Cover Array String Reconstruction

Maxime Crochemore[1,2], Costas Iliopoulos[1,3], Solon Pissis[1], German Tischler[1,4]

[1]King's College London, UK, [2]Université Paris-Est, France, [3]Curtin University of Technology, Perth, Australia, [4]Newton Fellow

CPM 2010

Outline

Problem definition

Properties of minimal-cover arrays

String Construction and Validity Checking

Open Problems

Definition (Cover)

Consider non empty string y of length $|y| = n$ over alphabet Σ.

Cover

A proper factor u of y (i.e. a factor u of y s.t. $u \neq y$) is a *cover* (or *quasiperiod*) of y, iff every position of y lies in an occurence of u in y. In particular every cover of y is a border of y.

Example

aba is a cover of ababa

Cover is a generalization of *period*.

Minimal/Maximal Cover

If y has a cover, then it has a unique minimal (shortest) and maximal (longest) cover.

Definition (Minimal-Cover array)

Minimal-Cover Array

Integer array $C^{\mathrm{m}}[0\,..\,n-1]$ is the *Minimal-Cover Array* of y, if for each $i = 0, \ldots, n-1$ the value $C^{\mathrm{m}}[i]$ denotes the length of the minimal cover of $y[0\,..\,i]$ if such cover exists and 0 otherwise.

Computation of C^{m}

There exists an *on-line linear time* algorithm computing C^{m} from y.
(cf. D. Breslauer, *An on-line string superprimitivity test*. Inform. Process. Lett. 44 6 (1992), pp. 345–347)

Definition (Maximal-Cover array)

Maximal-Cover Array

Integer array $C^{\mathrm{M}}[0\,..\,n-1]$ is the *Maximal-Cover Array* of y, if for each $i = 0, \ldots, n-1$ the value $C^{\mathrm{M}}[i]$ denotes the length of the maximal cover of $y[0\,..\,i]$ if such cover exists and 0 otherwise.

Computation of C^{M}

There exists an *on-line linear time* algorithm computing C^{M} from y.
(cf. Y. Li and W. F. Smyth, *Computing the Cover Array in Linear Time*, Algorithmica 32 1 (2002), pp. 95-106)

Example (Cover array)

The following table provides the minimal-cover array C^{m} and the maximal-cover array C^{M} of the string

$$y = \mathtt{abaababaababaabaababababaaba}$$

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
$y[i]$	a	b	a	a	b	a	b	a	a	b	a	b	a	a	b	a	a	b	a	b	a	a	b	a
$C^{\mathrm{m}}[i]$	0	0	0	0	0	3	0	3	0	5	3	7	3	9	5	3	0	5	3	0	3	9	5	3
$C^{\mathrm{M}}[i]$	0	0	0	0	0	3	0	3	0	5	6	7	8	9	10	11	0	5	6	0	8	9	10	11

Problem definition

Let A denote an integer array of length n

Minimal/Maximal Validity Problem

Decide, whether A is the minimal-cover/maximal-cover array of some string.

Minimal/Maximal Construction Problem

If A is the valid minimal-cover/maximal-cover array of some string, construct a string x (over an unbounded alphabet) whose minimal-cover/maximal-cover array is A.

Properties of minimal-cover arrays

Simple properties

- First entry in a cover array always 0
- Value 1 only for prefixes of type a^k for $k > 1$

Subsequently assume $n > 1$.

Properties of minimal-cover arrays

Transitivity

If u and v cover y and $|u| < |v|$, then u covers v.

Lemma 1

If

$$C[i] \neq 0$$

for $0 \leq i < n$, then

$$C[C[i] - 1] = 0$$

Proof

Immediate from transitivity.

Properties of minimal-cover arrays

Lemma 2

Let i and j be positions s.t. $C[i] \neq 0 \neq C[j]$ and

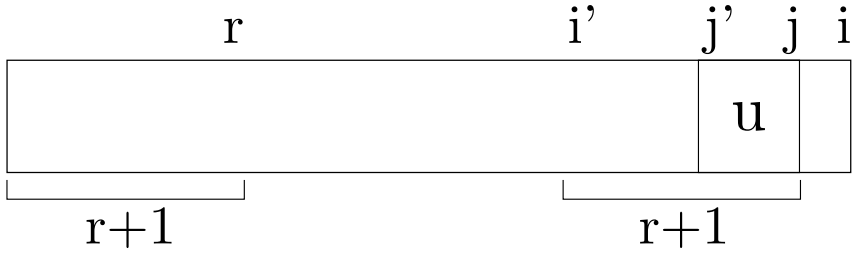
$$i - C[i] + 1 \leq j - C[j] + 1 < j < i$$

i.e. $\{j - C[j] + 1 \,..\, j\} \subset \{i - C[i] + 1 \,..\, i\}$. Let $r = j - (i - C[i] + 1)$. Then

$$C[r] = \begin{cases} 0 & \text{if } i - C[i] + 1 = j - C[j] + 1 \\ C[j] & \text{otherwise} \end{cases}$$

Properties of minimal-cover arrays

Illustration



where $i' = i - C^{\mathrm{m}}[i] + 1 \leq j' = j - C^{\mathrm{m}}[j] + 1$ and $u = y[j' \,..\, j]$

Proof

- $i' = j'$:

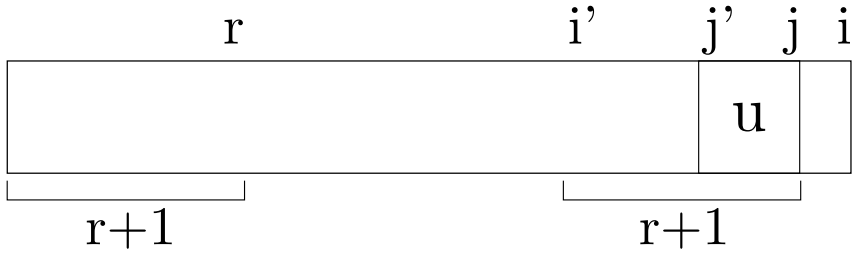
$$\begin{aligned} C^{\mathrm{m}}[r] &= C^{\mathrm{m}}[j - (i - C^{\mathrm{m}}[i] + 1)] \\ &= C^{\mathrm{m}}[j - (j - C^{\mathrm{m}}[j] + 1)] \\ &= C^{\mathrm{m}}[C^{\mathrm{m}}[j] - 1] \\ &= 0 \end{aligned}$$

due to transitivity.

Properties of minimal-cover arrays

Proof (followed)

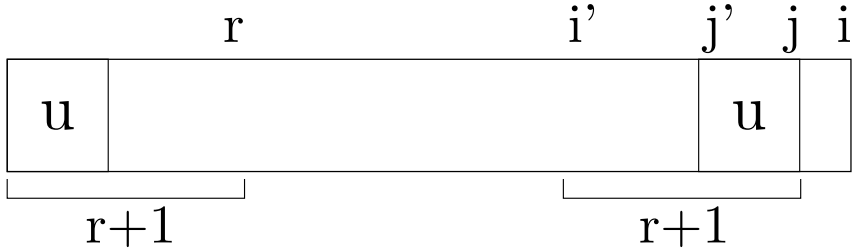
- $i' \neq j'$:



Properties of minimal-cover arrays

Proof (followed)

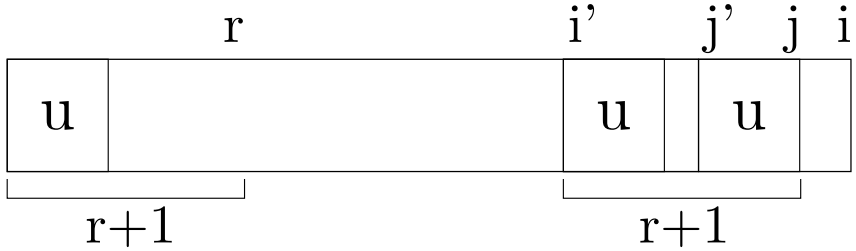
- $i' \neq j'$: $y[j' \,..\, j]$ is cover:



Properties of minimal-cover arrays

Proof (followed)

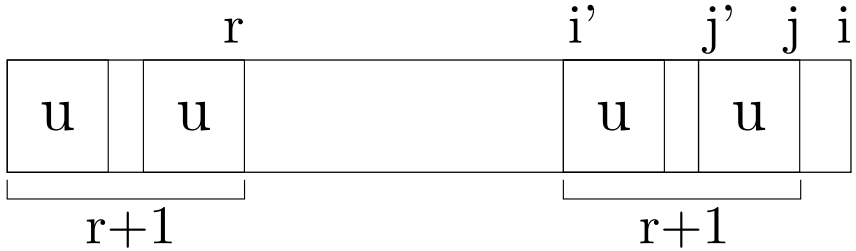
- $i' \neq j'$: $y[i' \,..\, i]$ is cover:



Properties of minimal-cover arrays

Proof (followed)

- $i' \neq j'$: $y[i' \,..\, i]$ is cover:



There is a copy of u ending at position $r > |u|$, thus $C^{\mathrm{m}}[r] \neq 0$ as u is a cover. Obtain $C^{\mathrm{m}}[r] = C^{\mathrm{m}}[j]$ by transitivity.

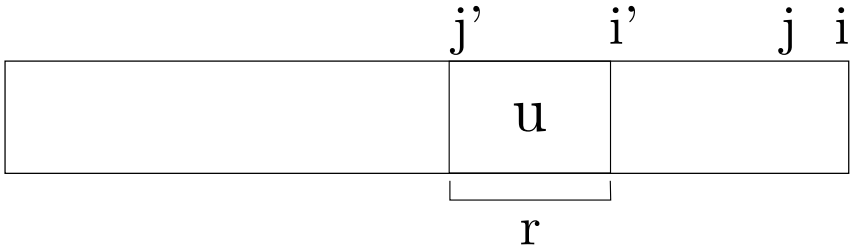
Properties of minimal-cover arrays

Lemma 3

Let i and j be positions s.t. $j < i$ and $j - C^{\mathrm{m}}[j] < i - C^{\mathrm{m}}[i]$. Then $r = (i - C^{\mathrm{m}}[i]) - (j - C^{\mathrm{m}}[j]) > C^{\mathrm{m}}[j]/2$.

Proof

Assume $r \leq C^{\mathrm{m}}[j]/2$. Illustration:
$(i' = i - C^{\mathrm{m}}[i] + 1, j' = j - C^{\mathrm{m}}[j] + 1, u = y[j' \,..\, i' - 1])$



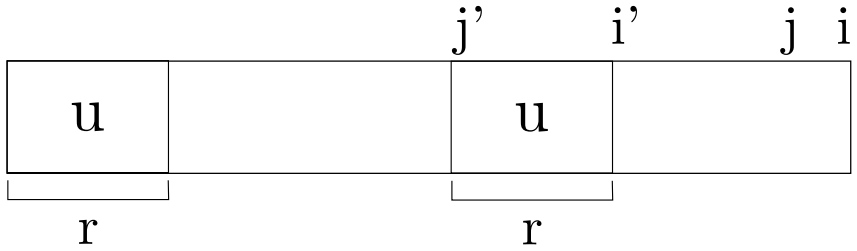
Properties of minimal-cover arrays

Lemma 3

Let i and j be positions s.t. $j < i$ and $j - C^{\mathrm{m}}[j] < i - C^{\mathrm{m}}[i]$. Then $r = (i - C^{\mathrm{m}}[i]) - (j - C^{\mathrm{m}}[j]) > C^{\mathrm{m}}[j]/2$.

Proof

Assume $r \leq C^{\mathrm{m}}[j]/2$. Illustration:
$(i' = i - C^{\mathrm{m}}[i] + 1, j' = j - C^{\mathrm{m}}[j] + 1, u = y[j' \,..\, i' - 1])$



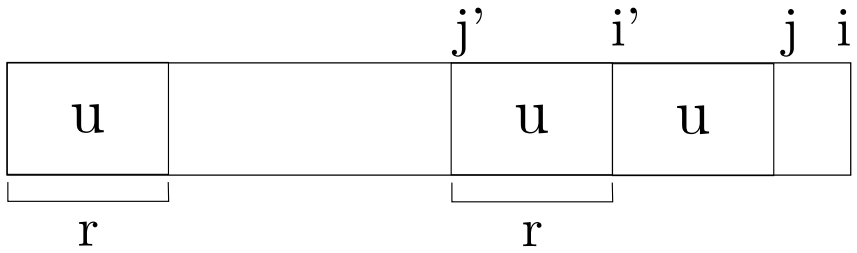
Properties of minimal-cover arrays

Lemma 3

Let i and j be positions s.t. $j < i$ and $j - C^{\mathrm{m}}[j] < i - C^{\mathrm{m}}[i]$. Then $r = (i - C^{\mathrm{m}}[i]) - (j - C^{\mathrm{m}}[j]) > C^{\mathrm{m}}[j]/2$.

Proof

Assume $r \leq C^{\mathrm{m}}[j]/2$. Illustration:
$(i' = i - C^{\mathrm{m}}[i] + 1, j' = j - C^{\mathrm{m}}[j] + 1, u = y[j' \,..\, i' - 1])$



$y[j' \,..\, j] = u^e$ for some $e \geq 2$. But $u^{1+e-\lfloor e \rfloor}$ is a shorter valid cover!

Properties of minimal-cover arrays

Totally covered position

Position j is totally covered, if there exists a position $i \neq j$ such that

$$i - C^{\mathrm{m}}[i] + 1 \leq j - C^{\mathrm{m}}[j] + 1 \leq j < i$$

Pruned minimal-cover array

Pruned minimal-cover array C^{p} obtained from C^{m} by setting entries of all totally covered positions to 0.

Properties of minimal-cover arrays

Lemma 4

$$\sum_{i=0}^{n-1} C^{\mathrm{p}}[i] \leq 2n$$

Proof

Let

$$I[i] = \begin{cases} \{i - C^{\mathrm{m}}[i] + 1, \ldots, i\} & \text{if } C^{\mathrm{m}}[i] \neq 0 \\ \emptyset & \text{otherwise} \end{cases}$$

Let $I'[i]$ lower half of $I[i]$. First halfs do not overlap (Lemma 3), thus $\sum |I'[i]| \leq n$ and $\sum |I[i]| \leq 2n$.

Properties of minimal-cover arrays

Bound of Lemma 4 is asymptotically tight. For $k > 1$, let

$$x_k = (a^k b a^{k+1} b)^{n/(2k+3)}$$

For $k = 2$ and $n = 23$ we get:

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
$y[i]$	a	a	b	a	a	a	b	a	a	b	a	a	a	b	a	a	b	a	a	a	b	a	a
$C^{\mathrm{p}}[i]$	0	1	0	0	0	0	0	0	5	0	0	0	0	7	0	5	0	0	0	0	7	0	5

- All segments of length $2k + 3$ of C^{p} contain values $2k + 1$ and $2k + 3$, except at the beginning of the string.
- Thus sum of elements in C^{p} is $(4k + 4)(\frac{n}{2k+3} - 1) + 1$, which tends to $2n$ when k (and n) goes to infinity.

Dependency Graph

Dependency

If we find $C[i] \neq 0$, then

$$y[i - C[i] + 1 + k] = y[k]$$

for $k = 0, 1, \ldots, C[i] - 1$. Respective positions are *dependent*.

Dependency Graph

Undirected graph (V, E) where

$$V = \{0, 1, \ldots, n - 1\}$$

(vertices are positions on y) and an edge exists between positions p_0 and p_1 iff p_0 and p_1 are dependent.

Dependency Graph

Observations

- If positions i and j connected in graph, then $y[i] = y[j]$
- Pruning C^{m} does not change connected components in the induced graph.
- Dependency graph can be built from C^{p} in time $O(n)$, as it has n vertices and at most $2n$ undirected edges
- A string y can be built from dependency graph in time $O(n)$ (compute connected components by DFS, assign different character to each component)

Pruning a Minimal-Cover Array

```
PRUNE(C, n)
1  ℓ ← 0
2  for i ← n − 1 downto 0 do
3      if ℓ ≥ C[i] then
4          C[i] ← 0
5      ℓ ← max(0, max(ℓ, C[i]) − 1)
6  return C
```

- Scan array from right to left
- Keep remaining length of leftmost ending interval *I* in variable ℓ
- Erase totally covered positions

String Construction from C^{m}

Theorem 1

If C^{m} is a valid minimal-cover array, a string y such that the minimal-cover array of y is C^{m} can be computed from C^{m} in linear time $O(n)$.

Method

- Compute pruned array C^{p} from C^{m}
- Construct dependency graph induced by C^{p}
- Deduce string y from induced graph

Maximal-Cover to Minimal-Cover Conversion

$\textsc{Maxtomin}(C, n)$

```
1  for i ← 0 to n − 1 do
2      if C[i] ≠ 0 and C[C[i] − 1] ≠ 0 then
3          C[i] ← C[C[i] − 1]
```

Method

- Scan array from left to right
- Consider position i such that $C^{\mathrm{M}}[i] \neq 0$
  1. $C^{\mathrm{M}}[C^{\mathrm{M}}[i] - 1] = 0$: $C^{\mathrm{M}}[i]$ is minimal, as there is no shorter cover (if there were, it would cover $y[0\,..\,C^{\mathrm{M}}[i] - 1]$)
  2. $C^{\mathrm{M}}[C^{\mathrm{M}}[i] - 1] \neq 0$: use minimal-cover found at position $C^{\mathrm{M}}[C^{\mathrm{M}}[i] - 1]$ (minimal by induction over length of considered prefix)

String Construction from C^{M}

Theorem 2

If C^{M} is a valid minimal-cover array, a string y such that the minimal-cover array of y is C^{M} can be computed from C^{M} in linear time $O(n)$.

Method

- Convert C^{M} to C^{m}
- Use method for C^{m}

Validity checking for Minimal-Cover

Let A be an integer array of length n.

Method

- Check, whether all values in A produce only valid edges in the dependency graph (i.e. $i - A[i] + 1 \geq 0$ for all positions i)
- Check, whether first halves of intervals I are overlap free (sufficient to check adjacent intervals)
- Construct string y from A assuming that A is a minimal-cover array
- Check whether minimal-cover array of y is A

Theorem 3

Let A be an integer array of length n. The above method verifies in linear time $O(n)$, whether A is a valid minimal-cover array.

Validity checking for Maximal-Cover

Let A be an integer array of length n.

Method

- Apply the maximal- to minimal-cover algorithm
- Apply checks as for minimal-cover case
- Construct string y
- Check whether maximal-cover array of y equals the original input array

Theorem 4

Let A be an integer array of length n. The above method verifies in linear time $O(n)$, whether A is a valid maximal-cover array.

Open Probems

- Construct string of minimal alphabet size from minimal-/maximal-cover array
- Solve problems for generalized definitions of cover (e.g. seeds)

Thank you!