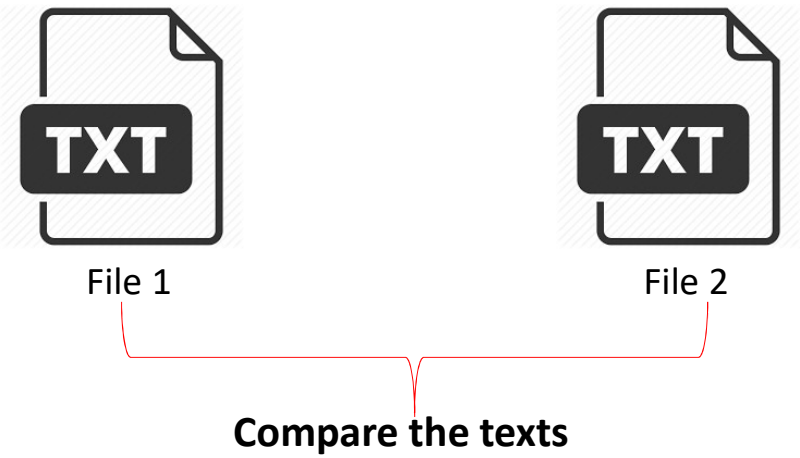
# Aggregating a Data Set: Rankings to Strings

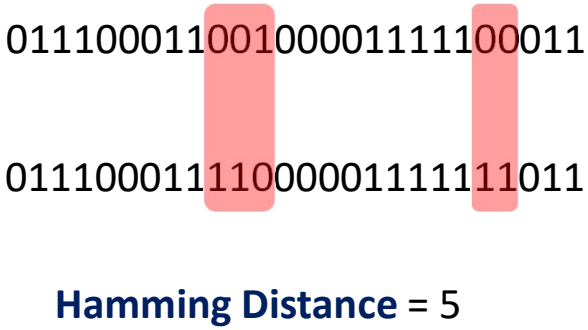
Diptarka Chakraborty

(National University of Singapore)

**Recent Trends in Algorithms 2022**

# String Similarity



# String Similarity


File 1


File 2

Hey Diptarka! How is your life at NUS?

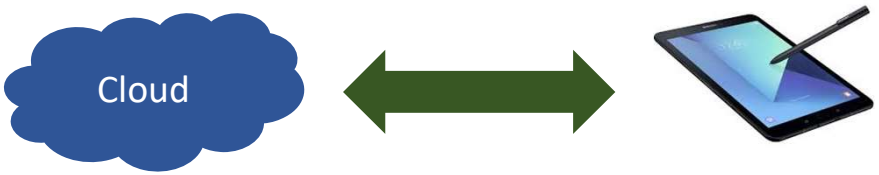
Hey ! How is your life at NUS?

**Compare the texts**

Count min. number of “edit operations”, say deletions, insertions, substitutions

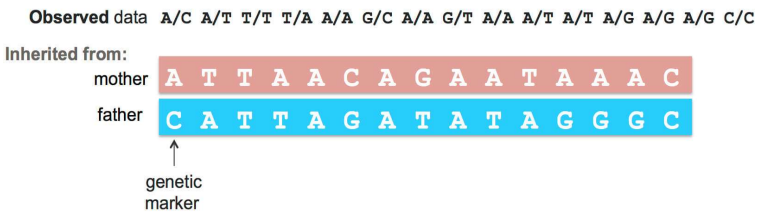
# Applications of Edit Distance

- File synchronization




- NLP (e.g., auto spell-correction)


- Pattern recognition
- Computation biology (DNA matching)

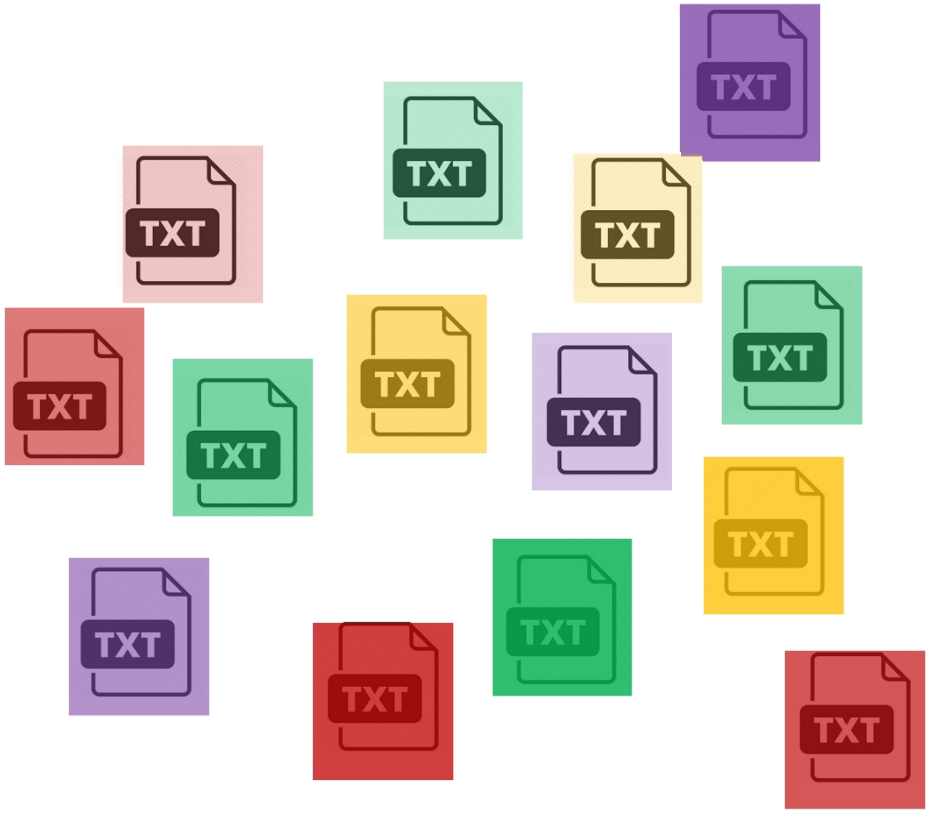



- Database systems
- Many more……

# Computing Edit Distance

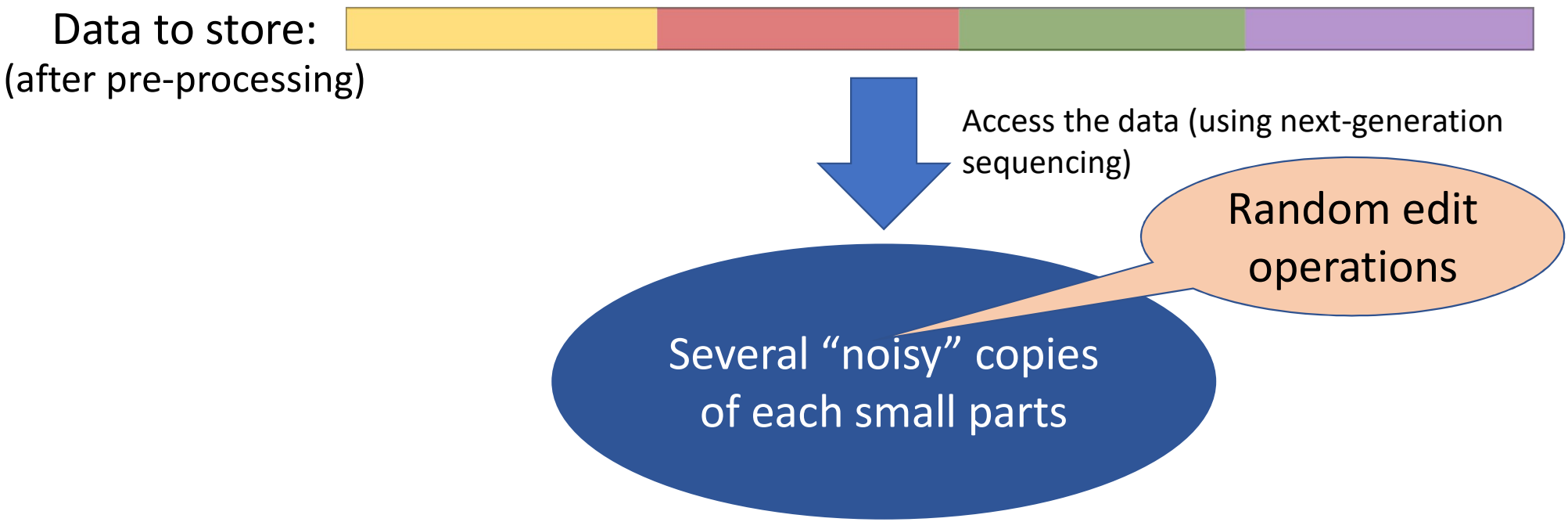
- For two strings basic Dynamic Programming solves in Quadratic time
- Many results on approximating the edit distance

# Clustering




Question 1: Can we partition them efficiently so that “similar” strings are in the same partition?

# One Application: DNA-Storage System



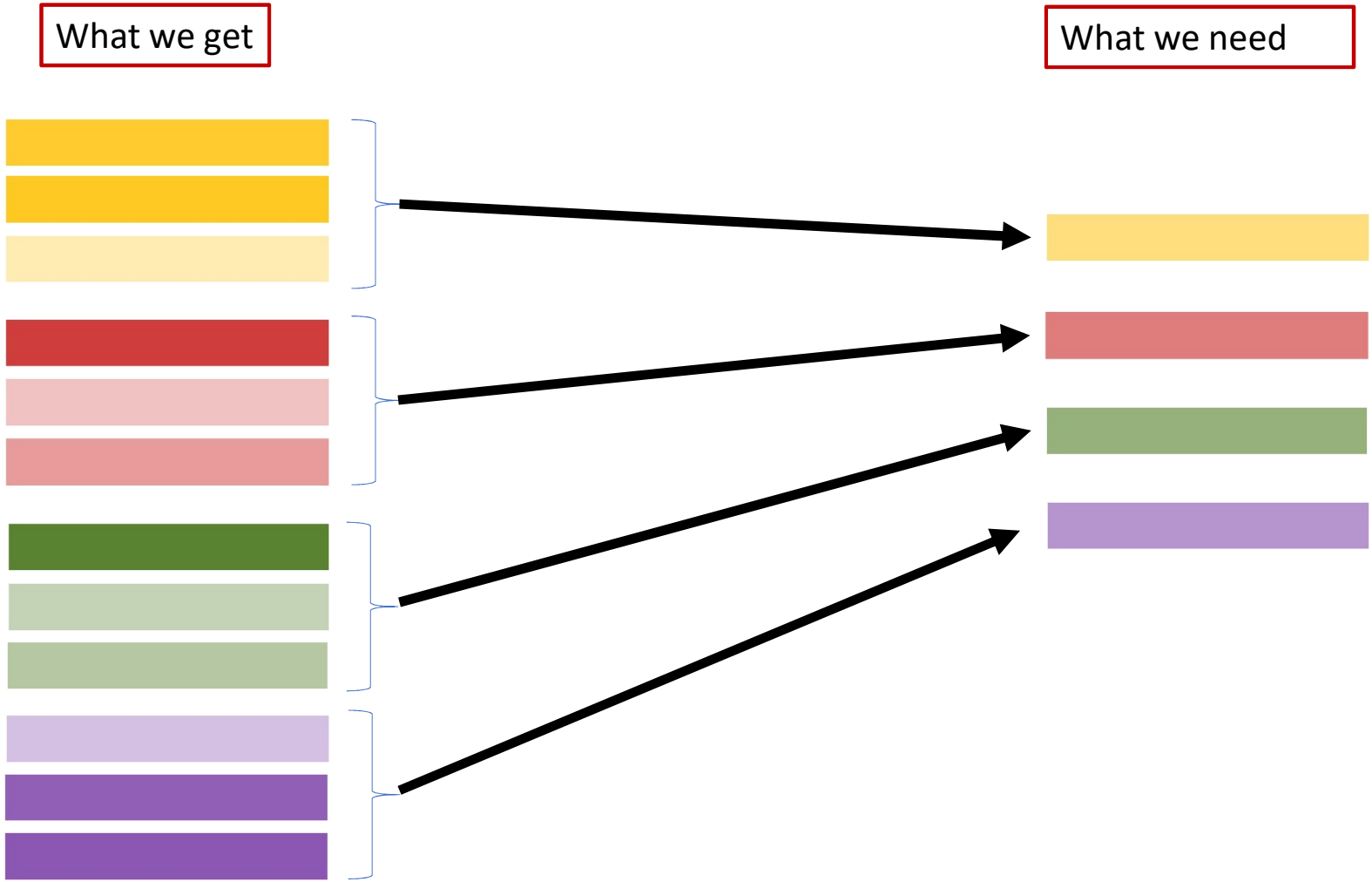
# One Application: DNA-Storage System

# DNA-Storage: Step 1 - Clustering

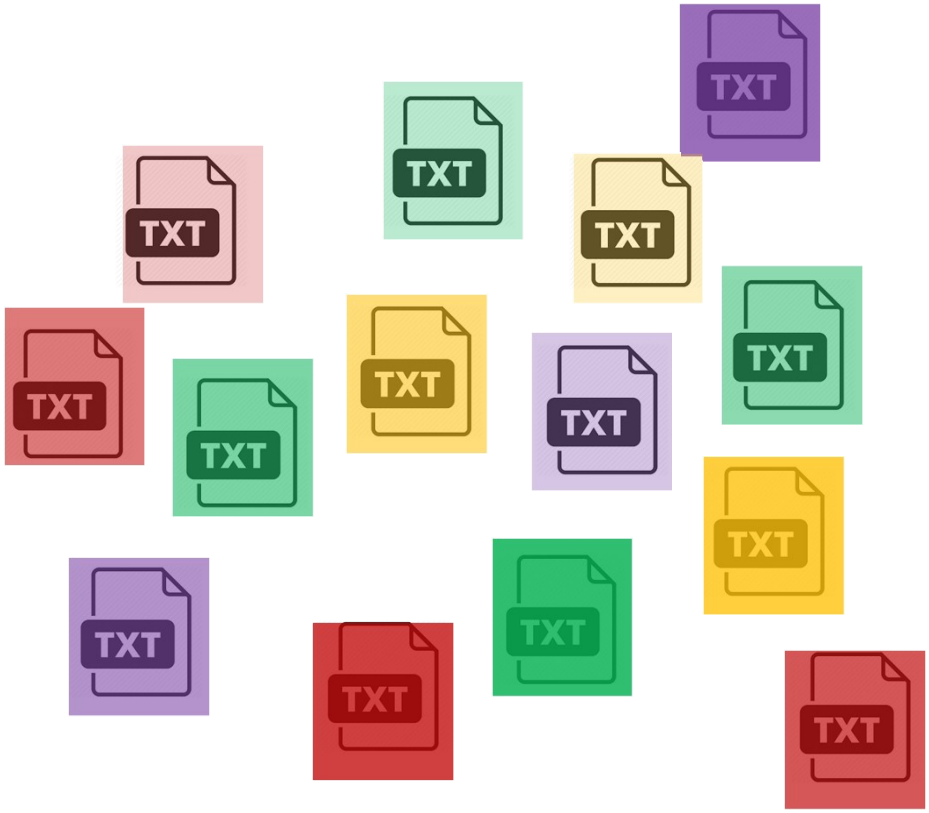
What we get

What we need

# DNA-Storage: Step 1 - Clustering



# Clustering

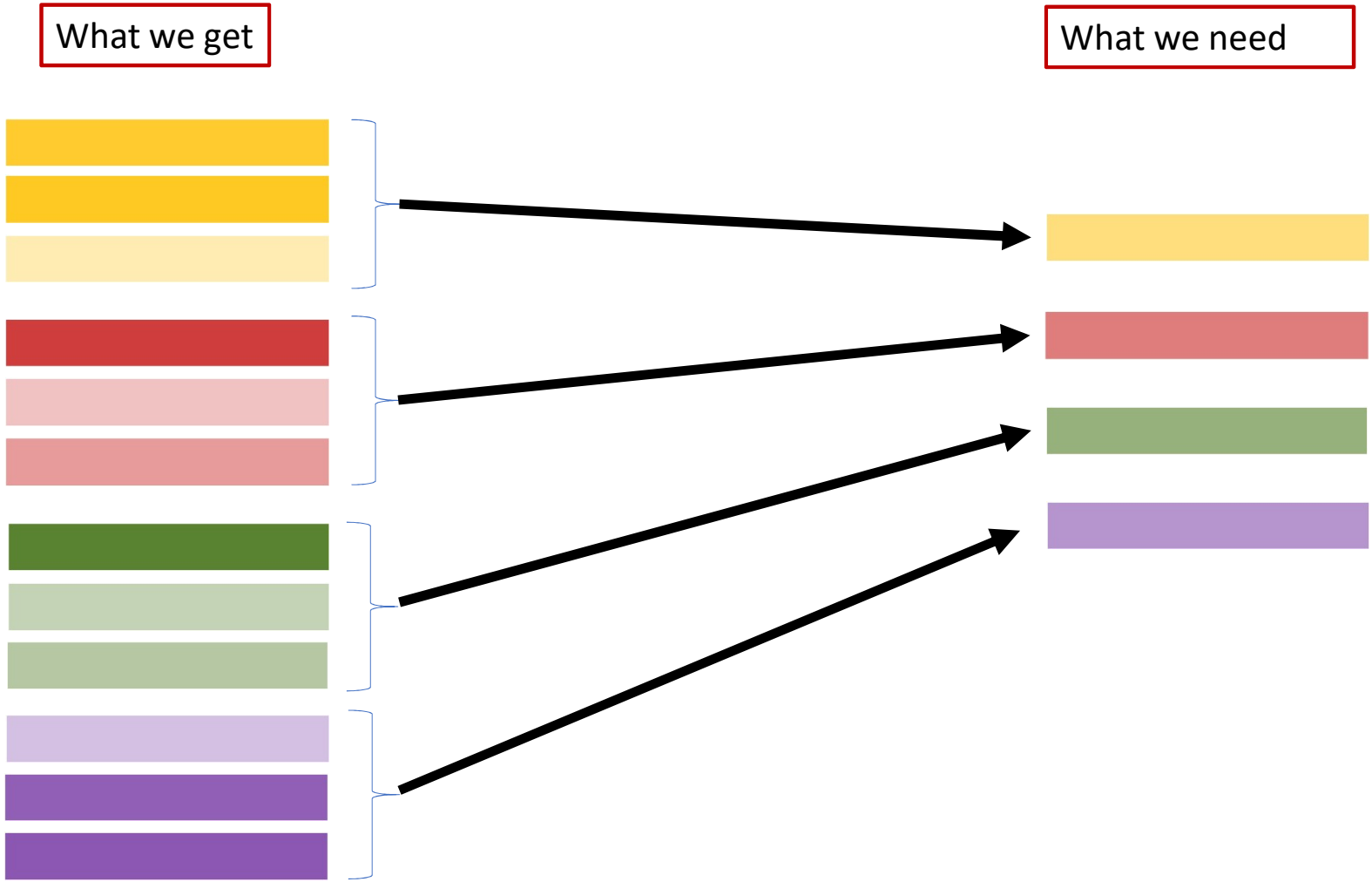



Question 1: Can we partition them efficiently so that “similar” strings are in the same partition?

# Clustering – What is known?

- Nothing much non-trivial is known, except
- One attack on clustering for noisy data [Rashtchian et al.'17]
- **Objective:** For DNA-storage application we need algorithm much faster than $O(n^2)$ time, where $n$ is the data size

# DNA-Storage: Step 2 – How?



# Trace Reconstruction

- 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

$x = 011000101$ → Noise Channel → $x_i = 0100110101$

- **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

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

Roughly suffices for DNA storage application

# What is Known (for exact reconstruction)?

- <u>Worst-case</u>
  - **Upper Bound:** $2^{O(n^{1/5})}$ traces suffices [Chase '21]
  - **Lower Bound:** $\widetilde{\Omega}(n^{3/2})$ traces necessary [Chase '21]
- <u>Average-case</u>
  - **Upper Bound:** $\exp(O(\log^{1/3} n))$ traces suffices ($n^{1+o(1)}$ running time) [Holden, Pemantle, Peres, Zhai '20]
  - **Lower Bound:** $\widetilde{\Omega}(\log^{5/2} n)$ traces necessary [Chase '21]

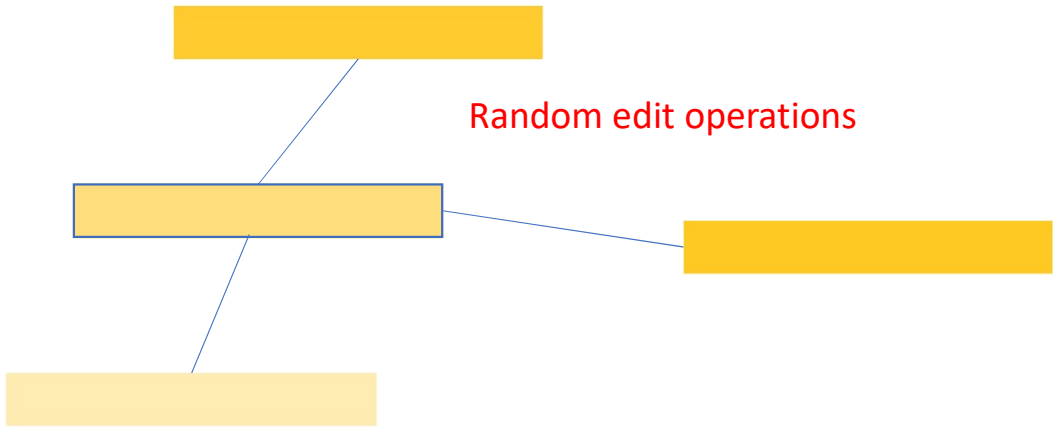
# What about Approximation?

- 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?
Will see tomorrow

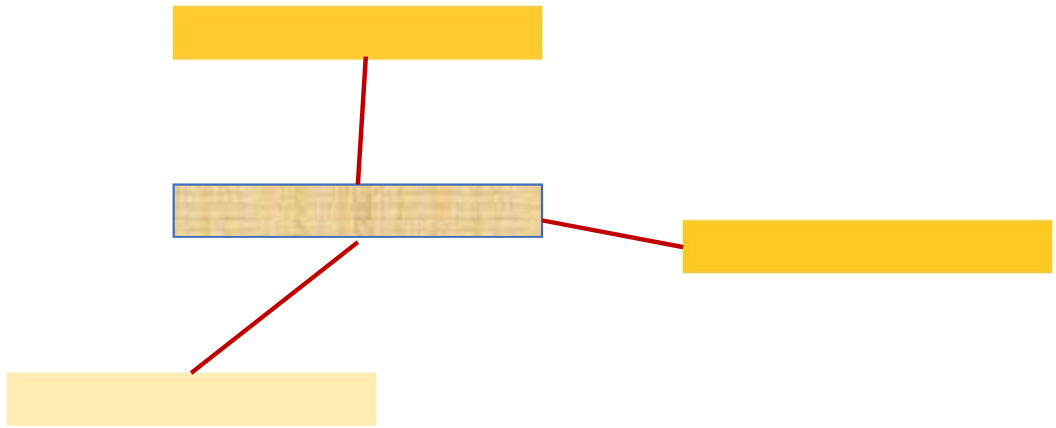
Getting $\sim pn$ edit distance is trivial (any input trace works)

# DNA-Storage: Step 2 – Trace Reconstruction


Proposed Heuristic:
Find a representative

# DNA-Storage: Step 2 – Finding Median

Proposed Heuristic:
Find a representative

**Minimizing the sum of distances**

# Finding a Median String

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

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

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

# Finding a Median String

Can we do better with approximation?

- The problem is NP-hard

- A standard dynamic programming finds a median in time $O(2^m n^m)$, where each of $m$ input strings is of length at most $n$ [Sankoff '75]

- No $O(n^{m-\epsilon})$ time algorithm assuming **S**trong **E**xponential **T**ime **H**ypothesis (**SETH**) [Hoppenworth, Bentley, Gibney, Thankachan '20]

# Approximate Median

- Let $OPT(S) = Obj(S, y^*)$, where $y^*$ is a median string

- A string $\bar{y}$ is a c-approximate median iff $Obj(S, \bar{y}) \leq c \cdot OPT(S)$

# What is the connection?

- Is there any connection between the approximate trace reconstruction and the approximate median problem?

- A common heuristic for trace reconstruction (in practice) is to find a median (or **multi-sequence alignment**)

- To think: Is there any definite connection between these two problems?

Will come back tomorrow

# Questions encountered so far

- 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?

# Approximate Median

- Can we get a constant factor approximation in polytime?

- 2-approximation is easy (Why?)
  - Output the best input (i.e., $z \in S$ with the minimum $Obj(S, z)$)
  - Use <u>triangle inequality</u> (holds for any metric)

# Approximate Median

- Can we get a constant factor approximation in polytime?

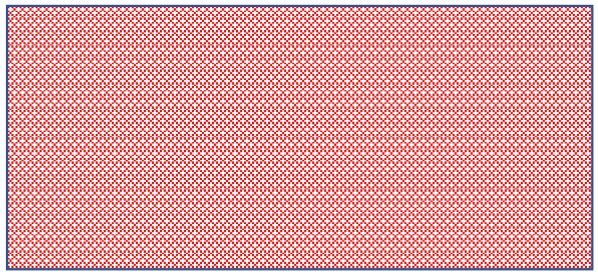
- 2-approximation is easy (Why?)
  - Output the best input (i.e., $z \in S$ with the minimum $Obj(S, z)$)
  - Use <u>triangle inequality</u> (holds for any metric)

- Question: What about PTAS? (Or even breaking below 2?)

# What about Hamming?

- Easy (Why?)

$x_1 =$ 1
$x_2 =$ 0
$x_3 =$ 0
$x_4 =$ 0



$$y^* = 1010001000001000000$$

Output coordinate-wise majority (break ties arbitrarily)

# Permutations – a special case?

- Suppose input strings are permutations over $[n]$ (instead of arbitrary $n$-length strings)

- Consider the edit distance between two permutations (as the min. number of insertions, deletions)

- Known as **Ulam metric**

e.g.

$$x_1 = 785693214$$
$$x_2 = 275693814$$

$$ED(x_1, x_2) = 4$$

# Permutations – a special case? Not Really

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

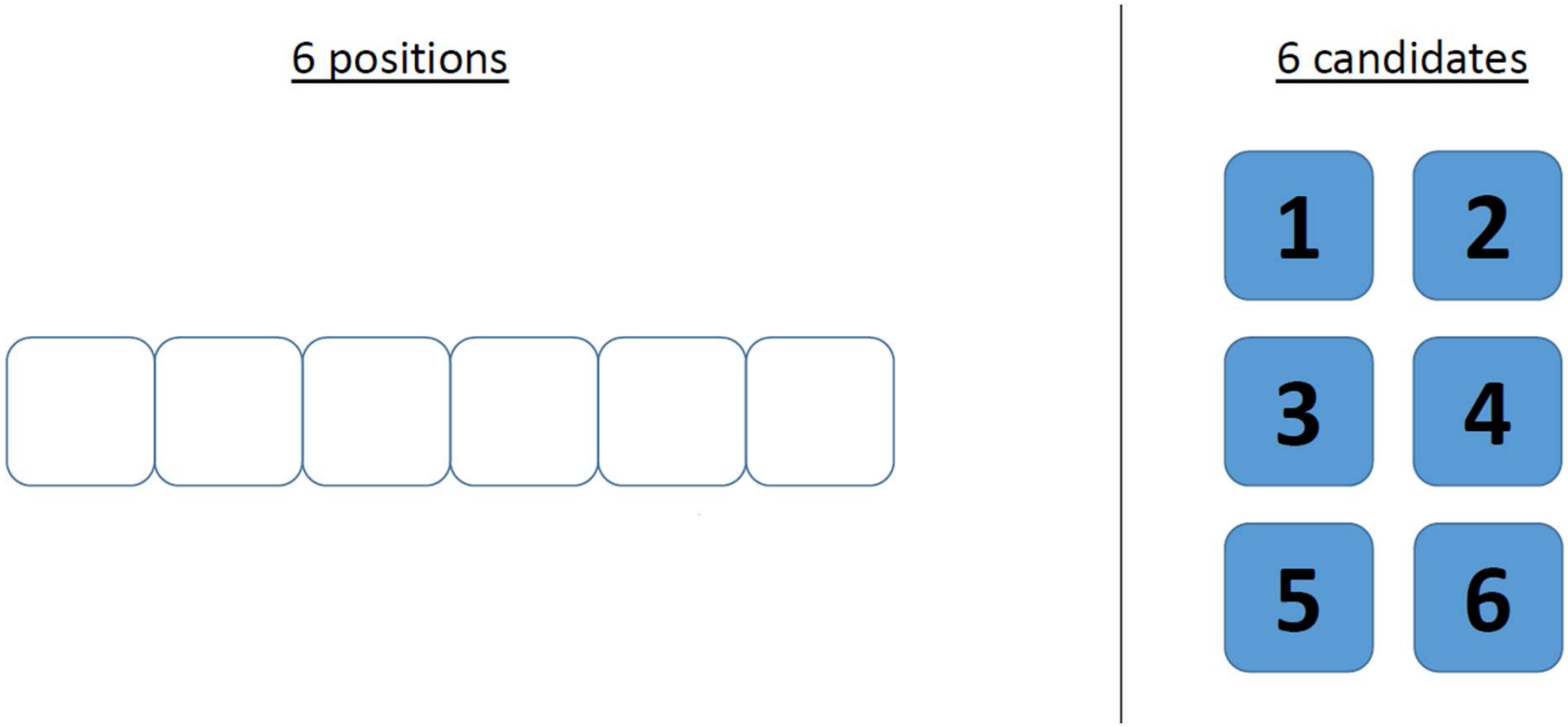
$$Obj(S, y) = \sum_{x_i \in S} ED(x_i, y)$$

# Why to study Ulam Median?

- First, it captures some of the inherent difficulties of the Edit metric

- Second, permutations can be viewed as rankings, and the Ulam distance is an interesting dissimilarity measure

# Rank Aggregation / (Ulam) Median



# Rank Aggregation / (Ulam) Median

| | | | | | | |
|---|---|---|---|---|---|---|
| Voter 1: | 1 | 6 | 5 | 3 | 4 | 2 |
| Voter 2: | 2 | 1 | 5 | 3 | 4 | 6 |
| Voter 3: | 5 | 3 | 1 | 4 | 6 | 2 |
| Voter 4: | 1 | 5 | 4 | 3 | 6 | 2 |

| | |
|---|---|
| 1 | 2 |
| 3 | 4 |
| 5 | 6 |

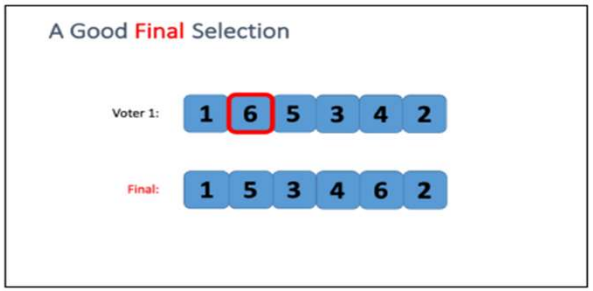
# Rank Aggregation / (Ulam) Median

Aggregated Rank:

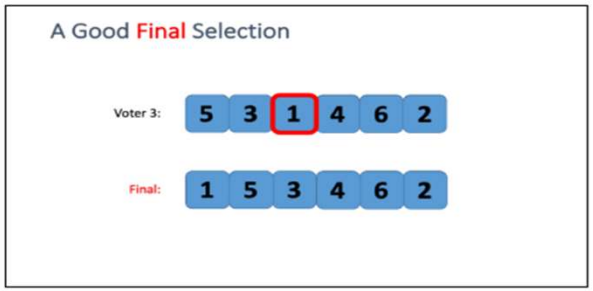
| | | | | | |
|---|---|---|---|---|---|
| 1 | 5 | 3 | 4 | 6 | 2 |

# Rank Aggregation / (Ulam) Median

Voter 1

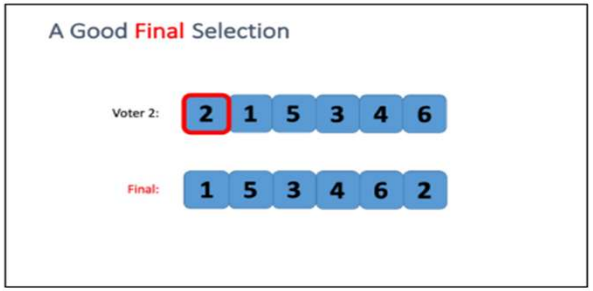



Voter 3

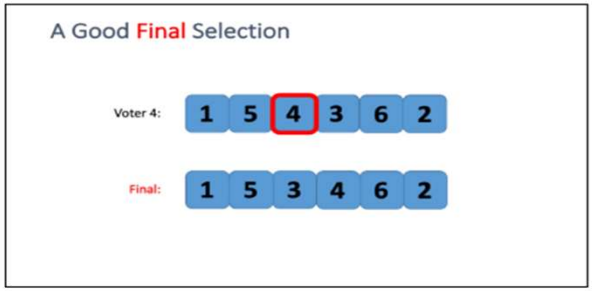



Aggregated Rank:


Voter 2




Voter 4



# Applications

- Social choice theory
- Sports
- Databases
- Statistics
- Internet
- Many more…

# Rank Aggregation

- Ulam distance is one of the dissimilarity measures among rankings/permutations

- Other popular measures include Kendall-tau / Kemeny, Spearman footrule,…

Counts the number of inversions

# 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)<br>For 3 inputs, NP-hard or P? |
| Ulam | $(2-\epsilon)$-approximation in polytime [C, Das, Krauthgamer '21]<br>For 3 inputs, in P | NP-hard? |

# Thank You!