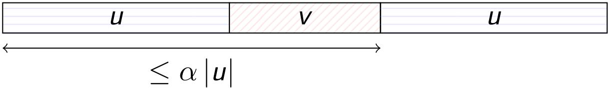
efficiently finding all maximal α-gapped repeats

Paweł Gawrychowski Tomohiro I Shunsuke Inenaga
Dominik Köppl Florin Manea

TU Dortmund

28th TCS mini-workshop 2015



accepted at STACS 2016

gapped repeats (gaprep)

- string w = TTCTACTAGAGACTAGCGA
- substring u = ACTA
- substring v = GAG

TTCTACTAGAGACTAGCGA

gapped repeats (gaprep)

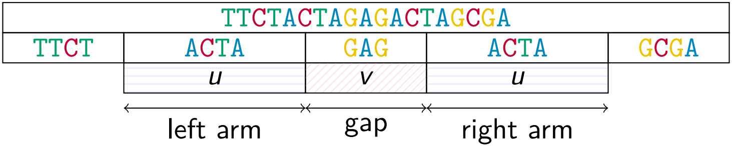
- string w = TTCTACTAGAGACTAGCGA
- substring u = ACTA
- substring v = GAG

TTCTACTAGAGACTAGCGA				
TTCT	ACTA	GAG	ACTA	GCGA

gapped repeats (gaprep)

- string w = TTCTACTAGAGACTAGCGA
- substring u = ACTA
- substring v = GAG

gaprep substring uvu of w ACTA GAG ACTA

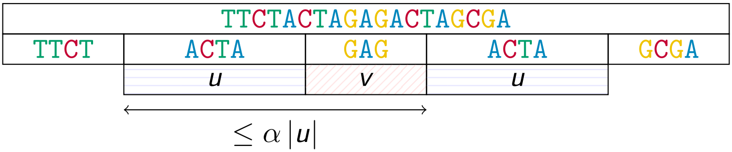


gapped repeats (gaprep)

- string w = TTCTACTAGAGACTAGCGA
- substring u = ACTA
- substring v = GAG
- $\alpha \in \mathbb{R}, \alpha \geq 1$

gaprep substring uvu of w ACTA GAG ACTA

α-gaprep $|uv| \leq \alpha |u|$ $|$ ACTA GAG $| = 7$, $|$ ACTA $| = 4$



gapped repeats (gaprep)

- string w = TTCTACTAGAGACTAGCGA
- substring u = ACTA
- substring v = GAG
- $\alpha \in \mathbb{R}, \alpha \geq 1$

gaprep substring uvu of w — ACTA GAG ACTA

α-gaprep $|uv| \leq \alpha |u|$ — $|$ ACTA GAG $| = 7$, $|$ ACTA $| = 4$

max α-gaprep cannot extend u to a longer α-gaprep

TTCTACTAGAGACTAGCGA				
TTCT	ACTA	GAG	ACTA	GCGA
	u	v	u	

$\leq \alpha |u|$

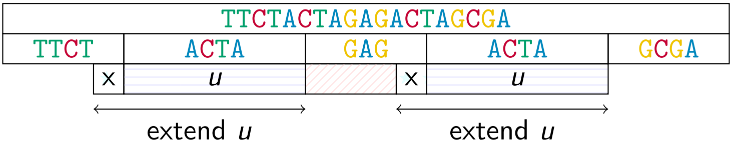
gapped repeats (gaprep)

- string w = TTCTACTAGAGACTAGCGA
- substring u = ACTA
- substring v = GAG
- $\alpha \in \mathbb{R}, \alpha \geq 1$

gaprep substring uvu of w — ACTA GAG ACTA

α-gaprep $|uv| \leq \alpha\,|u|$ — $|$ ACTA GAG $| = 7$, $|$ ACTA $| = 4$

max α-gaprep cannot extend u to a longer α-gaprep



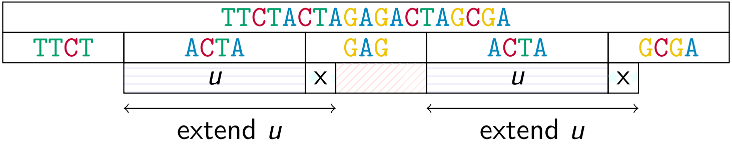
gapped repeats (gaprep)

- string w = TTCTACTAGAGACTAGCGA
- substring u = ACTA
- substring v = GAG
- $\alpha \in \mathbb{R}, \alpha \geq 1$

gaprep substring uvu of w ACTA GAG ACTA

α-gaprep $|uv| \leq \alpha |u|$ | ACTA GAG | = 7, | ACTA | = 4

max α-gaprep cannot extend u to a longer α-gaprep



why?

generalization:

- $\alpha = 1$: squares ACGT ACGT
- $\alpha = 2$: long armed repeats ACGT CATG ACGT

Problem

highest number of # max α-gaprep?

Crochemore et al.'06: $\text{occ} = \Omega(\alpha n)$ n: string length

$\alpha = 1$: Bannai et. al'15: $\text{occ} \leq n$

Question

$$\text{occ} = \Theta(\mathbf{?})$$

where?

in genomics:

base pairs

C–G

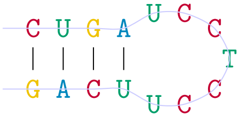
A–T

A–U (RNA)

- inverted repeats in DNA sequences

  . . . AAATCGG . . . CCGATTT. . .

- hairpin structures (RNA stem-loop)



Observation

DNA contains copies with small gaps.

repetitions in $w =$ AACAACACACAC

substring u has

- length $|u|$

AACAACACACAC	substring u	length	per	exp
____ACACACAC	AC AC AC AC	8		

repetitions in w = AACAACACACAC

substring u has

- length $|u|$
- minimal period $\text{per}(u)$

AACAACACACAC	substring u	length	per	exp
____ACACACAC	AC AC AC AC	8	2	

repetitions in w = AACAACACACAC

substring u has

- length $|u|$
- minimal period $\mathrm{per}(u)$
- exponent $\exp(u) = |u| / \mathrm{per}(u)$

AACAACACACAC	substring u	length	per	exp
____ACACACAC	AC AC AC AC	8	2	4

repetitions in w = AACAACACACAC

rule: $\exp(u) \geq 1$
def:

AACAACACACAC	substring	length	per	exp

repetitions in w = AACAACACACAC

rule: $\exp(u) \geq 1$
def:

primitive $\exp(u) = 1$

AACAACACACAC	substring	length	per	exp
___AACAC____	AACAC	5	5	1

repetitions in $w =$ AACAACACACAC

rule: $\exp(u) \geq 1$

def:

primitive $\exp(u) = 1$

repeat $\exp(u) \geq 2$

AACAACACACAC	substring	length	per	exp
___AACAC____	AACAC	5	5	1
AA__________	A A	2	1	2
AACAACA_____	AAC AAC A	7	3	7/3

repetitions in w = AACAACACACAC

rule: $\exp(u) \geq 1$
def:

primitive $\exp(u) = 1$

repeat $\exp(u) \geq 2$

What about $1 < \exp(u) < 2$?

AACAACACACAC	substring	length	per	exp
___AACAC____	AACAC	5	5	1
AA__________	A A	2	1	2
AACAACA_____	AAC AAC A	7	3	7/3
___AACA_____	AAC A	4	3	4/3

max δ-subrepetition $(0 \leq \delta < 1)$

- $1 + \delta \leq \exp(u) < 2$

Kolpakov et al'13

$$\#\text{max } \delta\text{-subrep} \leq \#\text{max } 1/\delta\text{-gaprep}$$

AACAACACACAC	substring	length	per	exp
___AACA_____	AAC A	4	3	4/3

AAC A is $\frac{1}{3}$-subrepetition

history

catching all max α-gaprep

authors	when	time	particularity
Brodal et al.	'99	$\mathcal{O}(n \lg n + \text{occ})$	$\alpha \equiv$ const.
Kolpakov et al.	'00	$\mathcal{O}(n \lg \alpha + \text{occ})$	$\|v\| = \alpha$
Kolpakov et al.	'14	$\mathcal{O}(\alpha^2 n + \text{occ})$	$\text{occ} = \mathcal{O}(\alpha^2 n)$
Tanimura et al.	Sep'15	$\mathcal{O}(\alpha n + \text{occ})$	$\Sigma \equiv$ const
Crochemore et al.	Sep'15	$\mathcal{O}(\alpha n + \text{occ})$	$\Sigma \equiv$ const
we	Sep'15	$\mathcal{O}(\alpha n)$	Σ integer

integer alphabets

w string

- alphabet Σ
- length n

Σ is

constant	$\lvert\Sigma\rvert \equiv \text{const}$
integer	$\lvert\Sigma\rvert = n^{\mathcal{O}(1)}$
effective	$\Sigma = \{w[i] : 1 \leq i \leq n\}$, $\lvert\Sigma\rvert \leq n$.

Lemma

*transform any Σ to **effective** alphabet by sorting*

recently

I, Inenaga, Köppl: Aug'15

$\#$ maximal α-gapped repeats $= \Theta(\alpha n)$

Gawrychowski, Manea: FCT'15

compute

$$\text{argmax}\,\{|u| : uvu \text{ is } \alpha - \text{gapped repeat}\}$$

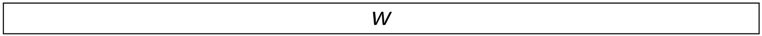
in $\mathcal{O}(\alpha n + \text{occ})$ time, integer alphabet

joint work

all max α-gaprep in $\mathcal{O}(\alpha n)$ time, integer alphabet.

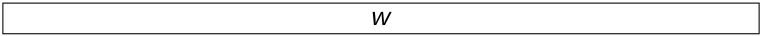
lcp data structure (lcp-DS)

- given string w
- construction in linear time
- answers in $\mathcal{O}(1)$ time



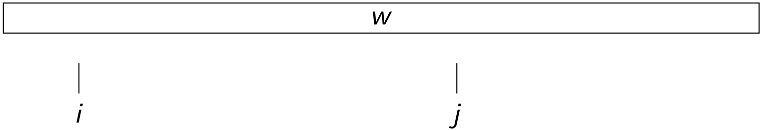
lcp data structure (lcp-DS)

- given string w
- construction in linear time
- answers in $\mathcal{O}(1)$ time
  - ***longest common prefix***
    $\text{lcp}(i, j) := \max \{\ell : w[i, i + \ell - 1] = w[j, j + \ell - 1]\}$



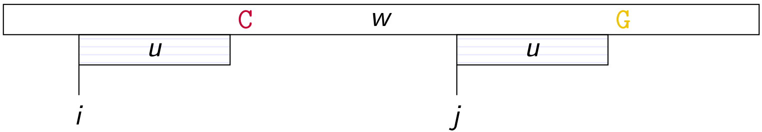
lcp data structure (lcp-DS)

- given string w
- construction in linear time
- answers in $\mathcal{O}(1)$ time
  - ***longest common prefix***
    $\text{lcp}(i, j) := \max\{\ell : w[i, i+\ell-1] = w[j, j+\ell-1]\}$



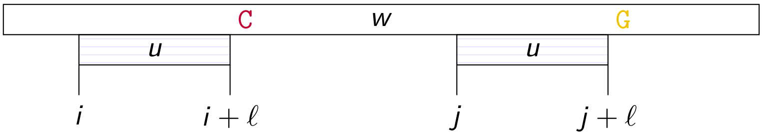
lcp data structure (lcp-DS)

- given string w
- construction in linear time
- answers in $\mathcal{O}(1)$ time
  - ***longest common prefix***
    $\mathrm{lcp}(i, j) := \max\{\ell : w[i, i+\ell-1] = w[j, j+\ell-1]\}$



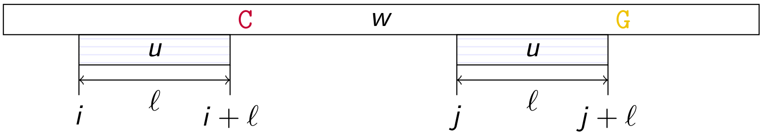
lcp data structure (lcp-DS)

- given string w
- construction in linear time
- answers in $\mathcal{O}(1)$ time
  - ***longest common prefix***
    $\mathrm{lcp}(i,j) := \max\{\ell : w[i, i+\ell-1] = w[j, j+\ell-1]\}$



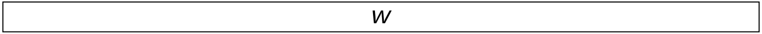
lcp data structure (lcp-DS)

- given string w
- construction in linear time
- answers in $\mathcal{O}(1)$ time
  - ***longest common prefix***
    $\mathrm{lcp}(i,j) := \max\{\ell : w[i, i+\ell-1] = w[j, j+\ell-1]\}$



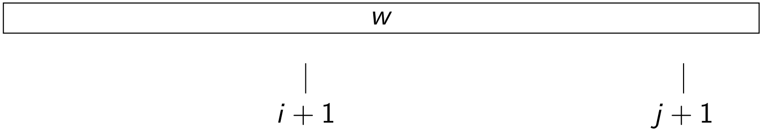
lcp data structure (lcp-DS)

- given string w
- construction in linear time
- answers in $\mathcal{O}(1)$ time
  - ***longest common prefix***
    $\operatorname{lcp}(i, j) := \max\{\ell : w[i, i+\ell-1] = w[j, j+\ell-1]\}$
  - ***longest common suffix***
    $\operatorname{lcs}(i, j) := \max\{\ell : w[i-\ell+1, i] = w[j-\ell+1, j]\}$



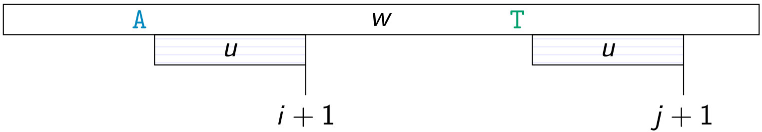
lcp data structure (lcp-DS)

- given string w
- construction in linear time
- answers in $\mathcal{O}(1)$ time
  - ***longest common prefix***
    $\mathrm{lcp}(i, j) := \max\{\ell : w[i, i+\ell-1] = w[j, j+\ell-1]\}$
  - ***longest common suffix***
    $\mathrm{lcs}(i, j) := \max\{\ell : w[i-\ell+1, i] = w[j-\ell+1, j]\}$



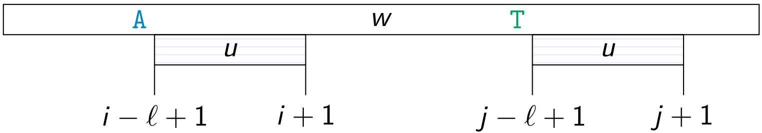
lcp data structure (lcp-DS)

- given string w
- construction in linear time
- answers in $\mathcal{O}(1)$ time
  - ***longest common prefix***
    $\mathrm{lcp}(i,j) := \max\{\ell : w[i, i+\ell-1] = w[j, j+\ell-1]\}$
  - ***longest common suffix***
    $\mathrm{lcs}(i,j) := \max\{\ell : w[i-\ell+1, i] = w[j-\ell+1, j]\}$



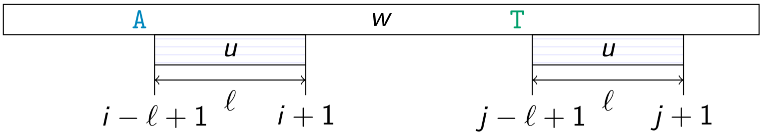
lcp data structure (lcp-DS)

- given string w
- construction in linear time
- answers in $\mathcal{O}(1)$ time
  - ***longest common prefix***
    $\mathrm{lcp}(i,j) := \max\{\ell : w[i, i+\ell-1] = w[j, j+\ell-1]\}$
  - ***longest common suffix***
    $\mathrm{lcs}(i,j) := \max\{\ell : w[i-\ell+1, i] = w[j-\ell+1, j]\}$



lcp data structure (lcp-DS)

- given string w
- construction in linear time
- answers in $\mathcal{O}(1)$ time
  - ***longest common prefix***
    $\mathrm{lcp}(i,j) := \max\{\ell : w[i, i+\ell-1] = w[j, j+\ell-1]\}$
  - ***longest common suffix***
    $\mathrm{lcs}(i,j) := \max\{\ell : w[i-\ell+1, i] = w[j-\ell+1, j]\}$



basic factors

w string, $|w| = m$.

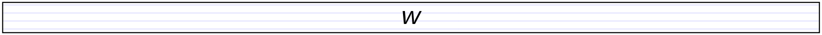
basic factor := 2^q-gram partition, $q \geq 1$

$$\sum_{q=1}^{q=\lfloor \lg m \rfloor} (m/2^q) \leq m \text{ many}$$

<table>
<tr><td colspan="8" align="center">w</td></tr>
<tr><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td><td>2</td></tr>
<tr><td colspan="2" align="center">4</td><td colspan="2" align="center">4</td><td colspan="2" align="center">4</td><td colspan="2" align="center">4</td></tr>
<tr><td colspan="4" align="center">8</td><td colspan="4" align="center">8</td></tr>
<tr><td colspan="8" align="center">16</td></tr>
</table>

getOcc(y, w): return where y found in w

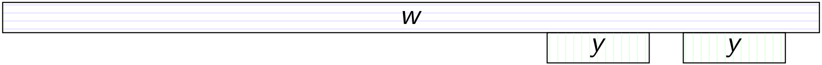
y occurs in w



getOcc(y, w): return where y found in w

y occurs in w

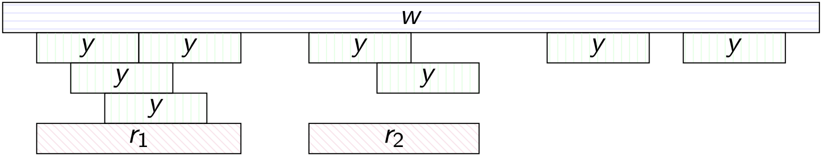
- lonely



getOcc(y, w): return where y found in w

y occurs in w

- lonely
- in a repetition

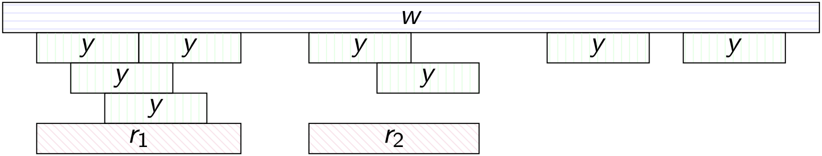


getOcc(y, w): return where y found in w

y occurs in w

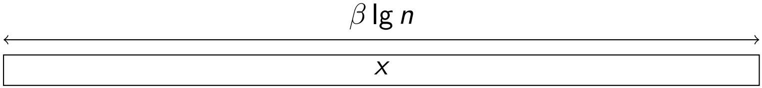
- lonely
- in a repetition

neglected here



- $2 \leq \gamma < \beta$ const.
- x string, $|x| = \beta \lg n$

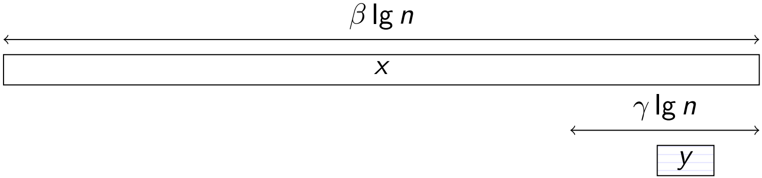
- $2 \leq \gamma < \beta$ const.
- x string, $|x| = \beta \lg n$



getOcc-index on x — P. Gawrychowski, F. Manea: FCT'15

- construction time $\mathcal{O}(\beta \lg n)$
- answers getOcc(y, z) in $\mathcal{O}(d)$ time for

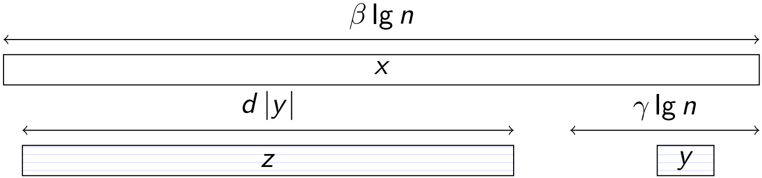
- $2 \leq \gamma < \beta$ const.
- x string, $|x| = \beta \lg n$



getOcc-index on x — P. Gawrychowski, F. Manea: FCT'15

- construction time $\mathcal{O}(\beta \lg n)$
- answers getOcc(y, z) in $\mathcal{O}(d)$ time for
  - y basic factor of x, y contained in last $\gamma \lg n$ characters of x

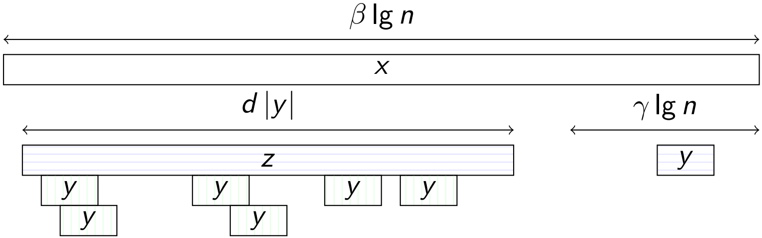
- $2 \leq \gamma < \beta$ const.
- x string, $|x| = \beta \lg n$



getOcc-index on x P. Gawrychowski, F. Manea: FCT'15

- construction time $\mathcal{O}(\beta \lg n)$
- answers getOcc(y, z) in $\mathcal{O}(d)$ time for
  - y basic factor of x, y contained in last $\gamma \lg n$ characters of x
  - substring z of x
  - $|z| = d\,|y|$, $d \equiv$ const. later: $d = \mathcal{O}(\alpha)$

- $2 \leq \gamma < \beta$ const.
- x string, $|x| = \beta \lg n$



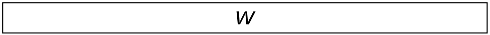
getOcc-index on x — P. Gawrychowski, F. Manea: FCT'15

- construction time $\mathcal{O}(\beta \lg n)$
- answers getOcc(y, z) in $\mathcal{O}(d)$ time for
  - y basic factor of x, y contained in last $\gamma \lg n$ characters of x
  - substring z of x
  - $|z| = d\,|y|$, $d \equiv$ const. later: $d = \mathcal{O}(\alpha)$

search $u_\lambda v u_\rho$

task: search left arm u_λ

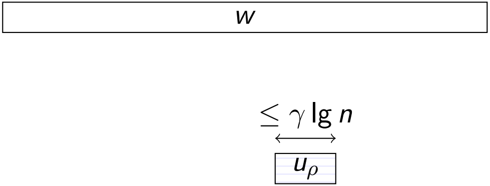
λ : left, ρ : right



search $u_\lambda v u_\rho$

task: search left arm u_λ λ : left, ρ : right

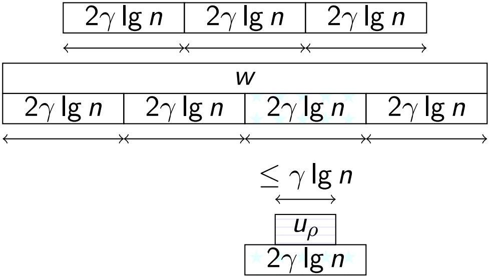
- assume: $|u_\rho| \leq \gamma \lg n$



search $u_\lambda v u_\rho$

task: search left arm u_λ — λ : left, ρ : right

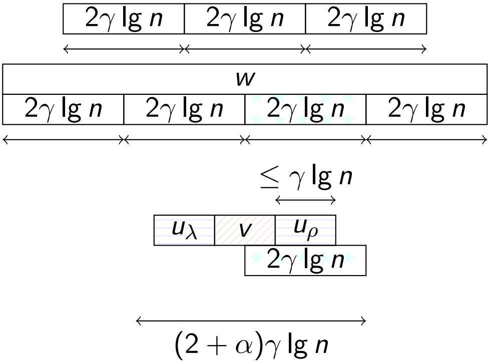
- assume: $|u_\rho| \leq \gamma \lg n$
- blocks of length w in $2\gamma \lg n$



search $u_\lambda v u_\rho$

task: search left arm u_λ λ : left, ρ : right

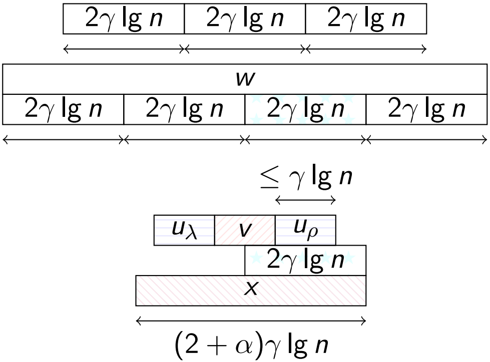
- assume: $|u_\rho| \leq \gamma \lg n$
- blocks of length w in $2\gamma \lg n$
- by def: $|u_\lambda v| \leq \alpha\, |u_\lambda| \Leftrightarrow |v| \leq \alpha(\gamma - 1) \lg n$



search $u_\lambda v u_\rho$

task: search left arm u_λ λ : left, ρ : right

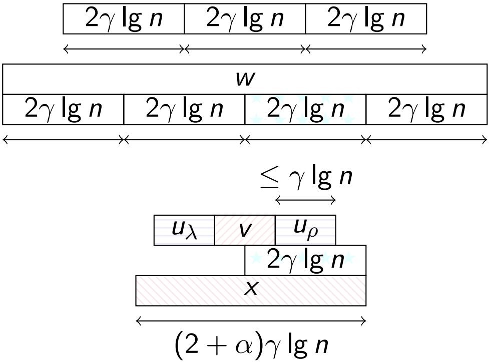
- assume: $|u_\rho| \leq \gamma \lg n$
- blocks of length w in $2\gamma \lg n$
- by def: $|u_\lambda v| \leq \alpha\, |u_\lambda| \Leftrightarrow |v| \leq \alpha(\gamma - 1) \lg n$
- x: block extended to left : $u_\lambda v u_\rho \subset x$



search $u_\lambda v u_\rho$

task: search left arm u_λ λ : left, ρ : right

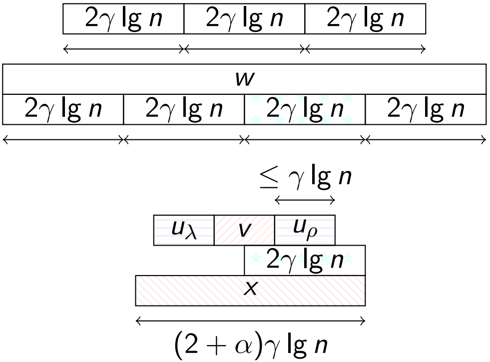
- assume: $|u_\rho| \leq \gamma \lg n$
- blocks of length w in $2\gamma \lg n$
- by def: $|u_\lambda v| \leq \alpha |u_\lambda| \Leftrightarrow |v| \leq \alpha(\gamma - 1) \lg n$
- x: block extended to left : $u_\lambda v u_\rho \subset x$
- x: m-th ***superblock***



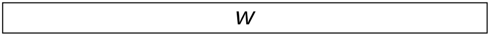
search $u_\lambda v u_\rho$

task: search left arm u_λ — λ : left, ρ : right

- assume: $|u_\rho| \leq \gamma \lg n$
- blocks of length w in $2\gamma \lg n$
- by def: $|u_\lambda v| \leq \alpha\, |u_\lambda| \Leftrightarrow |v| \leq \alpha(\gamma - 1) \lg n$
- x: block extended to left : $u_\lambda v u_\rho \subset x$
- x: m-th ***superblock***
- u_ρ contained in last $2\gamma \lg n$ chars of x

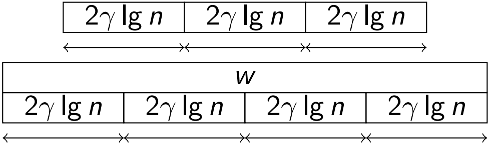


precompute superblocks $\{x_m\}_m$



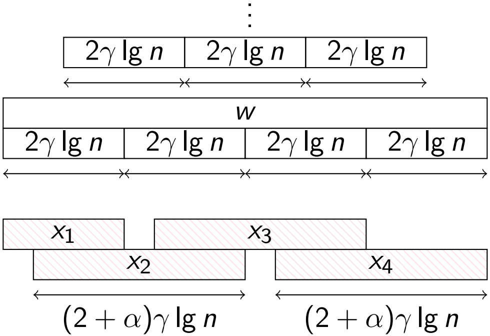
precompute superblocks $\{x_m\}_m$

- create $2\gamma \lg n$-partition



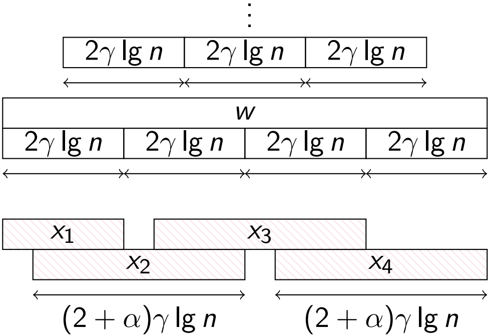
precompute superblocks $\{x_m\}_m$

- create $2\gamma \lg n$-partition
- create $\{x_m\}_m$

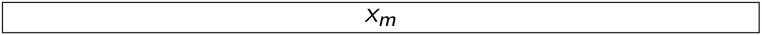


precompute superblocks $\{x_m\}_m$

- create $2\gamma \lg n$-partition
- create $\{x_m\}_m$
- on each superblock x_m build
  - lcp-DS
  - getOcc-DS

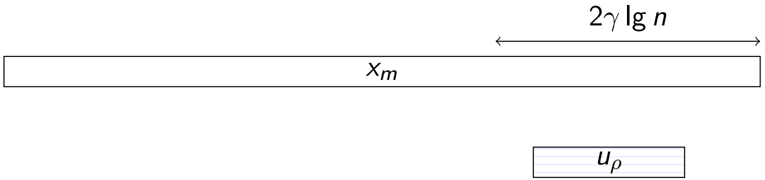


using superblock x_m



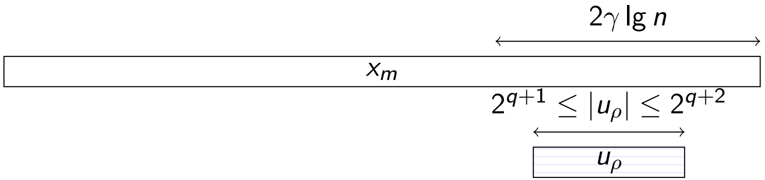
using superblock x_m

- u_ρ in the last $2\gamma \lg n$ characters of x_m



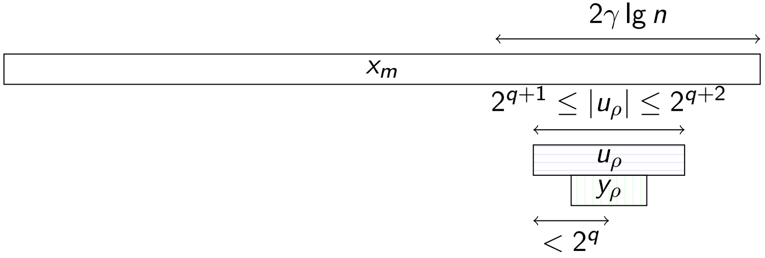
using superblock x_m

- u_ρ in the last $2\gamma \lg n$ characters of x_m
- $\exists q : 2^{q+1} \leq |u_\rho| \leq 2^{q+2}$



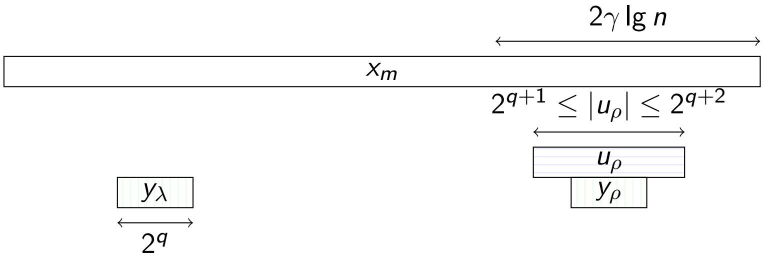
using superblock x_m

- u_ρ in the last $2\gamma \lg n$ characters of x_m
- $\exists q : 2^{q+1} \leq |u_\rho| \leq 2^{q+2}$
- $\exists 2^q$-basic factor $y_\rho : \mathrm{b}(y_\rho) \in [\mathrm{b}(u_\rho), \mathrm{b}(u_\rho) + 2^q]$



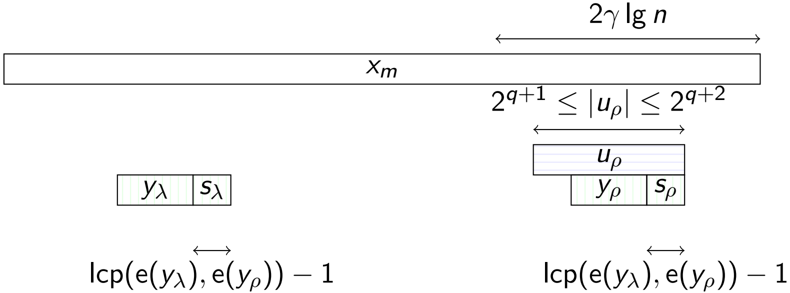
using superblock x_m

- u_ρ in the last $2\gamma \lg n$ characters of x_m
- $\exists q : 2^{q+1} \leq |u_\rho| \leq 2^{q+2}$
- $\exists 2^q$-basic factor $y_\rho : \mathrm{b}(y_\rho) \in [\mathrm{b}(u_\rho), \mathrm{b}(u_\rho) + 2^q]$
- use getOcc-index:
  - $\exists y_\lambda \equiv y_\rho : y_\lambda \subset u_\lambda$
  - $\# y_\lambda = \mathcal{O}(\alpha)$ many.



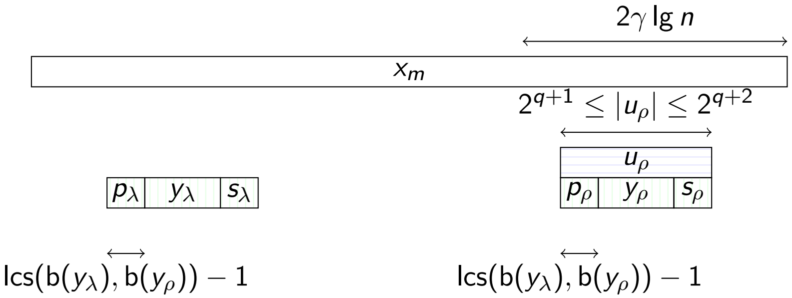
using superblock x_m

- u_ρ in the last $2\gamma \lg n$ characters of x_m
- $\exists q : 2^{q+1} \leq |u_\rho| \leq 2^{q+2}$
- $\exists 2^q$-basic factor $y_\rho : \mathrm{b}(y_\rho) \in [\mathrm{b}(u_\rho), \mathrm{b}(u_\rho) + 2^q]$
- use getOcc-index:
  - $\exists y_\lambda \equiv y_\rho : y_\lambda \subset u_\lambda$
  - $\# y_\lambda = \mathcal{O}(\alpha)$ many.
- for each y_λ use lcp-DS
  - extend right – $\mathrm{lcp}(\mathrm{e}(y_\lambda), \mathrm{e}(y_\rho))$



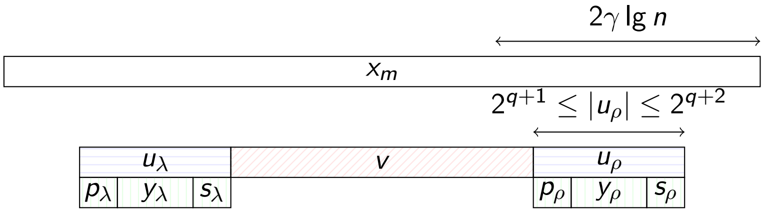
using superblock x_m

- u_ρ in the last $2\gamma \lg n$ characters of x_m
- $\exists q : 2^{q+1} \leq |u_\rho| \leq 2^{q+2}$
- $\exists 2^q$-basic factor $y_\rho : \mathrm{b}(y_\rho) \in [\mathrm{b}(u_\rho), \mathrm{b}(u_\rho) + 2^q]$
- use getOcc-index:
  - $\exists y_\lambda \equiv y_\rho : y_\lambda \subset u_\lambda$
  - $\# y_\lambda = \mathcal{O}(\alpha)$ many.
- for each y_λ use lcp-DS
  - extend right – $\mathrm{lcp}(\mathrm{e}(y_\lambda), \mathrm{e}(y_\rho))$
  - extend left – $\mathrm{lcs}(\mathrm{b}(y_\lambda), \mathrm{b}(y_\rho))$



using superblock x_m

- u_ρ in the last $2\gamma \lg n$ characters of x_m
- $\exists q : 2^{q+1} \leq |u_\rho| \leq 2^{q+2}$
- $\exists 2^q$-basic factor $y_\rho : \mathrm{b}(y_\rho) \in [\mathrm{b}(u_\rho), \mathrm{b}(u_\rho) + 2^q]$
- use getOcc-index:
  - $\exists y_\lambda \equiv y_\rho : y_\lambda \subset u_\lambda$
  - $\# y_\lambda = \mathcal{O}(\alpha)$ many.
- for each y_λ use lcp-DS
  - extend right – $\mathrm{lcp}(\mathrm{e}(y_\lambda), \mathrm{e}(y_\rho))$
  - extend left – $\mathrm{lcs}(\mathrm{b}(y_\lambda), \mathrm{b}(y_\rho))$



time complexity

// time

foreach $0 \leq m \leq n/\lg n$ **do**
  compute DS on superblock x_m // $\mathcal{O}(|x_m|) = \mathcal{O}(\alpha \lg n)$
  foreach $0 \leq q \leq \lg(\gamma \lg n)$ **do**
    foreach y_λ **do** // $\mathcal{O}(\alpha + \mathtt{occ}) = \mathcal{O}(\alpha)$
      compute lcp, lcs // $\mathcal{O}(1)$
      output if gaprep found // $\mathcal{O}(1)$

total $\mathcal{O}\left(\frac{n}{\lg n} \cdot \alpha \lg n\right)$ = $\mathcal{O}(\alpha n)$ time

- $\gamma = \mathcal{O}(1)$
- constr. time of lcp-DS and getOcc-DS: linear

summary

in $\mathcal{O}(\alpha\,|w|)$ time we

- find $\{$maximal α-gapped repeats of $w\}$
- find $\{$maximal α-gapped palindromes of $w\}$ (paper)

due to

- $\#$ maximal α-gapped repeats $= \mathcal{O}(\alpha\,|w|)$
- $\#$ maximal α-gapped palindroms $= \mathcal{O}(\alpha\,|w|)$

summary

in $\mathcal{O}(\alpha\,|w|)$ time we

- find {maximal α-gapped repeats of w}
- find {maximal α-gapped palindromes of w} (paper)

due to

- $\#$ maximal α-gapped repeats $= \mathcal{O}(\alpha\,|w|)$
- $\#$ maximal α-gapped palindroms $= \mathcal{O}(\alpha\,|w|)$

Thank you for listening. Any questions are welcome!