PS Ähnlichkeitssuche in großen Datenbanken

Task 2 - Prefix Filter

Thomas Hütter, WS 2020

Task 1

Length Filter

- Collection R ordered by size and global order among all set elements.

```
Algorithm 1: setsimjoin(R, t_J):
input: R collection of sets, t_J similarity threshold
output: res set of result pairs (similarity at least t_J)
res = [];    //stores the result pairs
for i = 0 to |R| - 1:
    for j = i + 1 to |R| - 1:
        if |R[j]| is to big for |R[i]| to reach t_J:
            break;
        if (R[i] ∩ R[j] / R[i] ∪ R[j]) ≥ t_J:
            res = res U (i, j);
return res;
```

Exercise

Complexity Analysis

- Given: function verify(r, s, t_J) from Algorithm 1.
- Task: analyze the runtime complexity.

Exercise

Set Similarity Self Join Algorithm

- Given: two sets r and s with |r| = 100 resp. |s| = 100 and Jaccard threshold $t_J = 0.95$.

r = | 7 | 10 | 24 | 32 | 44 | 57 | 61 | 72 | 84 | 90 | ⋯

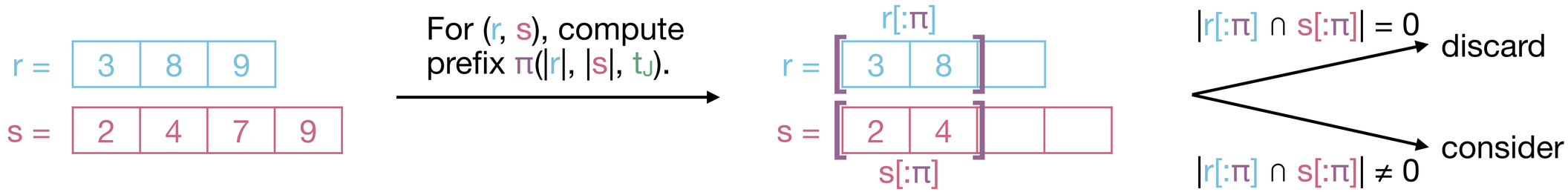
s = | 1 | 13 | 22 | 37 | 46 | 51 | 62 | 75 | 87 | 94 | ⋯

- Task: argue whether the pair (r, s) may (or not) be part of the result.

Prefix Filter

Optimization

- Motivation: verifying large set pairs is expensive.
- Idea: discard a pair without looking at all elements.
- Prefix filter: if there is no common element between the first π elements (arbitrary but fixed order) of sets r and s, denoted $r[:\pi]$ resp. $s[:\pi]$, then (r, s) cannot be part of the result. π depends on the set sizes and threshold t_J.



Pre-processing

Inverted Token Frequency

- Goal: introduce a common input format that can be processed efficiently and leveraged to optimize the algorithm.
- Idea: optimize prefix filter by having infrequent elements first.

Original Dataset

Hobbies
{guitar, swimming}
{hiking, guitar, singing}
{singing, guitar, swimming}
{guitar, skiing, swimming, hiking}

Inverted token frequency

1x skiing → 1, 2x hiking → 2, 2x singing → 3,
3x swimming → 4, 4x guitar → 5

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

Sort each set

Pre-processed Dataset

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

Homework

Task 2

- Add the Prefix Filter and implement verify(r, s, t_J).
  - Correctness: perform all tests within 45 seconds.
  - Compute the minimal prefix length. (1 point)
  - Efficient integration of the prefix filter in algorithm. (0.5 point)
- Verify your implementation with given datasets on the teaching website. Further datasets can be found at http://ssjoin.dbresearch.uni-salzburg.at/datasets.html.
- Follow the submission guidelines written on the teaching website.
- Submit via abgaben.cosy.sbg.ac.at until 24.11.2020, 23:55.