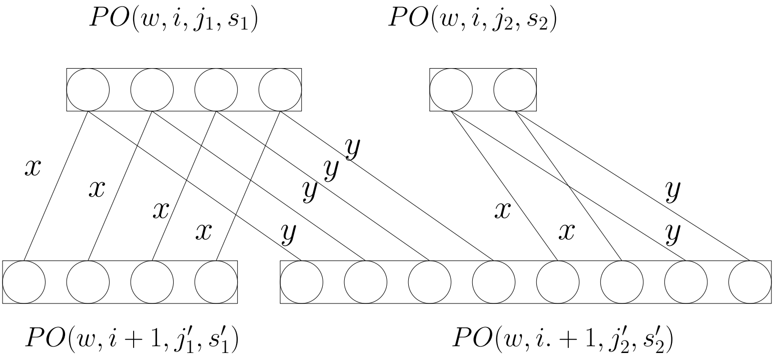
| | |
|---|---|
| $\mathrm{PO}(abbcabbc, 3, 0, cab)$ | $ccc, ccb, cbc, cac$ |
| $\mathrm{PO}(abbcabbc, 3, 1, cab)$ | $cc\mathbf{a}, cb\mathbf{a}$ |
| $\mathrm{PO}(abbcabbc, 3, 2, cab)$ | $c\mathbf{ab}$ |

# Counting the number of odd length palindromic bracelets

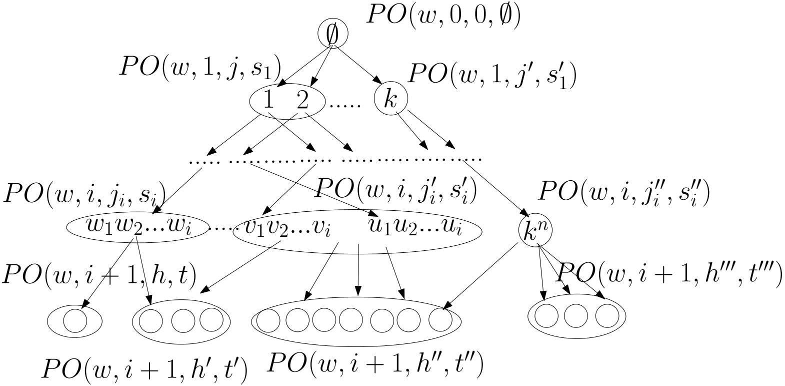
- Explicitly generating the whole prefix tree is impractical.
- Instead, we look at sets $\text{PO}(w, i, j, s)$ containing every word $u$ where:
    - $|u| = i$
    - j is the longest suffix of $u$ that is a prefix of $w$.
    - $s$ is the lexicographically greatest subword of $w$ that is less than or equal to $u$.

| | |
|---|---|
| $\text{PO}(abbcabbc, 3, 0, \textbf{cab})$ | $ccc, ccb, cbc, cac$ |
| $\text{PO}(abbcabbc, 3, 1, \textbf{cab})$ | $cca, cba$ |
| $\text{PO}(abbcabbc, 3, 2, \textbf{cab})$ | $cab$ |

# Computing the size of PO($w, i, j, s$)



# Counting the number of odd length palindromic bracelets



# Even Length palindromic bracelets

Lemma

*A bracelet $b$ of even length $n$ is palindromic if and only if it contains some word $w$ where either (1) $w = x\phi y\phi^R$ for symbols $x, y$ and word $\phi$ of length $\frac{n}{2} - 1$, or (2) $w = \phi\phi^R$ where $\phi$ is a word of length $\frac{n}{2}$.*

- The number of such words is counted in the same manner as in the odd case.
- The main difference is that these words are not unique, adding additional complexity to computing the number of palindromic bracelets of even length.

# Computing $RE(w)$

- Similarly to the Palindromic case, the first step in ranking the number of enclosing is defining a combinatorial structure to capture them.
- The canonical representation of every bracelet enclosing $w$ can be written in the form

$$w_1 w_2 \dots w_i x \phi$$

  where:
  - $w_1 w_2 \dots w_i$ is the prefix of $w$ of length $i$.
  - $x < w_{i+1}$
  - $\phi$ is a word such that every suffix of $x w_i w_{i-1} \dots w_1 \phi^R x w_i w_{i-1} \dots w_1$ starting at some symbol in $\phi$ is strictly greater than $w$.

# Counting Enclosing Bracelets

- Using the combinatorial characterisation, the number of enclosing words is counted by determining the number of possible values of $\phi$ for every pair $i \in 1 \dots n$ and $x \in \Sigma$.

$$\sum_{i=0}^{n-1} \sum_{x \in \Sigma} |\phi(w, i, x)|$$

- This is done in a recursive manner based on counting the number of words with no suffix greater than $w$ in increasing length.
- As each bracelet may only have a single such representation, by counting the number of representatives the exact number of enclosing bracelets can be counted.

# Unranking

- The inverse of the ranking process is unranking.
- This problem asks what the $i^{th}$ bracelet in the set $B(n, k)$ is.
- This problem was solved by Sawada and Williams [4] for Necklaces and Lyndon words in $O(n^3 \log k)$ time.
- We solve the unranking problem for bracelets in $O(n^5 \cdot k^2 \cdot \log(k))$ time.

# The Unranking Process

- To determine the $i^{th}$ bracelet, a binary search is performed using the ranking algorithm as a sub process.
- Note that the number of bracelets where the canonical representative has a prefix $\psi$ is given by $RB(\psi k^{n-|\psi|}) - RB(\psi^{n/|\psi|})$.
- Further, the rank of every bracelet with the prefix $\psi$ is between $RB(\psi^{n/|\psi|})$ and $RB(\psi k^{n-|\psi|})$.

# The Unranking Process

- The first symbol of $b$ must be the symbol $x$ for which $RB(x^n) \leq i \leq RB(xk^{n-1})$.
- This is found by performing a binary search over the alphabet.
- Similarly, the $2^{nd}$ symbol is the symbol $y$ where $RB((xy)^{n/2}) \leq i \leq RB((xy)k^{n-2})$.

# The Unranking Process

- The first symbol of $b$ must be the symbol $x$ for which $RB(x^n) \leq i \leq RB(xk^{n-1})$.
- This is found by performing a binary search over the alphabet.
- Similarly, the $2^{nd}$ symbol is the symbol $y$ where $RB((xy)^{n/2}) \leq i \leq RB((xy)k^{n-2})$.
- Let $\psi$ be the prefix of $b$ of length $j - 1$.
- The $j^{th}$ symbol of $b$ is the symbol $x$ such that $RB((\psi x)^{n/j}) \leq i \leq RB(\psi xk^{n-j})$.
- This can also be found by a binary search.

# Conclusion

- We have presented a $O(n^4 \cdot k^2)$ time algorithm to rank bracelets.
- This is complimented by a $O(n^5 \cdot k^2 \cdot \log k)$ time unranking algorithm.
- In order to rank bracelets in polynomial time, we have also developed algorithms to rank both Palindromic and Enclosing bracelets in $O(n^3 \cdot k \cdot \log^2(n))$ and $O(n^4 \cdot k^2)$ time respectively.
- These algorithms may be used to rank the aperiodic counterparts to Bracelets, Palindromic Bracelets, and Enclosing bracelets at an additional factor of $O(n)$ operations.

# Open Problems

- Is there a $O(n^3)$ or faster algorithm for ranking Bracelets (either with a fixed alphabet or in general)?
- Can fixed density bracelets be ranked in polynomial time, and if so does there exist an $O(n^5)$ time algorithm to do so?

P. Hartman and J. Sawada.
Ranking and unranking fixed-density necklaces and Lyndon words.
*Theoretical Computer Science*, 791:36–47, 2019.

T. Kociumaka, J. Radoszewski, and W. Rytter.
Computing k-th Lyndon word and decoding lexicographically minimal de Bruijn sequence.
In *Symposium on Combinatorial Pattern Matching*, pages 202–211. Springer, 2014.

S. Kopparty, M. Kumar, and M. Saks.
Efficient indexing of necklaces and irreducible polynomials over finite fields.
*Theory of Computing*, 12(1):1–27, 2016.

J. Sawada and A. Williams.
Practical algorithms to rank necklaces, Lyndon words, and de Bruijn sequences.

*Journal of Discrete Algorithms*, 43:95–110, 2017.