Maintaining the Size of LZ77 on Semi-dynamic Strings

Hideo Bannai
Panagiotis Charalampopoulos
Jakub Radoszewski

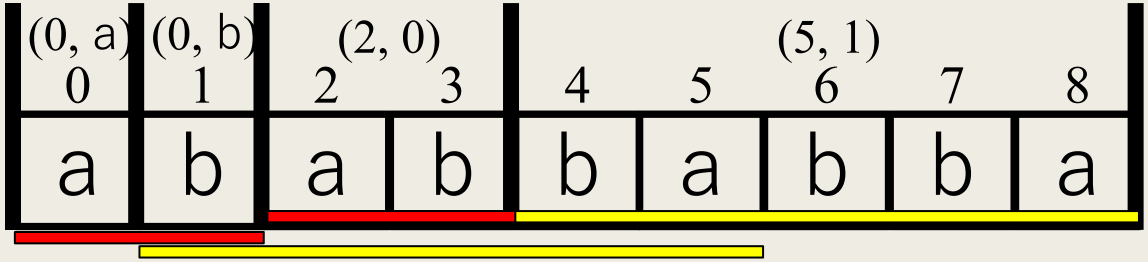
LZ77 [Ziv&Lempel 1977]

LZ77 factorization

Greedy partitioning of string into ***phrases***:

- first occurrence of symbol ➔(0, *c*)
- longest prefix of the rest, that has prev. occ. ➔(*len*, *src*)

Example



- One of the smallest expressions efficiently computable
- Size of LZ77 = # of phrases z : measure of compression
- We will require *src* to take *right-most* previous occurrence

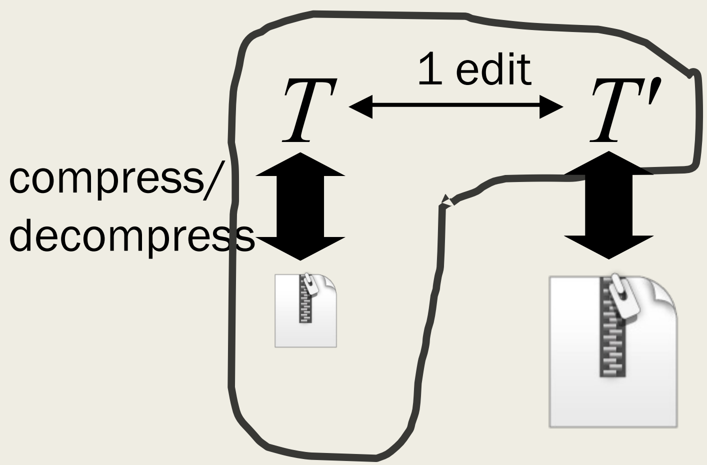
Motivation

Compression sensitivity [Akagi et al. 2023]

- How much can the size of LZ77 (or other compressed representations) change after an edit operation?
- Showed Upper/Lower bounds of additive/multiplicative change of various repetitiveness measures under ins/del/sub operations

➔ Other operations? (rotation?)

➔ Can we exploit this to get smaller representation?



Main Results

- Maintain LZ77 size in semi-dynamic setting:
  - pop_front: delete first symbol
  - push_back: append given symbol

  in $O(\sqrt{n}\log^2 n)$ amortized time for updates using $O(n)$ space
- Corollary:
  - $O(n\sqrt{n}\log^2 n)$ time algorithm for computing most compressible rotation
    - $O(n^2)$ time is straightforward
    - substring compression queries compute in $\tilde{O}(Z)$ time: Z = total number of LZ77 factors in all rotations (still quadratic in worst case)
- Bounds for sensitivity of LZ77 for rotation operation (1 pop_front and 1 push_back)

Longest Previous Factor (LPF) Tree

<u>Main Idea</u>: Maintain LPF tree

Node : position i (right most position = root)
Parent : $i + \max\{1, \mathrm{LPF}[i]\}$
Edge label : $i - \mathrm{SRC}[i] = \mathrm{DIST}[i]$

i	0	1	2	3	4	5	6	7	8	9
T	a	b	b	b	a	b	a	b	b	
LPF	0	0	2	1	2	3	3	2	1	
SRC	-	-	1	2	0	3	0	2	7	
DIST	-	-	1	1	4	2	6	5	1	

- # of edges on path from 0 to n (root) is LZ77 size.
- Use Link/cut trees: will allow update/path length queries in $O(\log n)$ time.
- Incoming edges come from range of consecutive positions.
  - range of consecutive incoming edges can be moved by simulating RB-tree split/merge with Link/cut trees with $O(\log n)$ time overhead.
- $O(\sqrt{n})$ updates will allow us to get $O(\sqrt{n}\log^2 n)$ time

Combinatorial Properties (1)

What can change with pop_front?

i	0	1	2	3	4	5	6	7	8	9
T	a	b	b	b	a	b	a	b	b	
LPF	0	0	2	1	~~2~~0	3	~~3~~2	2	1	
SRC	-	-	1	2	~~0~~-	3	~~0~~4	2	7	
DIST	-	-	1	1	~~4~~-	2	~~6~~2	5	1	

Only LPFs whose right most occurrence is at the beginning of the string can change

Combinatorial Properties (prefix LPF occ)

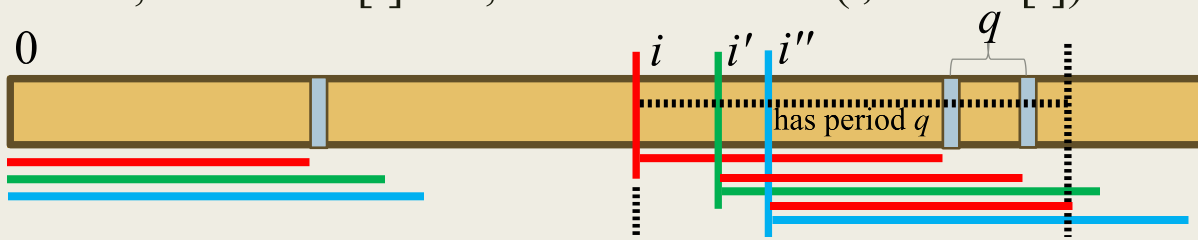
Claim

For any string, the number of right-most LPF occurrences that are at the beginning of the string is $O(\sqrt{n})$.

Proof

For $0 < i < i'$, right-most LPF occurrence at beginning implies $\mathrm{LPF}[i] < \mathrm{LPF}[i']$.

Show: $\forall i$, with $\mathrm{SRC}[i] = 0$, at most one $i' \in (i, i+\mathrm{LPF}[i])$ with $\mathrm{SRC}[i'] = 0$.



$T[i..i''+\mathrm{LPF}[i])$ is periodic with period $q = i' - i$. (By periodicity lemma: min period is divisor of q, and occurrences of $T[i..i+\mathrm{LPF}[i])$ form arithmetic progression)

Therefore $T[0..\mathrm{LPF}[i]] = T[i..i+\mathrm{LPF}[i]]$, which contradicts that $T[i..i+\mathrm{LPF}[i])$ is longest.

Combinatorial Properties (2)

What can change with `push_back`?

i	0	1	2	3	4	5	6	7	8	9	10
T	a	b	b	b	a	b	a	b	b	b	
LPF	0	0	2	1	2	3	~~3~~4	~~2~~3	~~1~~2	1	
SRC	-	-	1	2	0	3	0	~~2~~1	7	8	
DIST	-	-	1	1	4	2	6	~~5~~6	1	1	

LPFs that are suffixes (i.e. reach the root) can change by extending to the new root

- $\Theta(n)$ edges may need to change parents, but they are consecutive and can be done in a batch
- How to maintain DIST values?

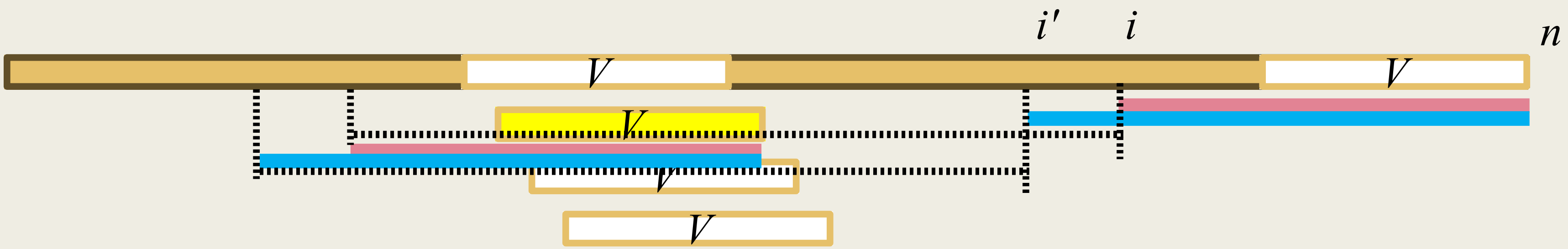
distinct distances for suffix LPFs

Claim

DIST[i] of suffix LPFs (i + LPF[i] = n) form a non-increasing sequence with O($\sqrt{n}$) different values.

Proof

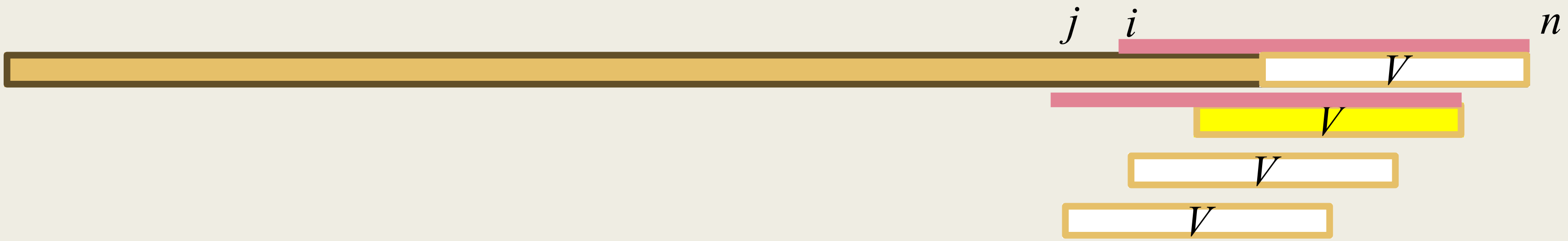
- Right-most occurrence of shorter suffix leads to non-increasing distance.
- For suffixes shorter than $\sqrt{n}$ ➔ at most $\sqrt{n}$ different DIST values.
- For longer suffixes: let V = length $\sqrt{n}$ suffix:
  - Occurrence of V in T form O($\sqrt{n}$) arithmetic progressions (AP).
  - DIST[i] = DIST[i'] for any i, i' are same for same *implied occurrence* of V.
  - Show: for each AP, at most two elements are implied occurrences.



distinct distances for suffix LPFs

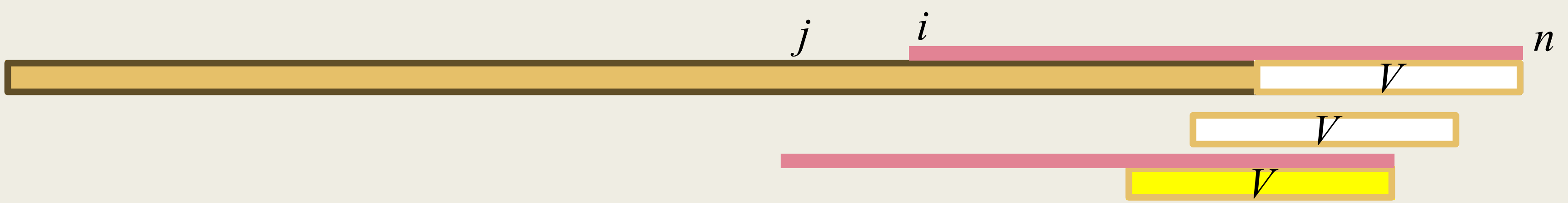
Case: If SRC[i] = j implies occ in suffix AP

subcase: periodicity of V extends to i: must use 2nd to last occ



subcase: periodicity of V doesn't extend to i: **impossible**

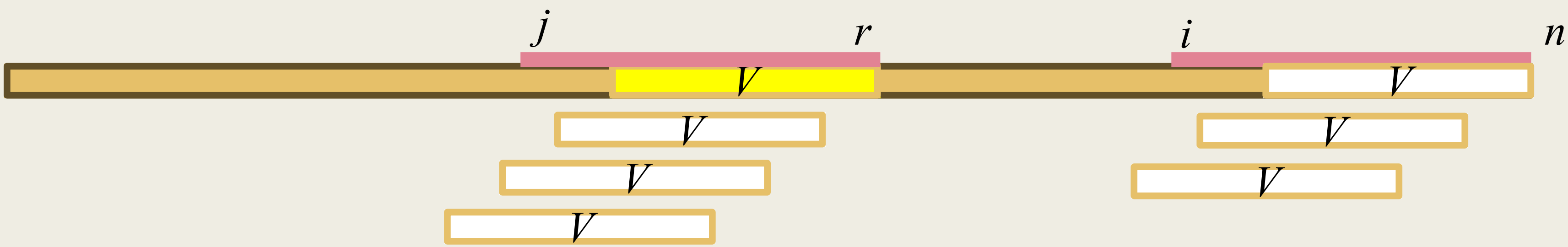
occ of V in $T[i..n]$ imply those in T[$j..j$+LPF[i]])



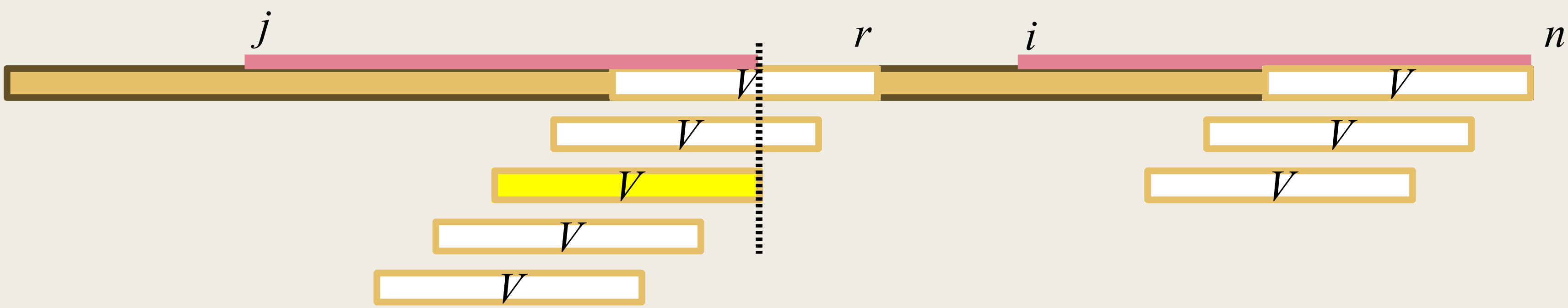
distinct distances for suffix LPFs

Case: If SRC[i] = j implies occ in non-suffix AP

subcase: i is in periodic suffix ➔ j is right most occ of AP



subcase: i is not in periodic suffix ➔ j is uniquely matching suffix



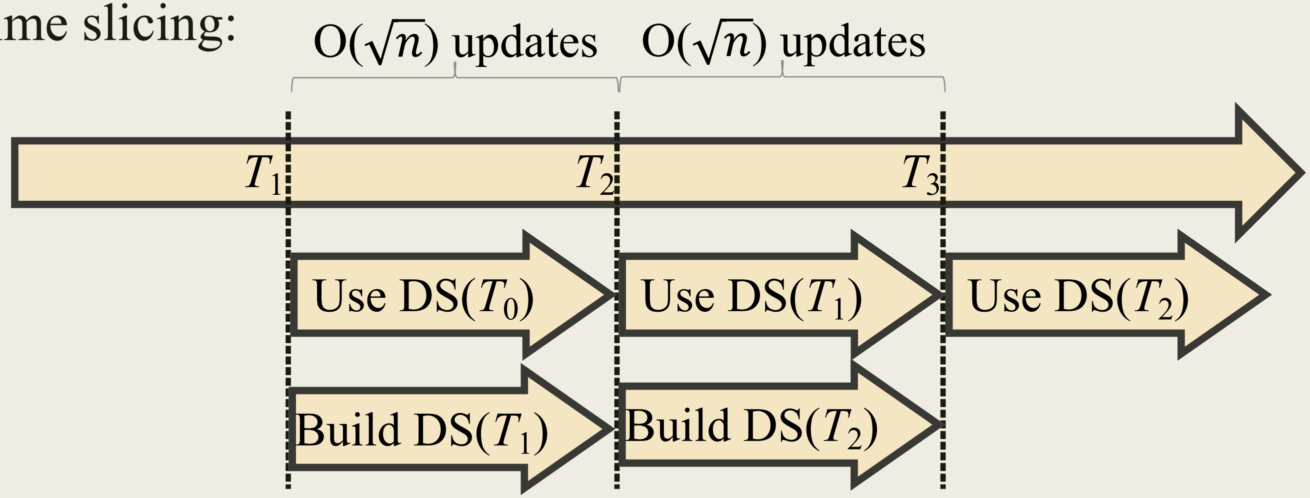
Semi-Dynamic Batch LPF queries

Claim: We can maintain a data structure for LPF/SRC/DIST queries

- $O(\sqrt{n \log n})$ update time
- $O(\sqrt{n \log n} + |Y| \log^{\epsilon} n)$ query time for all positions $y \in Y \subseteq [0, |T|)$.

Proof (sketch):

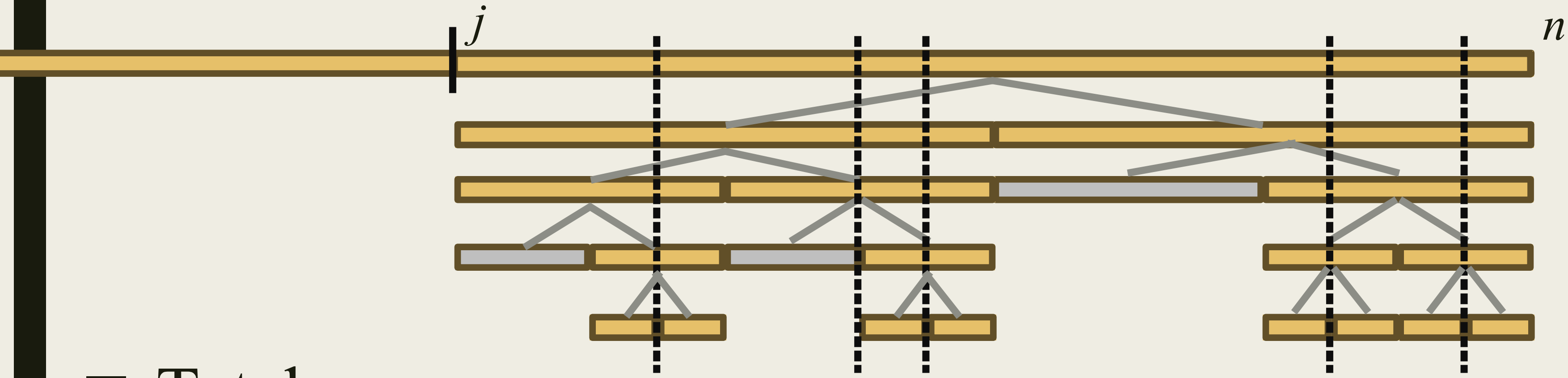
- Use static data structure with $O(\log^{\epsilon} n)$ query, $O(n\sqrt{\log n})$ build time [Keller et al. 2013, Belazzougui&Puglisi 2016]
- Time slicing:



At any point, static data structure for $T[0..|T| - x]$ ($x \leq 2\sqrt{n}$) available.
Use it to answer queries for $T[d..|T|)$

Finding the edges to update (push_back)

- Use LPF query to do binary search to find longest repeating suffix R = $T[j..n)$
- Recursively query endpoints of binary partitions
- Batch LPF queries ($|Y|=O(\sqrt{n})$) at each of log levels



- Total: $O((\sqrt{n\log n} + \sqrt{n}\log^{\epsilon} n)\log n)) = O(\sqrt{n}\log^{1.5} n)$

Finding the edges to update (pop_front)

See paper...

Maintain DIST information for finding edges that have SRC[i] = d, where d is the number of deletions used so far.

- For each value, determine smallest DIST value and store them in priority queue (min-type) ordered by SRC.
- When positions are deleted, if smallest element SRC is equal to current beginning of string, then all edges with same SRC must be updated.
- Since DIST can be updated in ranges, store key/value as: for each value of DIST, a predecessor DS holding maximal range of consecutive positions with that value

Sensitivity of rotation

rot(T): 1 pop_front (delete) and 1 push_back (insert)

$1/6|LZ(rot(T))| \leq LZ(T) \leq 6|LZ(rot(T))|$

follows from [Akagi et al. 2023] (factor 3 del/sub, 2 for ins)

We further show:

- There are infinitely many strings for which:
  $|LZ(rot(T))| \geq |LZ(T)| + \Theta\left(\sqrt{|T|}\right)$ and
  $|LZ(rot(T))| \geq 3/2\ |LZ(T)| - 2$

  Consider $S_m S_1 S_2 \ldots S_m$, where $S_i = a_1 \ldots a_i$.

- For any string T,
  $|LZ(T)| - 1 \leq |LZ(rot(T))| \leq |LZ(T)| + \Theta\left(\sqrt{|T|}\right)$
  and
  $|LZ(rot(T))| \leq 2|LZ(S)|$

Open Problems

- Can we do better?
  - Upper bound of $O(\sqrt{n})$ is for one rotation. Tighter bound for all rotations?
  - Is strictly sublinear update time possible in the fully dynamic setting?