Similarity Search

Windowed *pq*-Grams for Unordered Trees

Nikolaus Augsten
nikolaus.augsten@sbg.ac.at
Department of Computer Sciences
University of Salzburg

database
research group
http://dbresearch.uni-salzburg.at

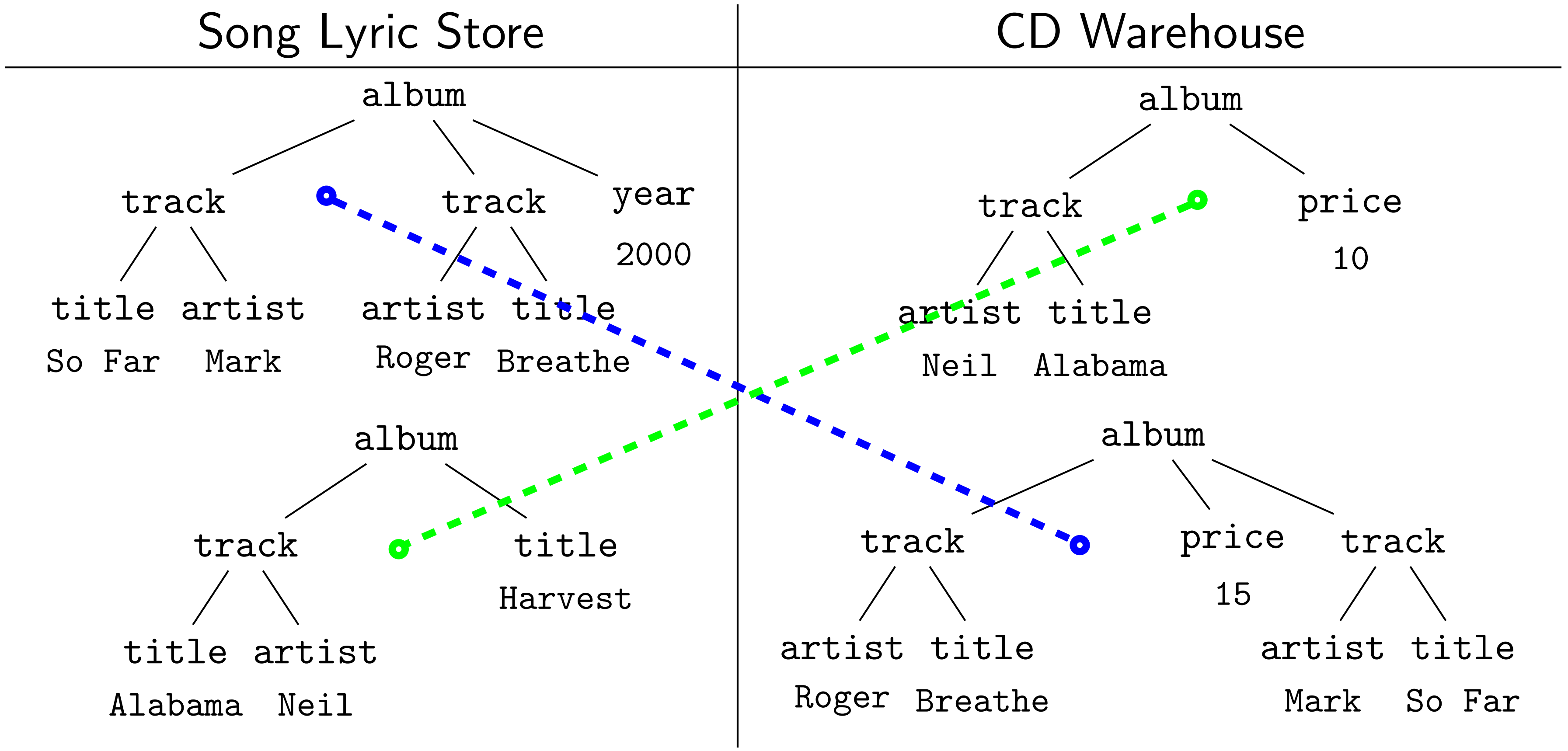
WS 2019/20

Version February 10, 2020

Outline

Outline

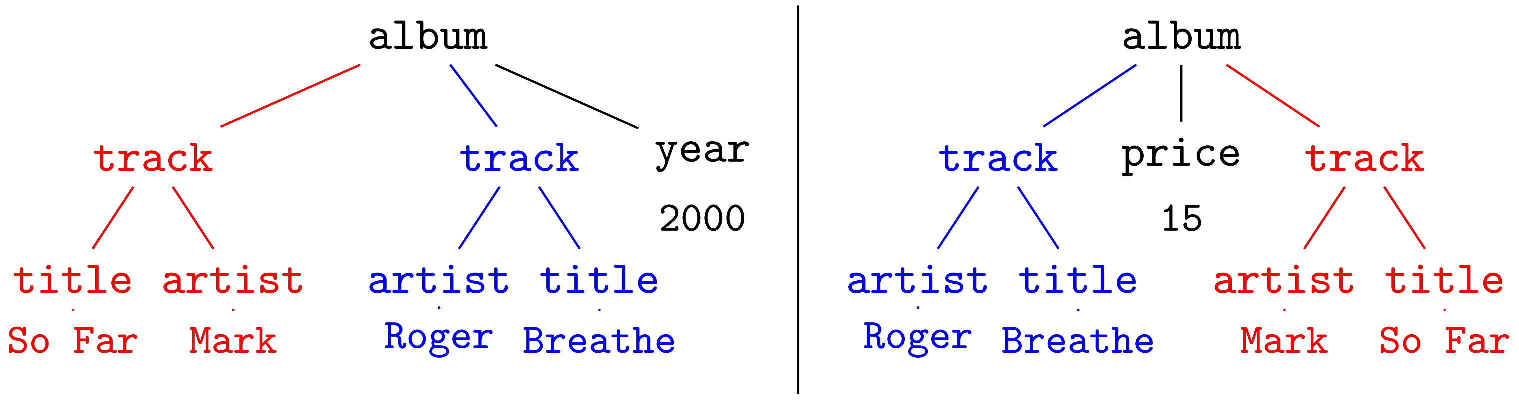
Similarity Join on Music CDs



- Query: Give me all album pairs that represent the same music CDs.

How similar are two XML items?

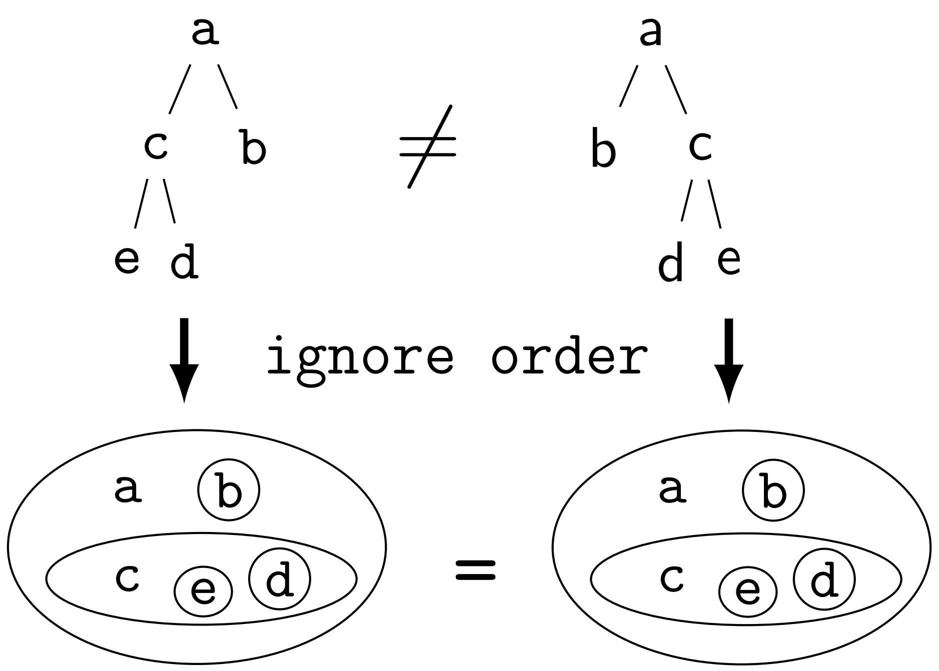
How Similar Are these XMLs?



- Standard solution $\mathbf{O(n^3)}$: tree edit distance
  Minimum number of node edit operations (insert, delete, rename) that transforms one ordered tree into the other.
- Problem: permuted subtrees are deleted/re-inserted node by node

Ordered vs. Unordered Trees

Ordered Trees
sibling order matters



- Edit distance between unordered trees: NP-complete
  $\rightarrow$ all sibling permutations must be considered!

Problem Definition

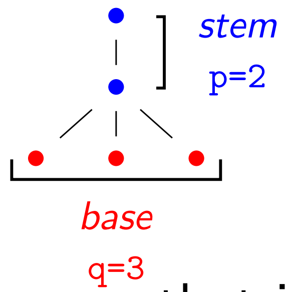
Find an effective distance for unordered labeled trees that is efficient for similarity joins.

Naive approaches that fail:

- unordered tree edit distance: NP-complete
- allow subtree move: NP-hard
- compute minimum distance between all permutations: runtime $O(n!)$, error $O(n)$
- sort by label and use ordered tree edit distance: error $O(n)$

Outline

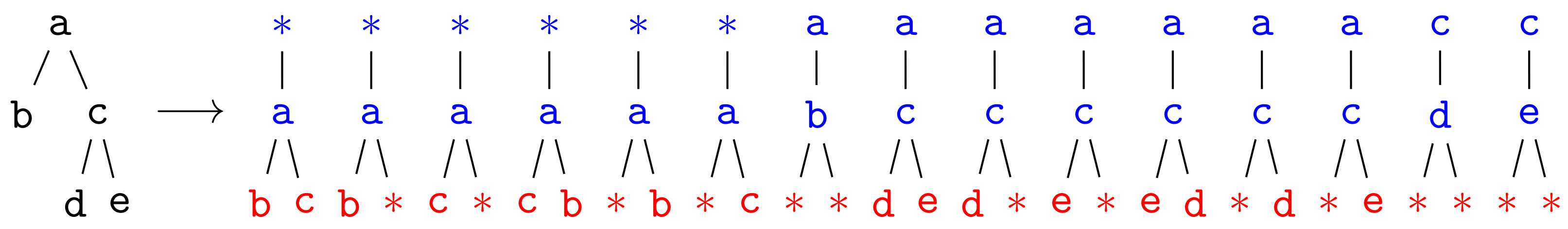
Solution: Windowed *pq*-Grams [ABDG08]



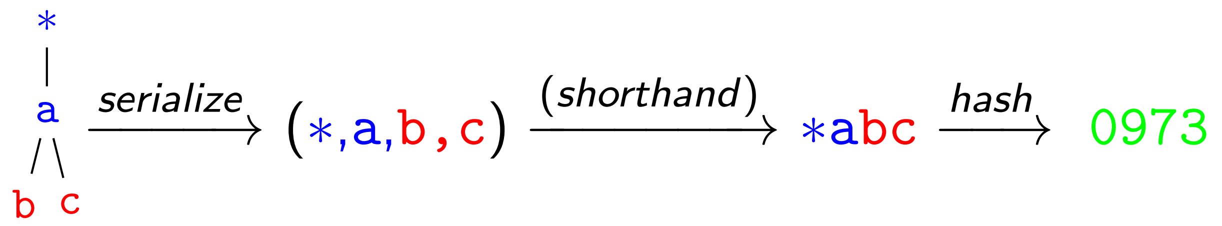
- Windowed pq-Gram: small subtree with stem and base
- Key Idea: split unordered tree into set of windowed *pq*-grams that is
  - not sensitive to the sibling order
  - sensitive to any other change in the tree
- Intuition: similar unordered trees have similar windowed *pq*-grams
- Systematic computation of windowed *pq*-grams
  1. sort the children of each node by their label (works OK for *pq*-grams)
  2. simulate permutations with a window
  3. split tree into windowed *pq*-grams

Implementation of Windowed *pq*-Grams

- Set of windowed *pq*-grams:



- Hashing: map *pq*-gram to integer:



label l	h(l)
∗	0
a	9
b	7
c	3
...	...

Note: labels may be strings of arbitrary length!

- Windowed *pq*-gram index: bag of hashed *pq*-grams

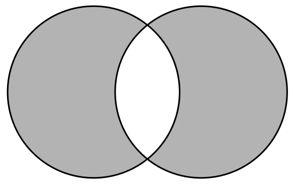
$$\mathcal{I}(\mathbf{T}) = \{0973, 0970, 0930, 0937, 0907, 0903, 9700, 9316, 9310, 9360, 9361, 9301, 9306, 3100, 3600\}$$

$\boxed{\text{Tree is represented by a bag of integers!}}$

The Windowed *pq*-Gram Distance

- The windowed *pq*-gram distance between two trees, **T** and **T**′:

$$\mathrm{dist}^{pq}(\mathbf{T},\mathbf{T}') = |\mathcal{I}(\mathbf{T}) \uplus \mathcal{I}(\mathbf{T}')| - 2|\mathcal{I}(\mathbf{T}) \Cap \mathcal{I}(\mathbf{T}')|$$



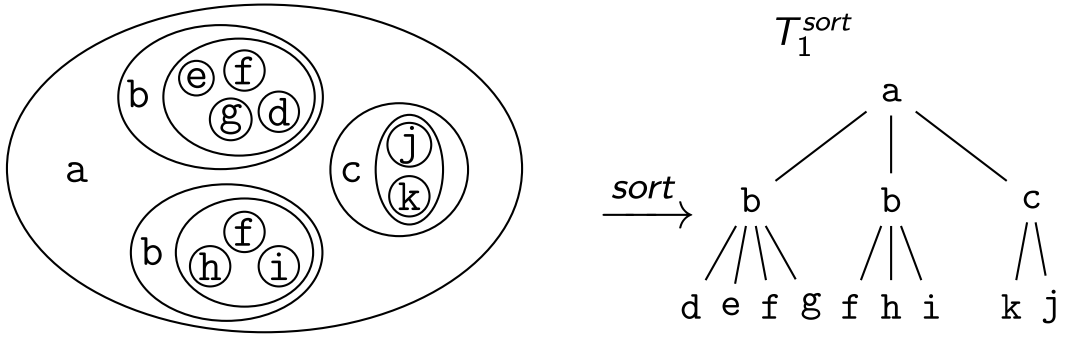
- Pseudo-metric properties hold:
  - ✓ self-identity: $x = y \cancel{\Leftarrow} \Rightarrow \mathrm{dist}^{pq}(x,y) = 0$
  - ✓ symmetry: $\mathrm{dist}^{pq}(x,y) = \mathrm{dist}^{pq}(y,x)$
  - ✓ triangle inequality: $\mathrm{dist}^{pq}(x,z) \leq \mathrm{dist}^{pq}(x,y) + \mathrm{dist}^{pq}(y,z)$

```
  b      b
  |     / \
  b    b   b
 / \   |
b   b  b
```

- Different trees may be at distance zero:
- Runtime for the distance computation is $O(n \log n)$.

Sorting the Tree?

- Idea:
  1. sort the children of each node by their label
  2. apply an ordered tree distance

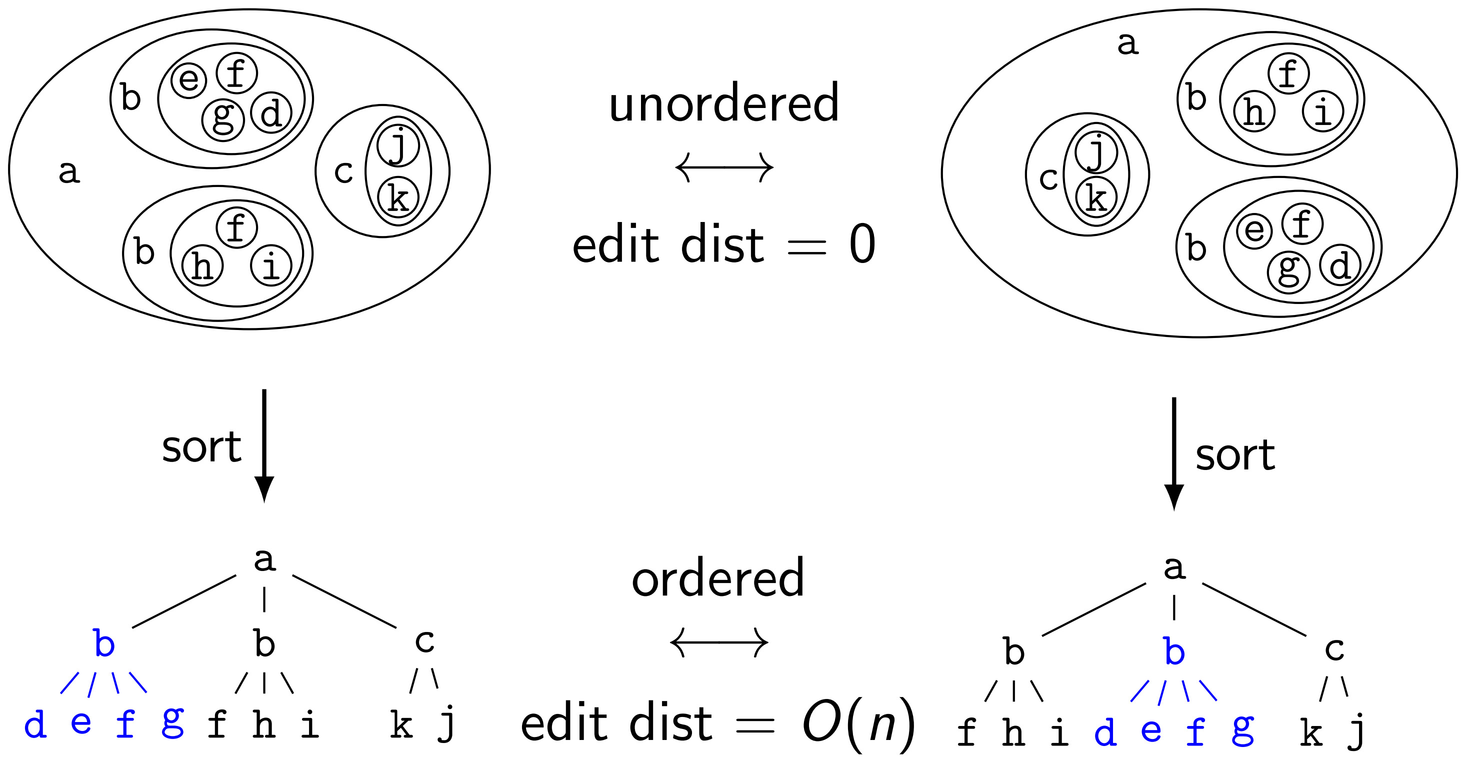


✘ Edit distance: tree sorting does not work

✓ Windowed *pq*-Grams: tree sorting works OK

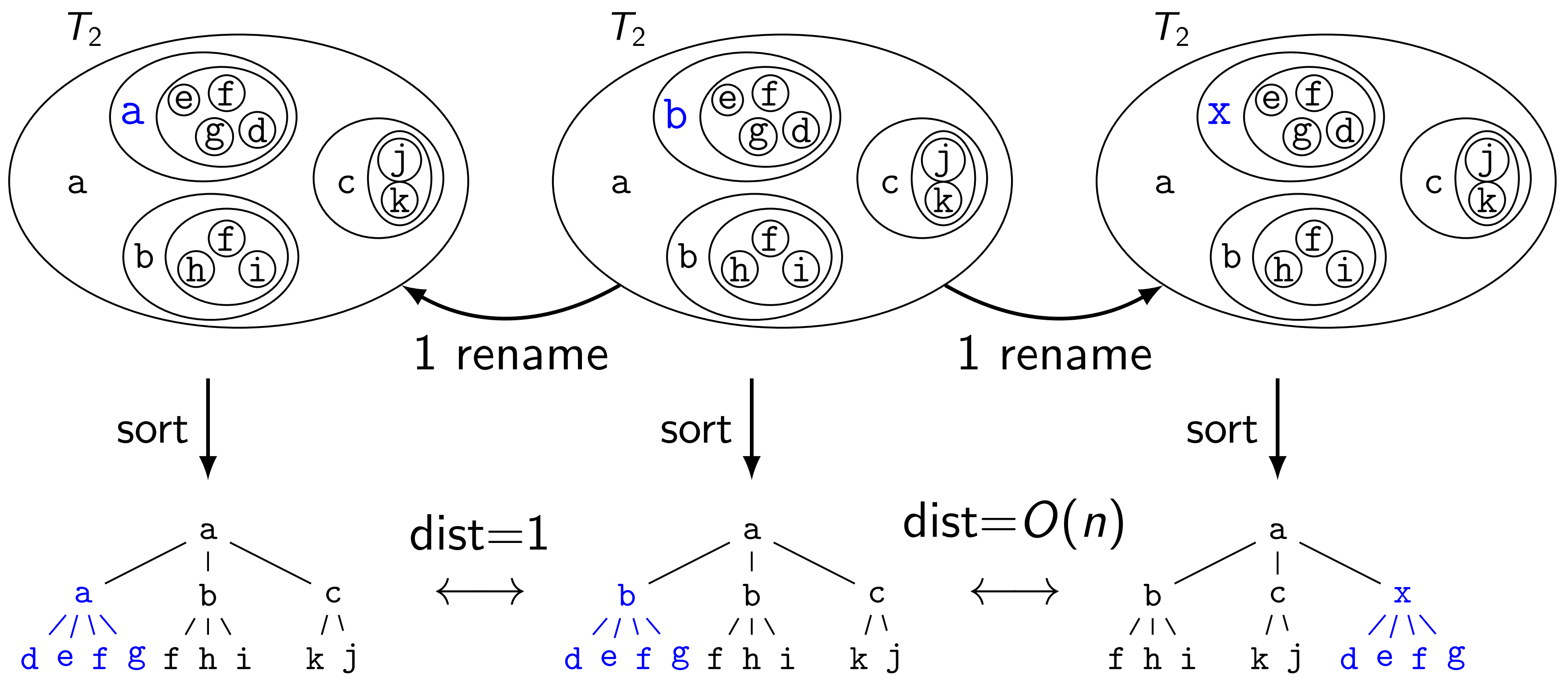
✘ Edit Distance: Tree Sorting Does Not Work

1. Non-unique sorting: edit distance $O(n)$ for identical trees



✘ Edit Distance: Tree Sorting Does Not Work

2. Node renaming: edit distance depends on node label

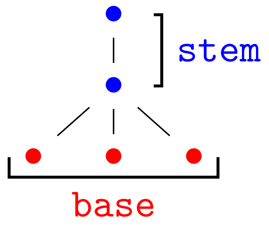


✓ Windowed *pq*-Grams: Tree Sorting Works OK

Theorem (Local Effect of Node Reordering)

If k children of a node are reordered, i.e., their subtrees are moved, only $O(k)$ windowed pq-grams change.

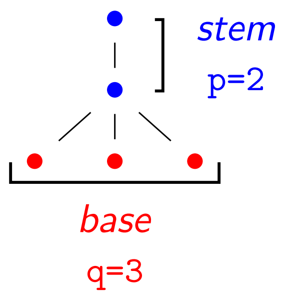
- Proof (idea):
  - *pq*-grams consist of a stem and a base
  - stems are invariant to the sibling order
  - bases: only the $O(k)$ *pq*-grams with the reordered nodes in the bases change



✓ Non-unique sortings are equivalent: distance is 0 for identical trees

✓ Node renaming is independent of the node label

How To Form Bases?



- Goal for windowed *pq*-grams:
  - not sensitive to the sibling order
  - sensitive to any other change in the tree
- Stems: ignore sibling order

```
  a
 / \
b   c            *  a  a  c  c
   / \    ⟶     |  |  |  |  |
  d   e          a  b  c  d  e
```

- Bases: do not ignore sibling order!

Requirements for Bases

- Requirements for bases:
  - detection of node moves
  - robustness to different sortings
  - balanced node weight
- Solution:
  - windows: simulate all permutations within a window
  - wrapping: wrap windows that extend beyond the right border
  - dummies: extend small sibling sets with dummy nodes

Solution: Windowed *pq*-Gram Bases

Algorithm 1: Form bases from a sorted (non-empty) sibling sequence

```
1 if sibling sequence < window then extend with dummy nodes;
2
3 initialize window: start with leftmost node;
4 repeat
5     form bases in window: all q-permutations that contain start node;
6     shift window to the right by one node;
7     if window extends the right border then wrap window;
8
9 until processed all window positions
```

- Example: stem, sorted sibling sequence, window $w = 3$

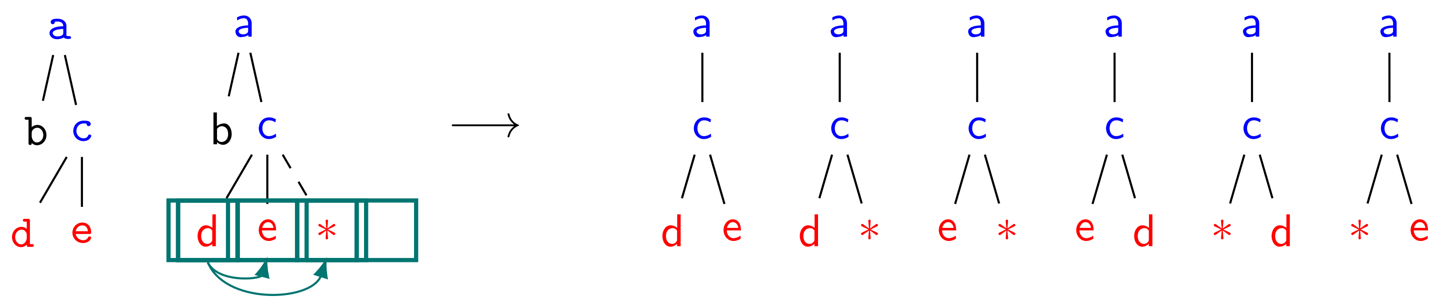
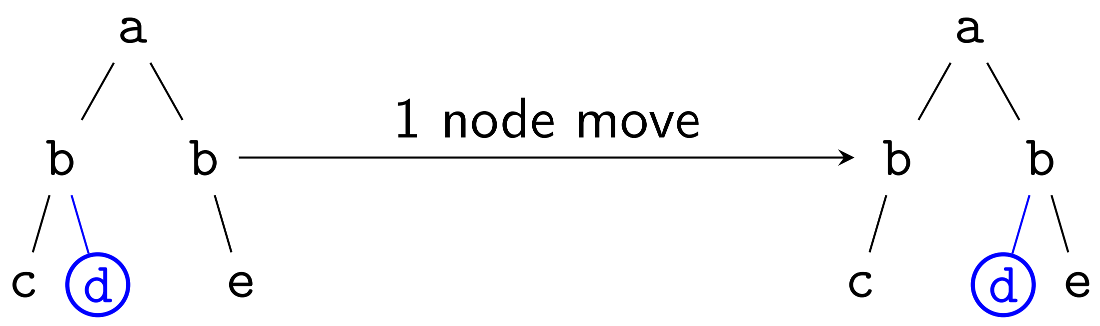


Illustration: Detection of Node Moves

- Single Node: each node forms a base of size $q = 1$
- Window: $q \geq 2$ nodes of a window form a base

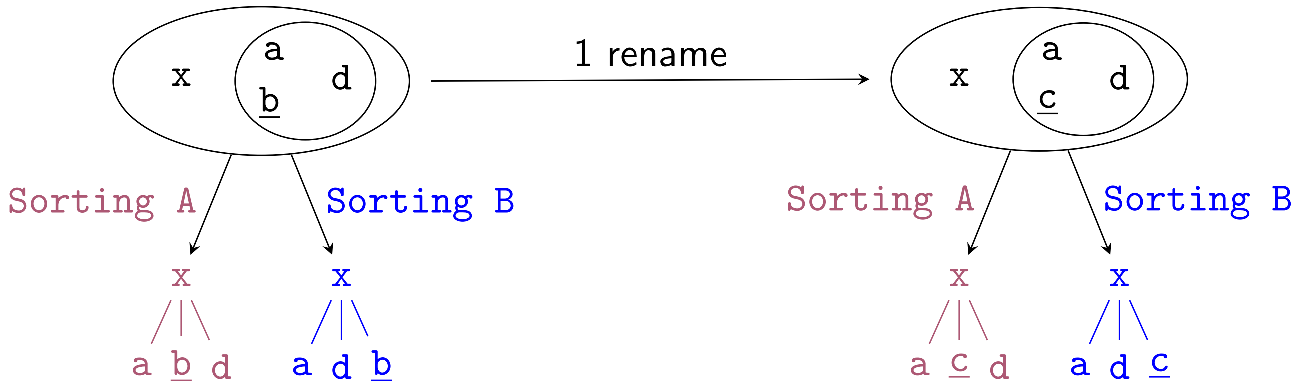


Goal:		bases must change	
✘ **Single Node:**	c, d, e	no bases change	c, d, e
✓ **Window:**	cd, c*, d*, dc, *c, *d, e*, ...	33% bases change	c*, c*, **, *c, *c, **, de, ...

Windowed *pq*-grams detect node moves.

Illustration: Robustness to Different Sortings

- Consecutive siblings form a base (no permutation)
- Window: all sibling permutations within the window form bases



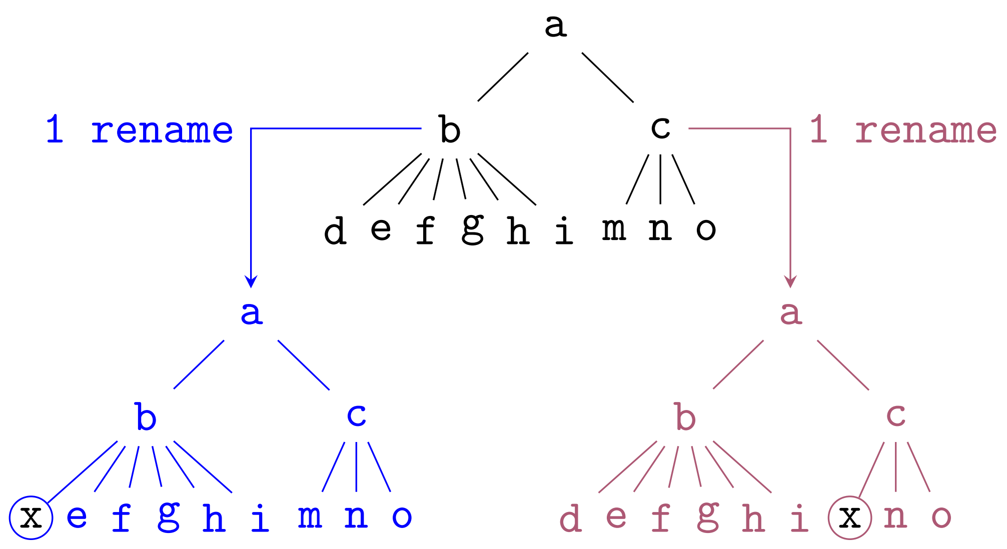
Goal: Same number of bases change for both sortings.

✘ **Consecutive:**	Sort A	a<u>b</u> <u>b</u>d	100% bases change	a<u>c</u> <u>c</u>d
	Sort B	ad d<u>b</u>	50% bases change	ad d<u>c</u>
✓ **Window:**	Sort A	ad a<u>b</u> d<u>b</u>...	33% bases change	ad a<u>c</u> d<u>c</u>...
	Sort B	ad a<u>b</u> d<u>b</u>...	33% bases change	ad a<u>c</u> d<u>c</u>...

Windowed *pq*-grams: Robust to different sortings.

Illustration: Balancing the Node Weight

- Permutations: all permutations of size q form a base
- Window: only permutations within window form a base

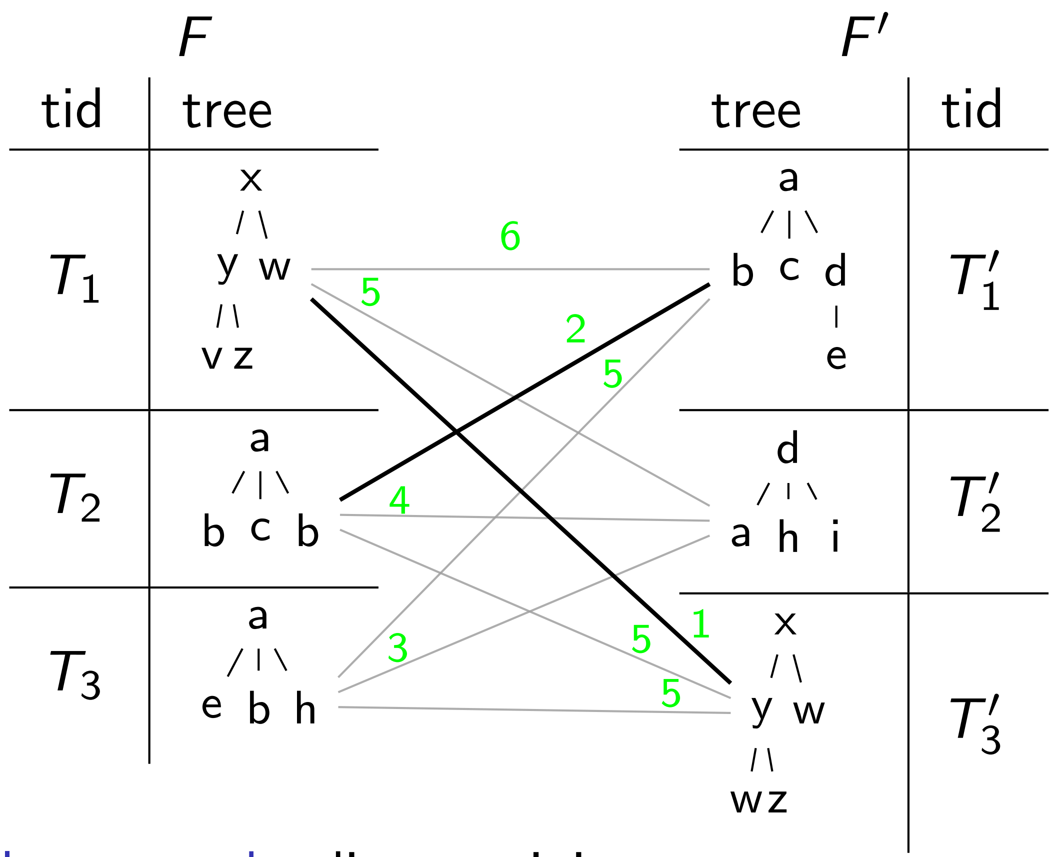


Goal:	Same number of bases change for both renames.	
✘ **Permutations:**	60/137 bases change	6/137 bases change
✓ **Window:**	12/51 bases change	12/51 bases change

Windowed *pq*-grams: Node weight is independent of sibling number.

Outline

Similarity Join



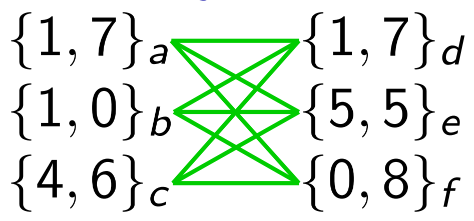
- Simple approach: distance join
  1. compute distance between all pairs of trees
  2. return document pairs within threshold
- Very expensive: N^2 distance computations!

Usual Join Optimization Does not Apply

- Distance join: expensive
  - nested loop join: evaluate distance function between every input pair
- Equality join: efficient
  - implementation as sort-merge or hash join
- Sort-merge and hash join:
  - *first step:* treat each join attribute in isolation (sort/hash)
  - *second step:* evaluate equality function
- Sort-merge and hash not applicable to distance join:
  - there is no sorting that groups similar trees
  - there is no hash function that partitions similar trees into buckets
- Solution: reduce distance join to equality join on *pq*-grams

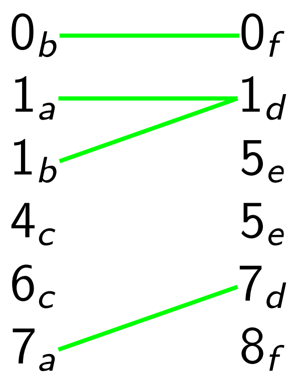
Reducing a Distance Join to an Equality Join

- Distance join between trees: N^2 intersections between integer bags



$$|a \cap d| = 2 \quad |a \cap e| = 0 \quad |a \cap f| = 0$$
$$|b \cap d| = 1 \quad |b \cap e| = 0 \quad |b \cap f| = 1$$
$$|c \cap d| = 0 \quad |c \cap e| = 0 \quad |c \cap f| = 0$$

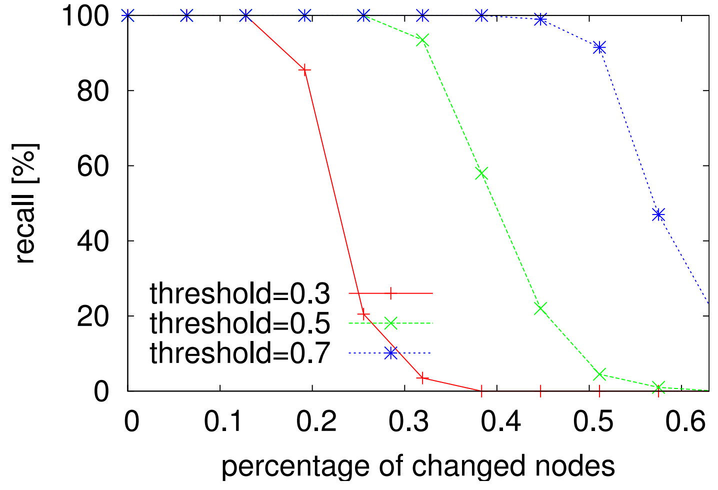
- Optimized *pq*-gram join: empty intersections are never computed!
  1. union
     $\{1_a, 7_a, 1_b, 0_b, 4_c, 6_c\}$ $\{1_d, 7_d, 5_e, 5_e, 0_f, 8_f\}$
  2. sort
  3. merge-join

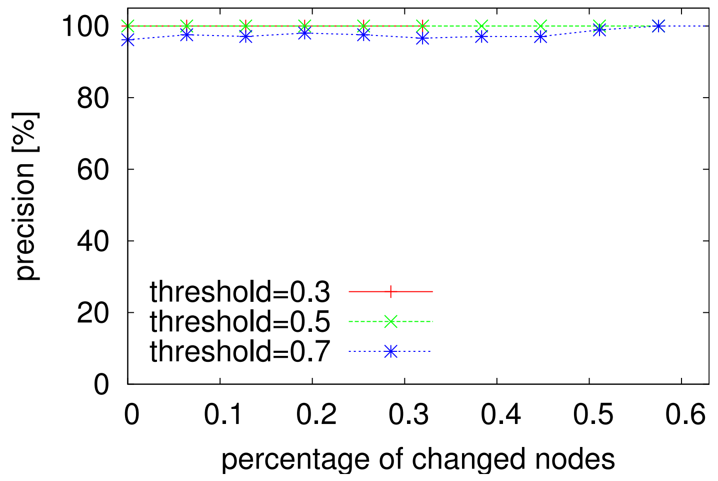


$|b \cap f|$ I
$|a \cap d|$ II
$|b \cap d|$ I

Outline

Effectiveness of the Windowed *pq*-Gram Join





Experiment: match **DBLP** articles

- add noise to articles (missing elements and spelling mistakes)
- similarity join between original and noisy data
- measure precision and recall for different thresholds

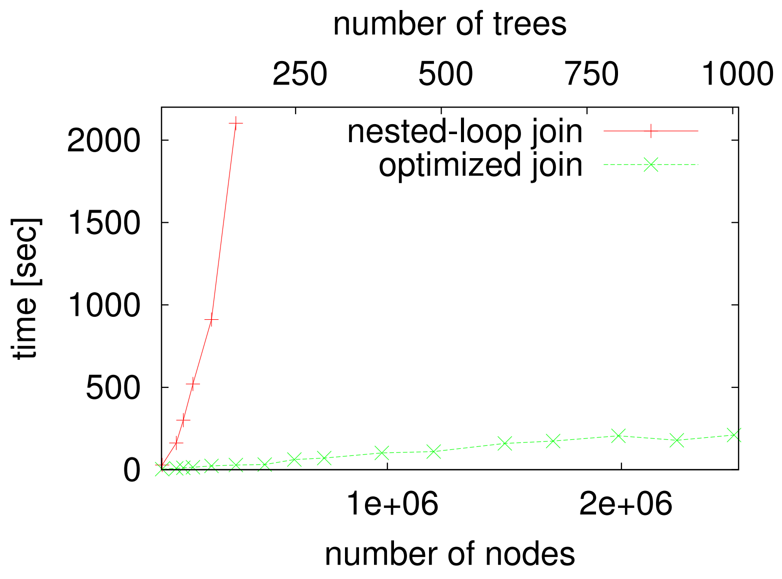
Datasets:

- **DBLP:** articles depth 1.9, 15 nodes (max 1494 nodes)
- **SwissProt:** protein descriptions depth 3.5, 104 nodes (max 2640 nodes)
- **Treebank:** tagged English sentences depth 6.9 (max depth 30), 43 nodes

Windowed *pq*-grams are effective for data-centric XML

Efficiency of the Optimized *pq*-Gram Join

Optimized *pq*-gram join: very efficient



- compute nested-loop join between trees
- compute optimized *pq*-gram join between trees
- measure wallclock time

Distances between Unordered Trees

Edit Distances between Unordered Trees

- [ZSS92]: proof for NP-completeness
- [KKSS04]: lower bound for a restricted edit distance
- [CGM97]: $O(n^3)$ heuristics
- Our solution: $O(n \log n)$ heuristics

Similarity Join

- [GIJ$^+$01]: efficient similarity join for strings

Nikolaus Augsten, Michael Böhlen, Curtis Dyreson, and Johann Gamper.
Approximate joins for data-centric XML.
In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 814–823, Cancún, Mexico, April 2008. IEEE Computer Society.

Sudarshan S. Chawathe and Hector Garcia-Molina.
Meaningful change detection in structured data.
In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 26–37, Tucson, Arizona, United States, May 1997. ACM Press.

Luis Gravano, Panagiotis G. Ipeirotis, H. V. Jagadish, Nick Koudas, S. Muthukrishnan, and Divesh Srivastava.
Approximate string joins in a database (almost) for free.
In *Proceedings of the International Conference on Very Large Databases (VLDB)*, pages 491–500, Roma, Italy, September 2001. Morgan Kaufmann Publishers Inc.

Karin Kailing, Hans-Peter Kriegel, Stefan Schönauer, and Thomas Seidl.
Efficient similarity search for hierarchical data in large databases.
In *Proceedings of the International Conference on Extending Database Technology (EDBT)*, volume 2992 of *Lecture Notes in Computer Science*, pages 676–693, Heraklion, Crete, Greece, March 2004. Springer.

Kaizhong Zhang, Richard Statman, and Dennis Shasha.
On the editing distance between unordered labeled trees.
Information Processing Letters, 42(3):133–139, 1992.