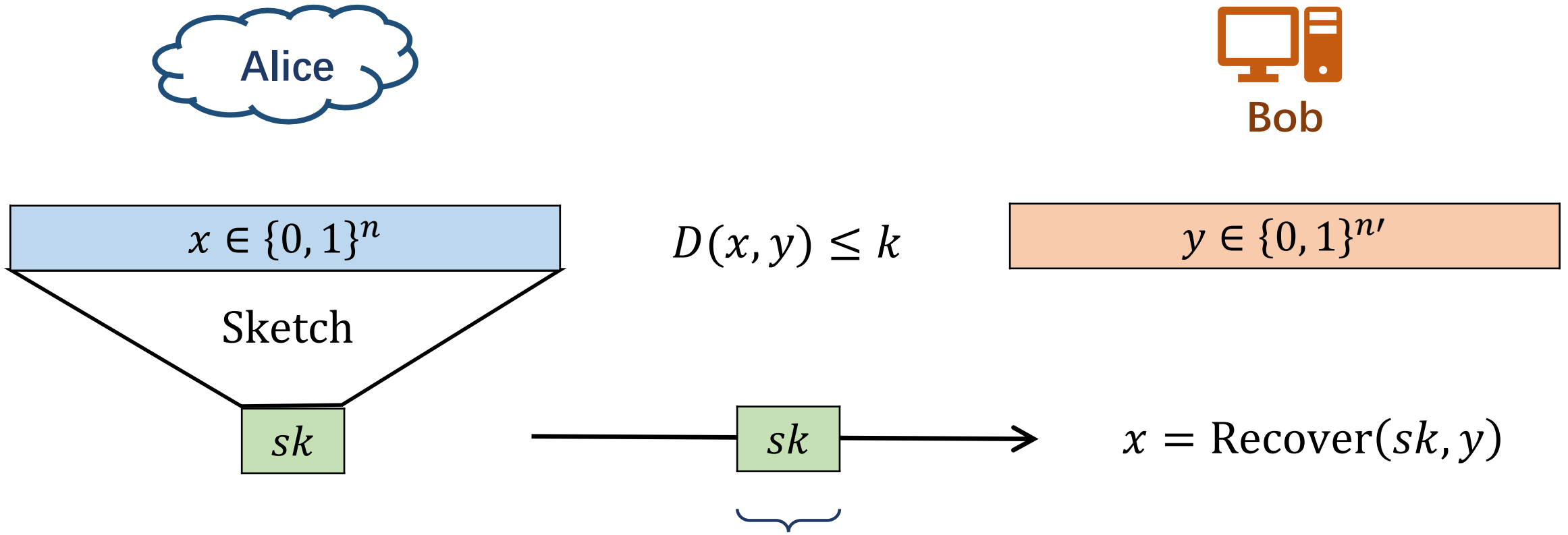
Block Edit Errors with Transpositions: Deterministic Document Exchange Protocols and Almost Optimal Binary Codes

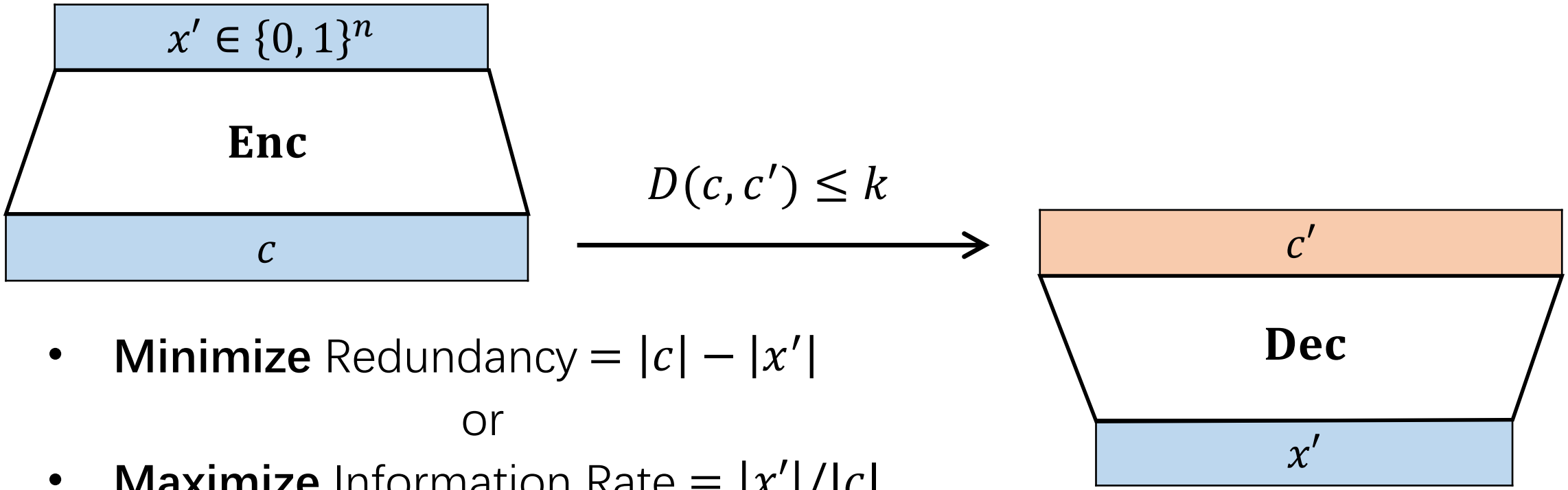
Kuan Cheng, **Zhengzhong Jin**, Xin Li, Ke Wu

Document Exchange Protocol



Minimize Sketch Size

Error Correcting Code



- **Minimize** Redundancy $= |c| - |x'|$

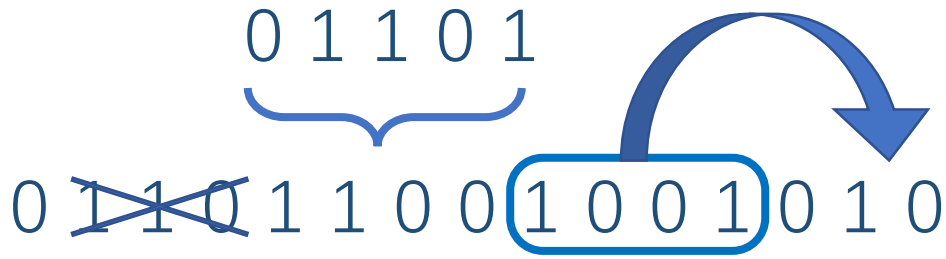
or

- **Maximize** Information Rate $= |x'|/|c|$

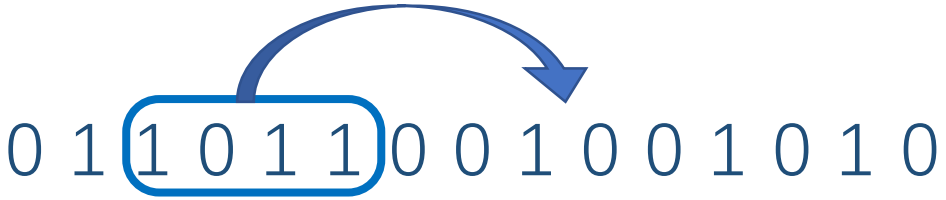
Background

Block Edit Errors with Transpositions

1. Burst of Insertion/Deletion
2. Block Transposition



- k : the total number of insertions, deletions, and **transpositions**
- t : the total number of bits inserted and deleted



- A strict generalization of edit error
- 1 block transposition may cause a lot of edit errors

We focus on binary alphabet.

Applications & Related Problems

- Applications:
  - Document Exchange: file synchronization, etc.
  - Error Correcting Code: protect data from corruptions.

- Related problems:
  - Metric embeddings: Edit distance, Ulam distance, etc.
  - Approximating Edit distance (with transpositions)

Previous Results

- **Document Exchange**
  - For **Edit Errors**:

    Exponential time protocol $O(k \log \frac{n}{k})$ (asymptotically optimal) [Orlitsky 91]

    Randomized Protocols $O(k \log^2 n)$ [Irmak, Mihaylov, Suel 05]

    $O(k(\log^2 k + \log n))$ [Belazzougui, Zhang 17]

    $O(k \log \frac{n}{k})$ (asymptotically optimal) [Haeupler 18]

    Deterministic Protocols $O(k^2 + k \log^2 n)$ [Belazzougui 15]

    $O(k \log^2(\frac{n}{k}))$ [Cheng,Jin,Li,Wu 18][Haeupler 18]

  - For **Edit Errors with Transpositions:**

    Randomized Protocols $O(k \log^2 n)$ [Irmak, Mihaylov, Suel 05]

    Deterministic Protocol: **Not known before**

Previous Results

- **Error Correcting Code**
  - For **Edit Errors:**

    Asymptotically Good Code [Schulman, Zuckerman 97]

    In terms of rate : $1 - \tilde{O}(\sqrt{k/n})$ [Guruswami, Li 16] [Guruswami, Wang 17] [Haeupler and Shahrasbi 18]

    $1 - \tilde{O}(k \,/\, n)$ [Cheng, Jin, Li, Wu 18][Haeupler 18]

    Small error regime, in terms of redundancy:

    $O(k^2 \log k \log n)$ for $k = O(1)$ [Brakensiek, Guruswami, Zbarsky 17]

    $O(k \log n)$ [Cheng, Jin, Li, Wu 18]
  - For **Edit Errors with Transpositions** :

    Asymptotically Good Code [Schulman, Zuckerman 97]

Our Results

- **Document Exchange**

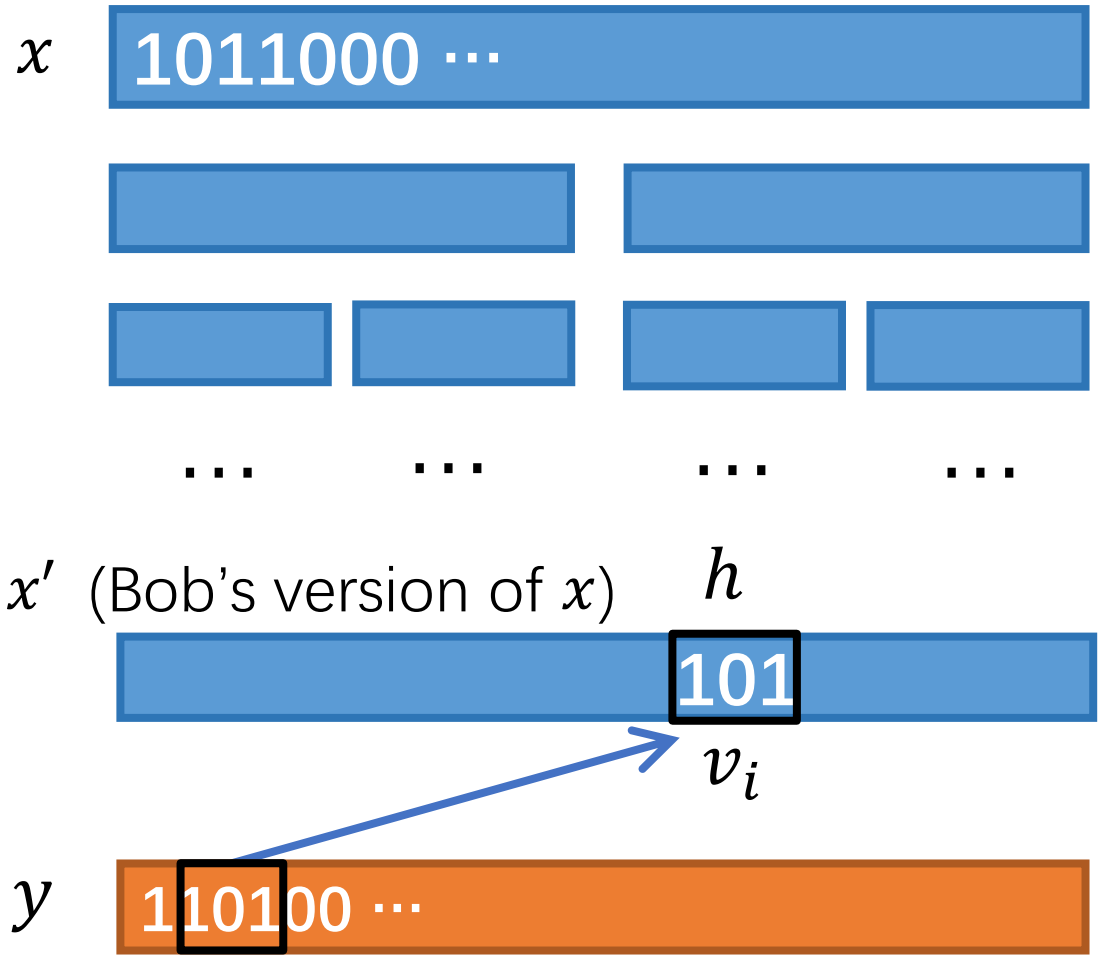
  Explicit deterministic protocol, sketch size $O\left((k \log n + t) \log^2 \frac{n}{k \log n + t}\right)$

- **Error Correcting Code**

  Explicit construction with redundancy $O(k \log n \log \log \log n + t)$

- **Information Theoretic Optimum** : $O(k \log n + t)$

Document Exchange Protocol: Overview
(Adapt from FOCS18)



Alice: $O(\log n)$ levels. In each level, divide each block evenly into 2. Hash each block with random function Send $sk(\text{hash value}), \text{hash function}$

Bob: $O(\log n)$ levels, in each level,
1. Recover hash values from sk
2. Use hashes to find maximum matching

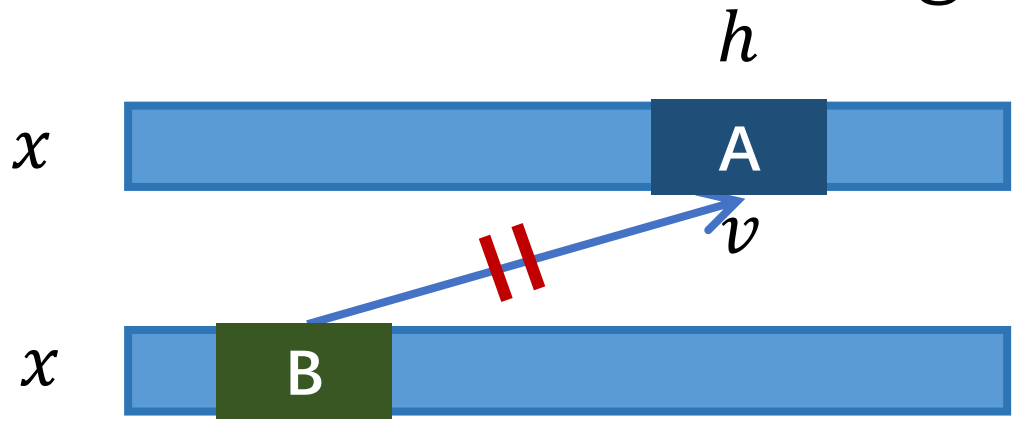
Hash Functions: Derandomization & Matching

- Collison Free Hash Functions

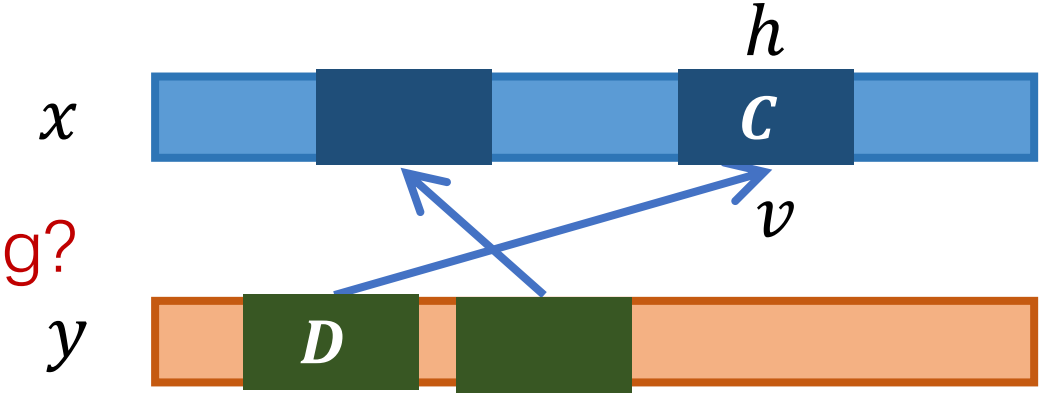
  For any two substrings of x: A $\neq$ B, $h($A$) \neq h($B$)$.

- Derandomization:

  Almost k-wise independence (seed length $O(\log n)$)



- **Question:** How does Bob find a maximum ***non-overlapping,*** (possibly) ***non-monotone*** matching?
- Exact solution? Seams Hard ☹
- Approximation? √☺

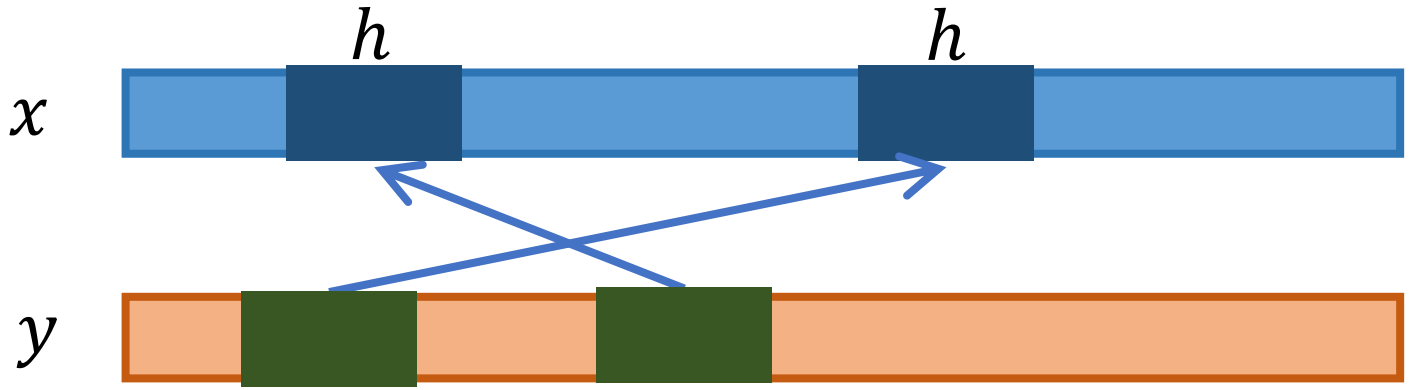


C and D are matched iff $h(C) = h(D)$

Bob's Matching Algorithm (Approximation)

- **First Attempt** A Greedy Matching Algorithm

  For each *unmatched block*, match it to the block (non-overlapping with existing matchings) with the same hash value, until cannot do this any more.



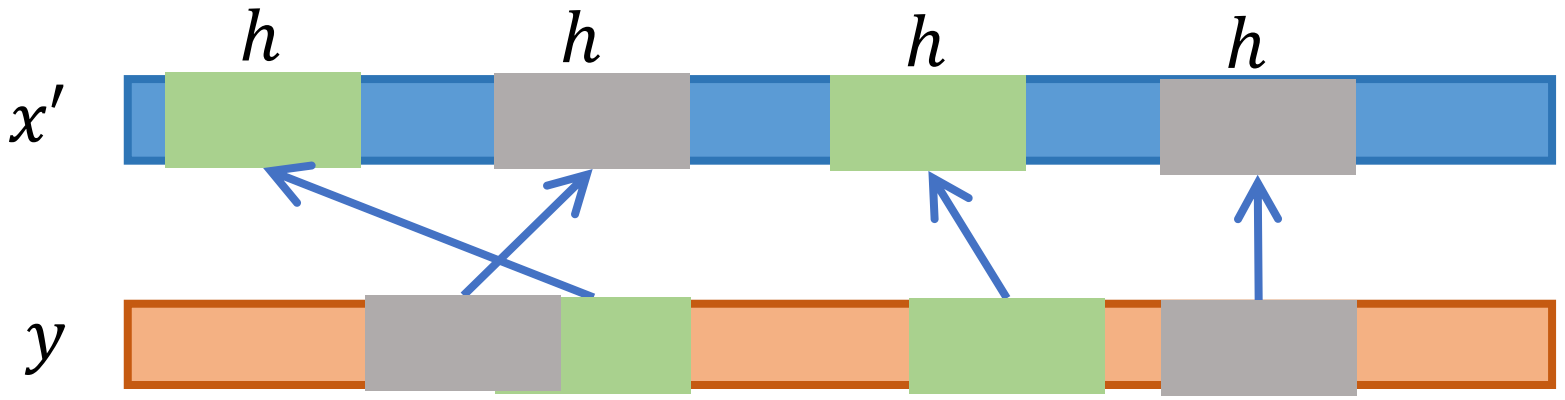
- < 2/3 fraction of *new matches* may still be *incorrect.*
- 2/3 is not enough, since each level we divide each block into 2, we need <1/2.

Bob's Matching Algorithm (Approximation)

Approximating Matching

Idea: Allow some overlapping to approximate the optimal matching

Apply *Greedy Matching* 3 times to unmatched blocks in x', reuse y.



- Some matched blocks in y may be overlapped for $O(1)$ times.

Error Correcting Code

Asymptotically Good Code for Edit Errors [Schulman and Zuckerman 97]

- Initial Idea:

$$\mathrm{Enc}(x') \stackrel{\mathrm{def}}{=} x' \,\|\, \mathrm{Enc}'(sk(x'))$$

- A Better Approach with Smaller Sketch: (adapt from FOCS'18)

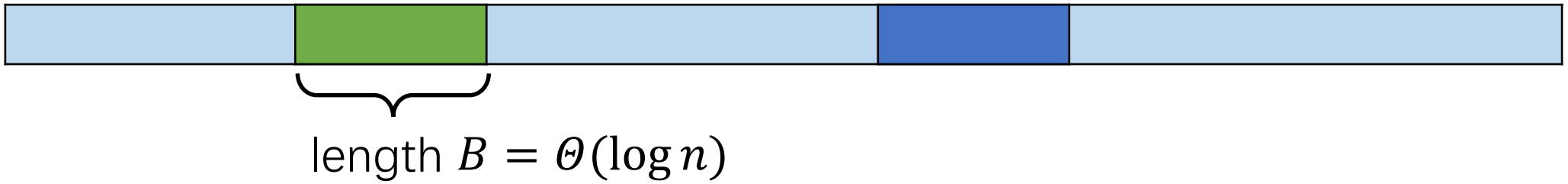
$$x' \oplus \mathrm{PRG}(r) = x$$

B-distinct Property

$$\mathrm{Enc}(x') \stackrel{\mathrm{def}}{=} x \,\|\, \mathrm{Enc}'(sk(x) \,\|\, r)$$

Document Exchange of B-distinct String

- B-distinct string: every pair of substrings of length B are different



- **Key Idea** : The content of the B-substrings serves as a kind of `index'.
- **Stage I :** *Partition* the string based on its content

  Send a sketch of the partition.

- **Stage II :** Apply the multi-level doc-exchange.

Partition B-distinct String

- The partition must based on `local' content of the string.
- [Cormode, Muthukrishnan 07] : *Edit Sensitive Parsing Tree*

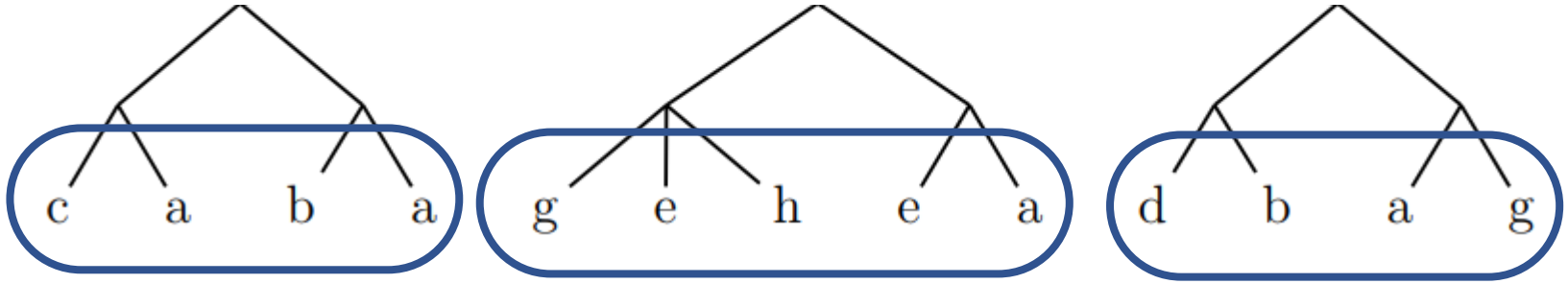


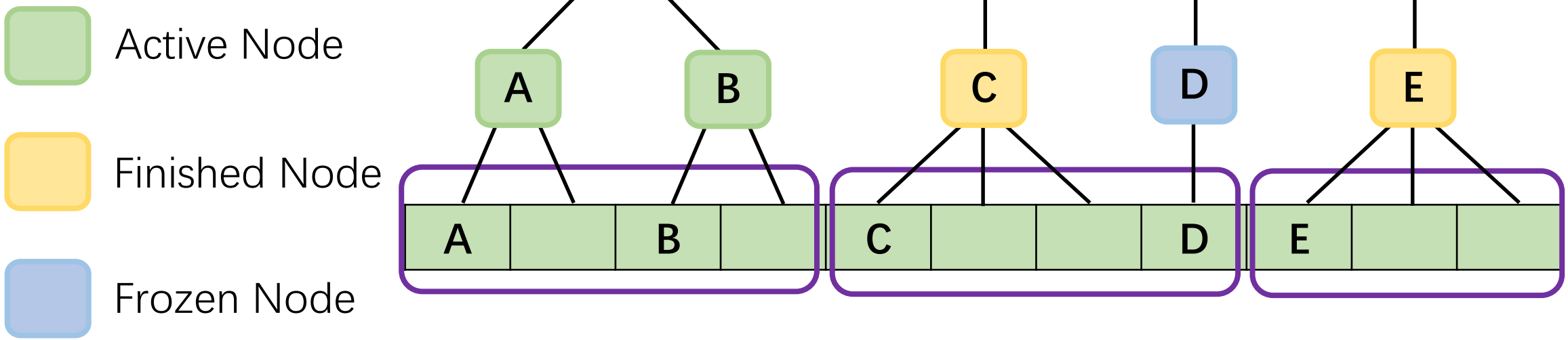
Figure copied from [Cormode, Muthukrishnan 07]

- Only partially construct the parsing tree
- Each tree is a block.

Partial Parsing Tree

Frozen Node: neighbors are all `finished'.

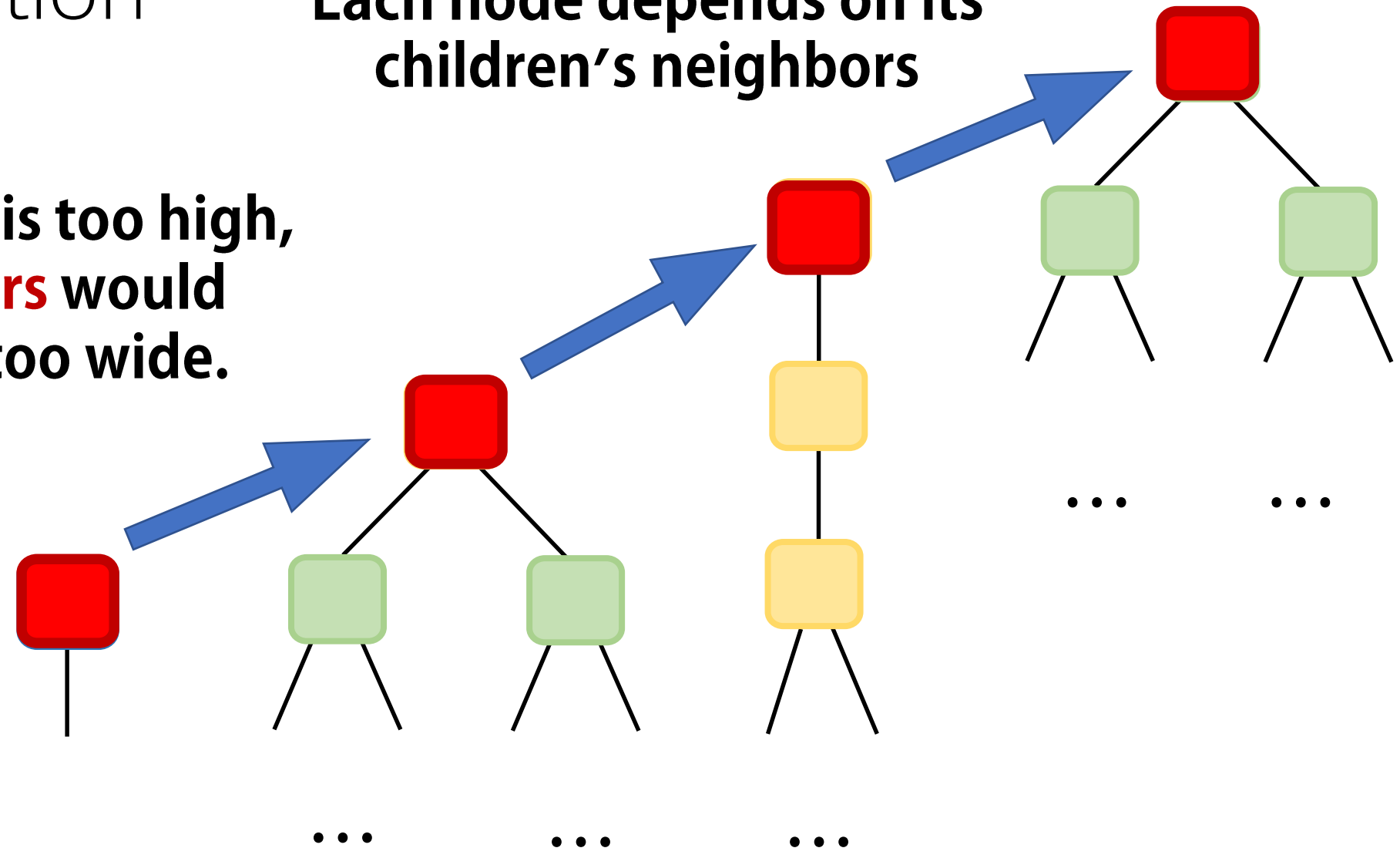
Finished Node: size ≥ a threshold T



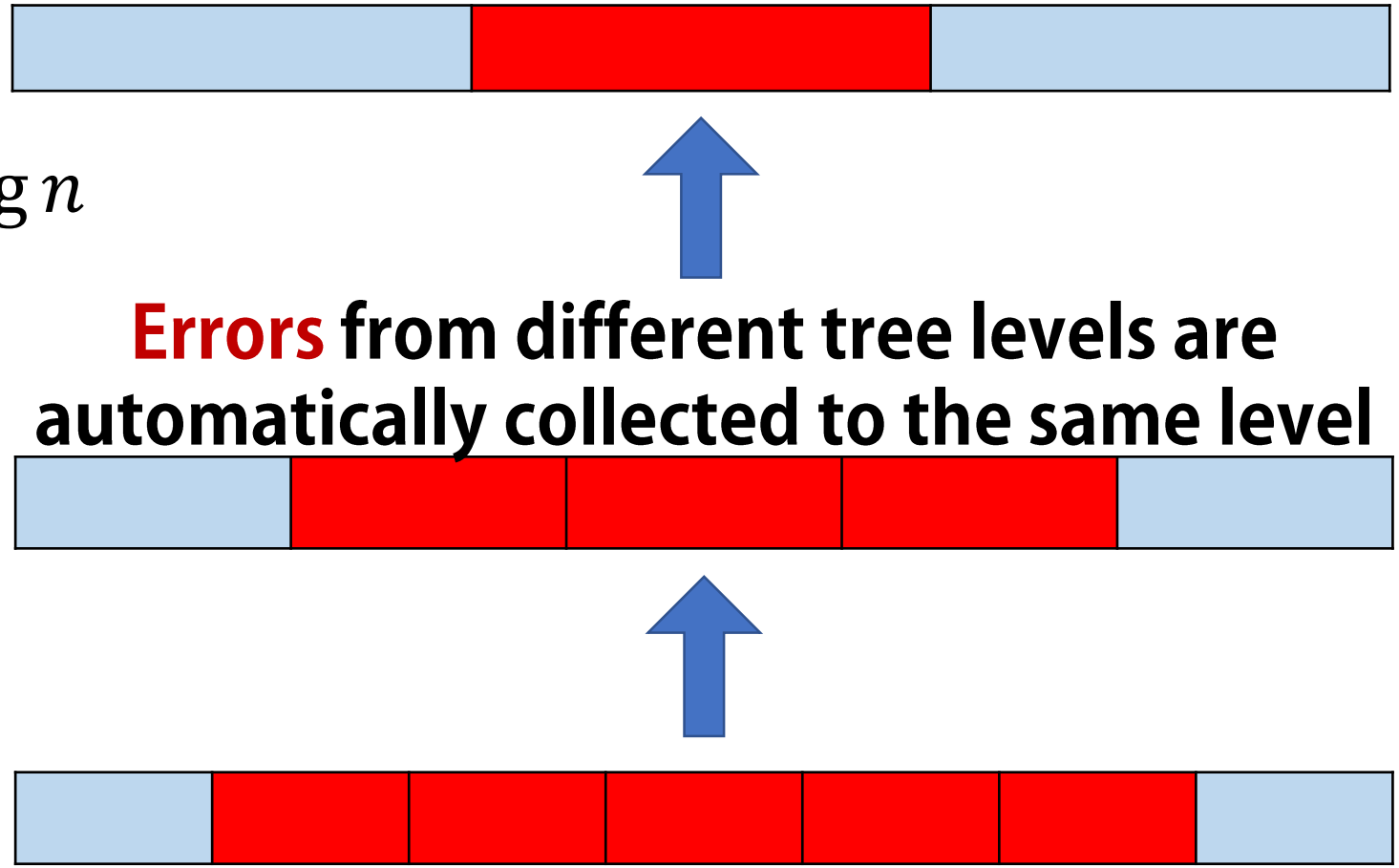
Observation

Each node depends on its children's neighbors

If the tree is too high, the errors would spread too wide.



Two-Stage Partition



Future Direction

- How to explicitly construct optimal document exchange protocols and error correcting codes for block edit errors & transpositions?

Thank you!

Ευχαριστώ