Aggregating a Data Set: Rankings to Strings

Diptarka Chakraborty

(National University of Singapore)

Recent Trends in Algorithms 2022

Finding a Median String (Recall)

- Given a set of strings $S = \{x_1, x_2, \ldots, x_m\}$ over alphabet Σ, the objective is to find a string $y \in \Sigma^*$ (not necessarily from S) that minimizes

$$Obj(S, y) = \sum_{x_i \in S} ED(x_i, y)$$

- Let y^* be a string that minimizes $Obj(S, y)$
- y^* is referred to as *median*

Questions encountered so far (Recall)

- How to do clustering efficiently? Not in this talk
- How to perform approximate trace reconstruction efficiently?
- What is the connection between approximate trace reconstruction and approximate median?
- How to find an approximate median efficiently?

Rank Aggregation (Recall)

- Given a set of permutations $S = \{x_1, x_2, \dots, x_m\}$ over $[n]$, the objective is to find a **permutation** y (not necessarily from S) that minimizes

$$Obj(S, y) = \sum_{x_i \in S} ED(x_i, y)$$

Rank Aggregation - What do we know?

Metric	Upper Bound	Lower Bound
Kendall-tau	PTAS ($(1+\epsilon)$-approximation in polytime) [Mathieu, Schudy '07]	NP-hard [Dwork et al. '01] (even for 4 inputs) For 3 inputs, NP-hard or P?
Ulam	$(2-\epsilon)$-approximation in polytime [C, Das, Krauthgamer '21] For 3 inputs, in P	NP-hard?

Ulam Median for three inputs

- Phase 1: Relax the constraint that the solution need not be a permutation

- Phase 2: Round the relaxed solution into a feasible one

Ulam Median – Phase 1 (Relax)

- The output need not be a permutation (can be an arbitrary string)

- Use dynamic programming for the Edit median, that runs in $O(2^m n^m)$ time (recall, now $m = 3$)

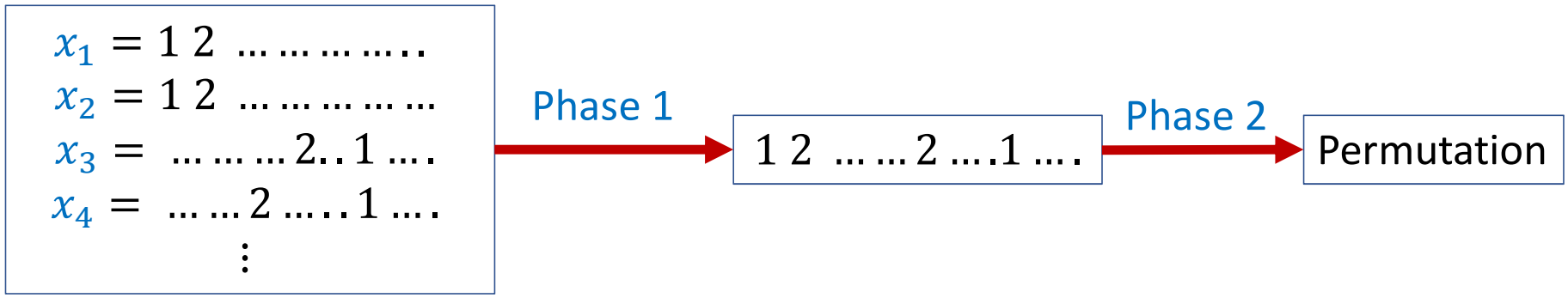
- Actually, find an n-length solution

Exercise 1: Design dynamic programming for the Edit median

Exercise 2: Modify it so that it outputs length restricted solution

Ulam Median – Phase 2 (Round)

- Round the relaxed solution to a feasible one (permutation)
- “Cleverly” delete duplicate symbols and insert missing ones



Ulam Median – High-level Argument

- Consider the following two quantities
  - $OPT_{rank}(S)$: Optimum value attained by a permutation (Our final target)
  - $OPT_{n-len}(S)$: Optimum value attained by a n-length string (Phase 1 gives)

- Observation: $OPT_{n-len}(S) \leq OPT_{rank}(S)$

- Suppose $\bar{y}$ attains $OPT_{n-len}(S)$. Now, $\bar{y} \xrightarrow{\text{Phase 2}} z$

c-approximation

- To show: $Obj(S, z) \leq c \cdot Obj(S, \bar{y})$

Ulam Median for three inputs

- We show: $Obj(S, z) \leq Obj(S, \bar{y})$ (i.e., $c = 1$), for output z for three inputs

- Fix an arbitrary optimal *alignment* between $\bar{y}$ and x_1, x_2, x_3
- For each duplicate symbol
  - Only keep the occurrence with maximum matches and delete the rest

Idea: Increase the Obj by 1 (for one input), but decrease by 2 (avoid deletions in two inputs)

Ulam Median for three inputs

- We show: $Obj(S, z) \leq Obj(S, \bar{y})$ (i.e., $c = 1$), for output z for three inputs

- Fix an arbitrary optimal *alignment* between $\bar{y}$ and x_1, x_2, x_3
- For each duplicate symbol
  - Only keep the occurrence with maximum matches and delete the rest
- For each missing symbol
  - Insert while maintaining the match with at least one input

Idea: Decrease the Obj by 1 (for one matched input), but increase by 2 (for the rest two inputs)

Ulam Median for three inputs

- We show: $Obj(S, z) \leq Obj(S, \bar{y})$ (i.e., $c = 1$), for output z for three inputs

- Delete each duplicate: Increase the Obj by 1 (for one input), but decrease by 2 (avoid deletions in two inputs)
- Insert each missing: Decrease the Obj by 1 (for one matched input), but increase by 2 (for the rest two inputs)

- No. of duplicate entries = No. of missing (since $\bar{y}$ has length n)

- Hence, Obj stays the same

Ulam Median for constant inputs

- Can be extended to $Obj(S, z) \leq \frac{3}{2} Obj(S, \bar{y})$ (i.e., $c \approx 3/2$), for $O(1)$ inputs [C, Das, Krauthgamer '21]

- Similar approach works for the center problem [C, Gajjar, Jha '21]

Max. Rank Aggregation / Ulam Center

- Given a set of permutations $S = \{x_1, x_2, \dots, x_m\}$ over $[n]$, the objective is to find a **permutation** y (not necessarily from S) that minimizes

$$Obj(S, y) = \max_{x_i \in S} ED(x_i, y)$$

- The Ulam center problem is NP-hard

Ulam Center for O(1) inputs [C, Gajjar, Jha '21]

- Phase 1: Find an n-length center string (need not be a permutation)

- Phase 2: Round the relaxed solution into a feasible one
- Interestingly, Phase 2 is essentially solving the Hamming center with *wildcards*

- **Theorem:** Achieve $3/2$-approximation for the Ulam center in polytime for $O(1)$ inputs (and exact for 3 inputs)

Ulam Median – What about arbitrary inputs?

- Phase 1: Relax the constraint that the solution need not be a permutation

Takes exponential (in the no. of inputs m) time

- Phase 2: Round the relaxed solution into a feasible one

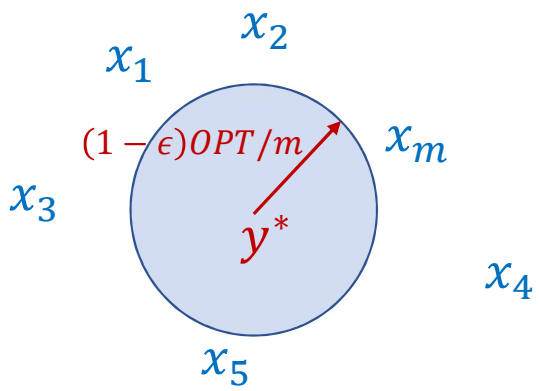
Ulam Median (High Regime)

- Consider the case when $OPT(S) \geq mn/5$ (Recall, m = no. of inputs, n = length of input)

- **Theorem:** Best input achieves $(2 - \epsilon)$-approximation, for some $\epsilon > 0$ (independent of n, m)

Ulam Median (High Regime)

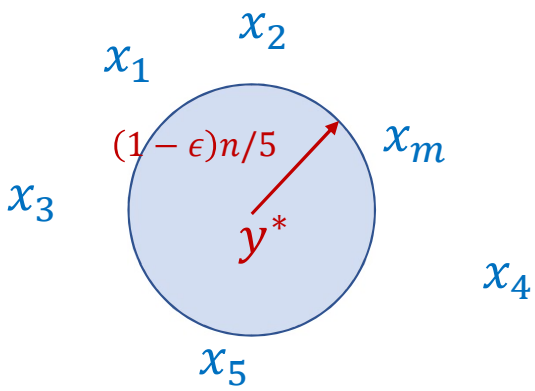
- **Assumption:** No input $x \in S$ is at distance $\leq (1-\epsilon)OPT/m$ from an (unknown) median y^*



- Else, that point achieves $(2-\epsilon)$-approximation

Ulam Median (High Regime)

- **High Regime:** $OPT \geq mn/5$



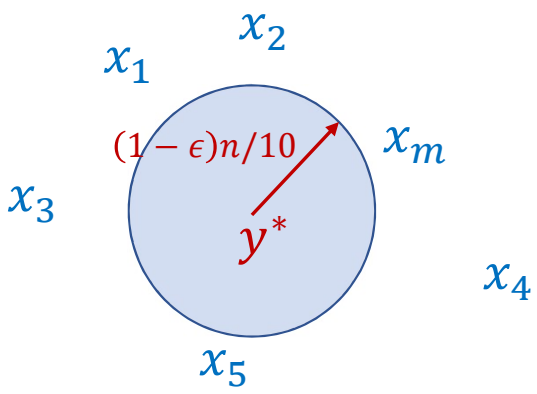
- For each point $x_i \in S$, there is a set I_i of edit (or unaligned) symbols, where $|I_i| \geq (1-\epsilon)n/10$ (for simplicity, $|I_i| \geq n/15$)

Ulam Median (High Regime)

- For each point $x_i \in S$, $|I_i| \geq n/15$

- **Claim:** There can be at most 30 such sets with pairwise intersection of size $\leq n/450$

(Follows from inclusion-exclusion)



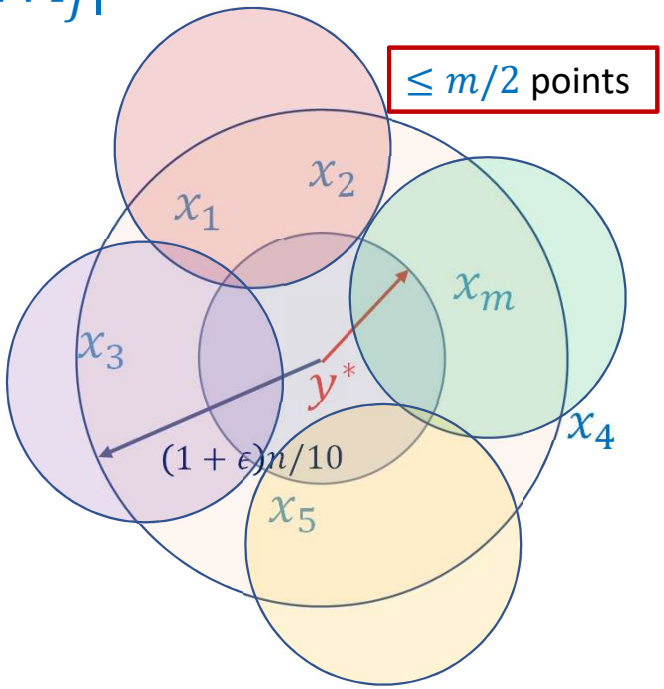
Ulam Median (High Regime)

- **Observation:** For any two x_i, x_j,

$$ED(x_i, x_j) \leq |I_i| + |I_j| - |I_i \cap I_j|$$

- So the points inside the outer ball forms ≤ 30 clusters (with a *buddy point*)

- Where the distance of any point from its buddy is roughly $\leq \left(2 - \frac{1}{30}\right) n/15$

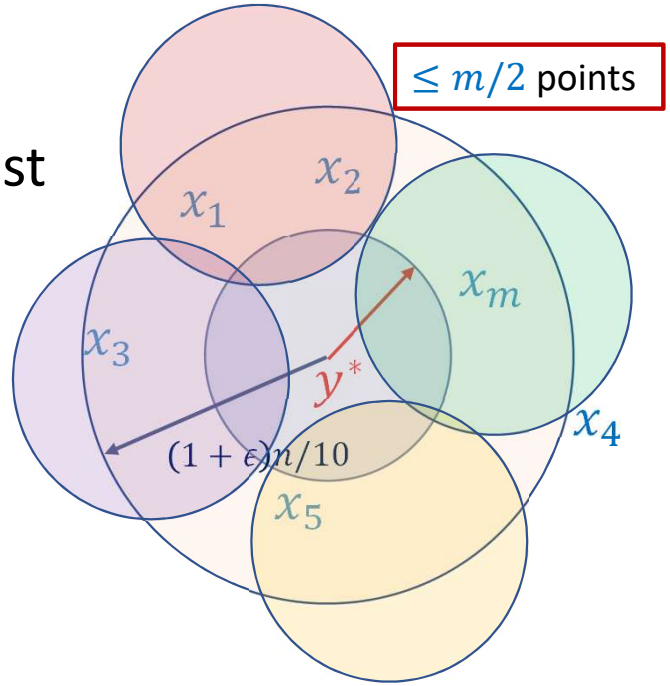


Ulam Median (High Regime)

- By averaging, one of the clusters must contributes $OPT/70$ mass to the OPT

- Next, consider the buddy point ($\bar{y}$) of the largest contributing cluster

- It breaks 2-factor for its own cluster, and roughly maintains 2-factor for other points

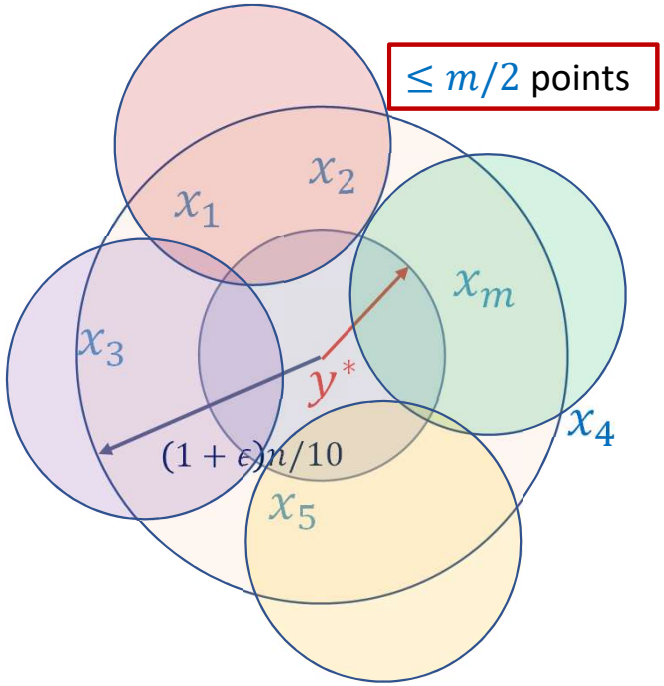


Ulam Median (High Regime)

- So $\bar{y}$ attains $(2 - 1/30)$-approx. for $OPT/70$ mass, and roughly 2-approx. for the rest

- Overall, it attains $(2 - \epsilon)$-approx.

- **Theorem:** Best input achieves $(2 - \epsilon)$-approx., for some $\epsilon > 0$ (independent of n, m)



Ulam Median (Small Regime)

- Consider the case when $OPT(S)$ is "small" (and distributed over all/many symbols)

- **Intuition:** Most pairs of symbols (a, b) maintain their relative ordering in "majority" (much more than 50%) of inputs, and thus can be retrieved.

- Of course, if we could identify relative ordering of all the pairs, we can identify the (unknown) median using topological ordering

Ulam Median (Small Regime)

- Issue: Relative ordering of some pair of symbols could be changed in majority of inputs (maybe because of "a few" *bad* symbols)
- Can be handled by removing (shortest) cycles from the graph with edges representing the relative ordering in "majority" inputs
- Another issue: How to merge high and small regime cases?
- Merging only leads us to $(2-\epsilon)$-approx. algorithm

Open Problems

- Can we get a PTAS for the Ulam median?
- Can we show the Ulam median is NP-hard? (For $O(1)$ inputs?)
- Can we extend the result to the Edit median?

Trace Reconstruction (Recall)

- Problem Statement: Reconstructing an unknown string from its noisy observable copies (aka. *traces*)

- There is an unknown string x of length n
- We observe a set of "noisy" copies (traces) $x_1, x_2, \ldots, x_m$
- The objective is to recover x

1. Use as few samples as possible
2. Minimize the "error"
3. Design an efficient algorithm

Types of Noises (Recall)

- **Substitution Channel** (Each symbol is flipped with probability p)
- **Deletion Channel** (Each symbol is deleted with probability p)
- **Insertion-Deletion Channel** (While scanning, keeps the next symbol as it is w.p. $1 - p$, deletes it w.p. $p/2$, and inserts a uniformly randomly chosen symbol before the next symbol w.p. $p/2$)

We consider this one

Two Cases w.r.t. Unknown Strings (Recall)

- **Worst-case:** Unknown string is arbitrary
- **Average-case:** Unknown string is a uniformly randomly chosen string

What about Approximation? (Recall)

- Many applications (including DNA storage system) do not need the exact reconstruction
- It suffices to recover a string z that is "close" to the unknown string x
- *Edit distance (ED)* is a natural closeness measure

Can we do better?

Getting $\sim pn$ edit distance is trivial (any input trace works)

Result 1 [C, Das, Krauthgamer '21]

Recall, unknown string = x of length n, and noise parameter = p

Only using *three* traces, we can recover a string z such that

$$\Pr\left[ED(x,z) \leq O\left(p^2 \log\frac{1}{p}\right)n\right] \geq 1 - \frac{1}{n}$$

in time $\tilde{O}(n)$ time (using a deterministic algorithm).

In this talk, we only consider Binary alphabet

Questions encountered so far (Recall)

- How to do clustering efficiently? Not in this talk
- How to perform approximate trace reconstruction efficiently?
- What is the connection between approximate trace reconstruction and approximate median?
- How to find an approximate median efficiently?

Result 2: They are the same (at least in average-case) [C, Das, Krauthgamer '21]

- Recall, in the average-case the unknown string x is **chosen uniformly at random**

- Let $\bar{y}$ be a $(1+\epsilon)$-approximate median of $x_1, x_2, \cdots, x_m$ (generated from x by the **insertion-deletion** channel), for $\epsilon \in \left[110\ p \log \frac{1}{p}, \frac{1}{6}\right]$

- Then w.h.p. $ED(x, \bar{y}) \leq O(\epsilon)\frac{OPT}{m}$

Result 2 ⟹ (almost) Result 1

- For three traces $S = \{x_1, x_2, x_3\}$, $OPT \leq 3\ p\ n$

($ED(x, x_i) \leq p\ n$ w.h.p., and thus $Obj(S, x) \leq 3\ p\ n$)

- Thus, w.h.p. $ED(x, \bar{y}) \leq O\left(p^2 \log \frac{1}{p}\right) n$

- Recall, a standard dynamic programming computes an exact median in time $O(n^3)$ for three input strings

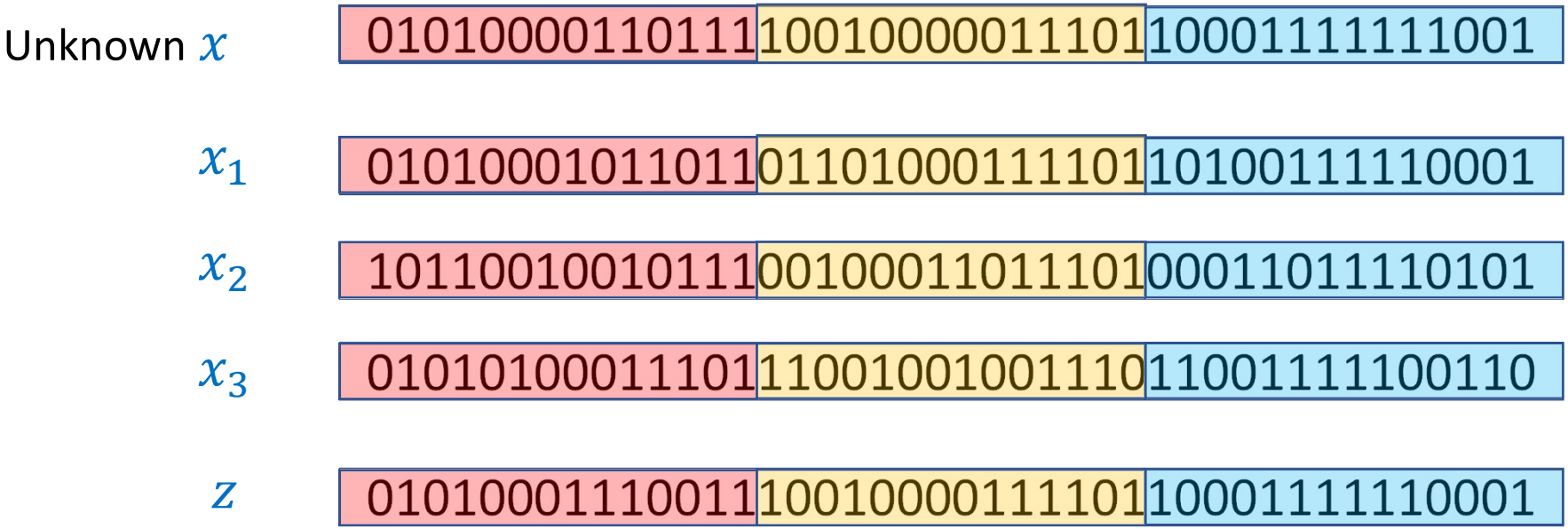
- So, we can reconstruct the unknown string x approximately in time $O(n^3)$ time

Achieving Near-linear Time

Main Idea:

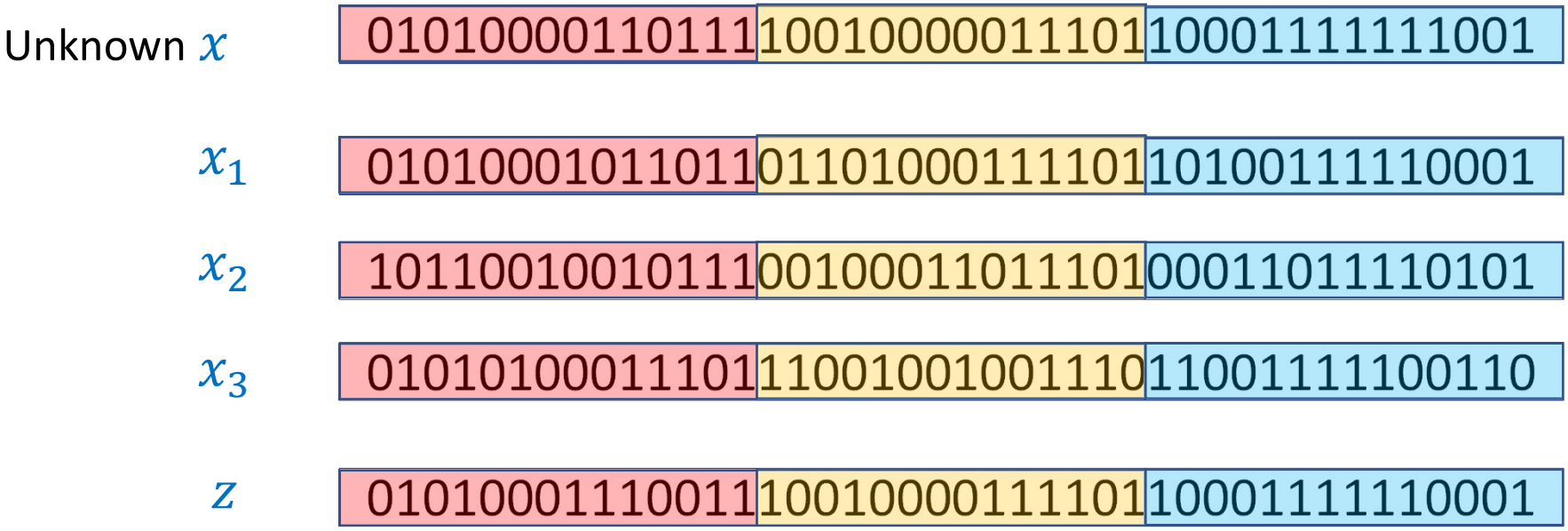
- Divide the input strings into small blocks,
- Then compute medians for those blocks independently, and
- Finally, concatenate those block medians

Achieving Near-linear Time (Intuition)



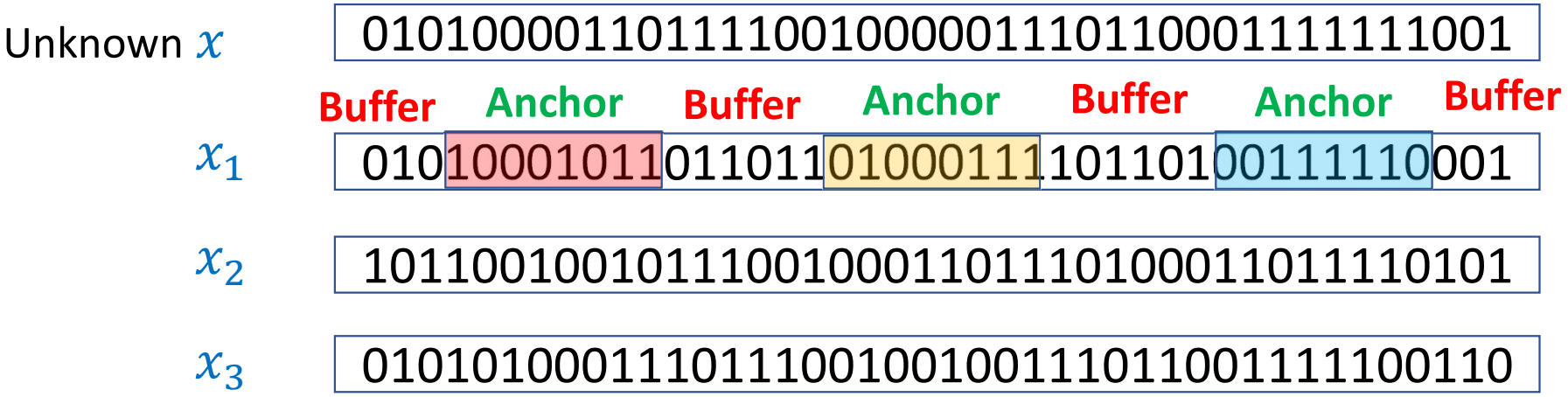
- Block-wise x and z are close (by Result 2), and thus as a whole they are close too

Achieving Near-linear Time (Intuition)



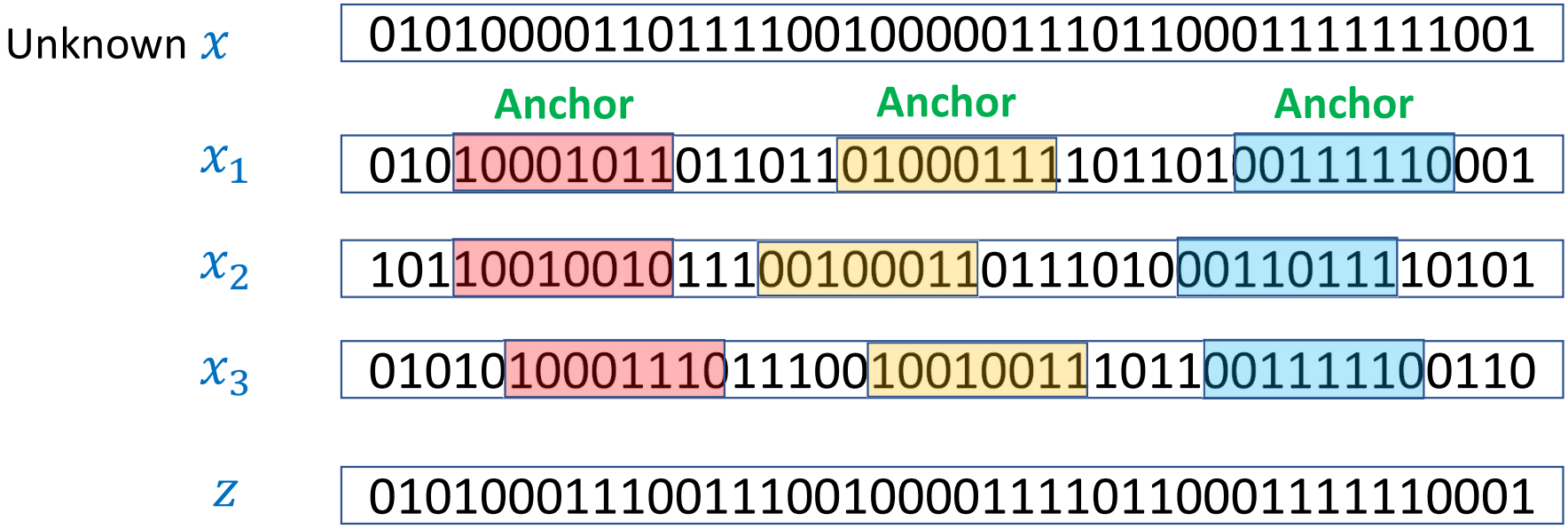
- **Challenges:** Induced partitions on x_1, x_2, x_3 may not be even. How do we identify them?

Achieving Near-linear Time



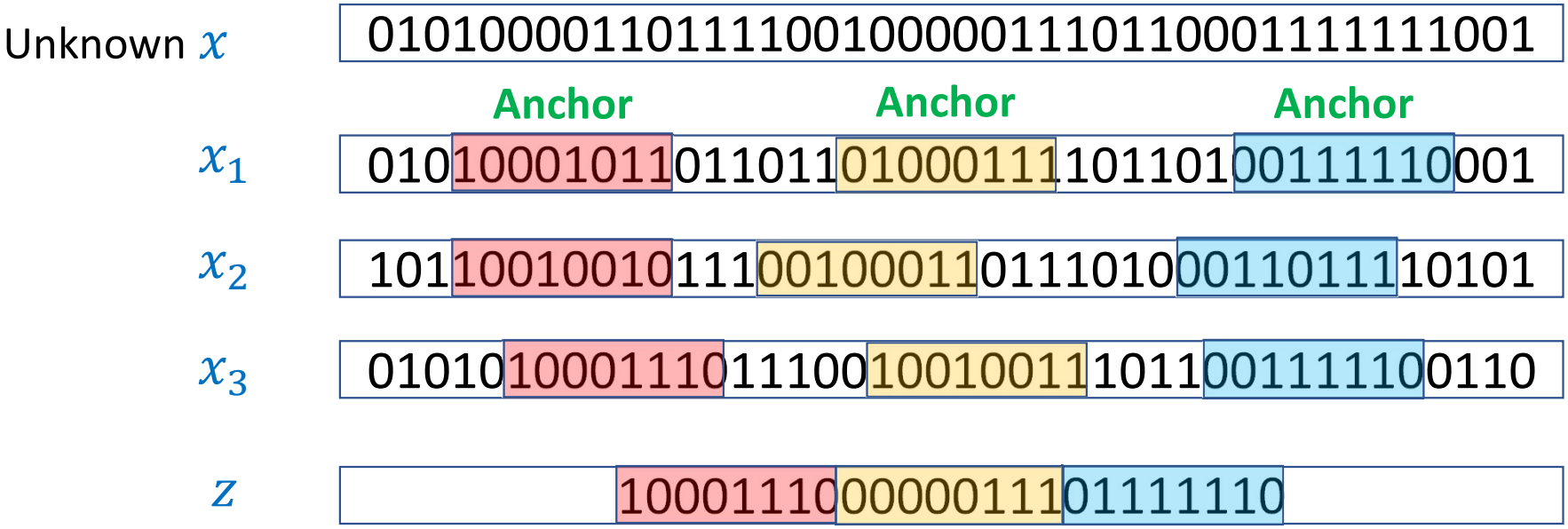
- Take **anchors** of size $\log^2 n$ in x_1

Achieving Near-linear Time



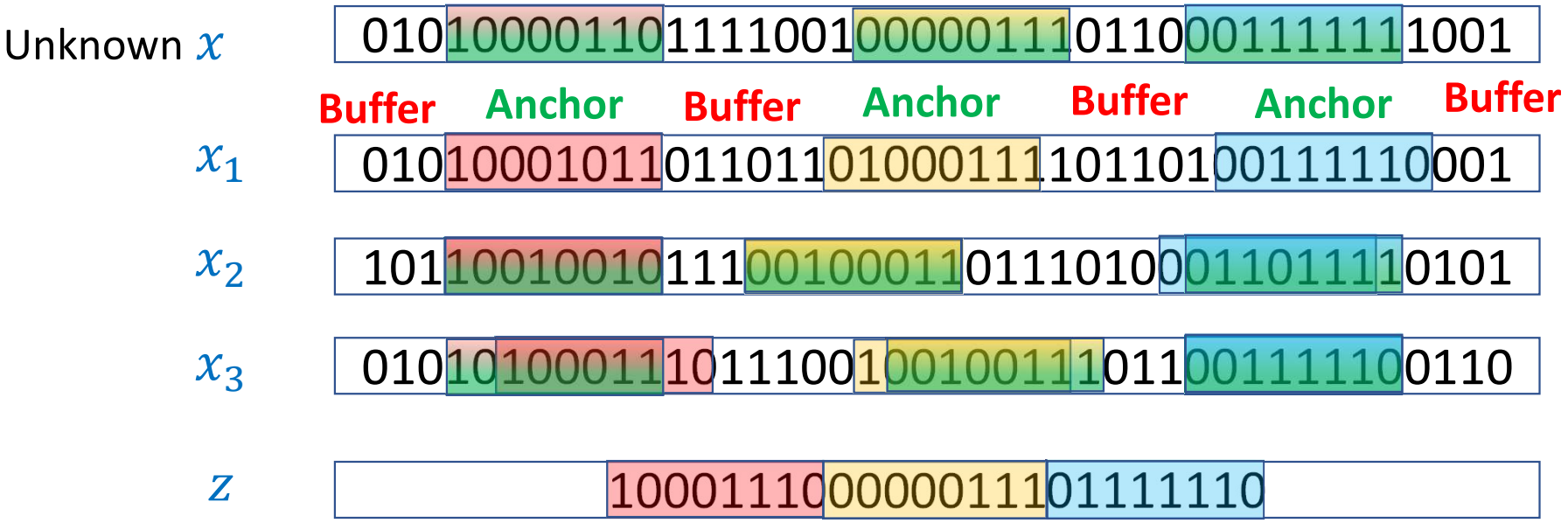
- Find the "**best match**" of the **anchors** in x_2, x_3

Achieving Near-linear Time



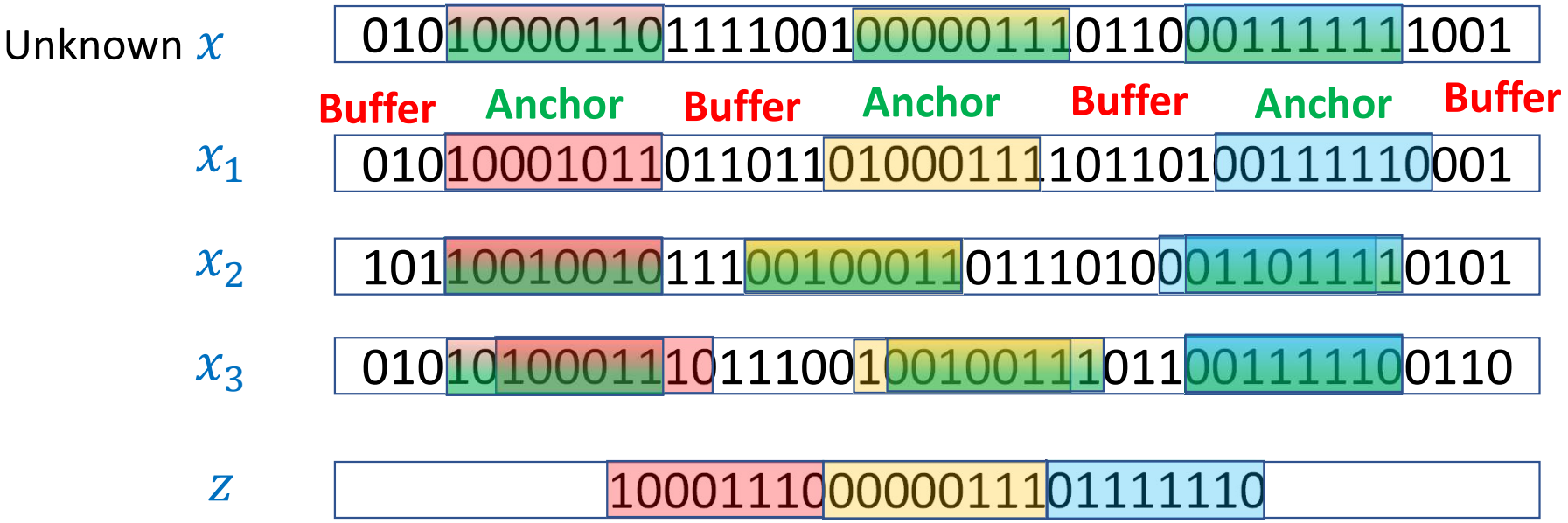
- Find an exact median of "**best match**" blocks of x_1, x_2, x_3, and concatenate them to form a string z

Achieving Near-linear Time (Correctness)



- For each **anchor**, look into its corresponding block in x
- Consider their "**true matches**" in x_2, x_3

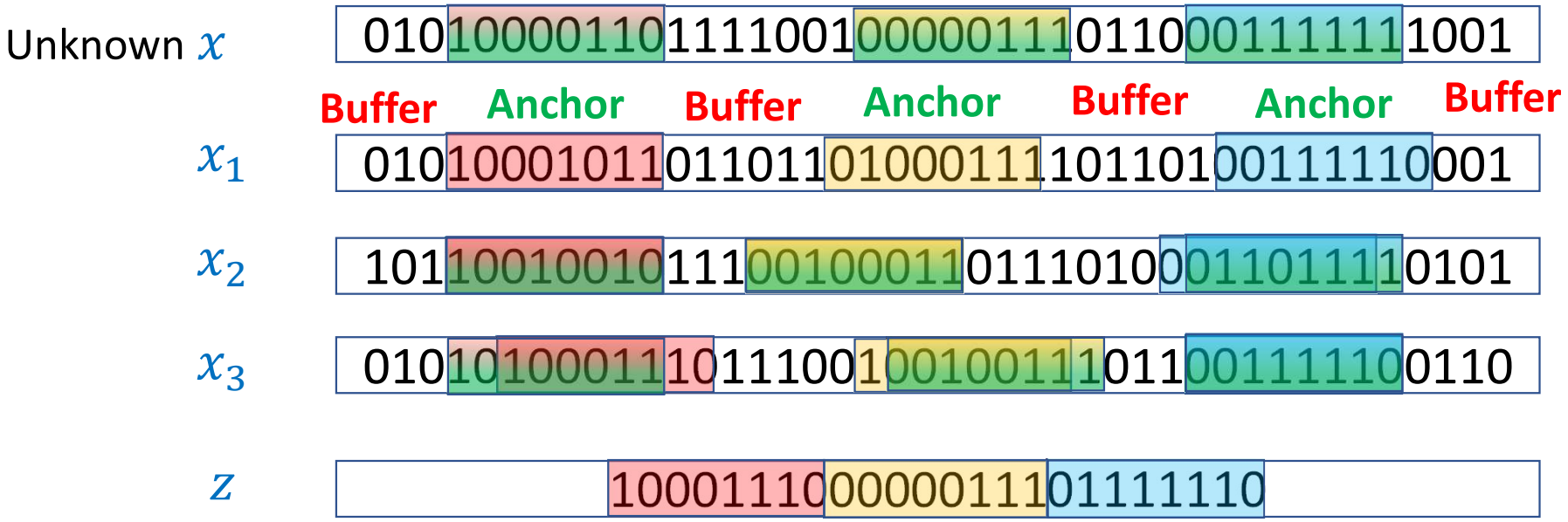
Achieving Near-linear Time (Correctness)



- **Claim:** For each **anchor**, **true matches** are close to the **best matches**

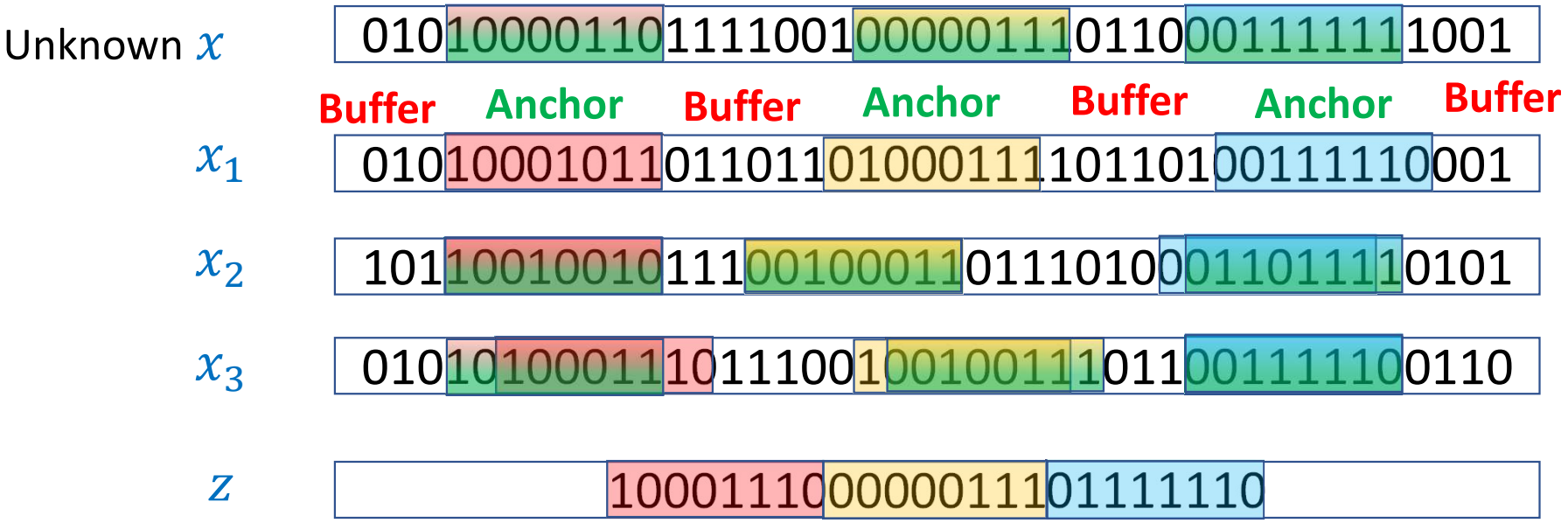
(It is crucial that x was chosen uniformly at random)

Achieving Near-linear Time (Correctness)



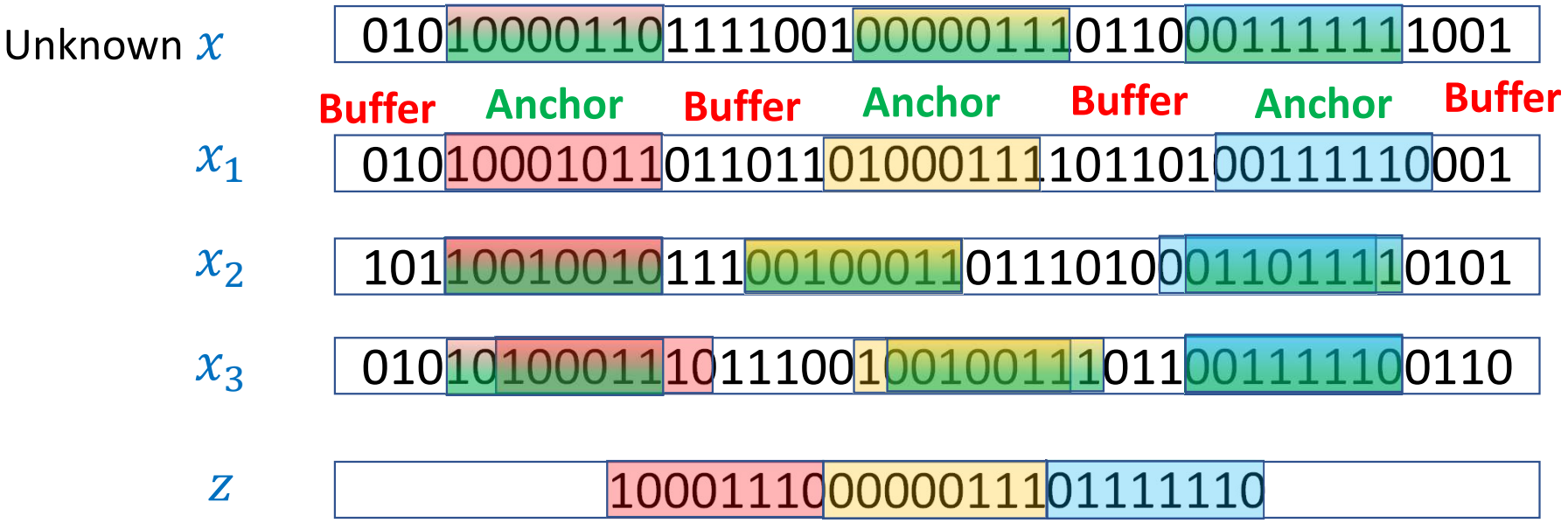
- So, finding median of the **best match blocks** is “almost same” as the finding median of the **true match blocks**

Achieving Near-linear Time (Correctness)



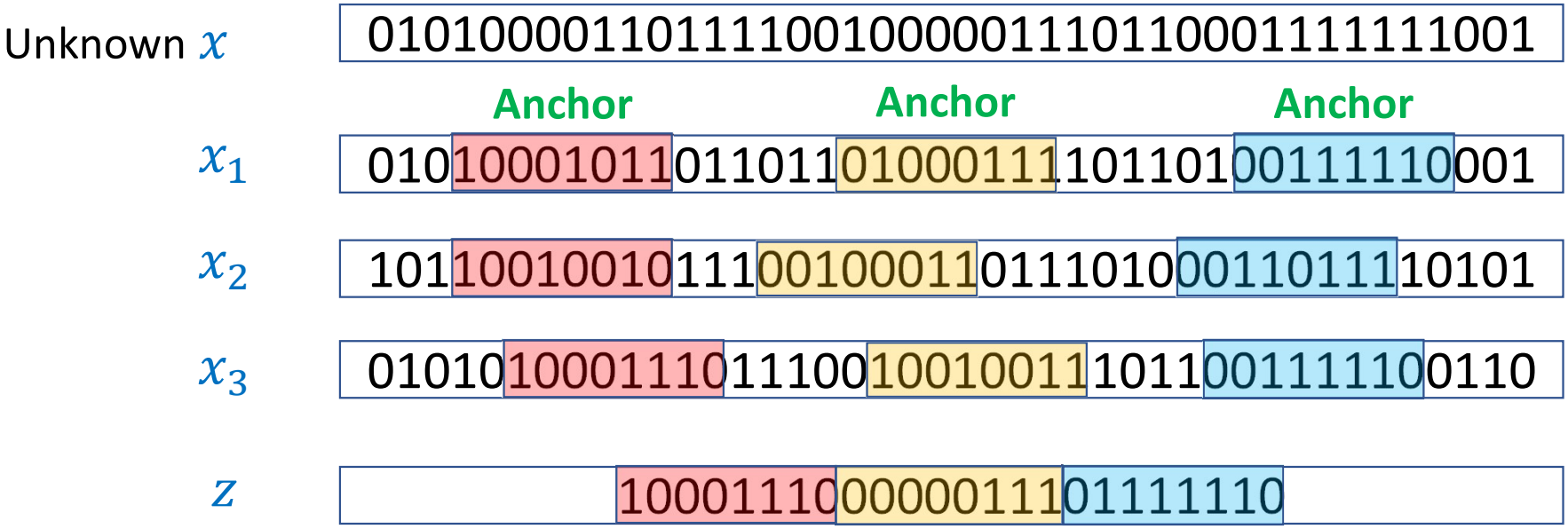
- **Buffer** (of size $\omega(\log n)$) helps to keep the best/true matched blocks well-separated

Achieving Near-linear Time (Correctness)



- Now, apply Result 2 block-wise
- Keep the buffer size much smaller compared to the anchors

Achieving Near-linear Time (Running time)



- There are $\frac{n}{\text{poly}\log n}$ **anchors**, and for each block-median computation takes $\text{poly}\log n$ time.

Our Result 1 (Recall)

Recall, unknown string = x of length n, and noise parameter = p

Only using *three* traces, we can recover a string z such that

$$\Pr\left[ED(x,z) \leq O\left(p^2 \log\frac{1}{p}\right)n\right] \geq 1 - \frac{1}{n}$$

in time $\tilde{O}(n)$ time (using a deterministic algorithm).

Our Result 2 (Recall)

- Recall, in the average-case the unknown string x is **chosen uniformly at random**
- Let $\bar{y}$ be an $(1+\epsilon)$-approximate median of $x_1, x_2, \cdots, x_m$ (generated from x by the **insertion-deletion** channel), for $\epsilon \in \left[110\ p \log\frac{1}{p}, \frac{1}{6}\right]$
- Then w.h.p. $ED(x, \bar{y}) \leq O(\epsilon)\frac{OPT}{m}$

Uniqueness of Alignment

$(1+\epsilon)$-approximate

- **Claim:** A near-optimal alignment (optimal set of edit operations) between x and any x_i is "almost" the same as that induced by the insertion-deletion noise channel

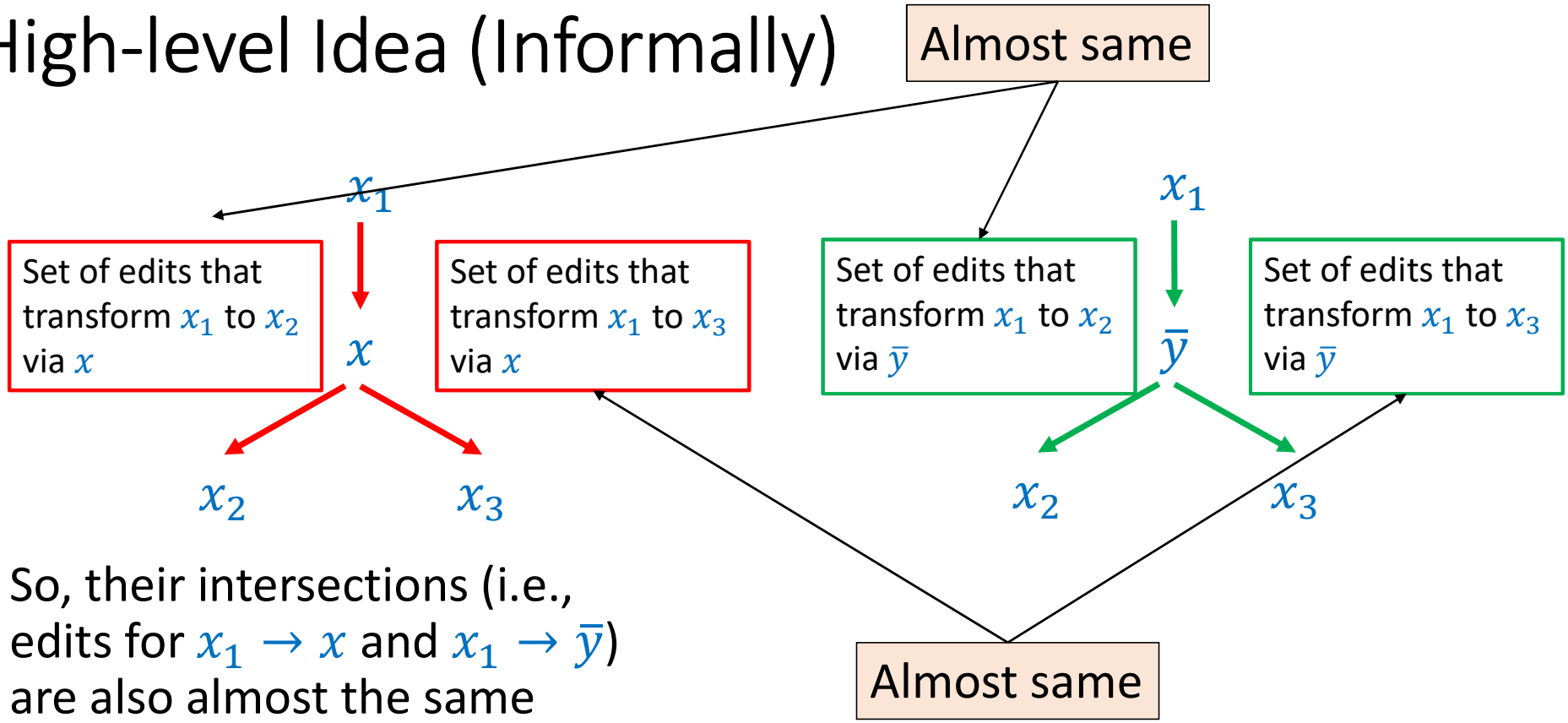
Can view as x_j generating from x_i with a higher noise rate ($\sim 2p - \Theta(p^2)$)

- **Corollary:** A near-optimal alignment between x_i and x_j is also "almost" the same as that induced by the insertion-deletion noise channel

Robustness of Approximate Median

- Let $\bar{y}$ be a $(1+\epsilon)$-approximate median
- Consider the set of edit operations that transforms $x_i \rightarrow \bar{y} \rightarrow x_j$
- We can prove that the **cost of the above alignment** (i.e., the number of edit operations) is **near-optimal** ($(1+O(\epsilon))$-approximation)
- Thus, this alignment is almost the same as that induced by the noise channel

High-level Idea (Informally)



- So, their intersections (i.e., edits for $x_1 \rightarrow x$ and $x_1 \rightarrow \bar{y}$) are also almost the same
- Hence, x and $\bar{y}$ are very similar (close in edit distance)

Concurrent Works

- For deletion channel (with noise rate p):
  - **Upper bound:** Given any $M \leq \Theta(1/p)$ traces, we can reconstruct up to edit distance $n \cdot (p\ M)^{\Omega(M)}$ in polynomial time
  - **Information-theoretic lower bound:** Given any $M \leq \Theta(1/p)$ traces, it is not possible to reconstruct up to edit distance better than $n \cdot (p\ M)^{O(M)}$ irrespective of the running time

  [Chen, De, Lee, Servedio, Sinha '21]

  - **Upper Bound:** $\exp\left(O_p\left(\left(\log\frac{1}{\epsilon}\right)^{\frac{1}{3}}\right)\right)$ traces suffices to reconstruct up to edit distance ϵn [Chase, Peres '21]

Open Problems

- Can we reduce the error by using more samples?
- **Conjecture:** $ED(x, \bar{y})$ could be reduced to $O(\epsilon n)$, for arbitrary small $\epsilon > 0$, using $poly(1/\epsilon)$ traces.

- Can we get a similar result for the worst-case trace reconstruction (i.e., when the unknown string is arbitrary)?

Thank You!