

Time-Space Trade-Offs for the Longest Common Substring Problem

(To appear at CPM 2013, preprint available at hww.dk)

Tatiana Starikovskaya¹ and Hjalte Wedel Vildhøj²

¹Moscow State University, Department of Mechanics and Mathematics,
tat.starikovskaya@gmail.com

²Technical University of Denmark, DTU Compute, hww@hww.dk

ARCO Spring 2013, SDU Odense
April 5, 2013

The Longest Common Substring Problem

Definition

Problem: Given T_1, T_2, \dots, T_m of total length n . Compute the longest substring, which appears in at least $2 \leq d \leq m$ strings.

Example

	1	2	3	4	5	6	7	8	9	10	11	12	13
$T_1 =$	a	g	g	c	t	a	g	c	t	a	c	c	t
$T_2 =$	a	c	a	c	c	t	a	c	c	c	t	a	g
$T_3 =$	a	c	t	a	g	t	a	a	t	g	c	a	t

The Longest Common Substring Problem

Definition

Problem: Given T_1, T_2, \dots, T_m of total length n . Compute the longest substring, which appears in at least $2 \leq d \leq m$ strings.

Example

$T_1 =$ 1 2 3 4 5 6 7 8 9 10 11 12 13
a g g c t a g c t a c c t

$T_2 =$ a c a c c t a c c c t a g

$T_3 =$ a c t a g t a a t g c a t

$d = 3 \Rightarrow \text{LCS} = \text{c t a g}$

The Longest Common Substring Problem

Definition

Problem: Given T_1, T_2, \dots, T_m of total length n . Compute the longest substring, which appears in at least $2 \leq d \leq m$ strings.

Example

$T_1 =$ 1 2 3 4 5 6 7 8 9 10 11 12 13
a g g c t a g c t a c c t

$T_2 =$ a c a c c t a c c c t a g

$T_3 =$ a c t a g t a a t g c a t

$d = 3 \Rightarrow \text{LCS} = \text{c t a g}$

$d = 2 \Rightarrow \text{LCS} = \text{c t a c c}$

The Longest Common Substring Problem

A patented solution



US006359574B1

(12) **United States Patent**
Yariv

(10) **Patent No.:** **US 6,359,574 B1**
(45) **Date of Patent:** **Mar. 19, 2002**

(54) **METHOD FOR IDENTIFYING LONGEST COMMON SUBSTRINGS**

(75) Inventor: **Shalom Yariv**, Bet Shemesh (IL)

(73) Assignee: **Proxell Systems Ltd.**, Shimshon (IL)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **09/821,054**

(22) Filed: **Mar. 30, 2001**

Related U.S. Application Data

(60) Provisional application No. 60/262,654, filed on Jan. 22, 2001.

(51) Int. Cl.⁷ **H03M 7/00**

(52) U.S. Cl. **341/50**; 341/51

(58) **Field of Search** 341/50, 51, 87, 341/107, 86, 63, 95, 106; 708/210, 203

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,978,795 A * 11/1999 Poutanen et al. 707/3

6,069,573 A * 5/2000 Clark, II et al. 341/50
6,121,901 A * 9/2000 Welch et al. 341/51
6,191,710 B1 * 2/2001 Waletzki 341/63
6,320,522 B1 * 11/2001 Satoh 341/51

* cited by examiner

Primary Examiner—Brian Young

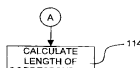
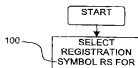
Assistant Examiner—John Nguyen

(74) *Attorney, Agent, or Firm*—AlphaPatent Associates Ltd.; Daniel J. Swirsky

(57) **ABSTRACT**

A method for identifying a longest common substring for a string T and a string R, including selecting a registration symbol that appears in both strings R and T, constructing a first relative distance vector R' from the appearance of the registration symbol in the string R, constructing a second relative distance vector T' from the appearance of the registration symbol in the string T, deriving a substring pair R_{CS} and T_{CS} in the strings R and T respectively for each common substring pair R_{CS} and T_{CS} in the vectors R' and T' respectively, and identifying the longest matching of the R_{CS} and T_{CS} substring pairs as the longest common substring for the string T and the string R.

10 Claims, 3 Drawing Sheets

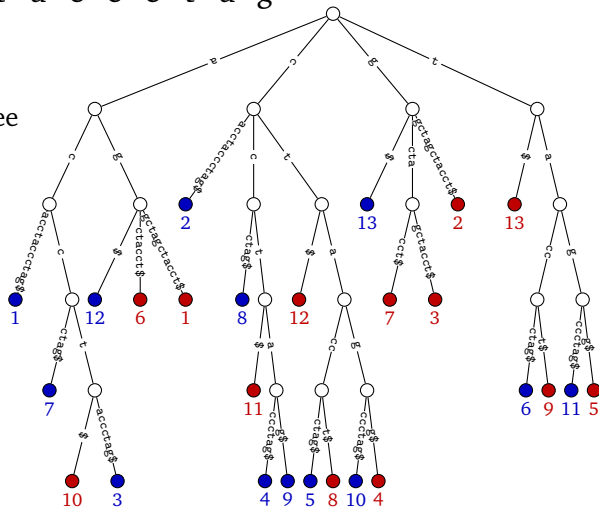


A Textbook Solution

$T_1 =$ a g g c t a g c t a c c t

$T_2 =$ a c a c c t a c c c t a g

Build Generalized Suffix Tree

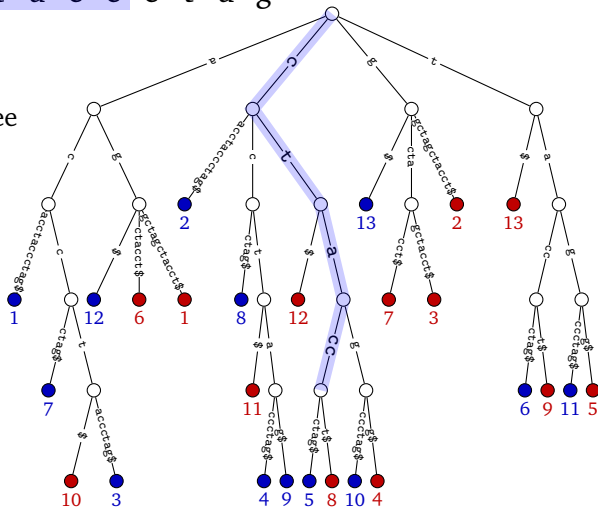


A Textbook Solution

$T_1 =$ a g g c t a g c t a c c t

$T_2 =$ a c a c c t a c c c t a g

Build Generalized Suffix Tree



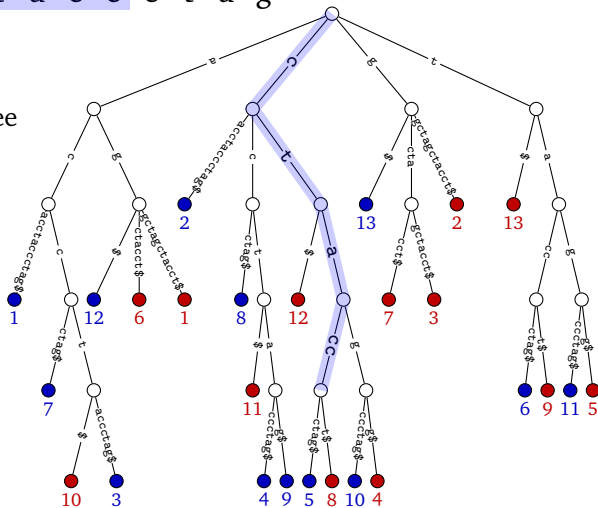
A Textbook Solution

$T_1 =$ a g g c t a g c t a c c t

$T_2 =$ a c a c c t a c c c t a g

Build Generalized Suffix Tree

Space: $\Theta(n)$



Our Results

Question

Can the LCS problem be solved (deterministically) in $O(n^{1-\epsilon})$ space and $O(n^{1+\epsilon})$ time for $0 \leq \epsilon \leq 1$?

Our Answer

Yes if $0 \leq \epsilon \leq \frac{1}{3}$. More precisely,

For two strings ($d = m = 2$), the problem can be solved in:

$$\begin{array}{l} \text{Time: } O(n^{1+\epsilon}) \\ \text{Space: } O(n^{1-\epsilon}) \end{array} \quad \text{for any } 0 < \epsilon \leq \frac{1}{3}.$$

In the general case ($2 \leq d \leq m$), the problem can be solved in:

$$\begin{array}{l} \text{Time: } O(n^{1+\epsilon} \log^2 n (d \log^2 n + d^2)) \\ \text{Space: } O(n^{1-\epsilon}) \end{array} \quad \text{for any } 0 \leq \epsilon < \frac{1}{3}.$$

A Solution for Two Strings

When the LCS is long

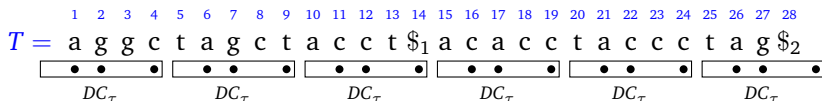
Idea: Preprocess a sparse sample of the n suffixes for LCP queries.

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28
 $T =$ a g g c t a g c t a c c t $\$_1$ a c a c c t a c c c t a g $\$_2$

A Solution for Two Strings

When the LCS is long

Idea: Preprocess a sparse sample of the n suffixes for LCP queries.

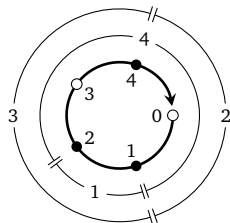


Difference Covers

A *difference cover modulo τ* is a set of integers $DC_\tau \subseteq \{0, 1, \dots, \tau - 1\}$ such that for any distance $d \in \{0, 1, \dots, \tau - 1\}$, DC_τ contains two elements separated by distance d modulo τ .

Ex: The set $DC_\tau = \{1, 2, 4\}$ is a difference cover modulo 5.

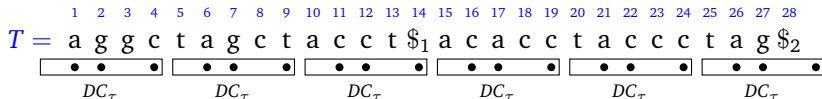
d	0	1	2	3	4
i, j	1, 1	2, 1	1, 4	4, 1	1, 2



A Solution for Two Strings

When the LCS is long

Idea: Preprocess a sparse sample of the n suffixes for LCP queries.

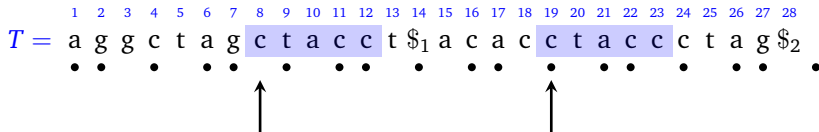


- ▶ Number of sampled suffixes: $O\left(\frac{n}{\tau} |DC_\tau|\right) = O\left(\frac{n}{\sqrt{\tau}}\right)$.

A Solution for Two Strings

When the LCS is long

Idea: Preprocess a sparse sample of the n suffixes for LCP queries.



- ▶ Number of sampled suffixes: $O\left(\frac{n}{\tau} |DC_{\tau}| \right) = O\left(\frac{n}{\sqrt{\tau}}\right)$.
- ▶ The LCS is the LCP of two suffixes.

A Solution for Two Strings

When the LCS is long

Idea: Preprocess a sparse sample of the n suffixes for LCP queries.

$T =$

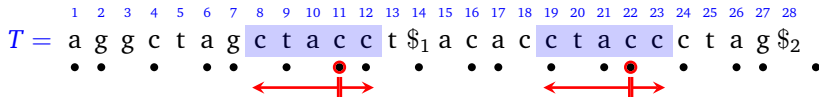
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	
	a	g	g	c	t	a	g	c	t	a	c	c	t	\$ ₁	a	c	a	c	c	t	a	c	c	c	t	a	g	\$ ₂	
	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•

- ▶ Number of sampled suffixes: $O\left(\frac{n}{\tau} |DC_{\tau}| \right) = O\left(\frac{n}{\sqrt{\tau}}\right)$.
- ▶ The LCS is the LCP of two suffixes.
- ▶ If $|LCS| \geq \tau$ one of the first τ characters of the LCS is sampled in both strings.

A Solution for Two Strings

When the LCS is long

Idea: Preprocess a sparse sample of the n suffixes for LCP queries.



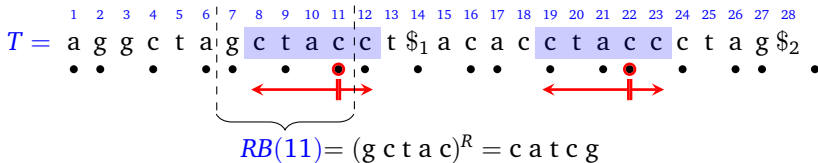
- ▶ Number of sampled suffixes: $O\left(\frac{n}{\tau} |DC_{\tau}|\right) = O\left(\frac{n}{\sqrt{\tau}}\right)$.
- ▶ The LCS is the LCP of two suffixes.
- ▶ If $|\text{LCS}| \geq \tau$ one of the first τ characters of the LCS is sampled in both strings.
- ▶ Hence the LCS corresponds to a pair (p_1^*, p_2^*) maximizing

$$\text{lcp}(RB(p_1), RB(p_2)) + \text{lcp}(T[p_1..], T[p_2..]) - 1$$

A Solution for Two Strings

When the LCS is long

Idea: Preprocess a sparse sample of the n suffixes for LCP queries.



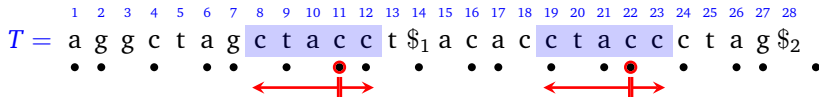
- ▶ Number of sampled suffixes: $O\left(\frac{n}{\tau} |DC_{\tau}|\right) = O\left(\frac{n}{\tau}\right)$.
- ▶ The LCS is the LCP of two suffixes.
- ▶ If $|LCS| \geq \tau$ one of the first τ characters of the LCS is sampled in both strings.
- ▶ Hence the LCS corresponds to a pair (p_1^*, p_2^*) maximizing

$$\text{lcp}(RB(p_1), RB(p_2)) + \text{lcp}(T[p_1..], T[p_2..]) - 1$$

A Solution for Two Strings

When the LCS is long

How to compute the pair (p_1^*, p_2^*) faster than $O(\frac{n^2}{\tau})$?



$$SA_{\tau} = [14, 21, 17, 26, 6, 1, 16, 22, 11, 12, 19, 24, 4, 27, 7, 2, 9]$$

$$LCP_{\tau} = [0, 3, 1, 2, 2, 0, 1, 2, 1, 2, 3, 4, 0, 1, 1, 0]$$

$$SA_{\tau}^R = [14, 1, 17, 21, 26, 6, 16, 22, 11, 19, 12, 24, 4, 2, 27, 7, 9]$$

$$LCP_{\tau}^R = [0, 1, 1, 4, 3, 0, 2, 4, 1, 3, 2, 1, 0, 2, 4, 0]$$

Main observation: $\text{lcp}(T[p_1^*..], T[p_2^*..]) \in [\ell_{\max} - \tau + 1; \ell_{\max}]$, so we can ignore all pairs with lcp values smaller than $\ell_{\max} - \tau + 1$.

A Solution for Two Strings

When the LCS is long

How to compute the pair (p_1^*, p_2^*) faster than $O(\frac{n^2}{\tau})$?

$T =$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
	a	g	g	c	t	a	g	c	t	a	c	c	t	\$ ₁	a	c	a	c	c	t	a	c	c	c	t	a	g	\$ ₂
	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•

$SA_\tau = [14, 21, 17, 26, 6, 1, 16, 22, 11, 12, 19, 24, 4, 27, 7, 2, 9]$

 \longleftarrow
 \longrightarrow

$LCP_\tau = [0, 3, 1, 2, 2, 0, 1, 2, 1, 2, 3, 4, 0, 1, 1, 0]$

$SA_\tau^R = [14, 1, 17, 21, 26, 6, 16, 22, 11, 19, 12, 24, 4, 2, 27, 7, 9]$

$LCP_\tau^R = [0, 1, 1, 4, 3, 0, 2, 4, 1, 3, 2, 1, 0, 2, 4, 0]$

Main observation: $\text{lcp}(T[p_1^*..], T[p_2^*..]) \in [\ell_{\max} - \tau + 1; \ell_{\max}]$, so we can ignore all pairs with lcp values smaller than $\ell_{\max} - \tau + 1$.

A Solution for Two Strings

When the LCS is long

How to compute the pair (p_1^*, p_2^*) faster than $O(\frac{n^2}{\tau})$?

$T =$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
$T =$	a	g	g	c	t	a	g	c	t	a	c	c	t	\$ ₁	a	c	a	c	c	t	a	c	c	c	t	a	g	\$ ₂
	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•

←→

←→

←→

$SA_\tau = [14, 21, 17, 26, 6, 1, 16, 22, 11, 12, 19, 24, 4, 27, 7, 2, 9]$

$LCP_\tau = [0, 3, 1, 2, 2, 0, 1, 2, 1, 2, 3, 4, 0, 1, 1, 0]$

$SA_\tau^R = [14, 1, 17, 21, 26, 6, 16, 22, 11, 19, 12, 24, 4, 2, 27, 7, 9]$

$LCP_\tau^R = [0, 1, 1, 4, 3, 0, 2, 4, 1, 3, 2, 1, 0, 2, 4, 0]$

Main observation: $\text{lcp}(T[p_1^*..], T[p_2^*..]) \in [\ell_{\max} - \tau + 1; \ell_{\max}]$, so we can ignore all pairs with lcp values smaller than $\ell_{\max} - \tau + 1$.

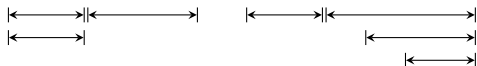
A Solution for Two Strings

When the LCS is long

How to compute the pair (p_1^*, p_2^*) faster than $O(\frac{n^2}{\tau})$?

$T =$

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
$T =$	a	g	g	c	t	a	g	c	t	a	c	c	t	$\$1$	a	c	a	c	c	t	a	c	c	c	t	a	g	$\$2$
	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•



$$SA_\tau = [14, 21, 17, 26, 6, 1, 16, 22, 11, 12, 19, 24, 4, 27, 7, 2, 9]$$

$$LCP_\tau = [0, 3, 1, 2, 2, 0, 1, 2, 1, 2, 3, 4, 0, 1, 1, 0]$$

$$SA_\tau^R = [14, 1, 17, 21, 26, 6, 16, 22, 11, 19, 12, 24, 4, 2, 27, 7, 9]$$

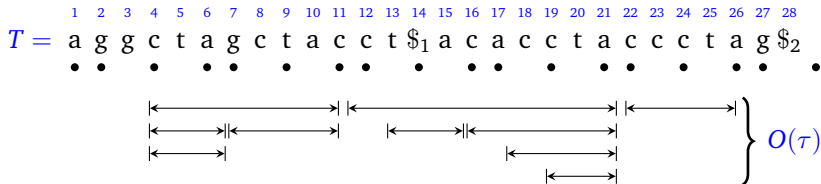
$$LCP_\tau^R = [0, 1, 1, 4, 3, 0, 2, 4, 1, 3, 2, 1, 0, 2, 4, 0]$$

Main observation: $\text{lcp}(T[p_1^*..], T[p_2^*..]) \in [\ell_{\max} - \tau + 1; \ell_{\max}]$, so we can ignore all pairs with lcp values smaller than $\ell_{\max} - \tau + 1$.

A Solution for Two Strings

When the LCS is long

How to compute the pair (p_1^*, p_2^*) faster than $O(\frac{n^2}{\tau})$?



$SA_\tau = [14, 21, 17, 26, 6, 1, 16, 22, 11, 12, 19, 24, 4, 27, 7, 2, 9]$

$LCP_\tau = [0, 3, 1, 2, 2, 0, 1, 2, 1, 2, 3, 4, 0, 1, 1, 0]$

Analysis (sketch): $O(\tau)$ rounds each using $O(n/\sqrt{\tau})$ time and space:

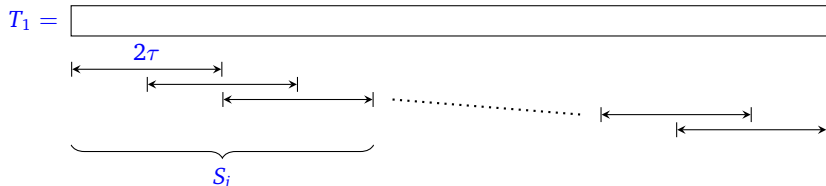
Time: $O(n\sqrt{\tau})$
 Space: $O(n/\sqrt{\tau})$

$\xrightarrow{\tau = n^{2\varepsilon}}$

Time: $O(n^{1+\varepsilon})$
 Space: $O(n^{1-\varepsilon})$ $0 < \varepsilon \leq \frac{1}{2}$

A Solution for Two Strings

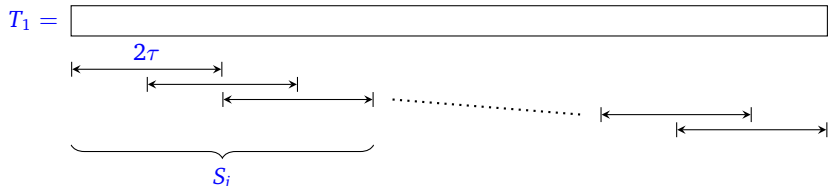
When the LCS is shorter than τ



- ▶ The LCS is a substring of one of the strings of length 2τ .
- ▶ Build the generalized suffix tree for a batch S_i of strings of total length $O(\frac{n}{\sqrt{\tau}})$.
- ▶ Traverse the suffix tree with T_2 in $O(n)$ time to find the node of greatest string depth.
- ▶ Repeat for all $O(\sqrt{\tau})$ batches.

A Solution for Two Strings

When the LCS is shorter than τ



- ▶ The LCS is a substring of one of the strings of length 2τ .
- ▶ Build the generalized suffix tree for a batch S_i of strings of total length $O(\frac{n}{\sqrt{\tau}})$.
- ▶ Traverse the suffix tree with T_2 in $O(n)$ time to find the node of greatest string depth.
- ▶ Repeat for all $O(\sqrt{\tau})$ batches.

Time: $O(n\sqrt{\tau})$
Space: $O(n/\sqrt{\tau})$

$\tau = n^{2\varepsilon}$

Time: $O(n^{1+\varepsilon})$
Space: $O(n^{1-\varepsilon})$ $0 \leq \varepsilon \leq \frac{1}{3}$

$\tau = O(n/\sqrt{\tau})$

Conclusion

Results

For two strings ($d = m = 2$), the LCS problem can be solved in:

$$\begin{array}{l} \text{Time: } O(n^{1+\varepsilon}) \\ \text{Space: } O(n^{1-\varepsilon}) \end{array} \text{ for any } 0 < \varepsilon \leq \frac{1}{3}.$$

In the general case ($2 \leq d \leq m$), the LCS problem can be solved in:

$$\begin{array}{l} \text{Time: } O\left(n^{1+\varepsilon} \log^2 n (d \log^2 n + d^2)\right) \\ \text{Space: } O(n^{1-\varepsilon}) \end{array} \text{ for any } 0 \leq \varepsilon < \frac{1}{3}.$$

Open Problems

Can the generalized solution be improved? Can the trade-off interval of our solutions be extended to $0 \leq \varepsilon \leq \frac{1}{2}$? Can the problem be solved in $O(n^{1+\varepsilon})$ time and $O(n^{1-\varepsilon})$ space for any $0 \leq \varepsilon \leq 1$?