MADALGO
CENTER FOR MASSIVE DATA ALGORITHMICS

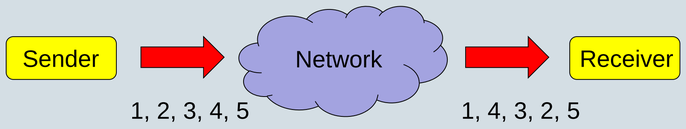

Lap-Kei Lee
Aarhus University

Danmarks Grundforskningsfond
Danish National Research Foundation

Edit Distance to Monotonicity in Sliding Windows

Motivations

Network Quality Monitoring: Suppose a sender sends a sequence of packets to a receiver through a network, where the packets have increasing sequence number.



Network congestion can distort the order of packets, and thus more distortions would imply a lower network quality. Therefore, it is desirable to answer:

How close is the sequence received (at the receiver) to the sorted sequence?

Relational Database: Operations are best performed when the relations are sorted or nearly sorted over the relevant attributes. Yet, sorting relations is expensive. Keeping an estimate on the sortedness help determining when sorting is needed .

Webpage Ranking (e.g., PageRank™), **Sorting Algorithms**, etc.

Problems and Data Stream Model

Problem: *How close* is a length-*n* sequence to monotonicity?
E.g., <1, 4, 3, 2, 5> to <1, 2, 3, 4, 5>

- **Edit Distance (ED):** Each step involves removing & reinserting an item.
  E.g., ED for <1, 4, 3, 2, 5> = 2: <1, 4, 3, 2, 5> → <1, 2, 4, 3, 5> → <1, 2, 3, 4, 5>
- **Length of Longest Increasing Subsequence (LIS)**
  E.g., LIS for <1, 4, 3, 2, 5> = 3

Note that **LIS** = *n* - **ED**.

Data Stream Model: The sequence is a stream of *n* items from the universe [1, *m*].

- Items are accessed *sequentially* in *one pass*.
- Since volume of data is massive, and exact computation of LIS and ED requires $\Omega(n)$ bits even for randomized algorithms [5]. Only *approximation* is possible.

Previous Results

Exact computation of LIS and ED
- Patience Sorting: O(LIS) space
- $\Omega(n)$ space for randomized algorithms [5]

Approximation (with multiplicative error) (note that LIS problem ≠ ED problem)**:**

(1+ε)-Approximation for LIS
- Deterministic upper bound using $O(\sqrt{n}/\varepsilon)$ space [5]
- Tight deterministic lower bound [3, 4]

Approximation for ED
- (4+ε)-approximate randomized algorithm using $O(\varepsilon^{-2}\log^2 n)$ space [5]
- (2+ε)-approximate deterministic algorithm using $O(\varepsilon^{-2}\log^2(\varepsilon n))$ space [3]

Upper Bound for ED in Sliding Window Model

We extend the results to the **sliding window model**:
- Motivation: In network quality monitoring, since network traffic is often bursty, network quality is better measured based on recent data.
- Let *w* be a positive integer denoting the window size.
  We are interested in *estimating ED of the latest w items*.
- (4+ε)-approximate deterministic algorithm using $O(\varepsilon^{-2}\log^2(\varepsilon w))$ space

Algorithm: Consider stream $a_1, a_2, \ldots, a_n$.

1. A *4-approximate estimator* of ED
- For each item index *i*, let inv(*i*) = { *j* | *j*<*i* but $a_j > a_i$}.
- The estimator in [5] is a subset *R* of *bad indices*:
  ∀ bad index *i* , ∃ *j* < *i* s.t. more than half of $\{a_j, a_{j+1}, \ldots, a_{i-1}\} \in$ inv(*i*).
- Index *j* is called the *witness* of the bad index *i*.
- *Property of R*: ED / 2 ≤ |*R*| ≤ 2 ED (4-approximation of **ED**)

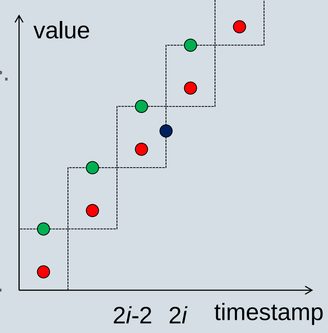
2. Approximating the estimator
- If (median of $\{a_j, a_{j+1}, \ldots, a_{i-1}\}$) > a_i, then *i* is a bad index.
- We use the data structure in [6] to *approximate the median*, and make sure that the witness of each bad index is the largest possible.
- The size of this set *R*' of bad indices is $4+\Theta(\varepsilon)$-approximation of **ED**.

3. Handling expiry of witnesses in the sliding window model
- A bad index is no longer bad when its largest witness expires.
- |*R*'| = the number of non-expired largest witnesses.
- Every time we find a bad index, we put its largest witness into a *witness stream*. Note that witnesses can be in arbitrary order.
- We use the data structure in [2] for *basic counting in out-of-order stream* to give an (1+ε)-approximate of |*R*'|.
- The estimate is $4+\Theta(\varepsilon)$-approximation of ED in sliding window.

Lower Bound for ED in Out-of-order Stream

Reduction from the 2-player communication problem **INDEX**:
- *Alice* has a bit string $x = (x_1, x_2, \ldots, x_k)$, $x_i \in \{0, 1\}$.
- *Bob* has an index $i \in \{1, 2, \ldots, k\}$.
- *Communication:* Alice sends a message to Bob.
- *Objective:* Bob outputs x_i.
- $\Omega(k)$ bits is required for randomized protocol.

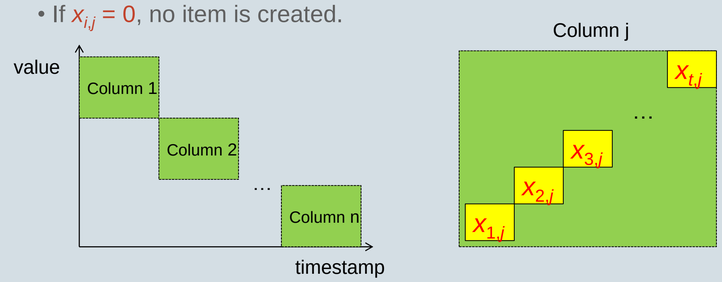
Use algorithm for **ED** to solve **INDEX**:
- Each item is a tuple <timestamp, value>.
- *Alice* constructs $\{a_1, a_2, \ldots, a_k\}$ s.t.
  If $x_j = 0$, a_j = <2*j*-1, 3*j*-2>;
  If $x_j = 1$, a_j = <2*j*-1, 3*j*>.
- *Bob* constructs a_{k+1} = <2*i*, 3*i*-1>
- If $x_j = 0$, ED = 0; if $x_j = 1$, ED = 1.
- Any O(1)-approximate randomized algorithm can distinguish the two cases.
- Space = $\Omega(k)$ bits



Lower Bound for LIS in Out-of-order Stream

Reduction from the *t*-players communication problem **DISJ**:
- *t players*: $P_1, P_2, \ldots, P_t$
- *Input*: *t* x *n* bit matrix ***x***
- *Player* P_i has the *i*-th row of ***x***: $(x_{i,1}, x_{i,2}, \ldots, x_{i,n})$, $x_{i,j} \in \{0, 1\}$.
- *Communication:*
  $P_1, P_2, \ldots, P_t$ sends a message to the next player in turn.
- *Objective:* P_t outputs
  - **disjoint**, if each column of ***x*** contains at most one 1-entry.
  - **uniquely intersecting**, if one column has all 1-entries and other column has at most one 1-entry.
- Total communication between all players is $\Omega(n/t)$ bits for randomized protocols.

Use algorithm for **LIS** to solve **DISJ**:
- $P_1, P_2, \ldots, P_t$ create items in an out-of-order stream in turn.
- Each item is a tuple <timestamp, value>.
- Player P_i has the *i*-th row of ***x***: $(x_{i,1}, x_{i,2}, \ldots, x_{i,n})$, $x_{i,j} \in \{0, 1\}$.
  - If $x_{i,j} = 1$, create an item <(*j*-1) *t* + *i*, (*n*-*j*) *t* + *i*>.
  - If $x_{i,j} = 0$, no item is created.



- If ***x*** is disjoint, each column has ≤ 1 item, and thus LIS ≤ 1.
- If ***x*** is uniquely intersecting, one column has *t* items, and thus LIS = *t*.
- An *t*/2-approximate randomized algorithm for LIS can distinguish the two cases, and thus its space $= \Omega(n/t \cdot \frac{1}{t}) = \Omega(n/t^2)$ bits.

References

[1] Chan, Lam, Lee, Pan, Ting, Zhang. *Edit distance to monotonicity in sliding windows.* ISAAC 2011.

[2] Cormode, Korn, Tirthapura. *Time-decaying aggregates in out-of-order streams.* PODS 2008.

[3] Ergun, Jowhari. *On distance to monotonicity and longest increasing subsequence of a data stream.* SODA 2008.

[4] Gal, Gopalan. *Lower bounds on streaming algorithms for approximating the length of the longest increasing subsequence.* SIAM Journal on Computing 2010.

[5] Gopalan, Jayram, Krauthgamer, Kumar. *Estimating the sortedness of a data stream.* SODA 2007.

[6] Lin, Lu, Xu, Yu. *Continuously maintaining quantile summaries of the most recent n elements over a data stream.* ICDE 2004.