Palindrome Recognition In The Streaming Model

Petra Berenbrink, Funda Ergün, Frederik Mallmann-Trenn, and Erfan Sadeqi Azer

January 15, 2014

Palindrome

A palindrome is a string which reads forwards the same as backwards.
For example:

1. kayak
2. radar
3. racecar
4. level
5. Hannah

Applications

Hairpins

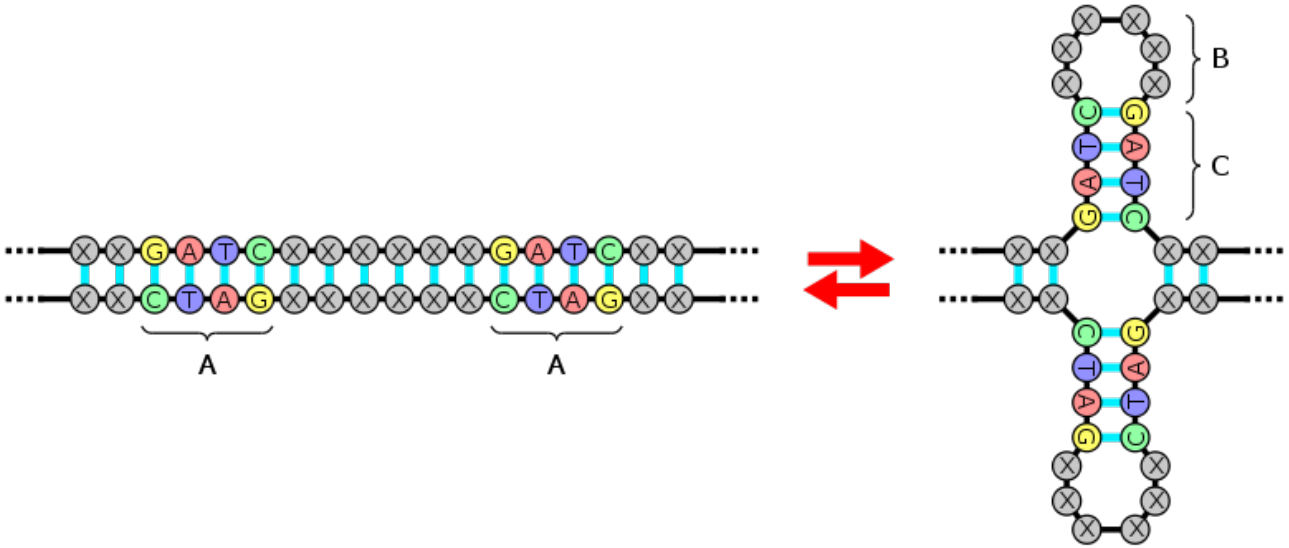
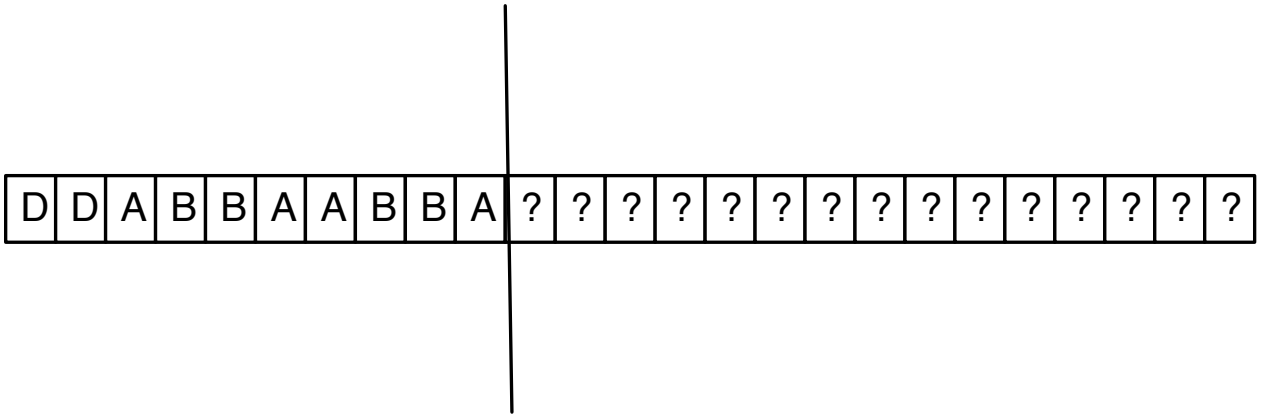


Figure : Palindrome of DNA structure A: Palindrome, B: Loop, C: Stem

The streaming model

The input is a stream S of n symbols.
The goal is to find an algorithm which uses only sublinear space to solve the task (in our case finds the longest palindrome).
The required time is less important.

Example of an input



What to store?

DDABBAABBADD

DDABBAABBAABBADD

DDABBAABBAABBADD

DDABBAABBAAAABBAABBADD

Results

$P[m]$ denotes the palindrome of maximal length centered at index m of S It's length is $\ell(m)$.
Example: *ABC PALINDROME EMORDNILAP ABC*
$P[n/2] = P[13] = \underbrace{\textit{PALINDROME}}_{\ell(m)}\ \textit{EMORDNILAP}$

Theorem (*ApproxSqrt*)

For any $\varepsilon \in [1/\sqrt{n}, 1]$ *Algorithm ApproxSqrt* (S, ε) *reports for every palindrome* $P[m]$ *in* S *its midpoint* m *as well as an estimate* $\tilde{\ell}(m)$ *(of* $\ell(m)$*) such that w.h.p.*

$$\ell(m) - \varepsilon\sqrt{n} < \tilde{\ell}(m) \leq \ell(m).$$

The algorithm makes one pass over S*, uses* $O(\frac{n}{\varepsilon})$ *time, and* $O(\frac{\sqrt{n}}{\varepsilon})$ *space.*

Results

Theorem (*Exact*)

Algorithm Exact reports w.h.p. ℓ_{max} *and* m *for all palindromes* $P[m]$ *with a length of* ℓ_{max}*. The algorithm makes two passes over* S*, uses* $O(n)$ *time, and* $O(\sqrt{n})$ *space.*

Theorem (*ApproxLog*)

For any ε *in* $(0, 1]$*, Algorithm ApproxLog reports w.h.p. an arbitrary palindrome* $P[m]$ *of length at least* $\ell_{max}/(1+\varepsilon)$*. The algorithm makes one pass over* S*, uses* $O(\frac{n \log(n)}{\varepsilon \log(1+\varepsilon)})$ *time, and* $O(\frac{\log(n)}{\varepsilon \log(1+\varepsilon)})$ *space.*

Fingerprints (1/2)

$S[i]$ is the symbol at index i of the input.
For a string S' we define the forward fingerprint $\phi^F_{r,p}$ (similar to [1]) and its reverse $\phi^R_{r,p}$ as follows.

$$\phi^F_{r,p}(S') = \left(\sum_{i=1}^{|S'|} S'[i] \cdot r^i \right) \mod p$$

$$\phi^R_{r,p}(S') = \left(\sum_{i=1}^{|S'|} S'[i] \cdot r^{l-i+1} \right) \mod p,$$

where p is an arbitrary prime number in $[n^4, n^5]$ and r is randomly chosen from $\{1, \ldots, p\}$.

[1] Breslauer, Galil 2011

Fingerprints (2/2)

$S = 314159$

$\phi^F_{r,p}(31415) = 3r + 1r^2 + 4r^3 + 1r^4 + 5r^5 \mod p$

1. Add a new symbol:
   $\phi^F_{r,p}(314159) = 3r + 1r^2 + 4r^3 + 1r^4 + 5r^5 + 9r^6 \equiv \phi^F_{r,p}(31415) + 9r^6 (mod\ p)$
2. Remove last symbol: $\phi^F_{r,p}(31415) = \phi^F_{r,p}(314159) - 9r^6 \mod p$
3. Remove first symbol: $\phi^F_{r,p}(14159) = \frac{1}{r}\phi^F_{r,p}(314159) - 3 \mod p$

Properties

Lemma (See [2])

For two arbitrary strings S_1 and S_2 with $S_1 \neq S_2$ the probability that $\phi^F(S_1) = \phi^F(S_2)$ is smaller than $1/n^c$ for an aribitrary $c \in \mathbb{N}$.

Lemma (See [2])

Let $S = s_1 s_2 \dots s_n$. Consider two substrings $S_1 = s_1 s_2 \dots s_k$ and $S_2 = s_{k+1} s_{k+2} \dots s_n$.

1. *Given $\phi^F(S_1)$, $\phi^F(S_2)$, $|S_1|$ and $|S_2|$ we can derive $\phi^F(S)$.*
2. *Given $\phi^F(S_1)$, $\phi^F(S)$, $|S_1|$ and $|S|$ we can derive $\phi^F(S_2)$.*
3. *Given $\phi^F(S)$, $\phi^F(S_2)$, $|S|$ and $|S_2|$ we can derive $\phi^F(S_1)$.*

[2] Breslauer, Galil 2011

Algorithm ApproxSqrt

Theorem (*ApproxSqrt*)

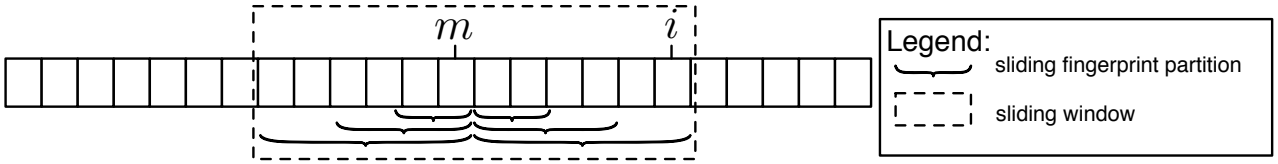
For any $\varepsilon \in [1/\sqrt{n}, 1]$ *Algorithm ApproxSqrt* (S, ε) *reports for every palindrome* $P[m]$ *in* S *its midpoint* m *as well as an estimate* $\tilde{\ell}(m)$ *(of* $\ell(m)$*) such that w.h.p.*

$$\ell(m) - \varepsilon\sqrt{n} < \tilde{\ell}(m) \leq \ell(m).$$

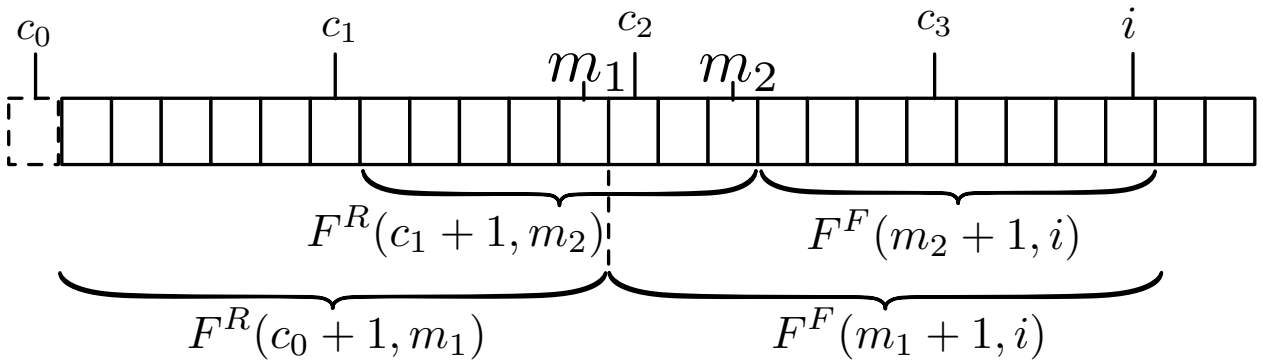
The algorithm makes one pass over S*, uses* $O(\frac{n}{\varepsilon})$ *time, and* $O(\frac{\sqrt{n}}{\varepsilon})$ *space.*

Algorithm ApproxSqrt

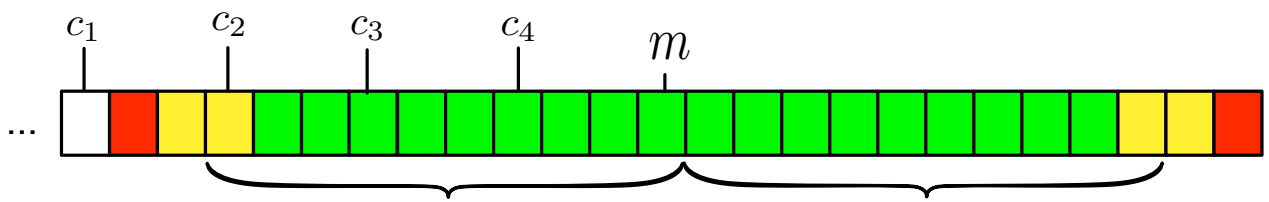
Short palindromes (lenght $\leq \sqrt{n}$).



Long palindromes (lenght $> \sqrt{n}$).



Long palindromes (0/4)



- Create checkpoints at distances which are multiples of $\varepsilon\sqrt{n}$

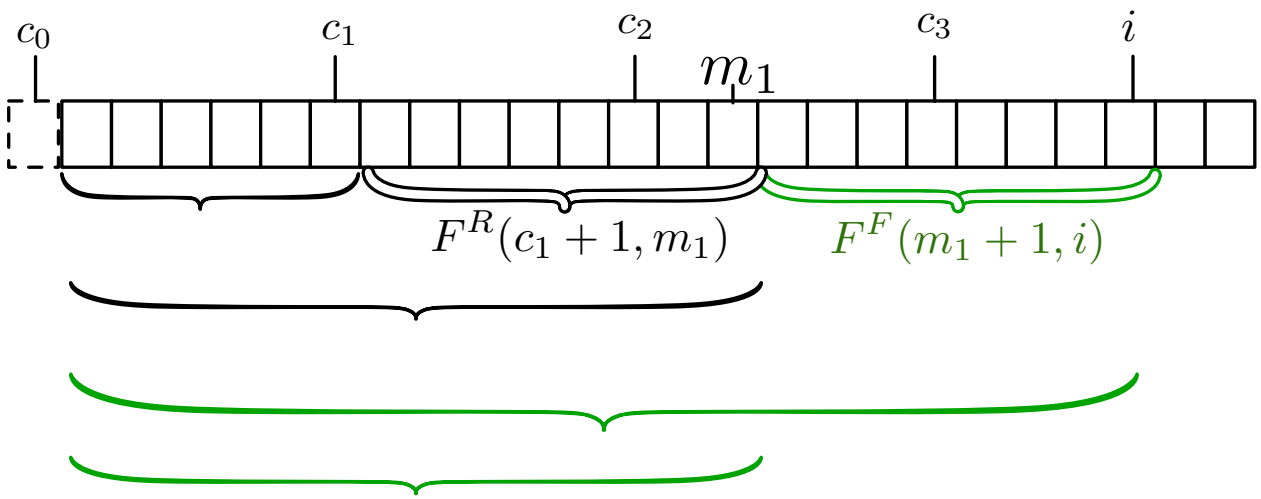
Long palindromes (1/4)

Idea for long palindromes. Let i be the the current index of the input.

- We always maintain the forward and backward fingerprint $\phi^F(S[1,i])$ and $\phi^R(S[1,i])$
- Every checkpoint c stores a backward fingerprint $\phi^R(S[1,c])$
- Every midpoint m stores a forward and a backward fingerprint $\phi^F(S[1,m])$ and $\phi^R(S[1,m])$
- This is possible when $i = m$ ($i = c$ respectively).

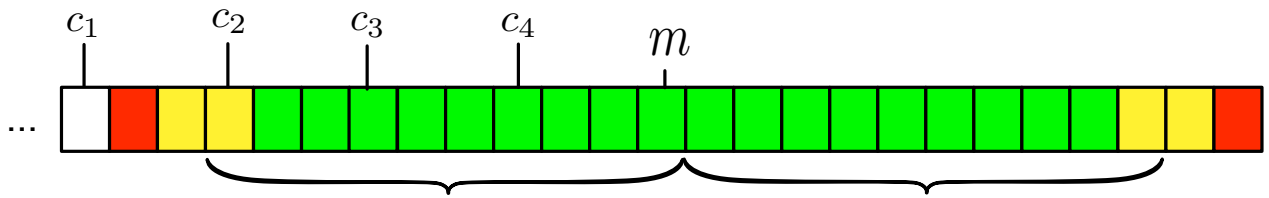
Long palindromes (2/4)

We can now do the following



This is possible whenever $m_1 - c = i - m_1$

Long palindromes (3/4)

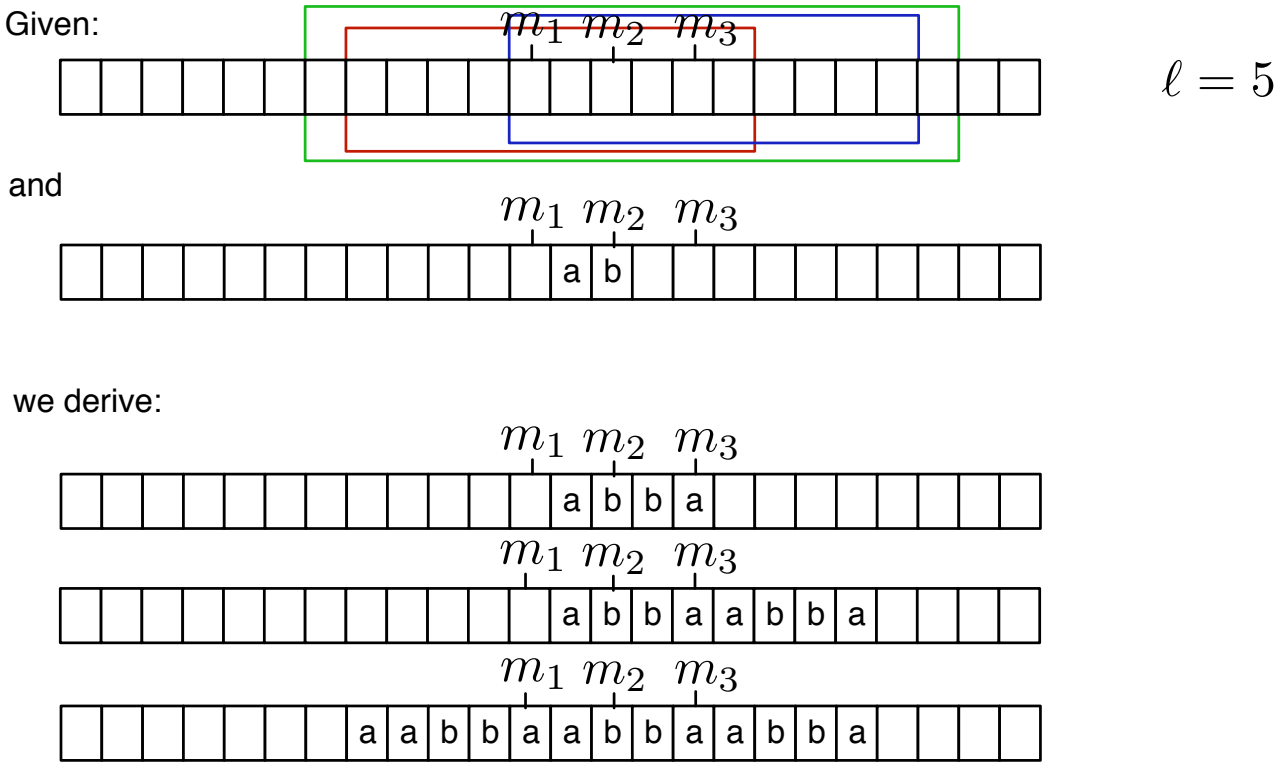


- $P[m]$ matched with: c_4, c_3, and c_2
- $P[m]$ mismatched with: c_1

Long palindromes (4/4)

- Storing every palindrome of a length of $\geq \sqrt{n}$ can still result in linear space. For this to happen the palindromes overlap eachother.
- ABAABAABAABAABAABAABAABAABAABAABAABAABA
- There is a midpoint of a palindome between every ABA
- We will take adventage of the overlapping

Pattern of palindromes



Pattern of palindromes

$$\overset{m_1}{} \quad \overset{m_2}{} \quad \overset{m_3}{} \quad \overset{m_4}{} \quad \overset{m_5}{}$$
$$w^R\, w\;\; w^R w\;\; w^R \,\Big|\, w \,\Big|\, w^R \Big|\, w \,\Big|\, w^R \Big|\, w\;\; w^R\, w\;\; w^R w$$

Length of a run

Definition (ℓ^*-Run)

Let ℓ^* be an arbitrary integer and $h \geq 3$. Let $m_1, m_2, m_3, \ldots, m_h$ be consecutive midpoints of ℓ^*-palindromes in S. $m_1, \ldots, m_h$ form an ℓ^*-*run* if $m_{j+1} - m_j \leq \ell^*/2$ for all $j \in \{1, \ldots, h-1\}$.

Lemma

At iteration i, let $m_1, m_2, m_3, ..., m_h$ be midpoints of a maximal ℓ^*-run in $S[1, i]$ for an arbitrary natural number ℓ^*. For any midpoint m_j, we have:

$$\ell(m_j, i) = \begin{cases} \ell(m_1, i) + (j-1) \cdot (m_2 - m_1) & j < \frac{h+1}{2} \\ \ell(m_h, i) + (h-j) \cdot (m_2 - m_1) & j > \frac{h+1}{2} \end{cases}$$

Analysis

1. short palindromes: We have everything in the slidingwindow
2. long palindromes:
   - Either they are far apart and we don't have too many.
   - If they are close together, then they form a run an we can compress (without loosing information) them

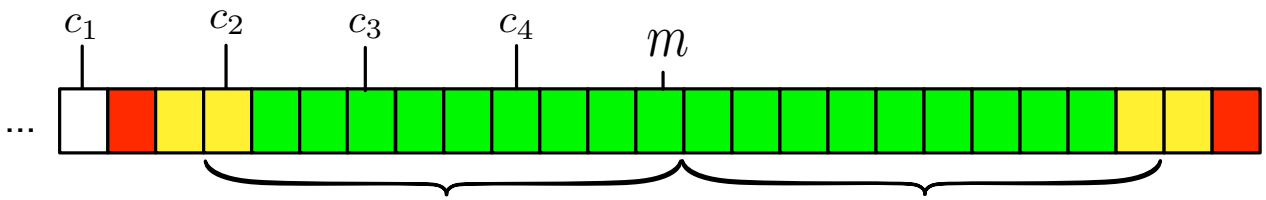
Algorithm Exact

Theorem

Algorithm Exact reports w.h.p. ℓ_{max} and m for all palindromes $P[m]$ with a length of ℓ_{max}. The algorithm makes two passes over S, uses $O(n)$ time, and $O(\sqrt{n})$ space.

The **idea** for the second run:

- For every run keep only the element(s) in the middle.
- Only keep the palindromes with the highest length estimate.
- Store the entire interval in which the palindrome starts



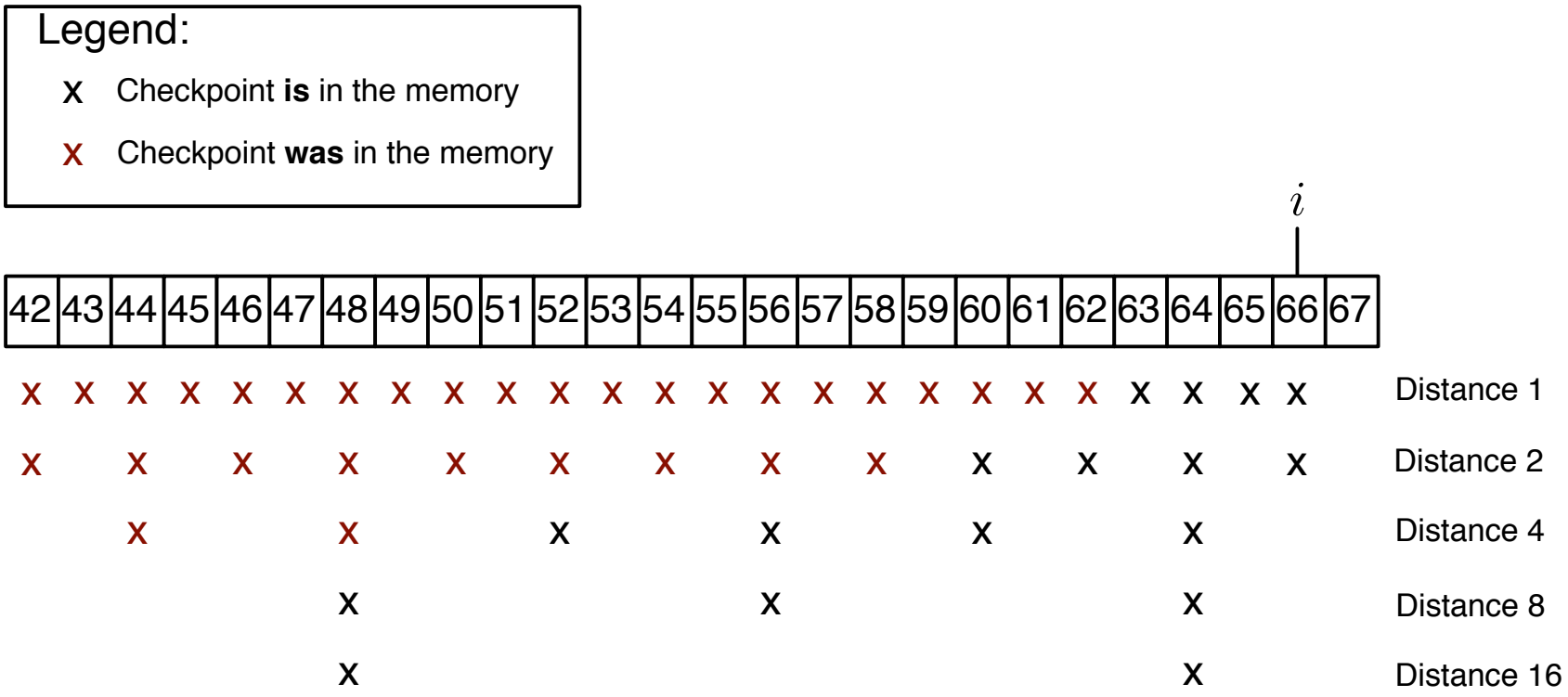
Algorithm ApproxLog

Theorem (*ApproxLog*)

For any ε in $(0, 1]$, Algorithm ApproxLog reports w.h.p. an arbitrary palindrome $P[m]$ of length at least $\ell_{max}/(1+\varepsilon)$. The algorithm makes one pass over S, uses $O(\frac{n \log(n)}{\varepsilon \log(1+\varepsilon)})$ time, and $O(\frac{\log(n)}{\varepsilon \log(1+\varepsilon)})$ space.

- This time we only care about **one** (a constant number) of the longest palindrome(s)
- This means that we might not find the **longest** palindrome.
- The error is now multiplicative as opposed to additive.
- The space is only logarithmic as opposed to $\sqrt{n}$.

Checkpointing



Future work

- It would be interesting to allow errors.
- It is probably extendable to a constant number of errors.
- What about a non-constant number of errors?
- Surprising that finding exact palindromes seems to require less space.

Retteb si flah dnoces eht tub, but the second half is better

Thank you

Question

Is there fingerprint allowing errors and satisfying:

Lemma (See [3])

For two arbitrary strings s and s' with $s \neq s'$ the probability that $\phi^F(s) = \phi^F(s')$ is smaller than $1/n^c$ for an aribitrary $c \in \mathbb{N}$.

Lemma (See [3])

Let $S = s_1 s_2 \dots s_n$. Consider two substrings $S_1 = s_1 s_2 \dots s_k$ and $S_2 = s_{k+1} s_{k+2} \dots s_n$.

1. *Given $\phi^F(S_1)$, $\phi^F(S_2)$, $|S_1|$ and $|S_2|$ we can derive $\phi^F(S)$.*
2. *Given $\phi^F(S_1)$, $\phi^F(S)$, $|S_1|$ and $|S|$ we can derive $\phi^F(S_2)$.*
3. *Given $\phi^F(S)$, $\phi^F(S_2)$, $|S|$ and $|S_2|$ we can derive $\phi^F(S_1)$.*

[3] Breslauer, Galil 2011