# Set Similarity Join

Dong Deng

# Similarity Query Processing: Re-cap

- String/Sequence, Edit Distance (first lecture) - ***Partition***
- Set, Overlap (second lecture) - ***Subset Enumeration***
- (Weighted) Set, Jaccard/Cosine Similarity and more (this lecture) - ***Prefix Filtering***
- Real-value Vectors, Euclidian Distance and more (next lecture) - ***Approximation***

# Programming Assignment I

# Near Duplicate Data

On one end, a winded Pete Sampras tried to summon enough energy to give the New York fans another memorable win to talk about it on the subway ride home. On the other side, Roger Federer wore a sly grin like he knew age was about to catch up to the former world No. 1 - the man who owns the record of 14 Grand Slams he wants.

GMA NEWS.TV BETA
The Official Website of GMA News And Public Affairs

Home | Nation | Regions | Business | Pinoy Ab

Breaking News » Militant group to Arroyo

Sports

Home > Sports > Top Stories

**Federer bests Sampras in exhibition at Madison Square Garden**

03/11/2008 | 11:28 AM

Email this | Email the Editor | Print | Digg this | Add to del.icio.us

NEW YORK – Past and present stood across the net from each other during a third-set tiebreaker at Madison Square Garden.

On one end, ergy to give the New York fans another mer . On the other side, Roger Federer wor to the former world No. 1 – the man who owns the record of 14 Grand Slams he wants.

Youth is served, indeed.

Current No. 1 Federer emerged with a 6-3, 6-7 (4), 7-6 (6) victory Monday night in an exhibition that featured a little bit of everything – some laughter, some stellar shots, uneven play and compelling tennis.

There was even a Tiger Woods sighting.

"It was a great night for tennis," Sampras said.

There were moments when, if you squinted a bit, you would have sworn that was the Sampras of old, rather than an old Sampras. There were moments when, if you

03/11/2008 | 11:28 AM

New User? Sign Up Sign In | Help

MA Boxing Golf Soccer Tennis Action Sports

Video Photos

**Roger Federer beats Pete Sampras in sold-out exhibition at Madison Square Garden**

By JAY COHEN, AP Sports Writer
Mar 11, 4:23 am EDT

Buzz Up Print

NEW YORK (AP)—Past and present stood across the net from each other during a third-set tiebreaker at Madison Square Garden.

On one end, Pete Sampras tried to summon enough energy to give the New York fans another memorable win to oger Federer wore a sly grin like he knew age was about t he record of 14 Grand Slams he wants.

Youth i

Curren Monday night in an exhibition that featured a little bit of ever nd compelling tennis.

There was even a Tiger Woods sighting.

ADVERTISEMENT

"It was a great night for tennis," Sampras said.

There were moments when, if you squinted a bit, you would have sworn that was the Sampras of old, rather than an old Sampras. There were moments when, if you listened to the whip of the racket through the air, you would have been absolutely sure Federer was giving it his all.

And then there were moments when, as you watched Sampras throw his racket to the ground in mock disgust or saw Federer raise an index finger to celebrate four

By JAY COHEN, AP Sports Writer
Mar 11, 4:23 am EDT

# Fuzzy Matching: A use case :-)

- Find speeches similar to Melania Trump's RNC speech

from: https://chenli.ics.uci.edu/

# Fuzz Matching: A use case :-)

**Melania**
Trump
RNC speech | July 18, 2016

From a young age, my parents impressed on me the values that you work hard for what you want in life, that your word is your bond and you do what you say and keep your promise, that you treat people with respect. They taught and showed me values and morals in their daily lives. That is a lesson that I continue to pass along to our son.

And we need to pass those lessons on to the many generations to follow. Because we want our children in this nation to know that the only limit to your achievements is the strength of your dreams and your willingness to work for them."

**Michelle**
Obama
DNC speech | August 25, 2008

And Barack and I were raised with so many of the same values: that you work hard for what you want in life; that your word is your bond and you do what you say you're going to do; that you treat people with dignity and respect, even if you don't know them, and even if you don't agree with them.

And Barack and I set out to build lives guided by these values, and to pass them on to the next generation. Because we want our children -- and all children in this nation -- to know that the only limit to the height of your achievements is the reach of your dreams and your willingness to work for them."

from: https://chenli.ics.uci.edu/

# Set Similarity Functions

- Set Similarity Function
  - $Jaccard(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$
  - $Cosine(X,Y) = \frac{|X \cap Y|}{\sqrt{|X||Y|}}$
  - $Dice(X,Y) = \frac{2|X \cap Y|}{|X|+|Y|}$

| $X$ = {A,B,C,D,E}<br>$Y$ = {B,C,D,E,F} |
| --- |
| 4/6 = 0.67 |
| 4/5 = 0.8 |
| 8/10 = 0.8 |

# Problem Definition

- Input:
  - A collection of sets $R$
  - A set similarity function $Sim$
  - A similarity threshold $\delta$
- Output
  - All pairs $(X, Y) \in R \times R$ such that $Sim(X, Y) \geq \delta$

# An Example

- Input:
  - $R$

| *id* | *The records* | |
|---|---|---|
| 1 | $\mathcal{X}_1$ | $\{x_1, x_2, x_5, x_6, x_7, x_{10}, x_{11}, x_{13}, x_{14}\}$ |
| 2 | $\mathcal{X}_2$ | $\{x_2, x_4, x_5, x_6, x_9, x_{11}, x_{13}, x_{14}, x_{15}\}$ |
| 3 | $\mathcal{X}_3$ | $\{x_1, x_3, x_6, x_7, x_9, x_{10}, x_{11}, x_{13}, x_{14}\}$ |
| 4 | $\mathcal{X}_4$ | $\{x_3, x_4, x_5, x_7, x_8, x_{10}, x_{12}, x_{13}, x_{14}\}$ |
| 5 | $\mathcal{X}_5$ | $\{x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_{10}, x_{11}, x_{13}, x_{14}\}$ |

  - *Sim* = Jaccard Similarity
  - $\delta = 0.73$
- Output
  - $\text{Jaccard}(X_1, X_5) = 0.82 \geq \delta$

# Applications

- For Web search engines:
  - Perform focused crawling
  - Increase the quality and diversity
  - Identify spams
- For Web mining:
  - Perform document clustering
  - Find replicate Web collections
  - Detect plagiarism

**Q**. What are the advantages of RAID5 over RAID4?
**A**. 1. Several write requests could be processed in parallel, since the bottleneck of a unique check disk has been eliminated. 2. Read requests have a higher level of parallelism. Since the data is distributed over all disks, read requests involve all disks, whereas in systems with a dedicated check disk the check disk never participates in read.

<NUMBER> drew lucky star winning numbers
<NUMBER> which consequently won in the 2ND

**Q**. What are the advantages of RAID5 over RAID4?
**A**. 1. Several write requests could be processed in parallel, since the bottleneck of a single check disk has been eliminated. 2. Read requests have a higher level of parallelism on RAID5. Since the data is distributed over all disks, read requests involve all disks, whereas in systems with a check disk the check disk never participates in read.

# Set Similarity Functions

- The above similarity functions can be equivalently converted to overlap
- Overlap:

$$O(X, Y) = |X \cap Y|$$

- Jaccard:

$$J(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \geq \theta \quad \leftrightarrow \quad O(X, Y) \geq \frac{\theta}{1+\theta}(|X| + |Y|) = t$$

- We can only consider the **overlap size,** as others can be transformed to the overlap similarity.

# Example

- Overlap threshold $t = 3$
- Result (RID pairs): $\{(1,3), (3,4), (3,5)\}$

| RID | Name |
|---|---|
| 1 | Database System Concepts |
| 2 | Database Concepts Techniques |
| 3 | Database System Programming Concepts Oracle Linux |
| 4 | Database Programming Concepts Illustrated |
| 5 | System Programming Concepts Techniques Oracle Linux |

# Naïve Solutions

- Enumerate all string pairs
- Verify whether the strings in the pair are similar

# Naïve Algorithm

need to compare all $O(n^2)$ pairs !!

| RID | Name |
|---|---|
| 1 | Database System Concepts |
| 2 | Database Concepts Techniques |
| 3 | Database System Programming Concepts Oracle Linux |
| 4 | Database Programming Concepts Illustrated |
| 5 | System Programming Concepts Techniques Oracle Linux |

$t = 3$

*Result:*
*(3, 1)*
*(4, 3)*
*(5, 4)*

| overlap of (*i*, *j*) | i=1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| j=1 | | 2 | 3 | 2 | 2 |
| 2 | | | 2 | 2 | 2 |
| 3 | | | | 3 | 5 |
| 4 | | | | | 2 |

# Framework

- for each $S_j \in S$
  - Candidates = $\phi$
  - Candidates = **getCandidates**($S_j$)
  - **Verify**($S_j$, Candidates)

**Verify**(x, {$y_1$, $y_2$, …})

for each $y_i$
  if overlap(x, $y_i$) ≥ t
    output(<x, $y_i$>)
end

- Algorithms differ in **getCandidate**s()
  - naïve alg: return {$S_i$ | i < j}

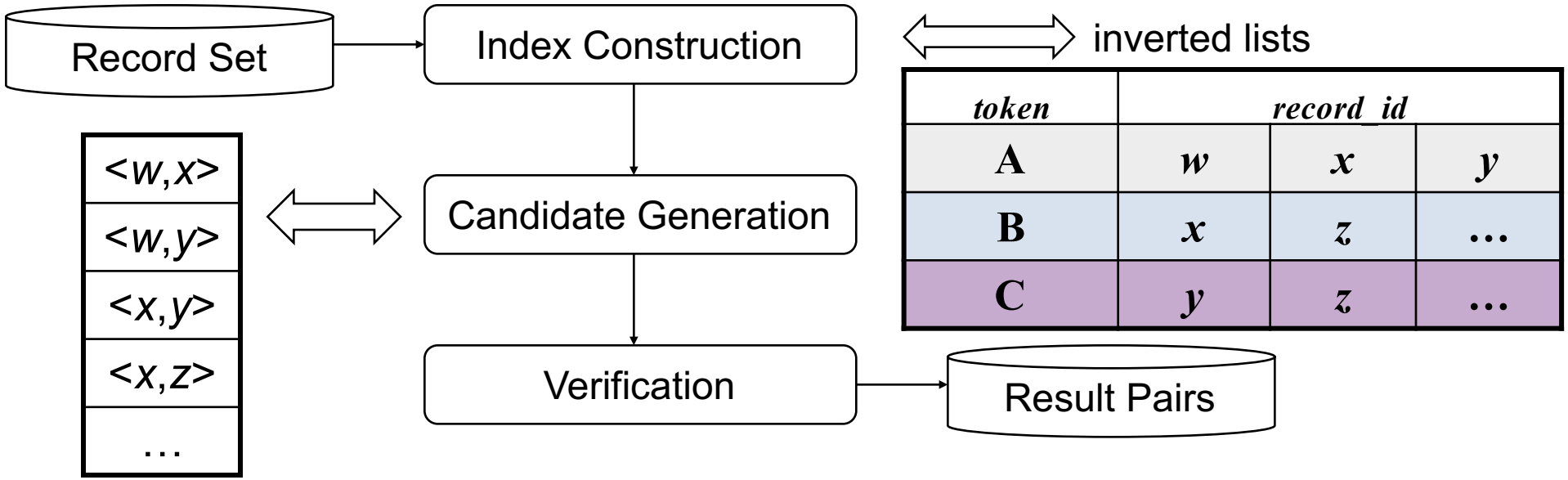
# Not-so-naïve Algorithm

- Observation: (*, 6) should not been compared
  - If x and y have no common token, they won't be in the result
- Idea: Use *inverted index* to consider promising candidate pairs only

| RID | Name |
|---|---|
| 1 | Database System Concepts |
| 2 | Database Concepts Techniques |
| 3 | Database System Programming Concepts Oracle Linux |
| 4 | Database Programming Concepts Illustrated |
| 5 | System Programming Concepts Techniques Oracle Linux |
| 6 | Harry Potter and the Sorcerer's Stone |

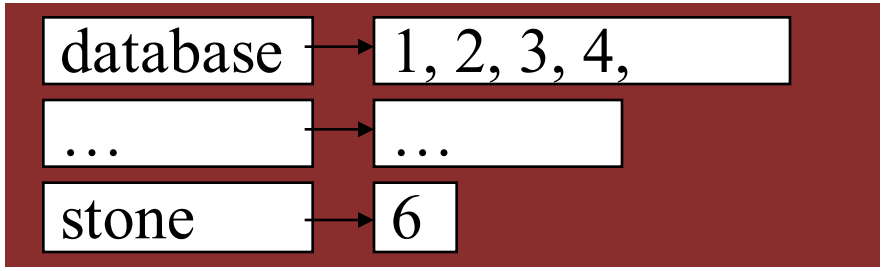
# Index-Based Algorithms

- Index-based Algorithm



# Inverted Index

- Conceptually, an inverted index has an inverted list for each token to be indexed from the document collection
  - An inverted list is just an sorted array of document identifiers (in our case, RIDs) such that the token appears in the corresponding document




| RID | Name |
|---|---|
| 1 | Database System Concepts |
| … | … … … |
| 6 | Harry Potter and the Sorcerer's Stone |

# Probe Count

| RID | Name |
|---|---|
| 1 | Database System Concepts |
| 2 | Database Concepts Techniques |
| 3 | Database System Programming Concepts Oracle Linux |
| 4 | Database Programming Concepts Illustrated |
| 5 | System Programming Concepts Techniques Oracle Linux |

| token | Inverted list |
|---|---|
| Database | {1, 2, 3, 4} |
| System | {1, 3, 5} |
| Concepts | {1, 2, 3, 4, 5} |
| Techniques | {2, 5} |
| Programming | {3, 4, 5} |
| Oracle | {3, 5} |
| Linux | {3, 5} |
| Illustrated | {4} |

| RID | Name |
|---|---|
| 1 | Database System Concepts |
| 2 | Database Concepts Techniques |
| 3 | |
| 4 | |
| 5 | |

| term | RID-list |
|---|---|
| Database | {1, 2, 3, 4} |
| System | {1, 3, 5} |
| Concepts | {1, 2, 3, 4, 5} |
| Techniques | {2, 5} |
| Programming | {3, 4, 5} |
| Oracle | {3, 5} |
| Linux | {3, 5} |
| Illustrated | {4} |

➔ {1, 2, 3, 4, 5}

1 needs to be compared with {2, 3, 4, 5}

1. Consider the terms in the current record and retrieve their inverted lists through the index
2. Merge these RID-lists
3. Repeat step 1 & 2 for every record

Note that RIDs in the inverted list are sorted

| RID | Name |
|---|---|
| 1 | Database System Concepts |
| 2 | Database Concepts Techniques |
| 3 | |
| 4 | |
| 5 | |

| term | RID-list |
|---|---|
| Database | {1, 2, 3, 4} |
| System | {1, 3, 5} |
| Concepts | {1, 2, 3, 4, 5} |
| Techniques | {2, 5} |
| Programming | {3, 4, 5} |
| Oracle | {3, 5} |
| Linux | {3, 5} |
| Illustrated | {4} |

{1/3, 2/2, 3/3, 4/2, 5/2}

RID / count. e.g., overlap(1,2)=2

1. Consider each term, *t*, in the current record and retrieve their inverted lists through the index
2. Count Filtering: Merge and filter these RID-lists
   - prune those whose count ≤ *t*
3. Repeat steps 1 & 2 for every record

Note that RIDs in the inverted list are sorted

# Framework

- for each $S_j \in S$
  - Candidates = $\phi$
  - Candidates = getCandidates($S_i$)
  - Verify($S_j$, Candidates)

- Algorithms differ in getCandidates()
  - **naïve alg: return $\{S_i \mid i < j\}$**
  - **index-based alg: return $\{S_i \mid i < j \ \wedge S_i \cap S_j \neq \phi\}$**

**// use inverted indexes**

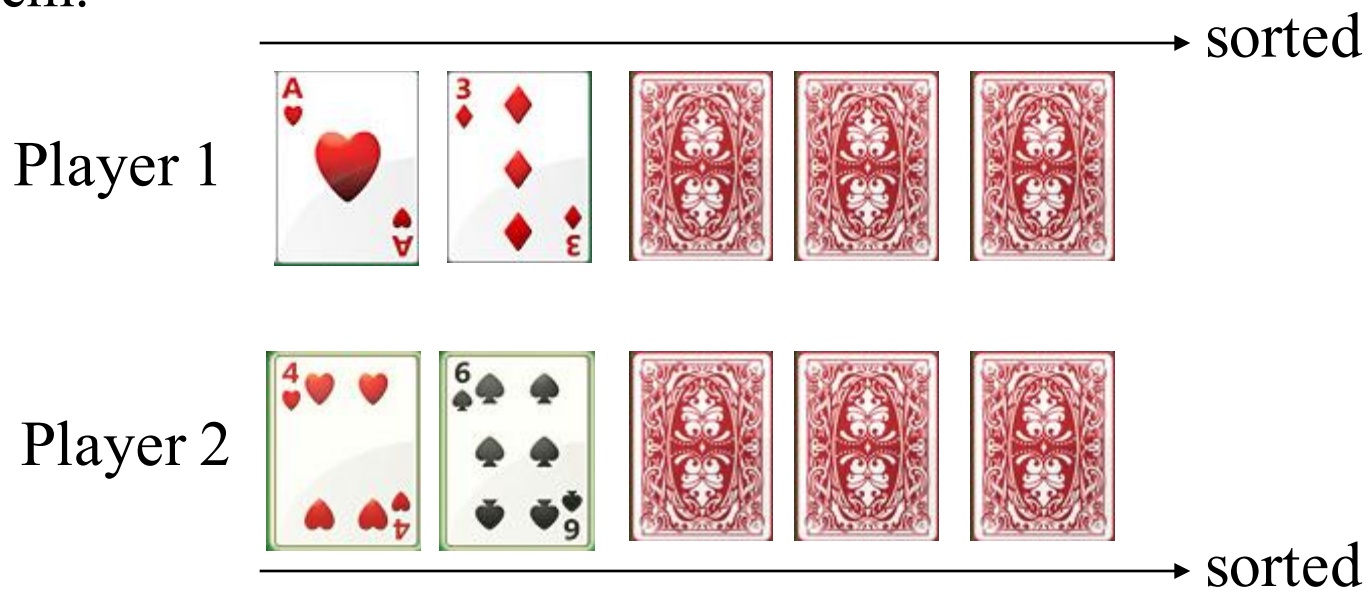
# Problems

- Still too many comparisons
  - tokens that appears in many documents not only results in large inverted lists, but also slow down the computation

| RID | Name |
|---|---|
| 1 | Database System Concepts |
| 2 | Database Concepts Techniques |
| 3 | Database System Programming Concepts Oracle Linux |
| 4 | Database Programming Concepts Illustrated |
| 5 | System Programming Concepts Techniques Oracle Linux |
| 6 | Database of Respiratory Diseases |

# Prefix Filtering

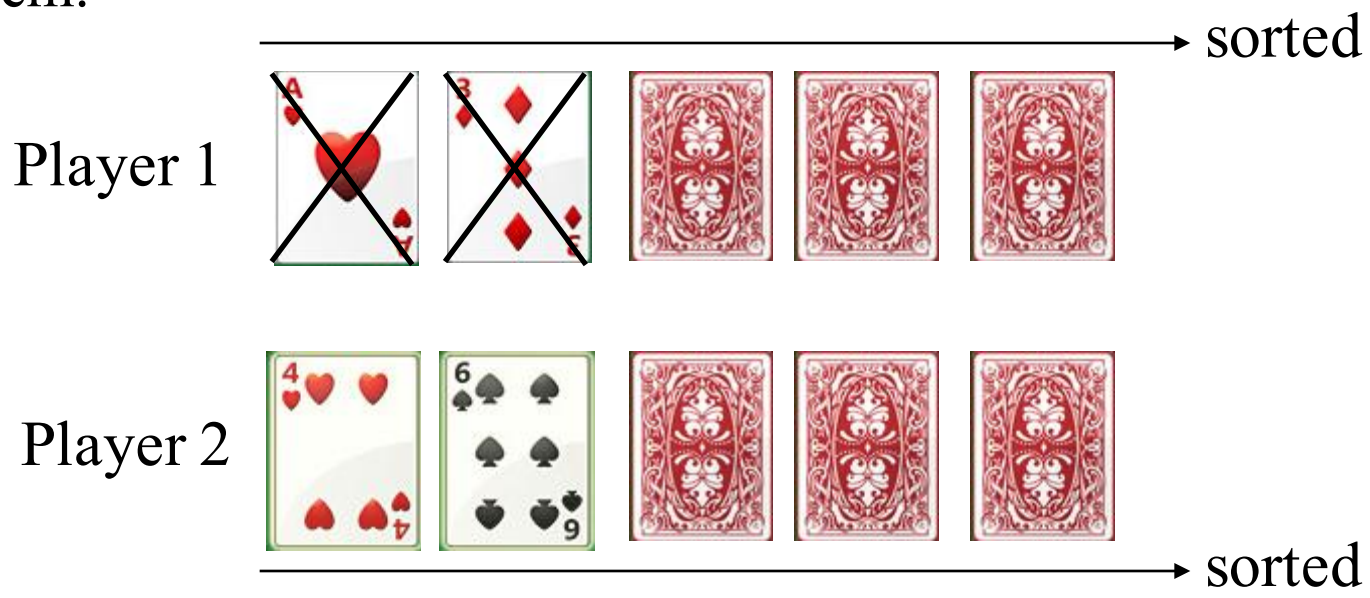
- Establish an upper bound of the overlap between two sets by looking at only *part of* them.




*what's the maximum possible number of cards held by both players (denomination not considered) ?*

# Prefix Filtering

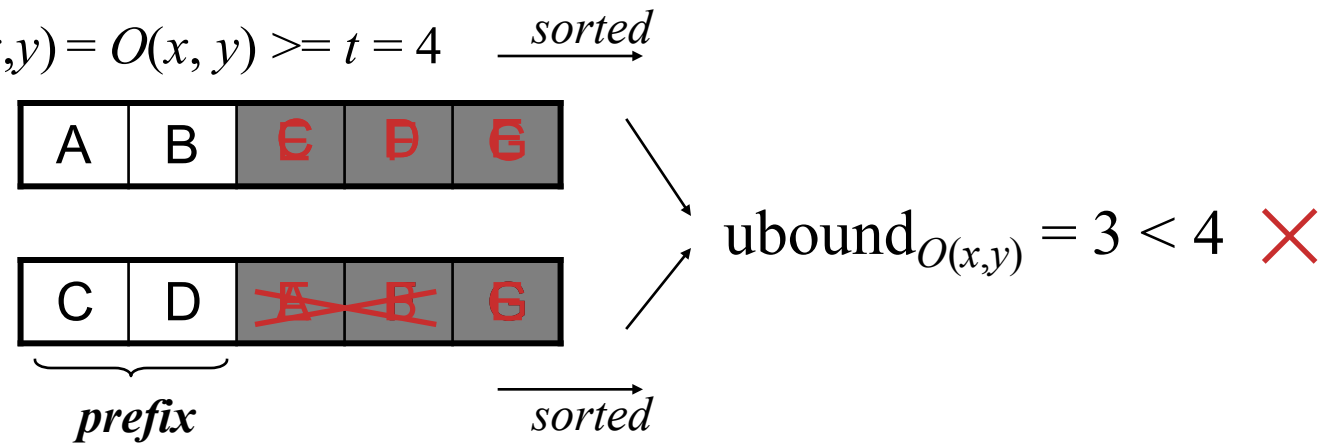
- Establish an upper bound of the overlap between two sets by looking at only *part of* them.




*what's the maximum possible number of cards held by both players (denomination not considered) ?*

# Prefix Filter

- Sort the tokens by a global ordering
  - increasing order of frequency
- Index the first few tokens (prefix) for each record
- Example:
  - suppose $\text{sim}(x,y) = O(x, y) >= t = 4$
  - $x =$
  - $y =$

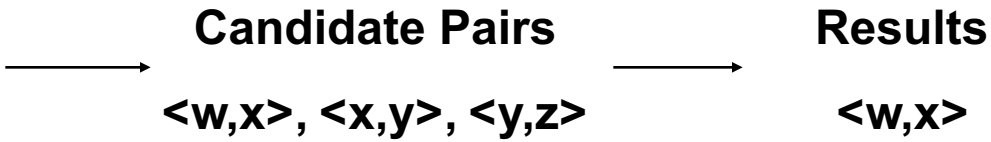



- Must share at least one token in prefix to be a candidate pair

# Prefix Filter

- *For x (or y)*
  - $O(x,y) >= t \rightarrow$ prefix length $= |x| - (t-1)$
  - $J(x,y) >= \delta \rightarrow O(x,y) >= \delta |x| \rightarrow$ prefix length $= \lfloor (1-\delta)|x| + 1 \rfloor$

- Example: suppose $\text{sim}(x,y) = J(x,y) >= \delta = 0.8$
  - w = {C, D, E, F}
  - x = {B, C, D, E, F}
  - y = {A, B, C, D, F}
  - z = {G, A, B, E, F}



# Prefix Filtering for Overlap

- Constraint: $\text{Overlap}(S_1, S_2) \geq t$
- Pre-requisite
  - All sets sorted in a global order
  - Usually the increasing **frequency** order ← facilitates the join
- Preprocessing
  - Set S → prefix(S), s.t., $|\text{prefix}(S)| = |S| - (t - 1)$
- Filter
  - If $\text{prefix}(S_1) \cap \text{prefix}(S_2) = \phi$, then $\text{Overlap}(S_1, S_2) < t$
  - i.e., $(S_1, S_2)$ can be filtered
- Refine: verify the survived pairs

# Framework

- for each $S_j \in S$
  - Candidates = $\phi$
  - Candidates = **getCandidates**($S_i$)
  - Verify($S_j$, Candidates)
- Algorithms differ in **getCandidates()**
  - naïve alg: return $\{S_i \mid i < j\}$
  - index-based alg: return $\{S_i \mid i < j \ \wedge S_i \cap S_j \neq \phi\}$ // use inverted index
  - **prefix-filtering-base alg: return $\{S_i \mid i < j \ \wedge$ prefix($S_i$) $\cap$ prefix($S_j$) $\neq \phi\}$ using inverted index on prefix set**

Pr

| RID | Name |
|---|---|
| 1 | Database System Concepts |
| 2 | Database Concepts Techniques |
| 3 | Database System Programming Concepts Oracle Linux |
| 4 | Database Programming Concepts Illustrated |
| 5 | System Programming Concepts Techniques Oracle Linux |

| token | Inverted list | df | idf | Order |
|---|---|---|---|---|
| Database | {1, 2, 3, 4} | 4 | 1/4 | 7 |
| System | {1, 3, 5} | 3 | 1/3 | 6 |
| Concepts | {1, 2, 3, 4, 5} | 5 | 1/5 | 8 |
| Techniques | {2, 5} | 2 | 1/2 | 4 |
| Programming | {3, 4, 5} | 3 | 1/3 | 5 |
| Oracle | {3, 5} | 2 | 1/2 | 3 |
| Linux | {3, 5} | 2 | 1/2 | 2 |
| Illustrated | {4} | 1 | 1/1 | 1 |

Pr

| RID | Name |
|---|---|
| 1 | *System* Database Concepts |
| 2 | ***Techniques*** Database Concepts |
| 3 | ***Linux Oracle Programming*** *System* Database Concepts |
| 4 | ***Illustrated Programming*** Database Concepts |
| 5 | ***Linux Oracle Techniques Programming*** System Concepts |

Order: Illustrated, Linux, Oracle, Techniques, Programming, System, Database, Concepts

t=3 ➔ prefix-len = |S| - 2

j=1

| token | Inverted list |
|---|---|
| System | {1} |

j=2

| token | Inverted list |
|---|---|
| System | {1} |
| Techniques | {2} |

j=3

| token | Inverted list |
|---|---|
| System | {1,3} |
| Techniques | {2} |
| Linux | {3} |
| Oracle | {3} |
| Programming | {3} |

Pr

| RID | Name | |
|---|---|---|
| 1 | *System* | Database Concepts |
| 2 | ***Techniques*** | Database Concepts |
| 3 | ***Linux Oracle Programming*** *System* | Database Concepts |
| 4 | ***Illustrated Programming*** | Database Concepts |
| 5 | ***Linux Oracle Techniques Programming*** | System Concepts |

t=3 ➔ prefix-len = |S| - 2

j=3

| token | Inverted list |
|---|---|
| System | {1,3} |
| Techniques | {2} |
| Linux | {3} |
| Oracle | {3} |
| Programming | {3} |

j=4

| token | Inverted list |
|---|---|
| System | {1,3} |
| Techniques | {2} |
| Linux | {3} |
| Oracle | {3} |
| Programming | {3, 4} |
| Illustrated | {4} |

j=5

| token | Inverted list |
|---|---|
| System | {1,3} |
| Techniques | {2, 5} |
| Linux | {3, 5} |
| Oracle | {3, 5} |
| Programming | {3, 4} |
| Illustrated | {4} |

# Framework

- for each $S_j \in S$
  - Candidates = $\phi$
  - Candidates = getCandidates($S_i$)
  - Verify($S_j$, Candidates)
- getCandidates()
  - **prefix-filtering-base algorithm:**
    - **Return $\{S_i \mid i < j \ \wedge \ prefix(S_i) \cap prefix(S_j) \neq \phi\}$**
    - **Using inverted lists**

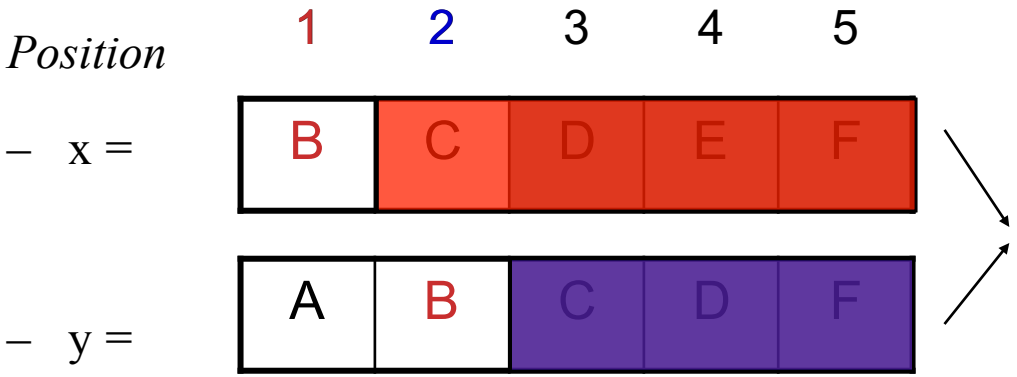
# Supporting two datasets

- Join R, S
- Generate prefix sets for R and S
- Build inverted lists for prefix sets of R
- for each $S_j \in S$
  - Candidates = $\phi$
  - Candidates = getCandidates($S_i$)
  - Verify($S_j$, Candidates)
- Algorithms differ in getCandidates()
  - **prefix-filtering-base algorithm:**
    - **Return $\{S_i \mid prefix(S_i) \cap prefix(R_j) \neq \phi\}$**
    - **Using inverted lists**

# Pruning Techniques for Jaccard Similarity

# Positional Filter within Prefix

- Index both tokens and their positions




$$\text{ubound}_{O(x,y)} = 1 + \min(4, 3) = 4$$

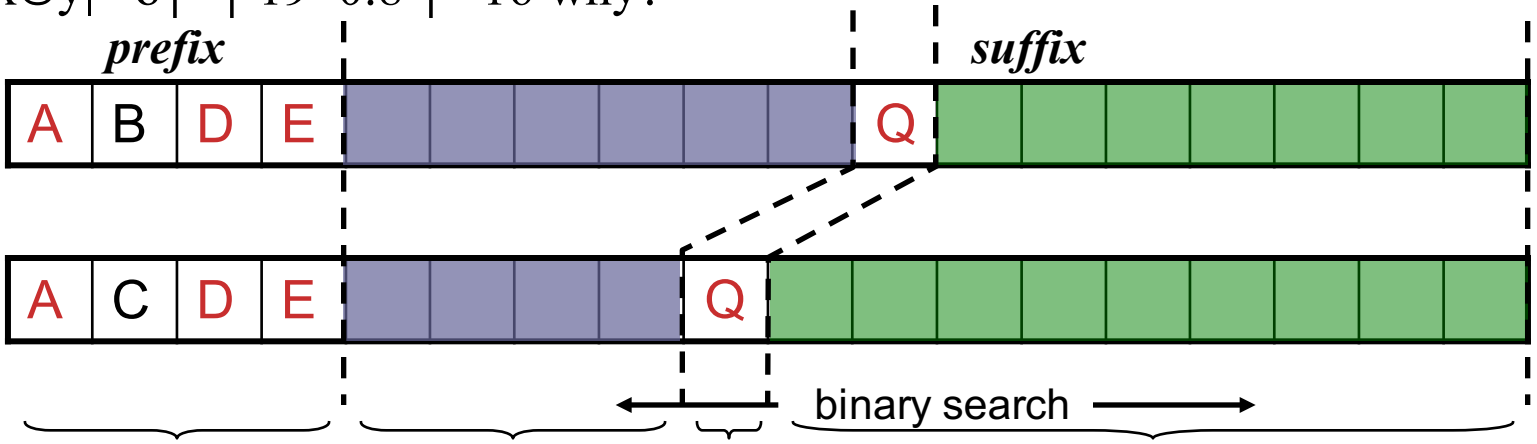
$$\text{lbound}_{|x \cup y|} = 2 + \max(4, 3) = 6$$

$$\text{ubound}_{J(x,y)} = \frac{|x \cap y|}{|x \cup y|} = \frac{4}{6} < \delta = 0.8$$

- $\text{ubound}_{O(x,y)} = p_x \cap p_y + \min(|x| - |p_x|, |y| - |p_y|)$ $p_x = B$
- $\text{lbound}_{x \cup y} = p_x \cup p_y + \max(|x| - |p_x|, |y| - |p_y|)$ $p_y = A\ B$

# Positional Filter within Suffix

- probe tokens in suffix, and compare their positions
- suppose $\text{sim}(x,y) = J(x,y) >= \delta = 0.8$
- $|x| = |y| = 18$, $|\text{prefix}| = \lfloor (1- \delta)*x+1 \rfloor = \lfloor (1-0.8)*18+1 \rfloor = 4$ why?
- $O(x,y) >= \lceil |x \cup y| * \delta \rceil = \lceil 19*0.8 \rceil = 16$ why?

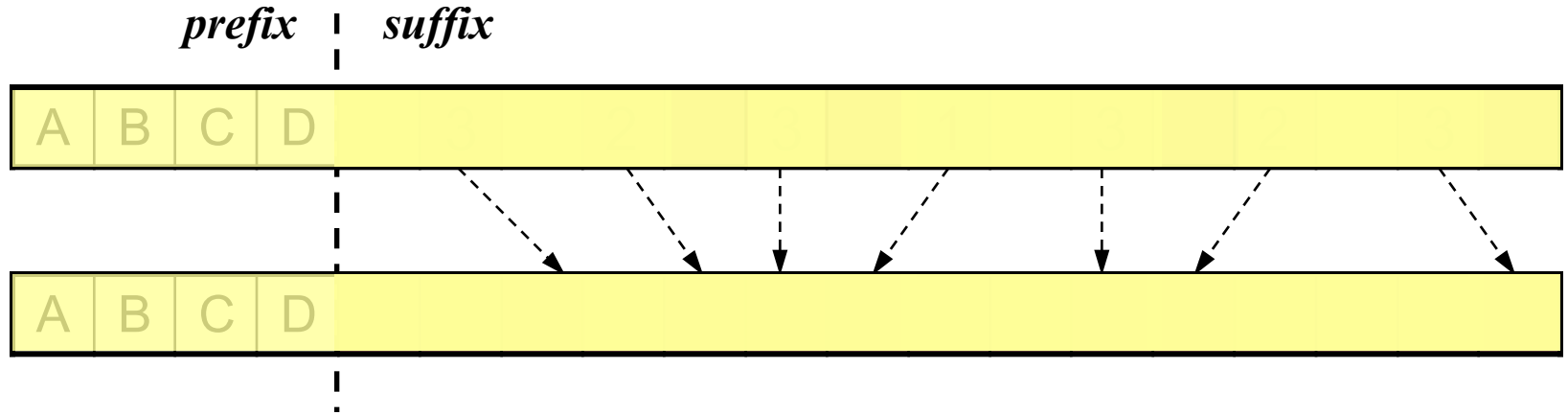



- $\text{ubound}_{O(x,y)} = 3 + 4 + 1 + 7$

$= 15 < 16$

# Positional Filter within Suffix

- Divide and Conquer




  - $ubound_{dep=1}$ = 4 + 6 + 1 + 7 = 18
  - $ubound_{dep=2}$ = 4 + 3 + 1 + 1 + 1 + 3 + 1 + 3 = 17
  - $ubound_{dep=3}$ = 4 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 = 15

- probe suffix recursively, until either candidate pair is pruned, or reach max-depth

# Effect of Filters

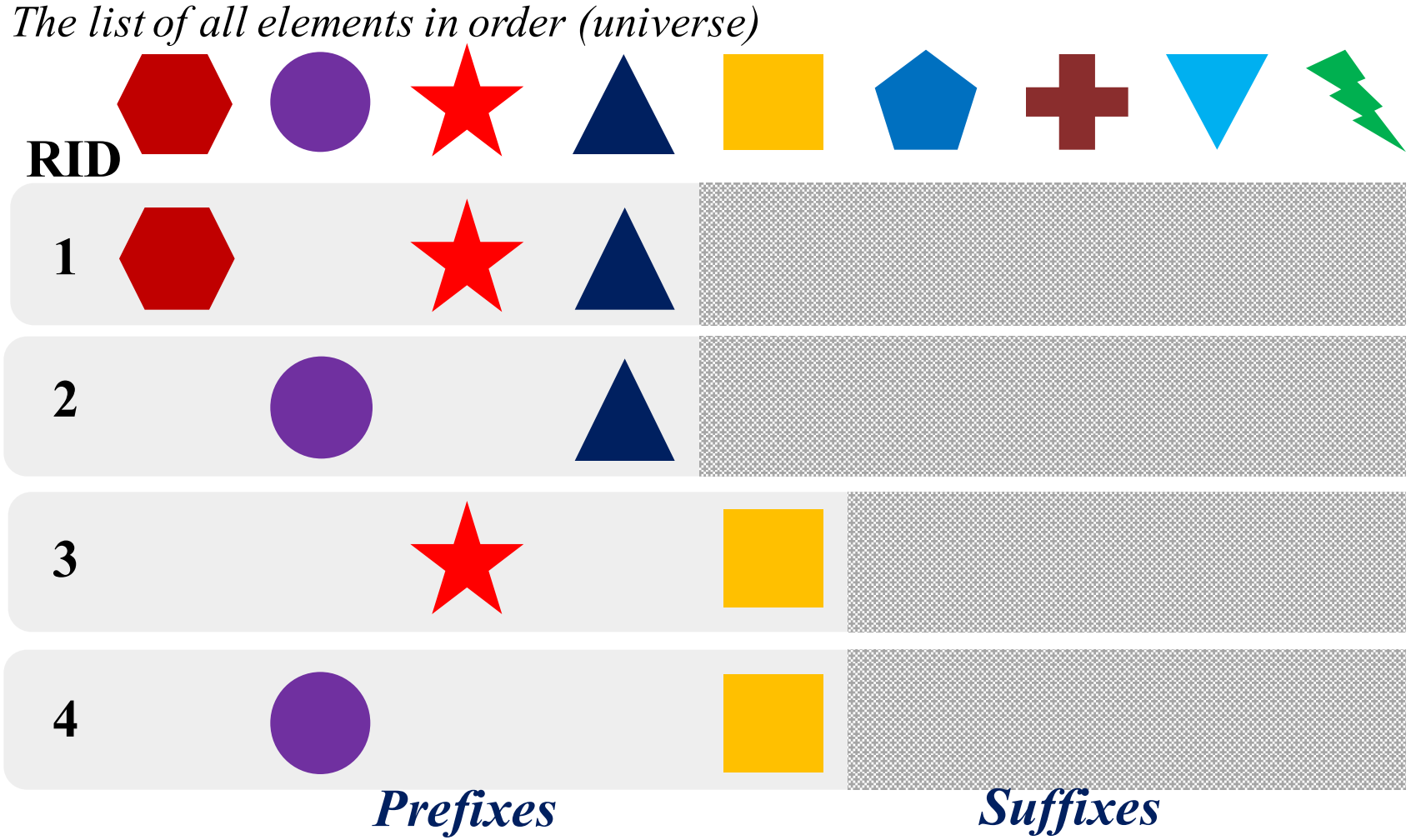
- $\text{sim}(x,y) = J(x,y) >= \delta = 0.8$
  - u = {C, D, E, F}
  - v = {B, C, D, E, F}
  - w = {A, B, C, D, F}
  - x = {G, A, B, E, F}
  - y = {A, B, D, E, F}
  - z = {G, A, C, D, E, F}
- after prefix filter:
  - <u,v>, <v,w>, <v,y>, <w,x>, <w,y>, <w,z>, <x,y>, <x,z>, <y,z>
- after prefix position filter:
  - <u,v>, <w,y>, <w,z>, <x,z>, <y,z>
- after suffix position + (max-depth = 1):
  - <u,v>, <x,z>, <y,z>
- real result:
  - <u,v>

# Similarity Join for Jaccard

- Generate tokens
- Compute frequency of token
- Filter: Generate Candidate Pairs
  - Using inverted lists
  - Using prefix sets
  - Using prefix sets with prefix-position pruning
  - Using suffix pruning
- Verify Candidate Pairs

# A Partition-based Method for Set Similarity Join

# Prefix Filter Framework




Prefix Filter: Sim(X, Y) $\geq \delta$ only if $Prefix(X) \cap Prefix(Y) \neq \emptyset$

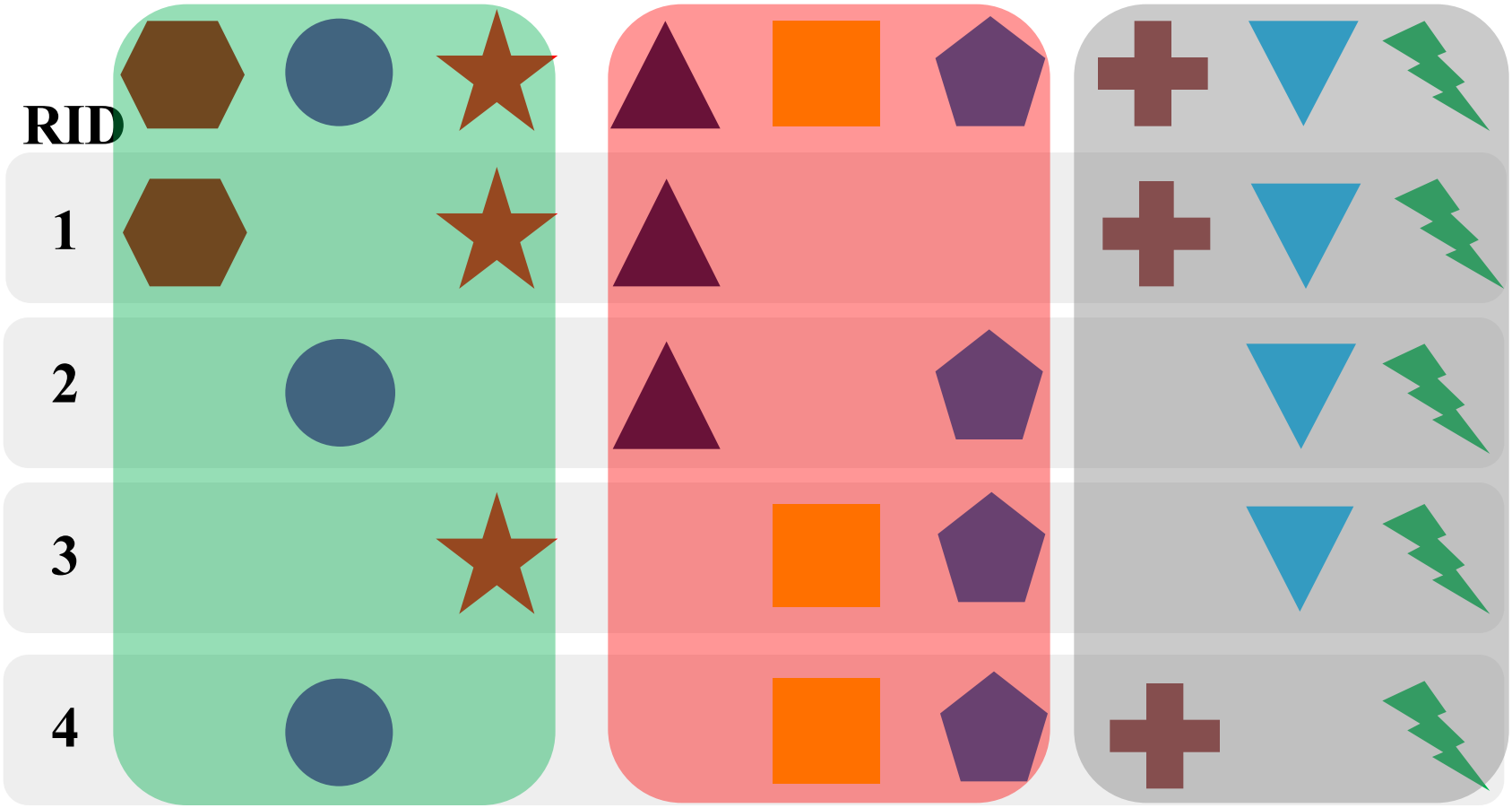
# Prefix Filter Framework

**the pruning power is limited!**

*two dissimilar sets are a candidate*
*if they share 1 element in their prefixes*

# Partition-based Framework

*The list of all elements in order (universe)*




$\text{Sim}(X, Y) \geq \delta$ only if $Subsets(X) \cap Subsets(Y) \neq \emptyset$

# Partition-based Framework

What is the minimum number of partitions that can guarantee completeness?

# The Number of Partitions

**Intuition:**

1: Deduce an overlap lower bound based on the similarity function and the threshold

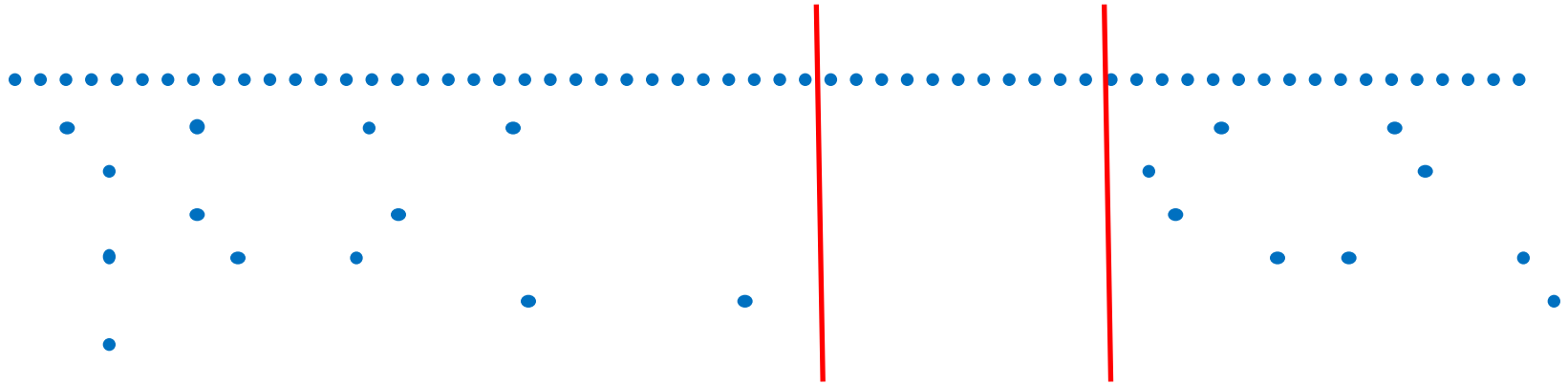
$$\mathbf{Sim}(\boldsymbol{X},\boldsymbol{Y}) \geq \boldsymbol{\delta} \quad \rightarrow \quad |\boldsymbol{X} \cap \boldsymbol{Y}| \geq \boldsymbol{m}$$

2: Partition them into $m + 1$ subsets

Then two similar sets must share at least 1 subset

# Element Skew Problem

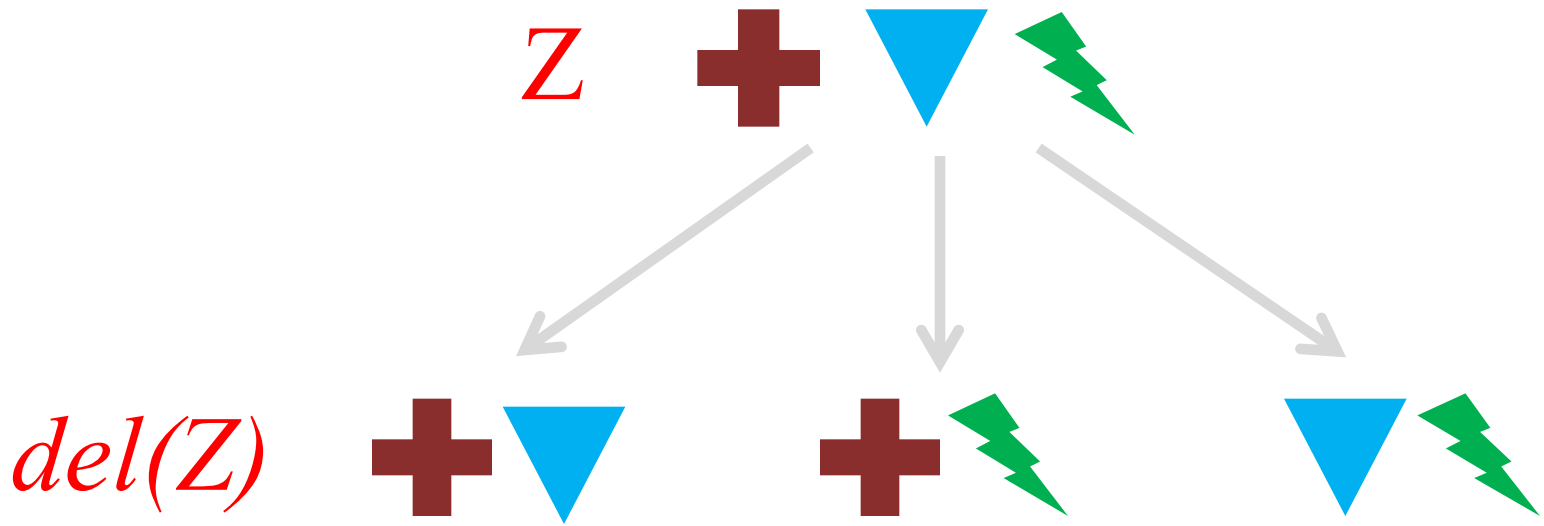
- Some subsets have limited number of elements
  - The ‘empty’ subsets yield quadratic candidates



- Solution: Add some flexibility
  - Skip the subsets with less elements
  - Select more signatures from subsets with more elements to guarantee completeness

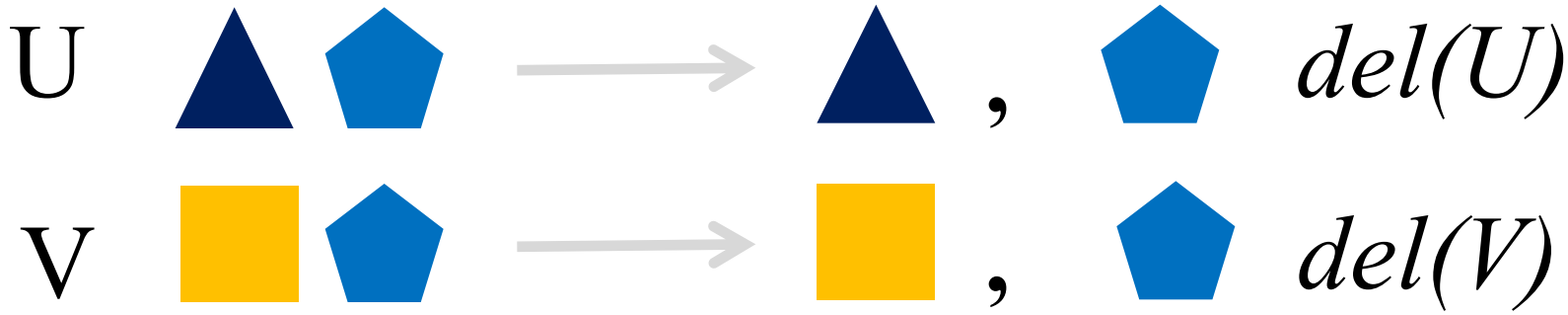
# Signatures: 1-deletion neighborhoods

- Given a non-empty set Z, its 1-deletion neighborhoods are its subsets with size of $|Z| - 1$, denote as *del(Z)*



# Using 1-deletion neighborhood

$$U \neq V, V \notin del(U), U \notin del(V) \rightarrow |U \Delta V| \geq 2$$




$$|U \Delta V| = 2$$

Skip $x$ subsets & select 1-deletion neighborhoods from another $x$ subsets

***guaranteed completeness !!***

# Optimal Allocation Strategy

$$v_i = \begin{cases} 0\text{: skip the } i^{th} \text{ subset} \\ 1\text{: only use the } i^{th} \text{ subset as signature} \\ 2\text{: use both the } i^{th} \text{ subset and 1-deletions} \end{cases}$$

$$\text{Constraint: } \sum_{i=1}^{m+1} v_i = m + 1$$

$$\text{Object: minimize } \sum_{i=1}^{m+1} c_{v_i}^i$$

$c_0^i = 0$ $\quad$ $c_1^i$: the # of sets sharing the $i^{th}$ subset

$c_2^i$: the # of sets sharing (subset or 1-deletion) signatures

# Time Complexity

- Dynamic Programming
  - Optimal: # of candidates
  - O($s^2$) time complexity as m = O(s) where $s$ is the set size

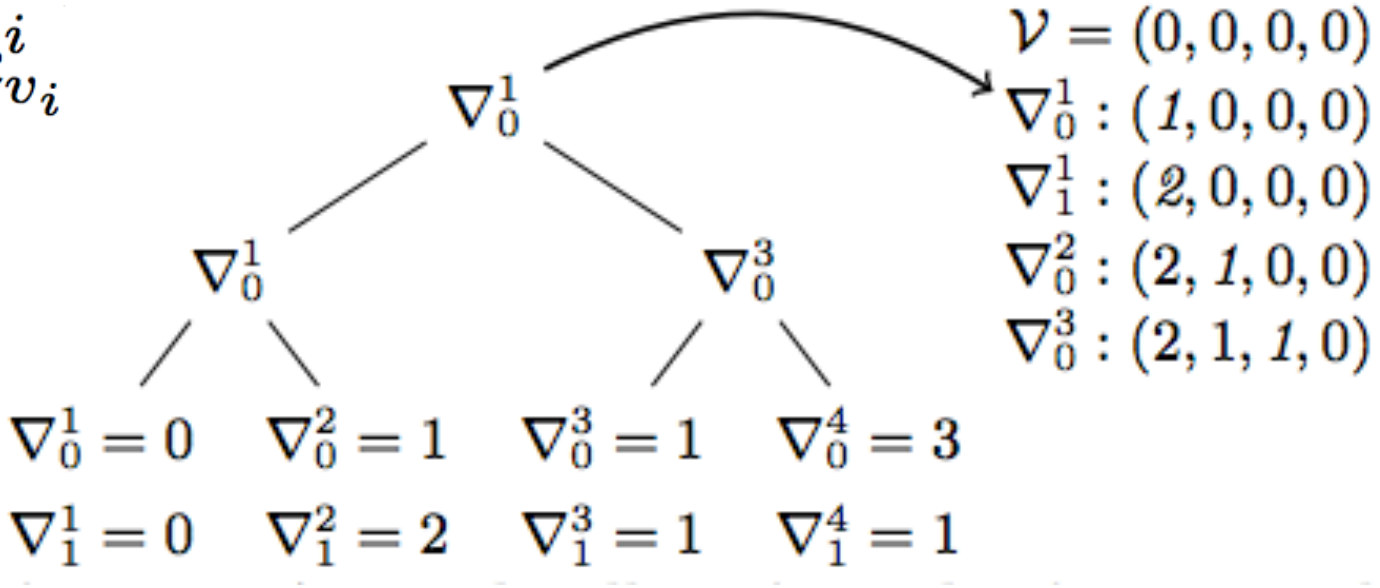
- Each set is partitioned $s - \delta s + 1 = O(s)$ times
- Allocation time complexity is O($s^3$) for each set

- Next we reduce it to O($s$ log $s$)

# Greedy Method for Allocation Selection

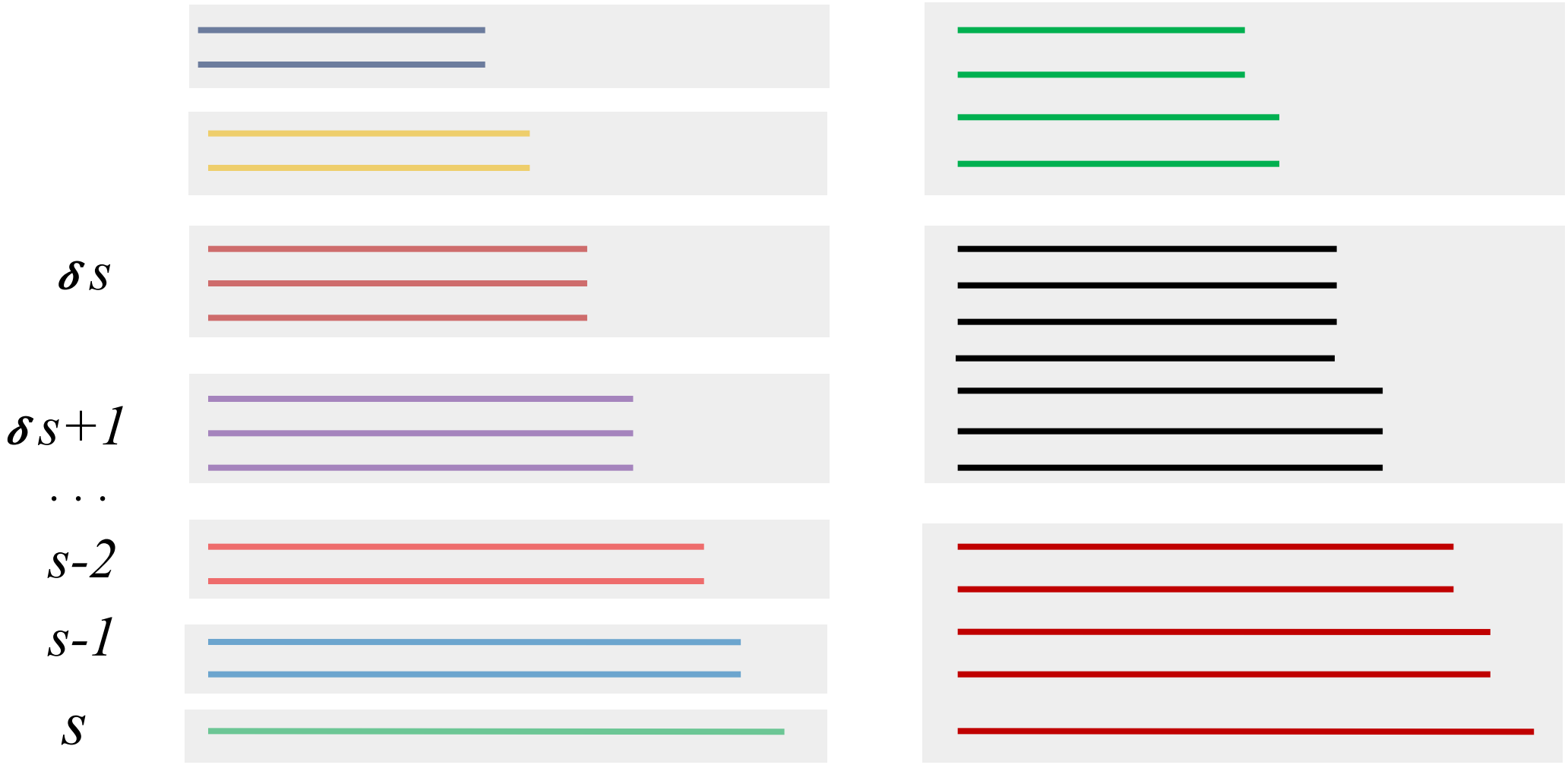
- Heap-based Method

$$\nabla_{v_i}^{i} = c_{v_i+1}^{i} - c_{v_i}^{i}$$




- 2-approximation algorithm
- Time complexity: $O(s^2)$ → $O(s \log s)$

# Adaptive Grouping



# Adaptive Grouping

- The $k$-th group includes all the sets with size within $[\frac{l_{min}}{\alpha^{k-1}}, \frac{l_{min}}{\alpha^{k}})$ where $\alpha \in [\frac{1}{2}, 1]$
- The size range becomes more and more ‘broader’
- The partition times is bounded by $log_{\alpha}\delta + 1 = O(1)$ for any set
- The time complexity is $O(s\ log\ s\ log_{\alpha}\delta) = O(s\ log\ s)$

# Prefix Filtering for Edit Distance

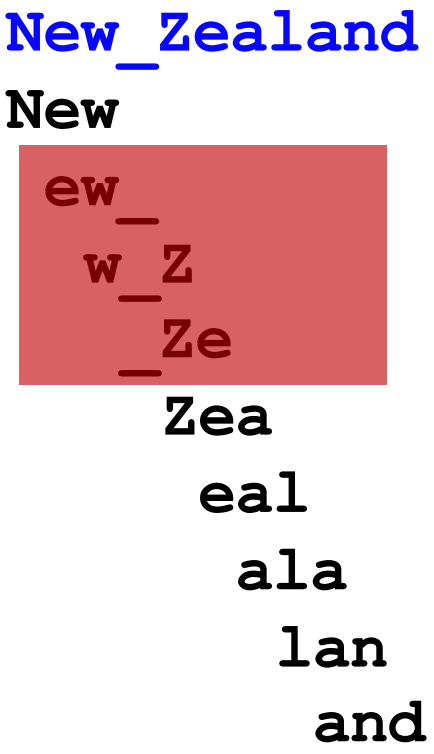
# q-gram Based Filtering

- Naïve algorithm
  - compute edit distance: $O(n^2)$ time complexity, where n is string length
  - do this for $N^2/2$ pairs, where N is the number of strings

- q-gram based filtering
  - q-gram is a substring of length q
  - filter-and-refine
  - length filtering
    - $| \text{len}(s) - \text{len}(t) | \leq \tau$

# Matching q-grams

- count filtering
  - at least LB($s$, $t$) common $q$-grams
  - $LB(s, t) = \max(|s|, |t|) - q + 1 - q*\tau$
- Bottleneck: generating candidate pairs which share at least LB($s$ ,$t$) matching q-grams



# Prefix Length

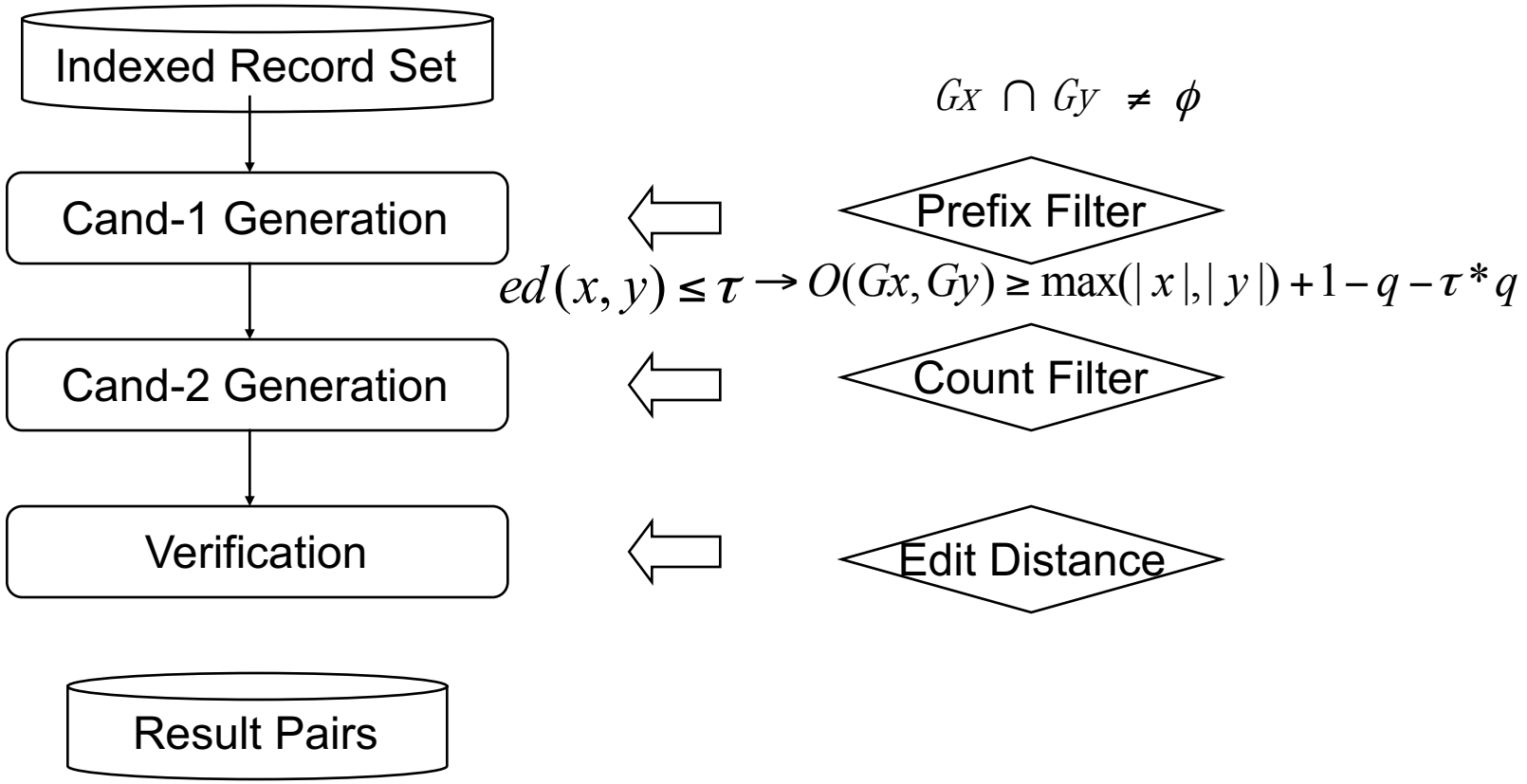
- Edit distance: $ed(x, y) \leq \tau$

$$\rightarrow O(Gx, Gy) \geq \max(|x|, |y|) + 1 - q - \tau * q$$

Gx and Gy are the gram sets of x and y respectively

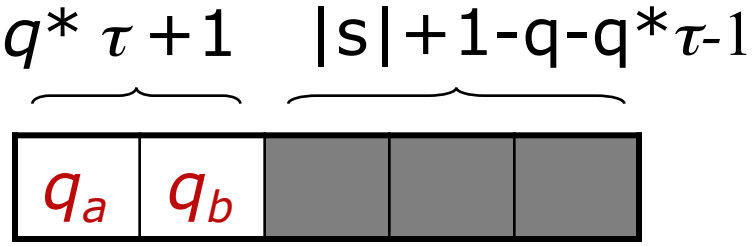
Prefix length = |x|+1-q-(O-1) = $\tau * q + 1$

# Similarity-Join Algorithm for ED

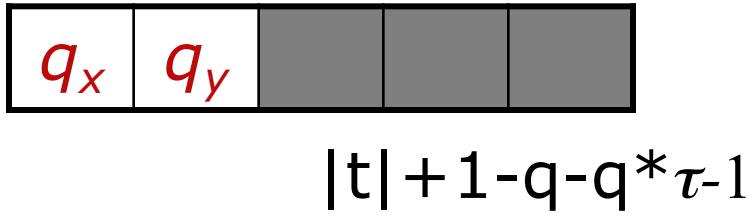


# Prefix Filter

- Prefix Filter
  - sort q-grams by global ordering, e.g., *frequency*
  - $Q_s$ =




  - $Q_t$ =



# Example – All-Pairs-Ed

- $\tau = 1, q = 2$ ⟶ prefix_len = $q*\tau+1$ = 3
  - a='Austria'
  - b='Australia'
  - c='Australiana'
  - d='New_Zealand'
  - e='New_Sealand'

Qa={ri, Au, us, …}
Qb={ra, li, Au, …}
Qc={na, ra, li, …}
Qd={_Z, Ze, Ne, …}
Qe={_S, Se, Ne, …}

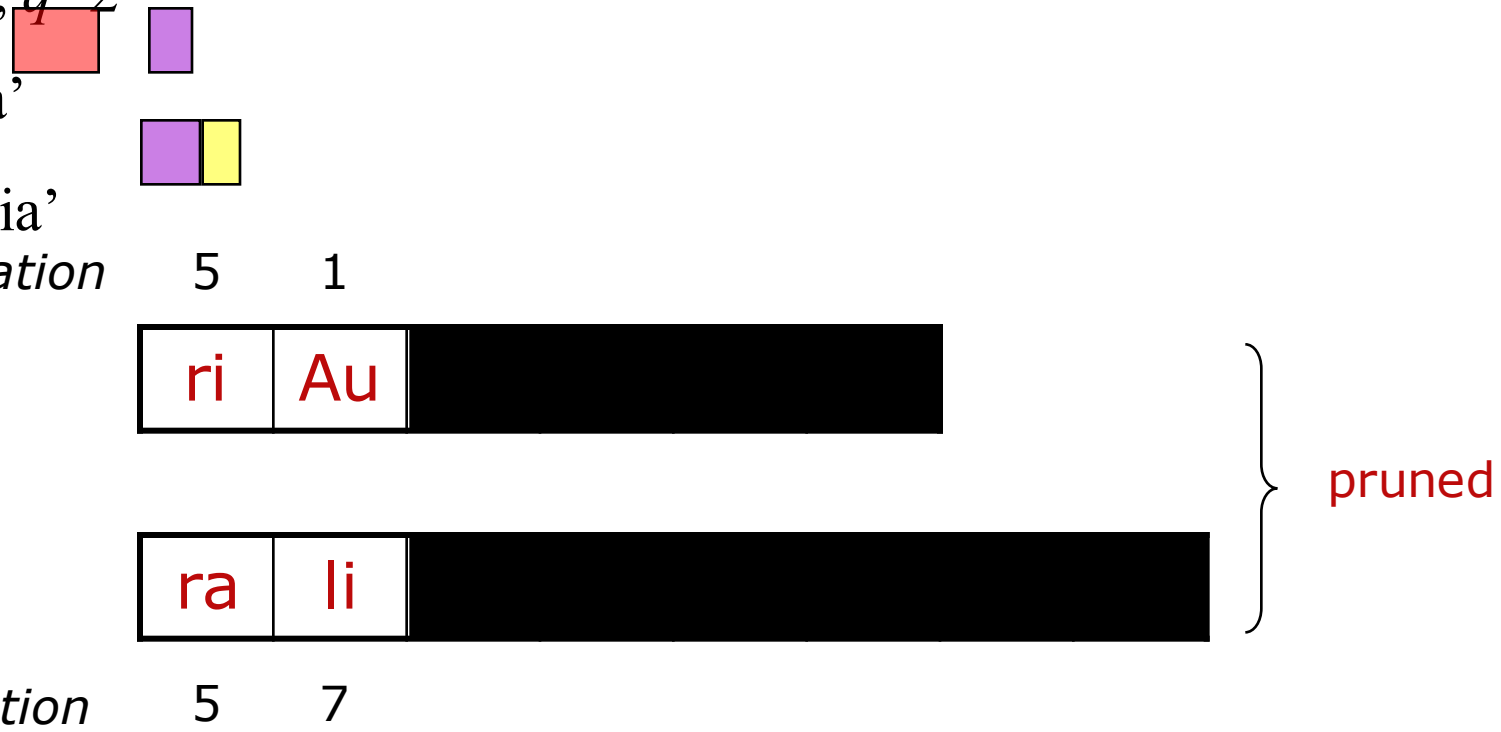
- after prefix filter: <a,b> <b,c> <d,e>
- after count filter: <b,c> <d,e>
- after edit distance verification: <d,e>

# Edit-Distance Join (Ed-Join)

- Idea
  - mismatching $q$-grams provide useful information

# Location-Based Filtering

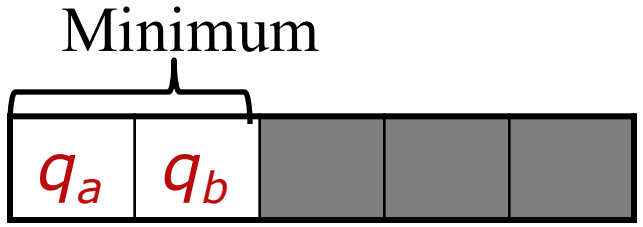
- Idea: reduce prefix length
- Example, $\tau = 1, q = 2$
  - $s$='Austria'
  - $t$='Australia'




  - $Q_s$=
  - $Q_t$=

# Find Minimum Prefix Length

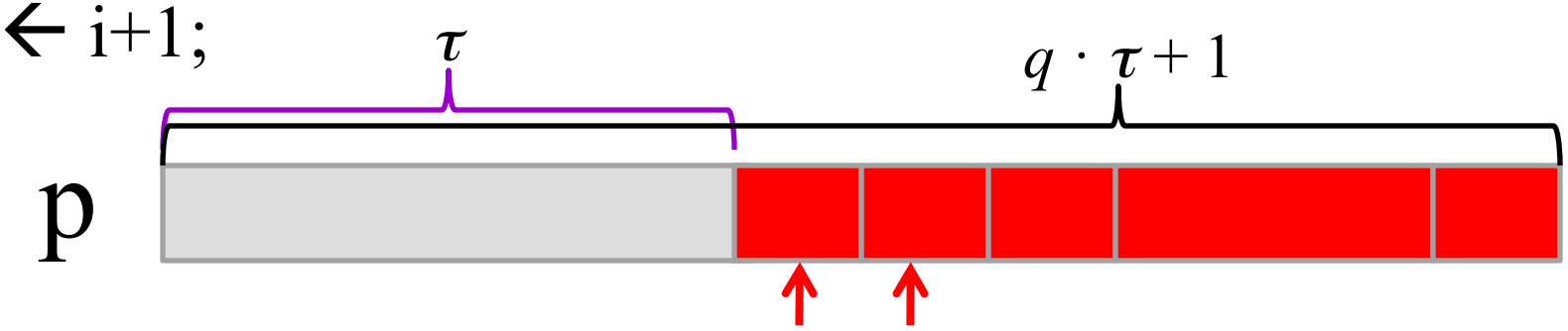
- Given a prefix set of q-grams, find the minimum prefix length that needs at least $\tau + 1$ edit operations to destroy them
- Qs =




- Example
  - *s*='Austria'
  - *t*='Australia'
  - $p_s$ = ri Au us (5 1 2) → ri Au
  - $P_t$ = ra li Au (5 7 1) → ra li

# Algorithms

- Find the minimum prefix length of p
- i ← $\tau$ + 1;  end  ← q · $\tau$ + 1;
- for i <= end do
- err ← MinEditErrors(p[1 . .i]);
- if err > $\tau$ then
- break;
- i ← i+1;

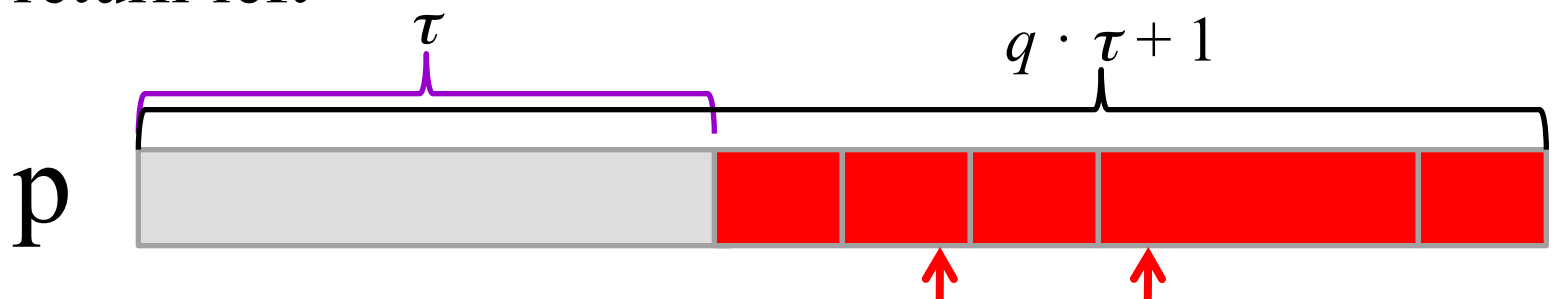


# Algorithms

- Find Minimum Edit Errors of a set of q-grams
- Sort q-grams in Q in increasing order of locations;
- cnt ← 0; loc ← 0;
- for i = 1 to |Q| do
- if Q[i].loc > loc then
- cnt ← cnt + 1;
- loc ← Q[i].loc + q − 1;
- return cnt
  - $p_s$ = ri Au us (5 1 2) → 1 2 5 → 2
  - $P_t$ = ra li Au (5 7 1) → 1 5 7 → 3

# Improved Algorithms

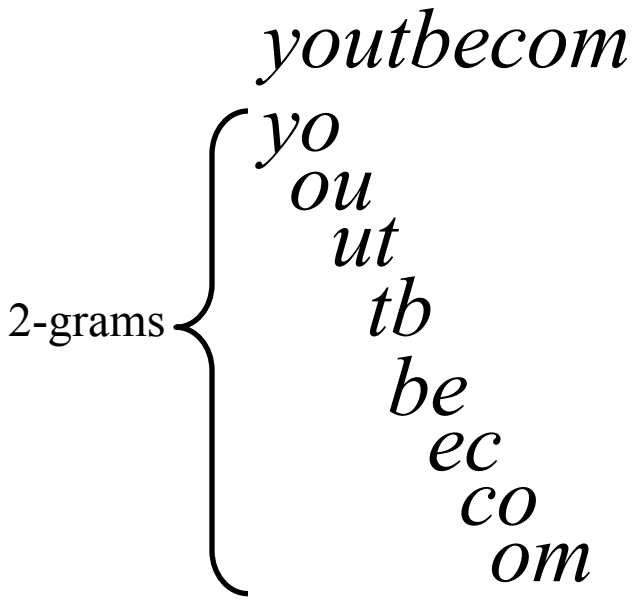
- Find the minimum prefix length
- left ← $\tau$ + 1; right ← q · $\tau$ + 1;
- while left < right do
- mid ← (left + right)/2;
- err ← MinEditErrors(x[1 . .mid]);
- if err <= $\tau$ then left ← mid + 1;
- else right ← mid;
- return left



# Pivotal Prefix Filter

# *q*-gram

- *q*-gram is the substring of length *q*



# *q*-gram

- 1 edit operation destroies at most $q$ grams.

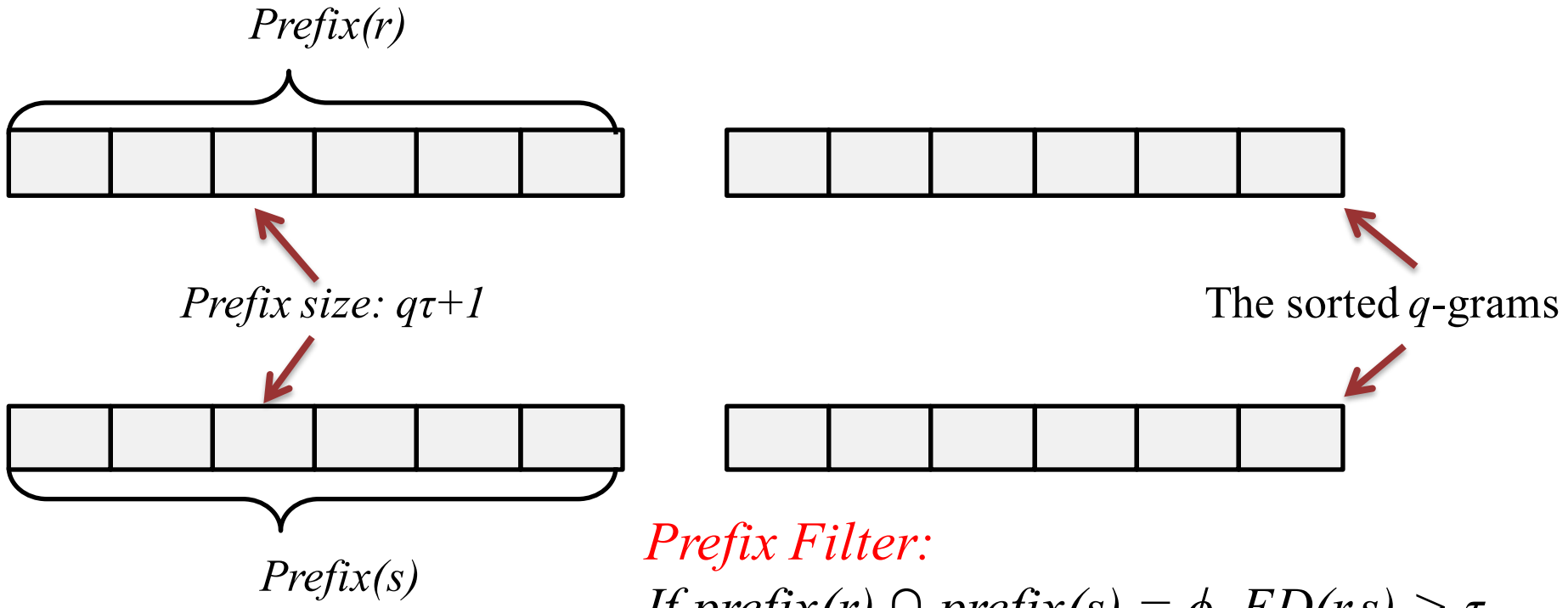
*youdecom*

*yo*
*ou*
*ut*
*td*
*de*
*ec*
*co*
*om*

- $\tau$ edit operations destroy *at most* $q\tau$ q-grams.
- if r and s have more than $q\tau$ mismatch q-grams, ED(r, s)>$\tau$

# Prefix Filter

Sort all q-grams by global ordering, such as *alphabetic order*

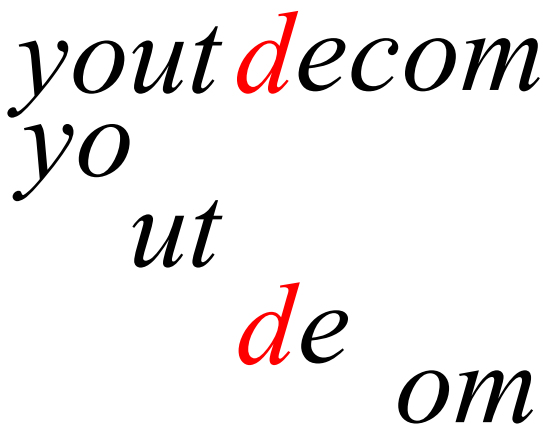



*Prefix Filter:*

*If* $prefix(r) \cap prefix(s) = \phi$, $ED(r,s) > \tau$

# Disjoint q-gram

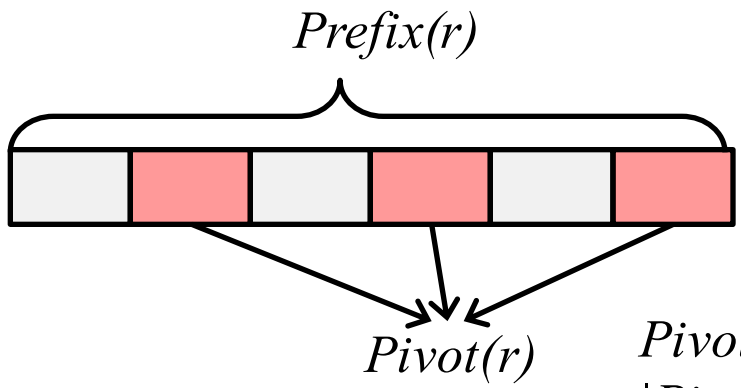
- One edit operation destroies at most *1* disjoint gram.



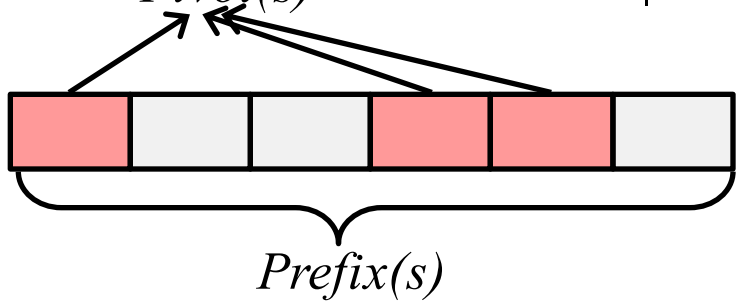

- τ edit operations destroy *at most* τ disjoint q-grams.
- if r and s have more than τ mismatch disjoint q-grams, ED(r, s)> τ

# Pivotal Prefix Filter

Sort all q-grams by global ordering, such as *alphabetical order*




*Pivot(•) is the pivotal prefix*
*|Pivot(•)|= τ+1 and the q-grams in Pivot(•) are disjoint*




*If Pivot(s) ∩ Prefix(r) = ϕ and*
*Pivot(r) ∩ Prefix(s) = ϕ, ED(r,s) > τ*

# Summary

- Prefix Filtering for Set Similarity Join
- Position Filter
- Suffix Filter
- Partition-based Framework
- 1-Deletion Neighborhood
- Optimal Allocation Algorithm
- 2-approximation Greedy Algorithm
- Adaptive Grouping Mechanism
- Prefix Filtering for Edit Distance
- Pivotal Prefix Filter