
# The Similarity Graph: Analyzing Database Access Patterns Within A Company

## PhD Preliminary Examination

Robert Searles

Committee

John Cavazos and Stephan Bohacek

University of Delaware

Computer and Information Sciences

In collaboration with JP Morgan Chase & Co.

UNIVERSITY OF DELAWARE

# What are Many Employees Doing?

- SQL queries!
- Some employees access the database
- Queries reveal information about employees

# Motivation

- Workforce efficiency
- Optimize Database
- Redundant work?
- New collaborations?

# Solution

- Find computationally friendly representations of queries
  - Directed graphs (trees)

- Graph similarity techniques and Visualization
  - Find relationships between users
  - Find relationships between business units

# What do SQL Queries Look Like?

- A query is a string of text
- SELECT <clause> FROM <clause> WHERE <clause>

# SQL Query Example 1

- **SELECT** A, B, C AS c, function1(D) AS d, E AS e, F AS f
  **FROM** table1_name
  **WHERE** table1_name.c1 ≤ 'STR' and table1_name.c1 < 'STR' AND (G='STR' AND C IN ('STR', 'STR' ))

# SQL Query Example 2

- **SELECT** A, H, E, C, F, G, I, D

  **FROM** table1_name

  **WHERE** table1_name.c1 ≤ 'STR' and table1_name.c1 < 'STR' AND (I>function2(p1, 'STR') AND I≤ function2(p1, 'STR'))

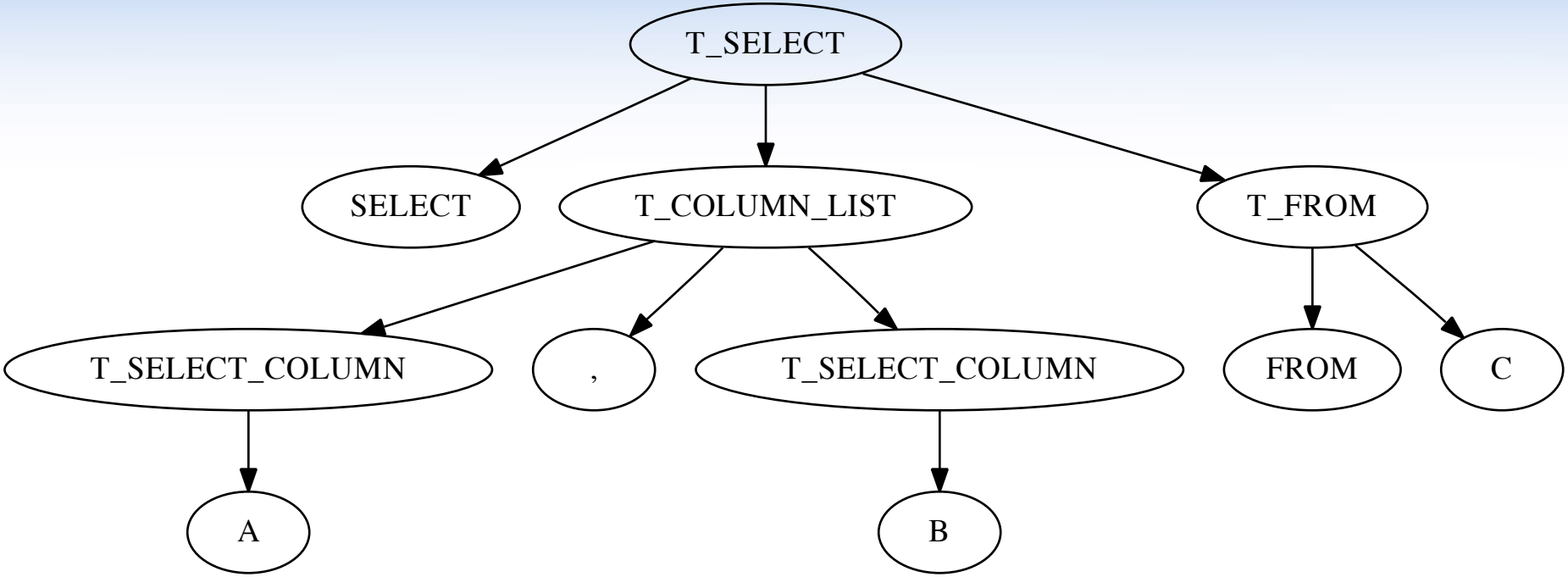
# Similarity Between SQL Queries

- **SELECT** A, B, C AS c, function1(D) AS d, E AS e, F AS f
  **FROM** table1_name
  **WHERE** table1_name.c1 ≤ 'STR' and table1_name.c1 < 'STR' AND (G='STR' AND C IN ('STR', 'STR' ))
- **SELECT** A, H, E, C, F, G, I, D
  **FROM** table1_name
  **WHERE** table1_name.c1 ≤ 'STR' and table1_name.c1 < 'STR' AND (I>function2(p1, 'STR') AND I≤ function2(p1, 'STR'))

# Representing SQL Queries

- Represent as graphs (parse trees)
  - No cycles
- Graphs are a more powerful representation
  - Similarity calculations
  - Structural relationships
- Graphs are visually appealing

# SQL Parse Tree Example




SELECT A,B FROM C

# Weisfeiler-Lehman

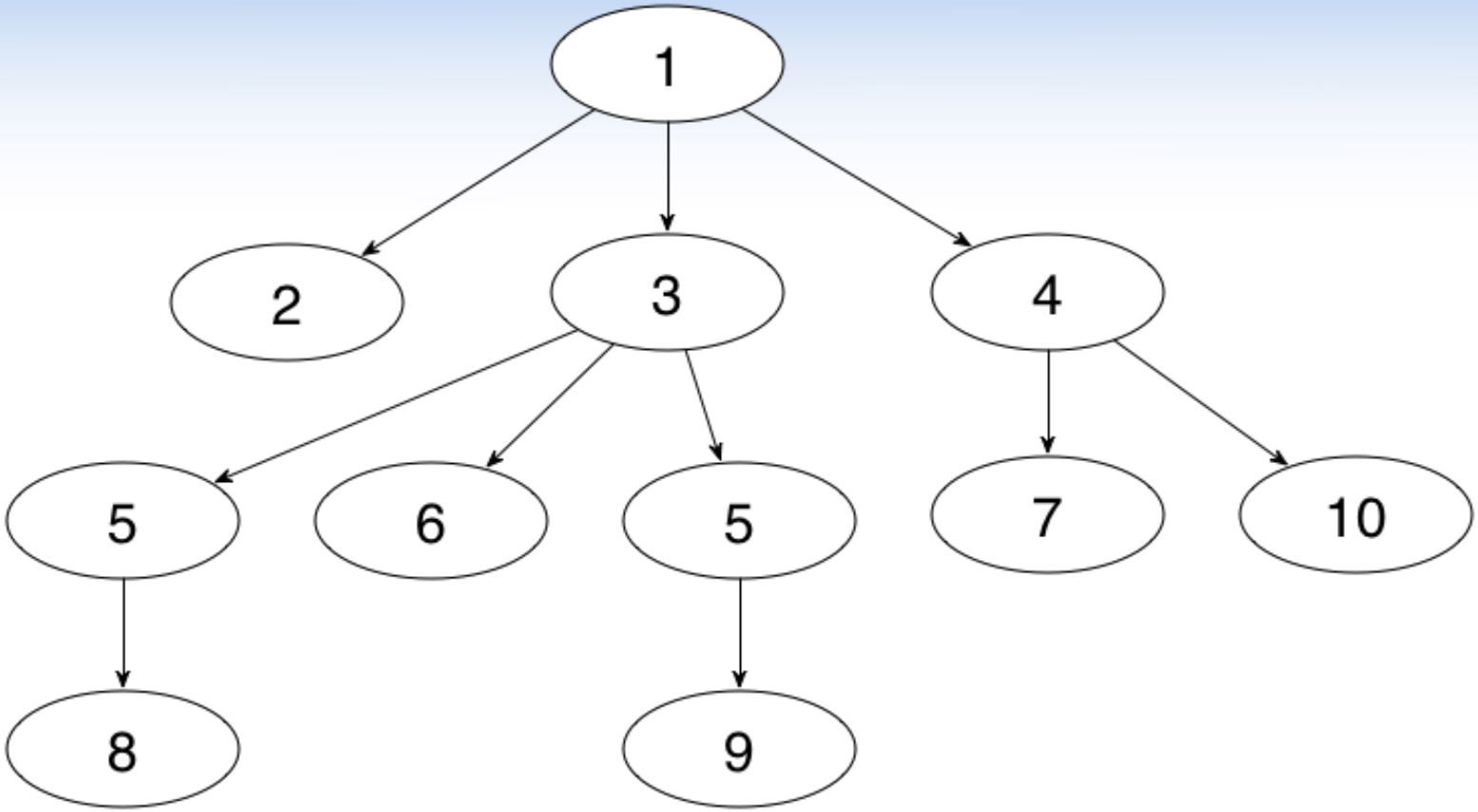
- Modified to work on ordered trees
  - Relabel trees according to global alphabet
  - Look for common labels and/or structures
  - Perform subtree analysis on pairs of trees

[4] N. Shervashidze, P. Schweitzer, E. J. van Leeuwen, K. Mehlhorn, and K. M. Borgwardt, "Weisfeiler-lehman graph kernels," *J. Mach. Learn. Res.*, vol. 999888, pp. 2539–2561, Nov. 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=2078183.2078187

# Global Alphabet

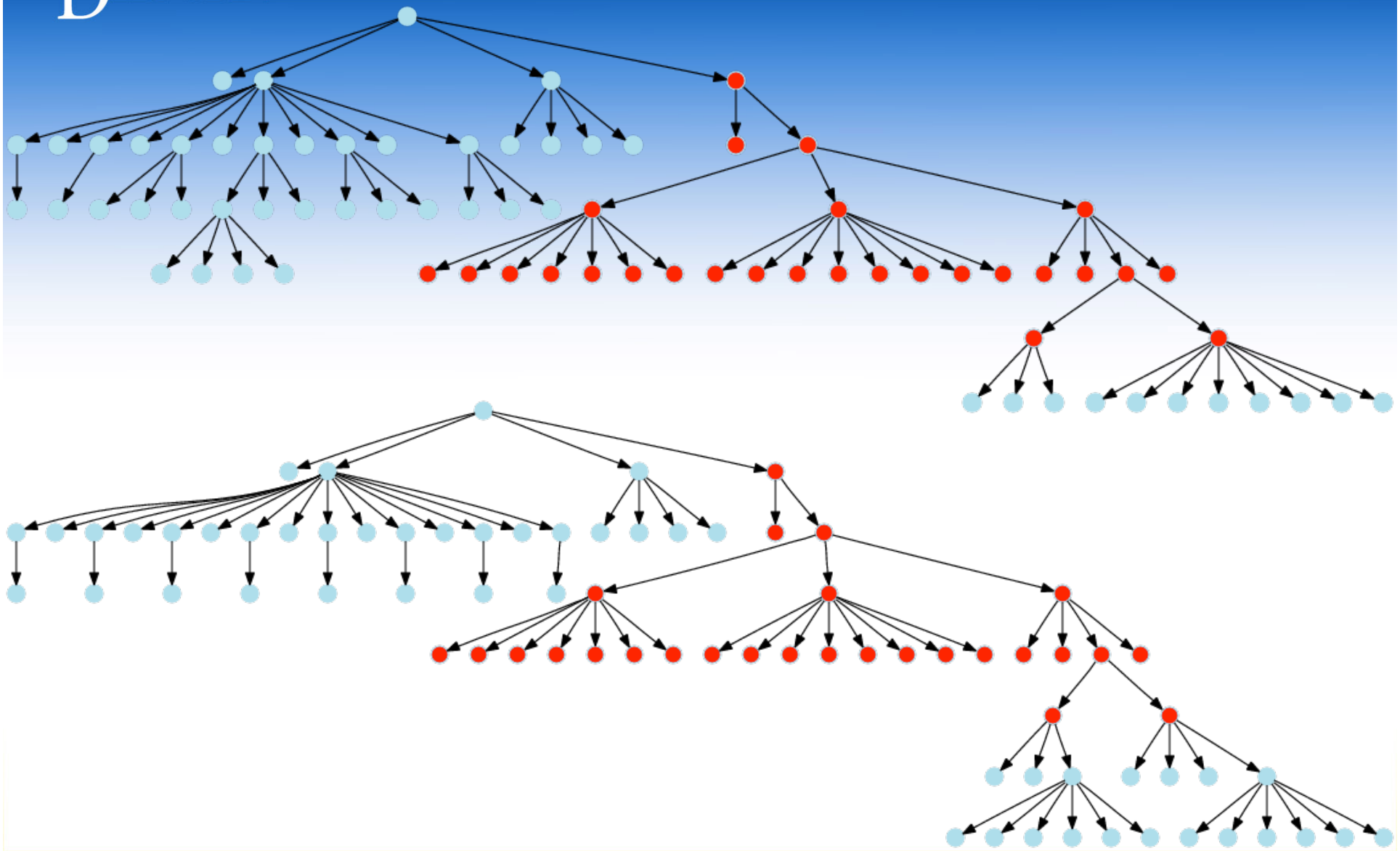
| | |
|---|---|
| • T_SELECT | 1 |
| • SELECT | 2 |
| • T_COLUMN_LIST | 3 |
| • T_FROM | 4 |
| • T_SELECT_COLUMN | 5 |
| • , | 6 |
| • FROM | 7 |
| • A | 8 |
| • B | 9 |
| • C | 10 |

# SELECT A,B FROM C



# Example

- **SELECT** A, B, C AS c, function1(D) AS d, E AS e, F AS f
  **FROM** table1 name
  **WHERE** table1 name.c1 ≤ 'STR' and table1 name.c1 < 'STR' AND (G='STR' AND C IN ('STR', 'STR' ))
- **SELECT** A, H, E, C, F, G, I, D
  **FROM** table1 name
  **WHERE** table1 name.c1 ≤ 'STR' and table1 name.c1 < 'STR' AND (I>function2(p1, 'STR') AND I≤ function2(p1, 'STR'))

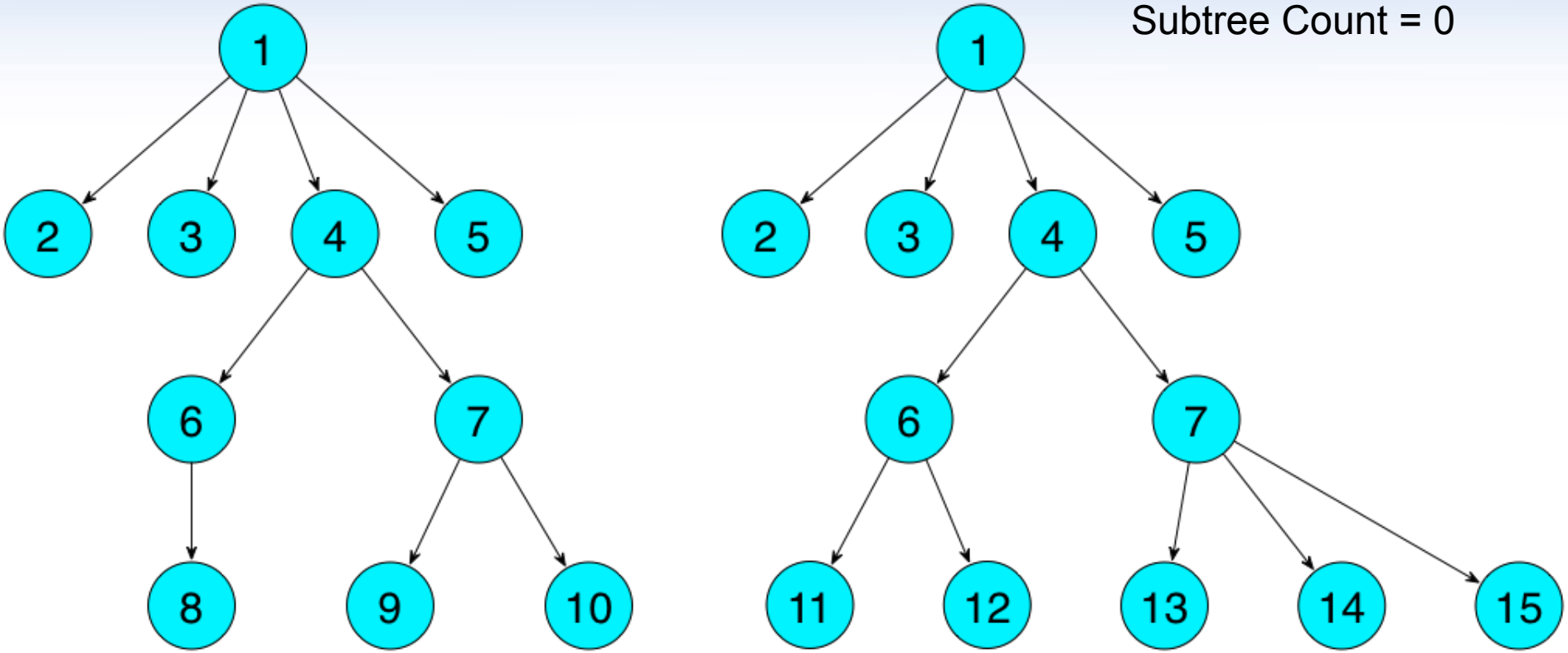


# Tree Similarity Algorithm

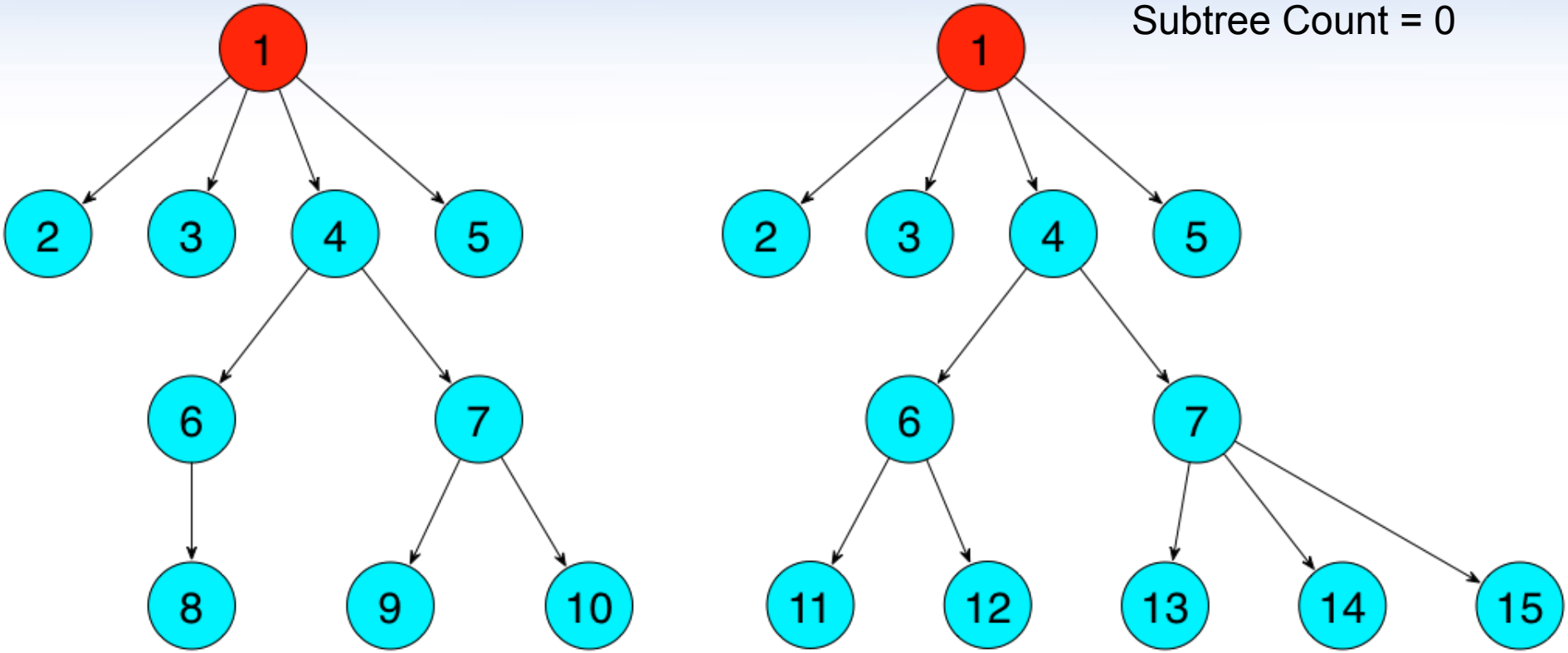
- Modified version of Weisfeiler-Lehman graph kernel
- Walk over nodes in trees T and T'
  - For each pair of nodes
    - Count number of matching subtrees (using labels)
    - For a match, multiply by the height of the subtree

- Normalize

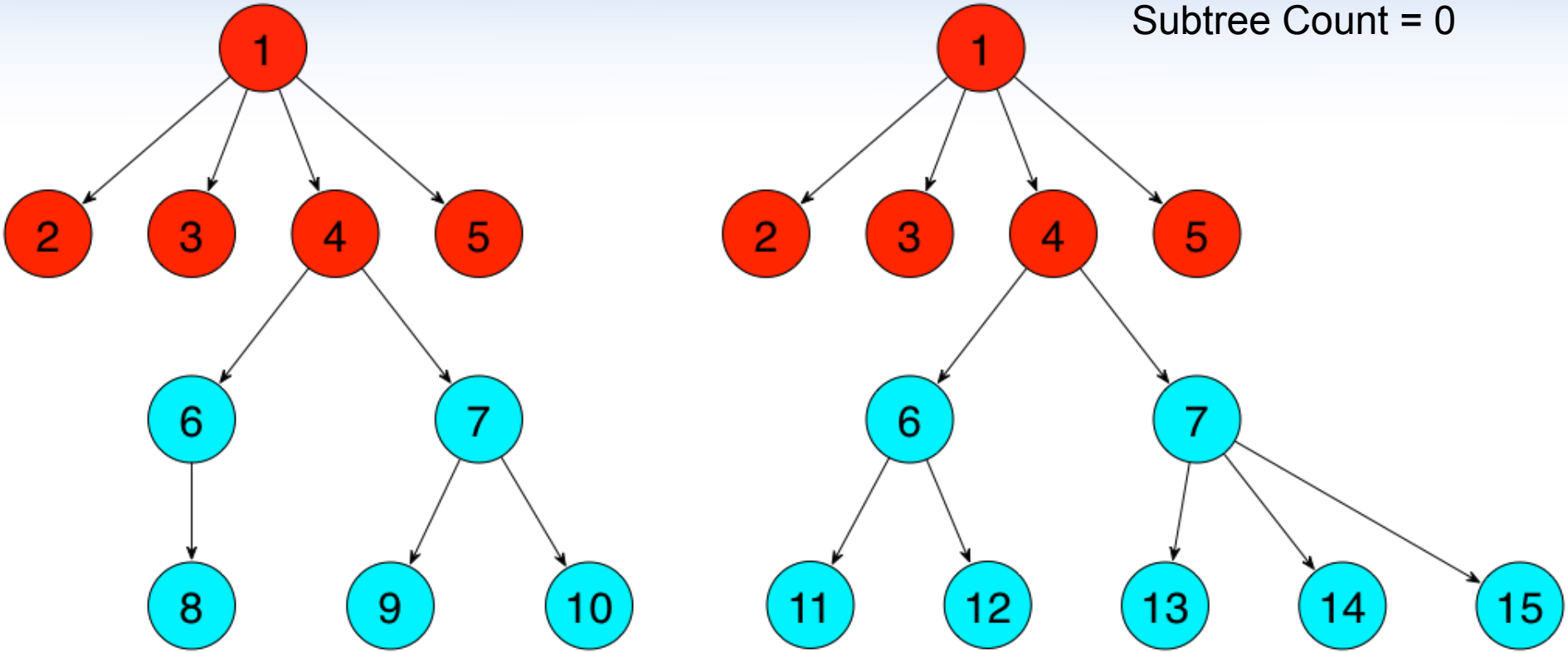
# Tree Similarity Algorithm



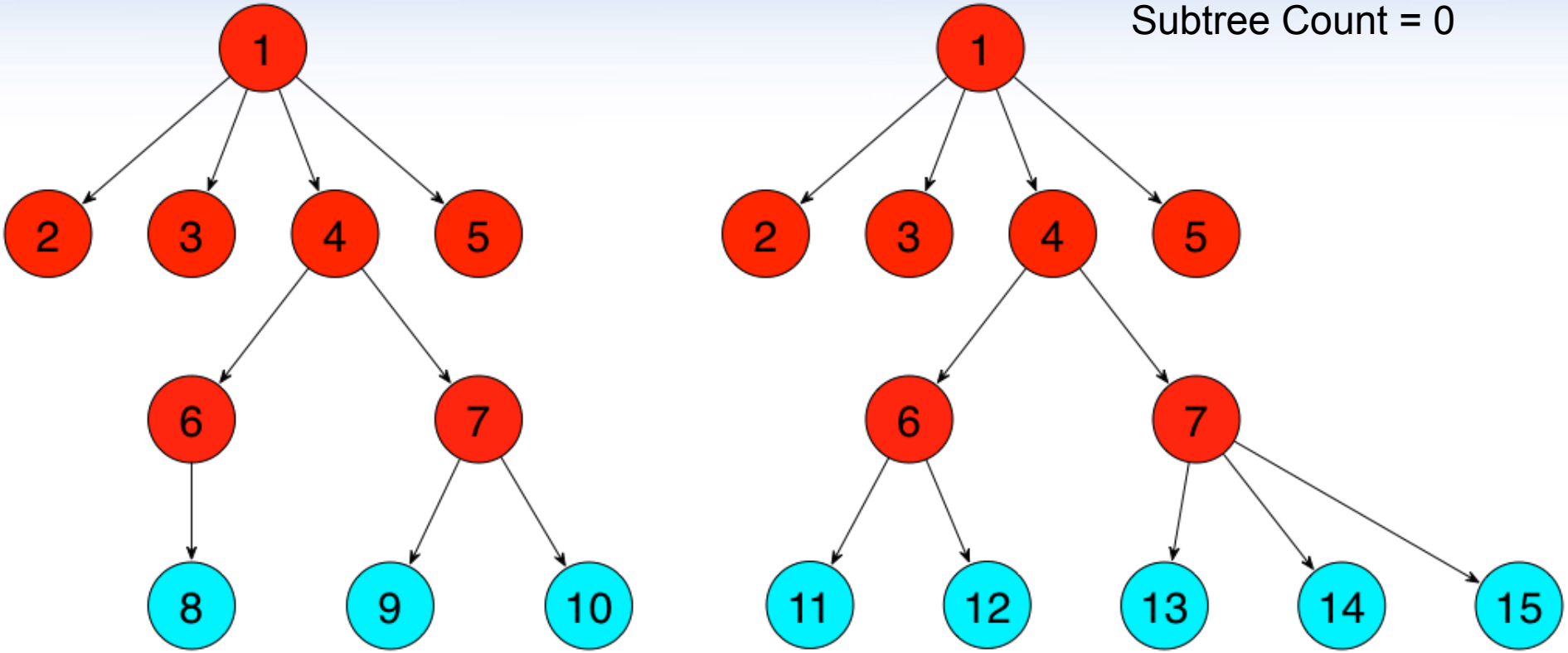
# Tree Similarity Algorithm



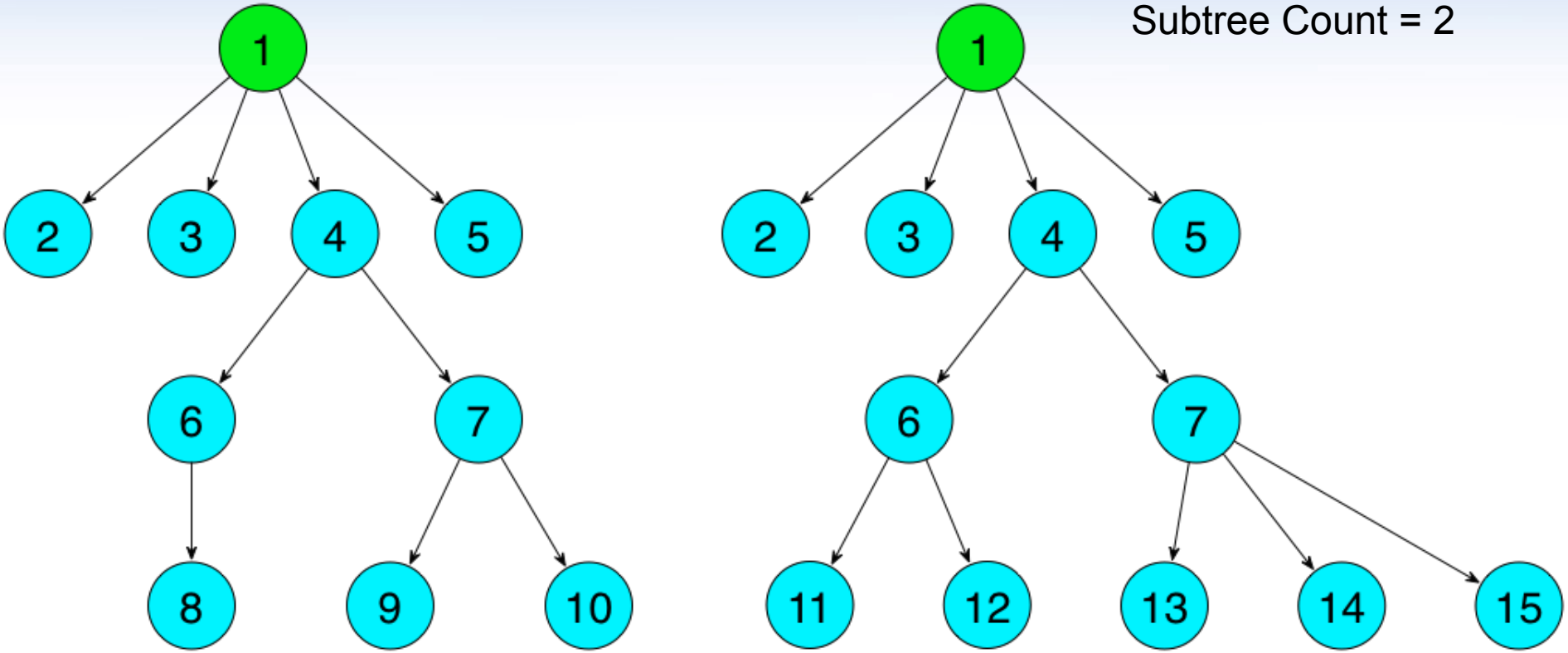
# Tree Similarity Algorithm



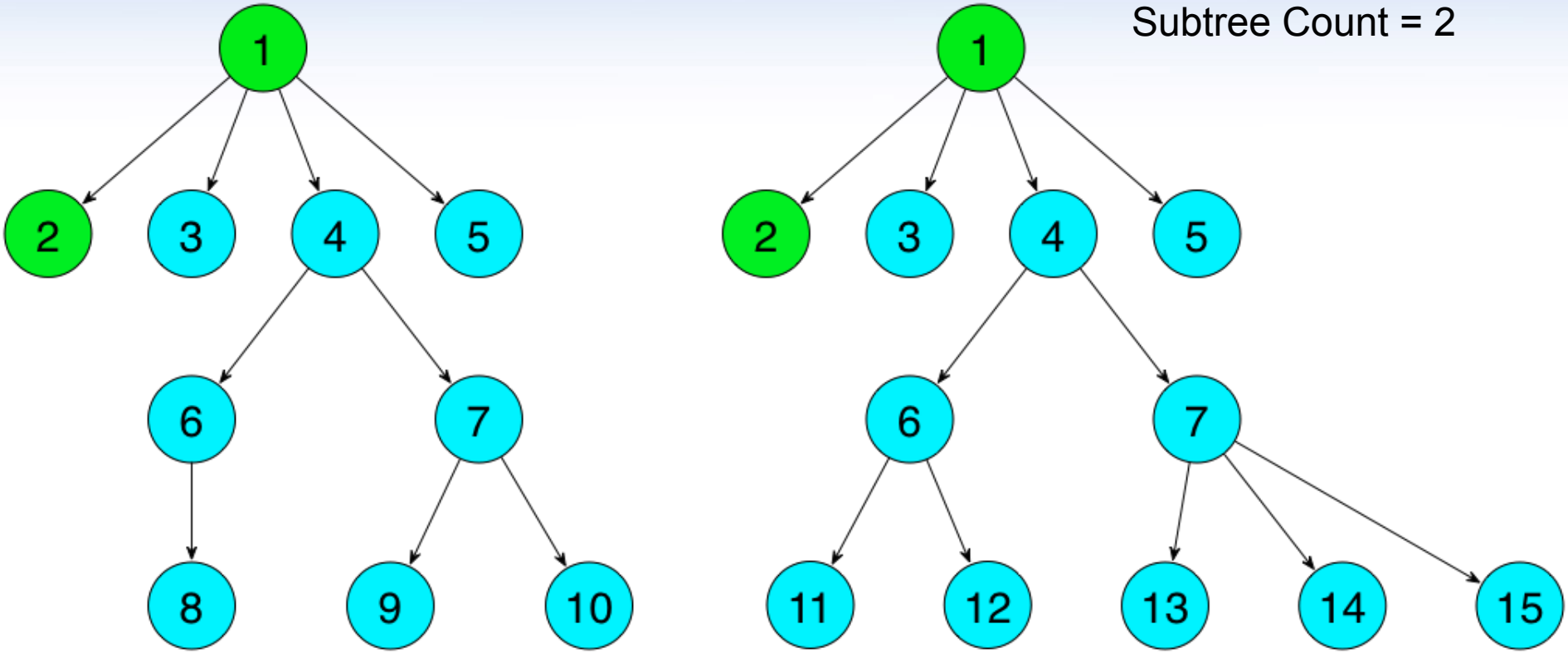
# Tree Similarity Algorithm



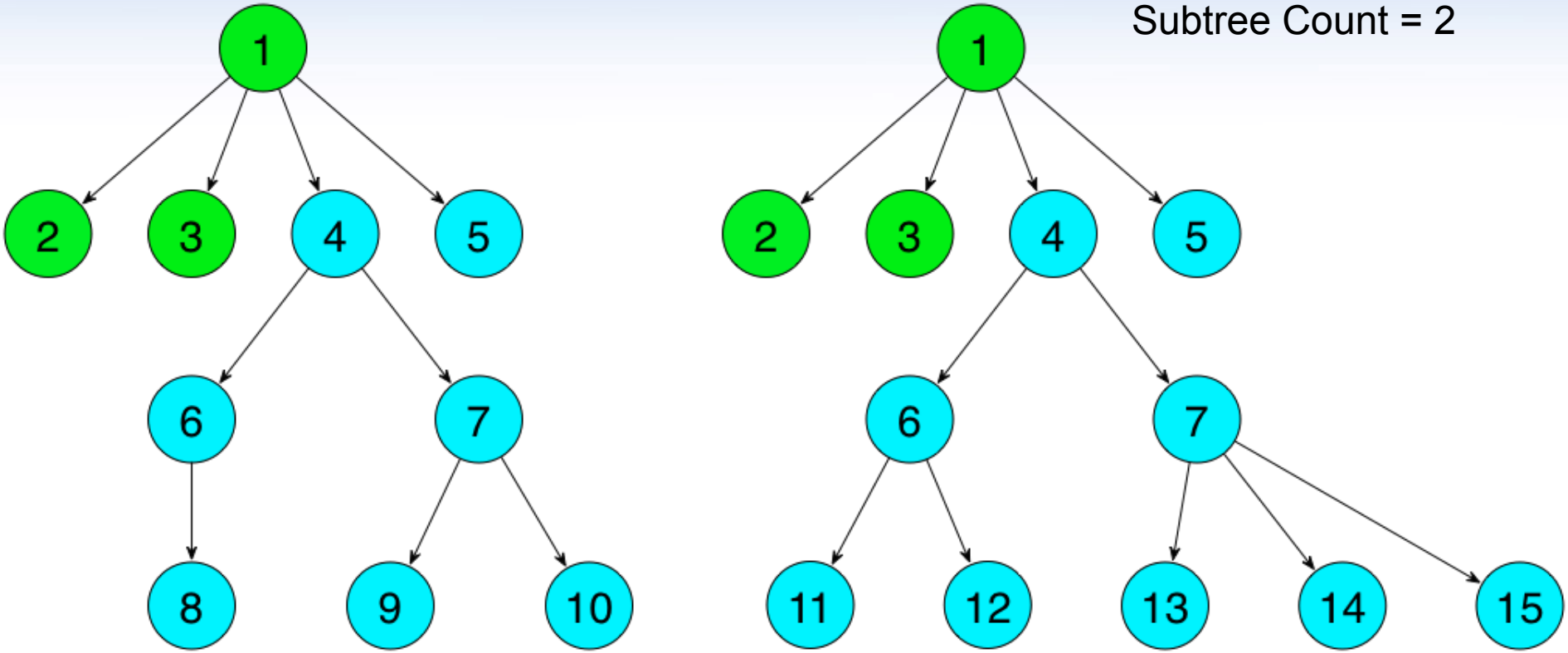
# Tree Similarity Algorithm



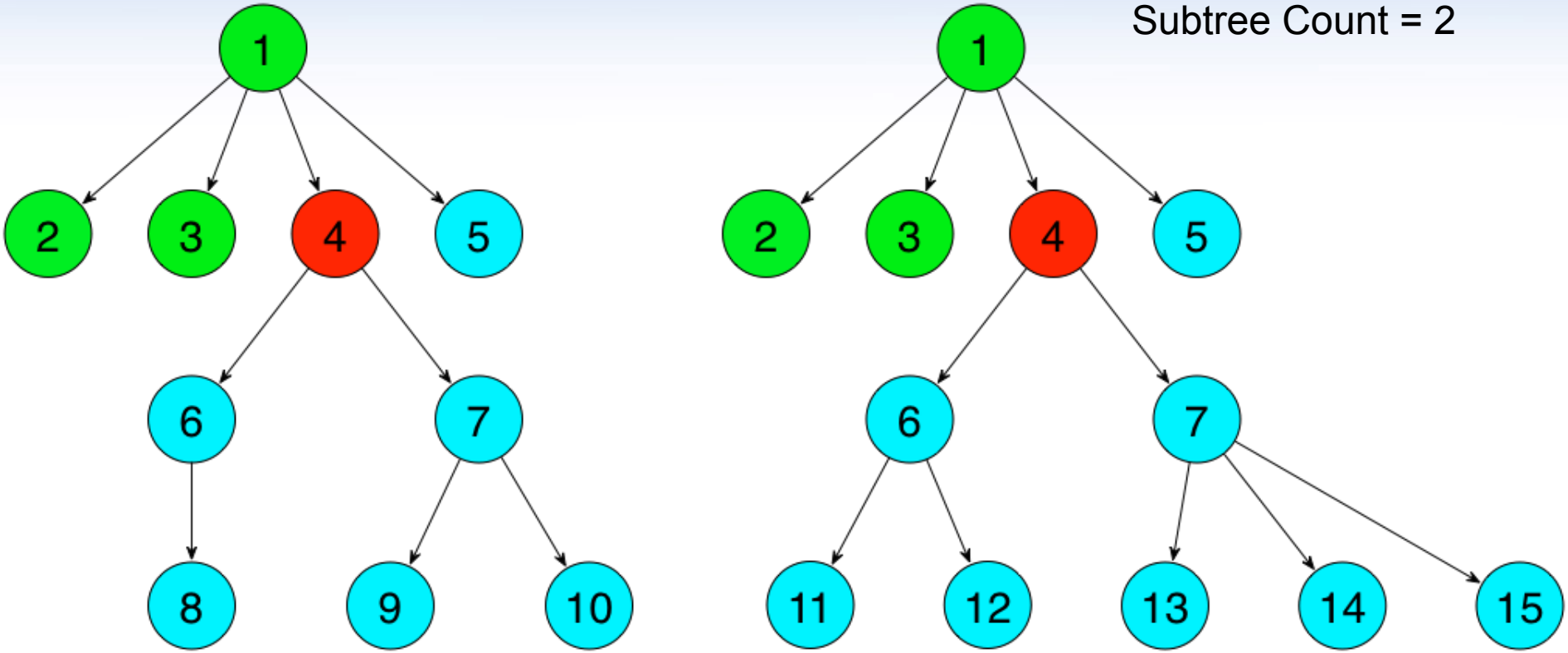
# Tree Similarity Algorithm



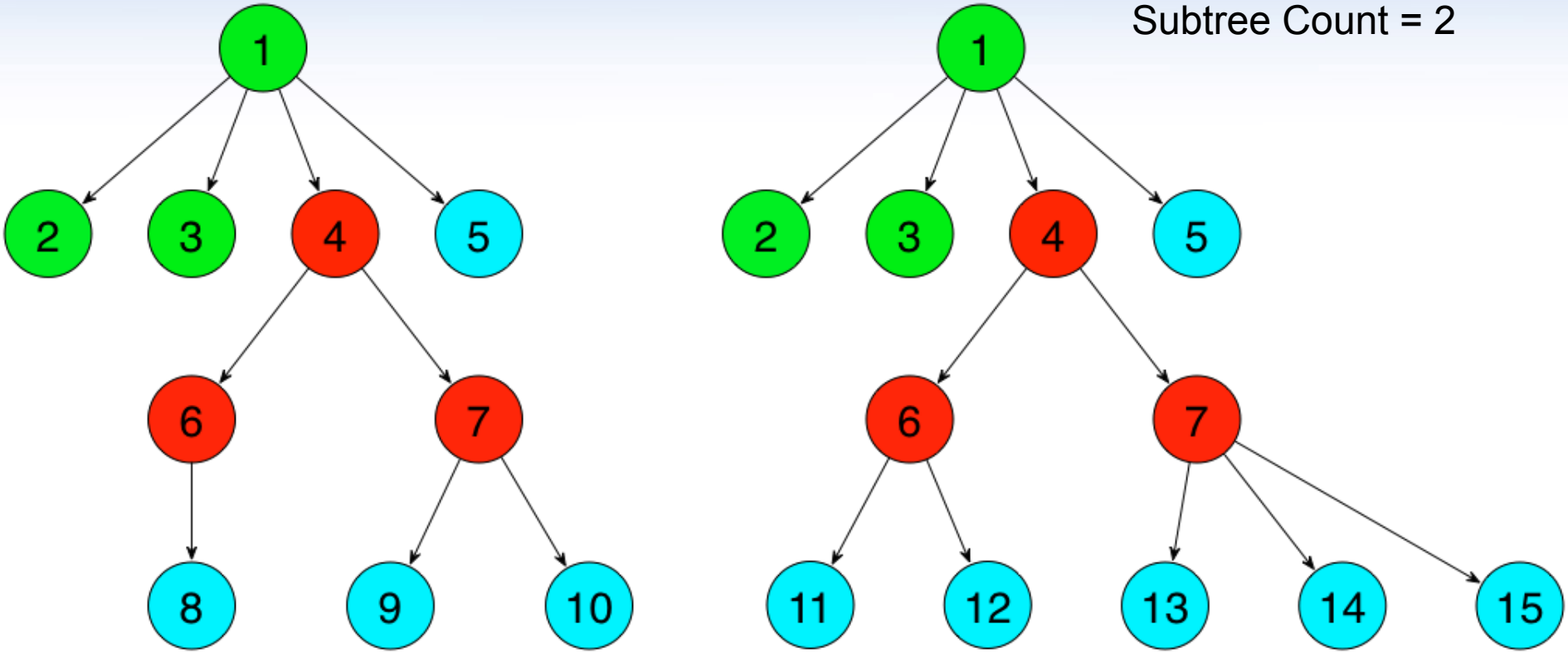
# Tree Similarity Algorithm



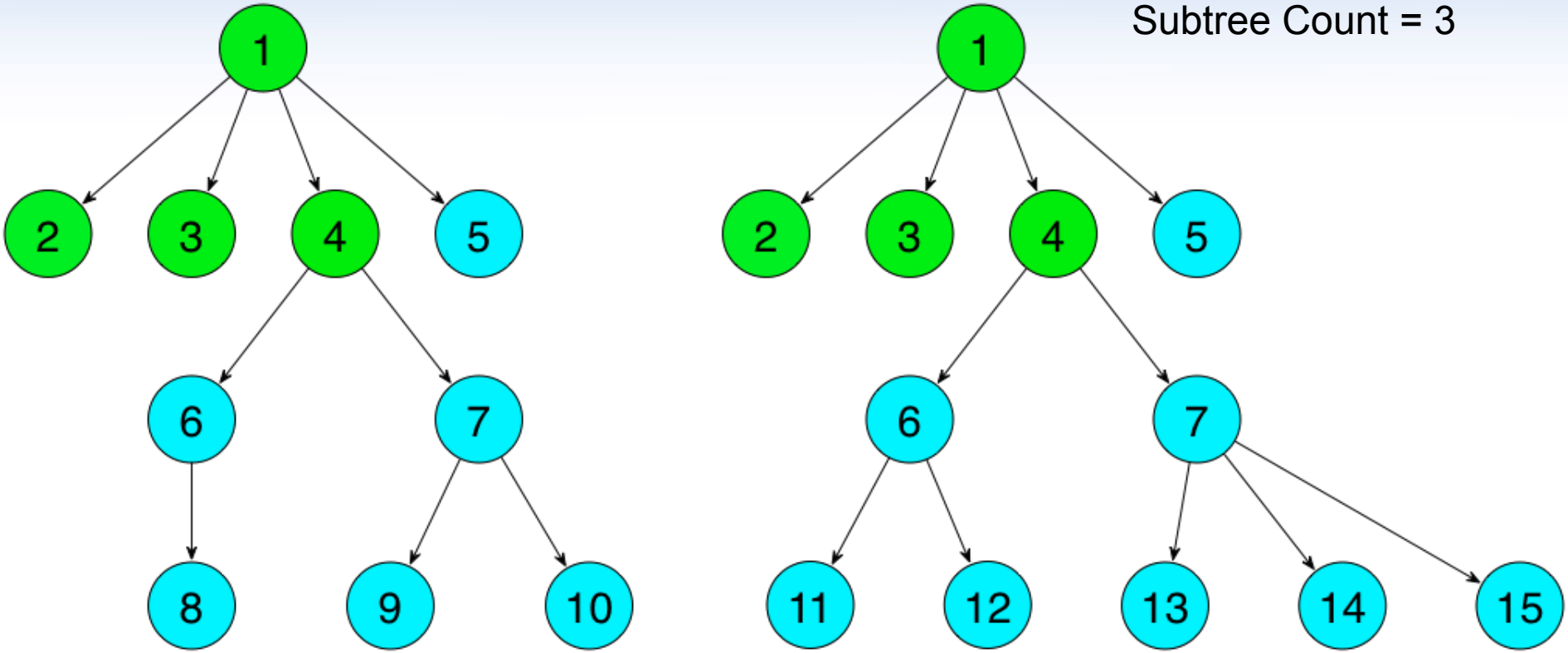
# Tree Similarity Algorithm



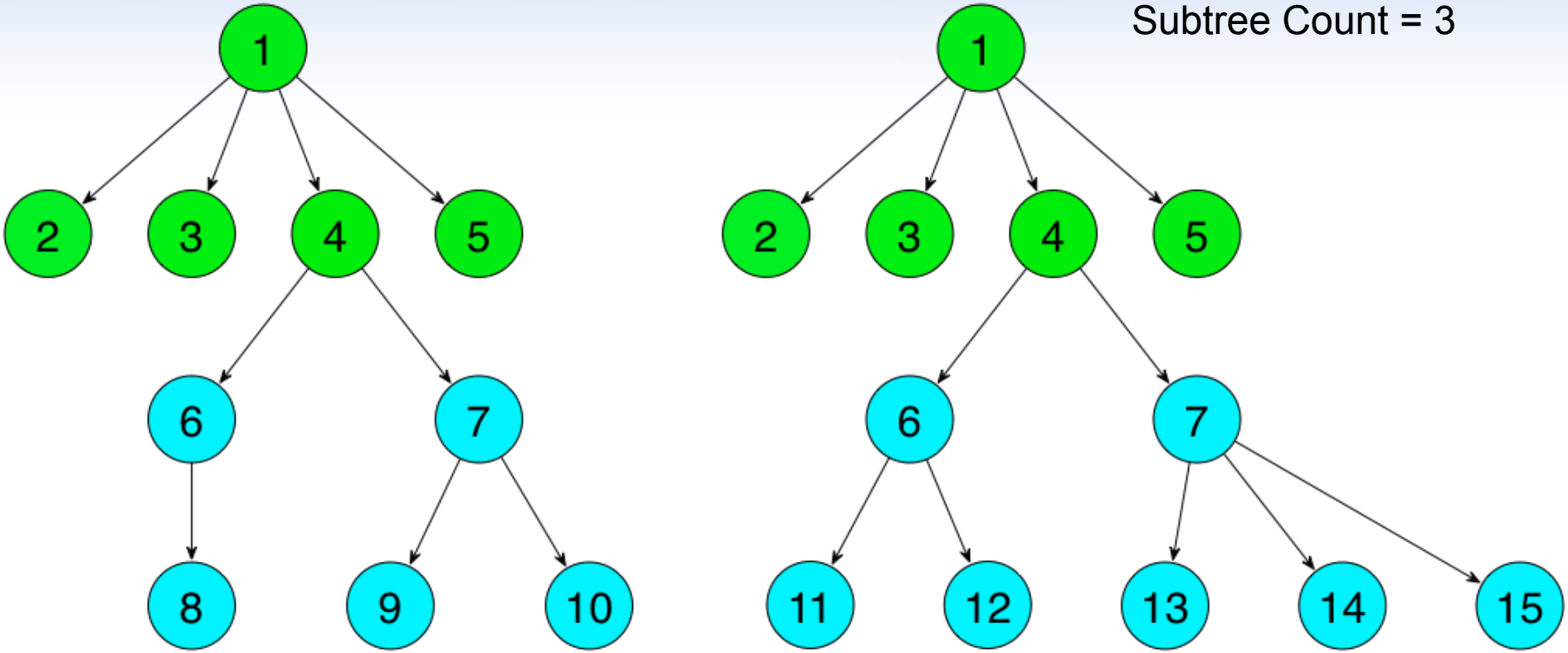
# Tree Similarity Algorithm



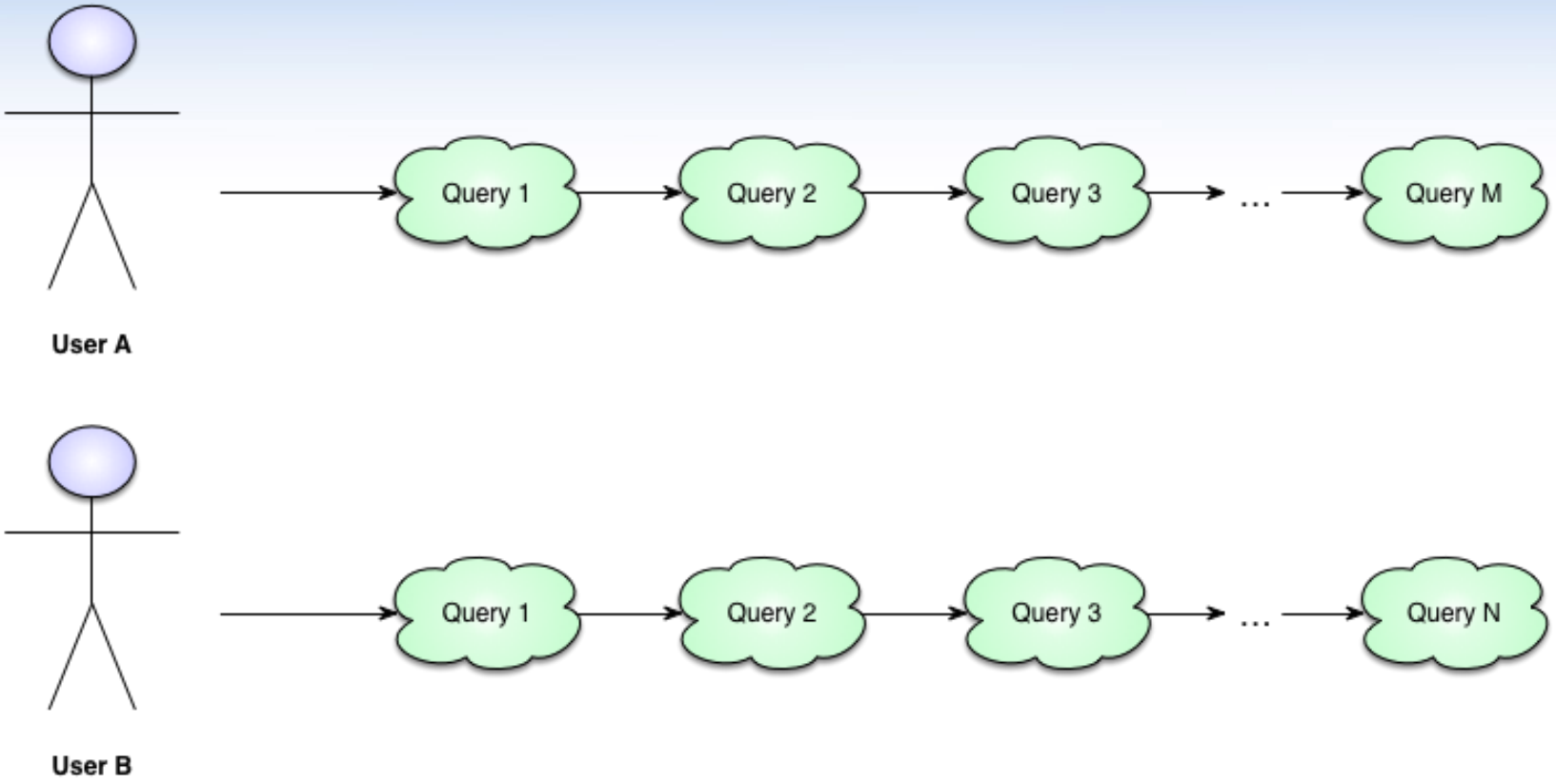
# Tree Similarity Algorithm



# Tree Similarity Algorithm



# User Similarity



# User Similarity and Betweenness Centrality

- Compare collections of queries from each individual
- User Similarity
  - Find strong similarities between pairs of individuals
- Betweenness Centrality
  - Find individuals who share commonalities with many other users

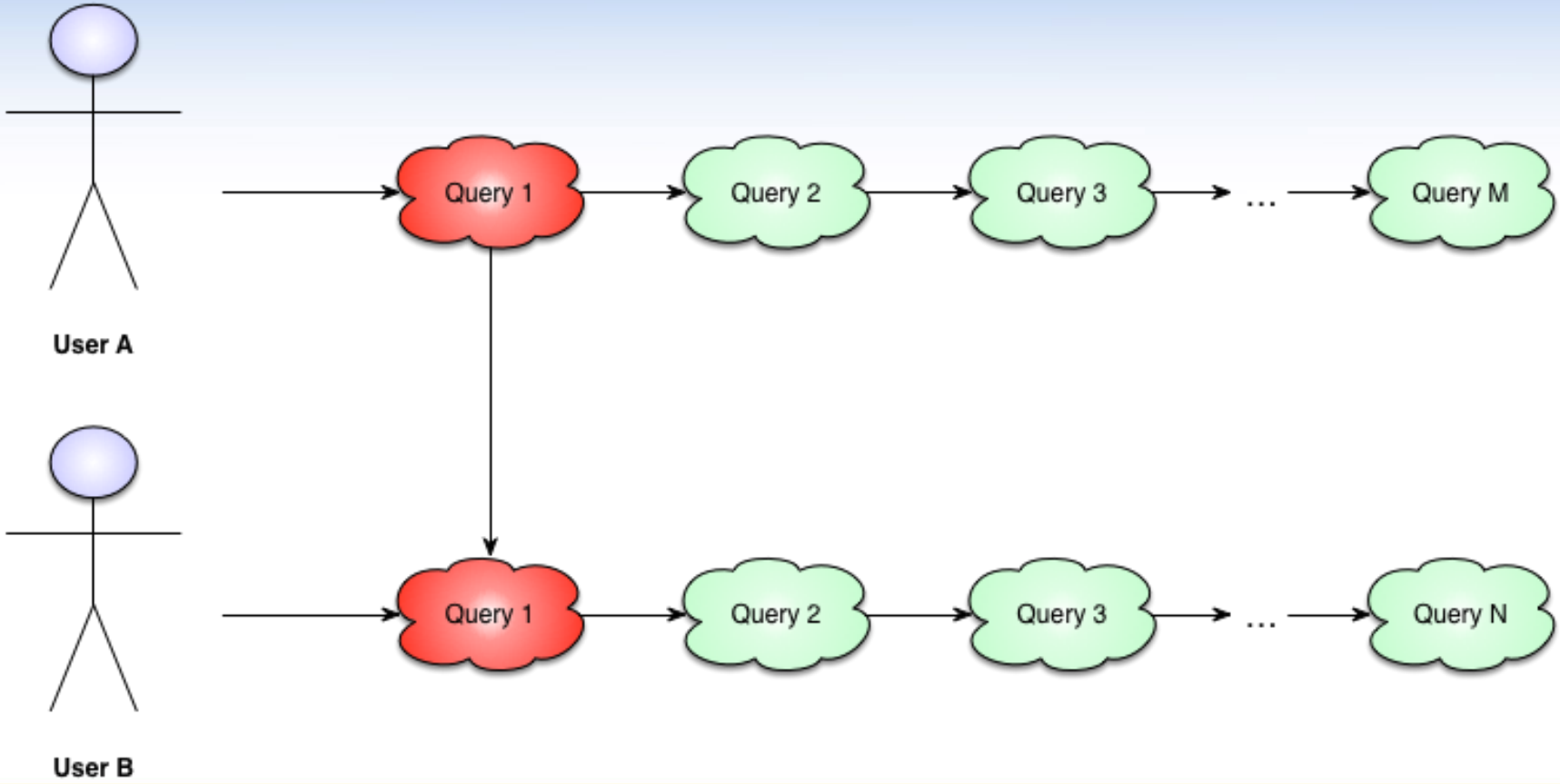
# User Similarity

- Let M, N denote the number of queries submitted by Users A, B respectively.

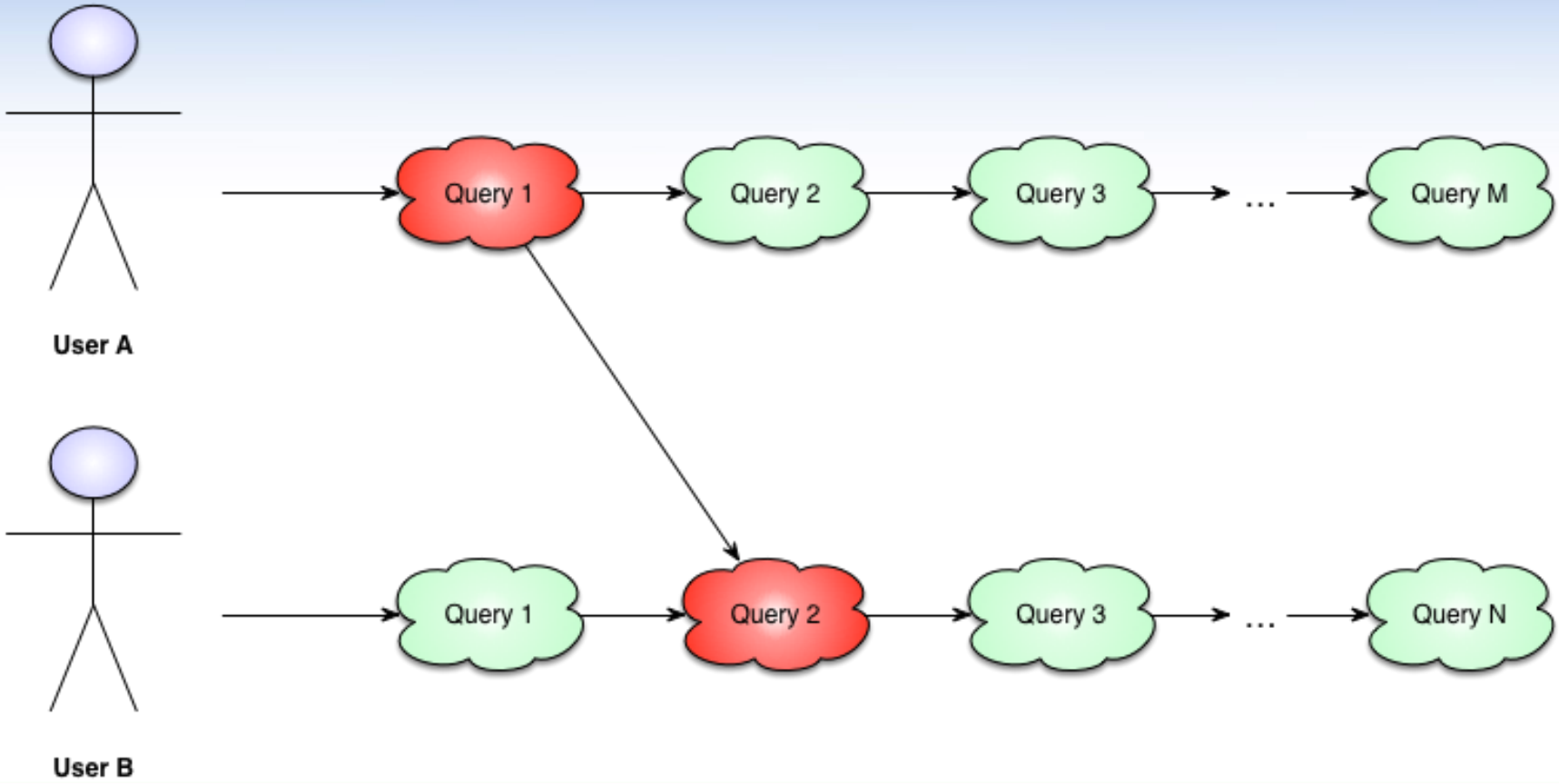
$$sim(A \rightarrow B) = \frac{\sum_{i \in M} \max(sim(i,j) | \forall j \in N)}{|M|}$$

- Note that we calculate (sim(A → B) + sim(B → A))/2 in order to obtain a symmetrical matrix.

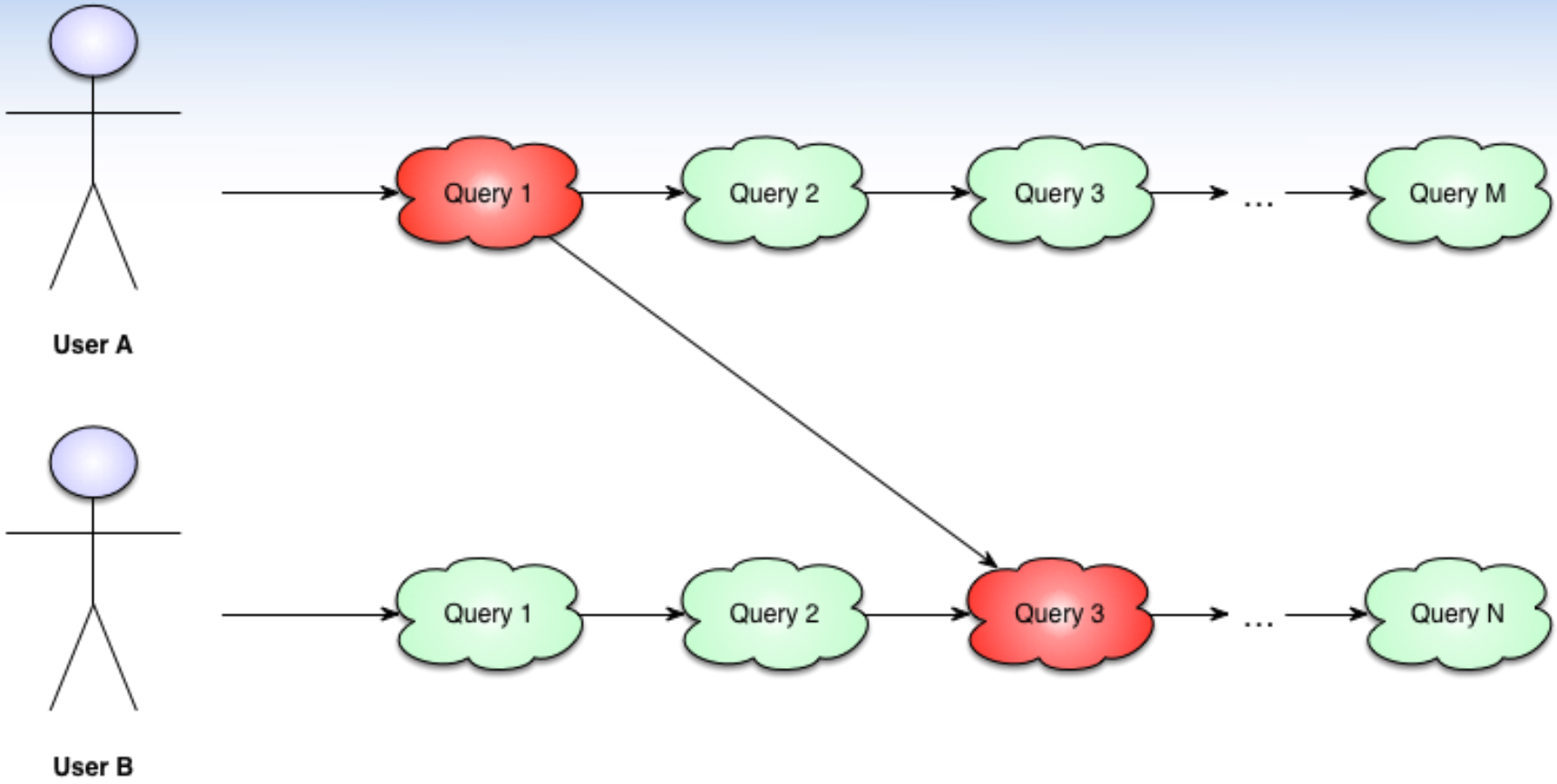
# User Similarity



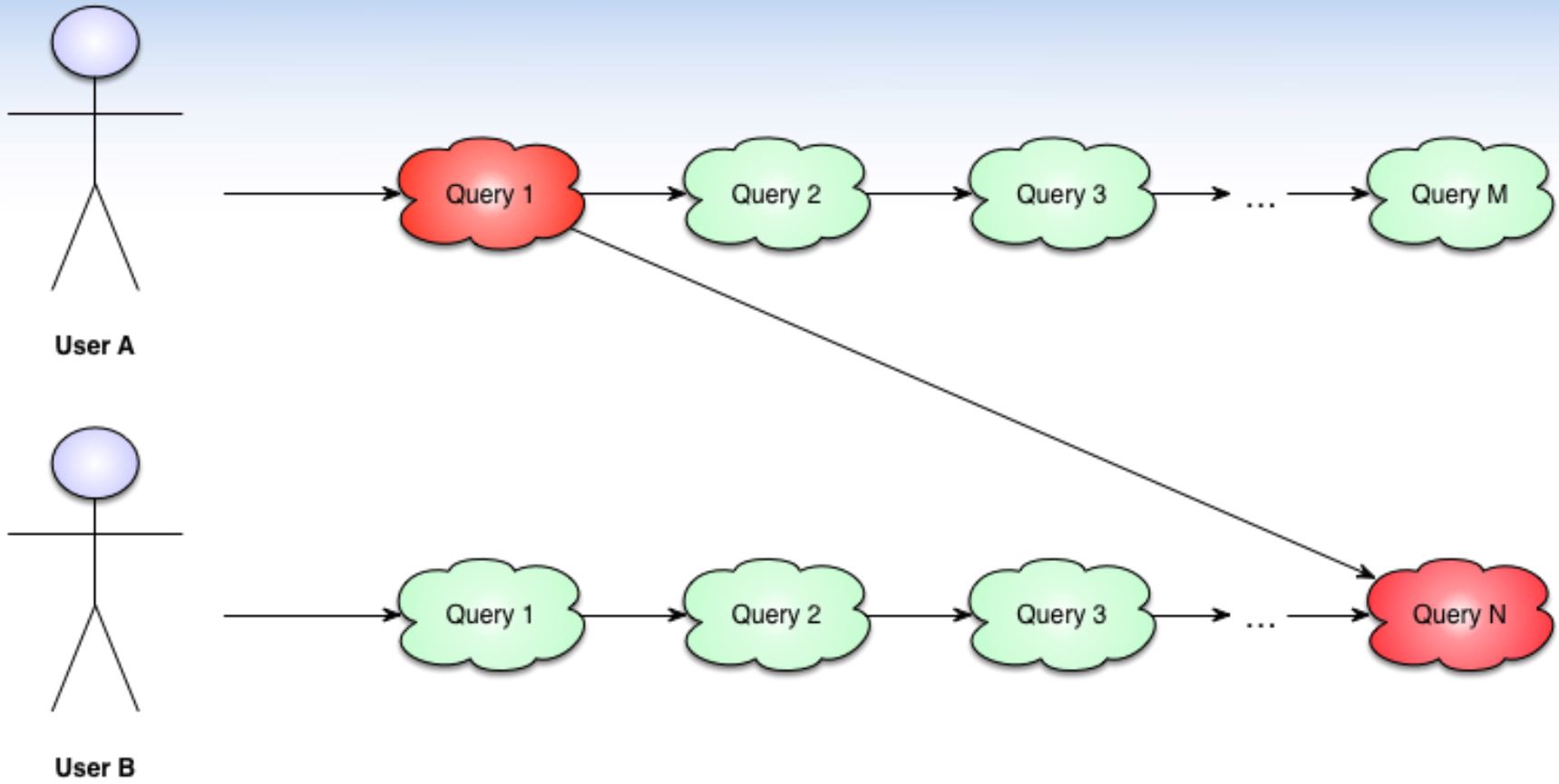
# User Similarity



# User Similarity



# User Similarity



# User Similarity

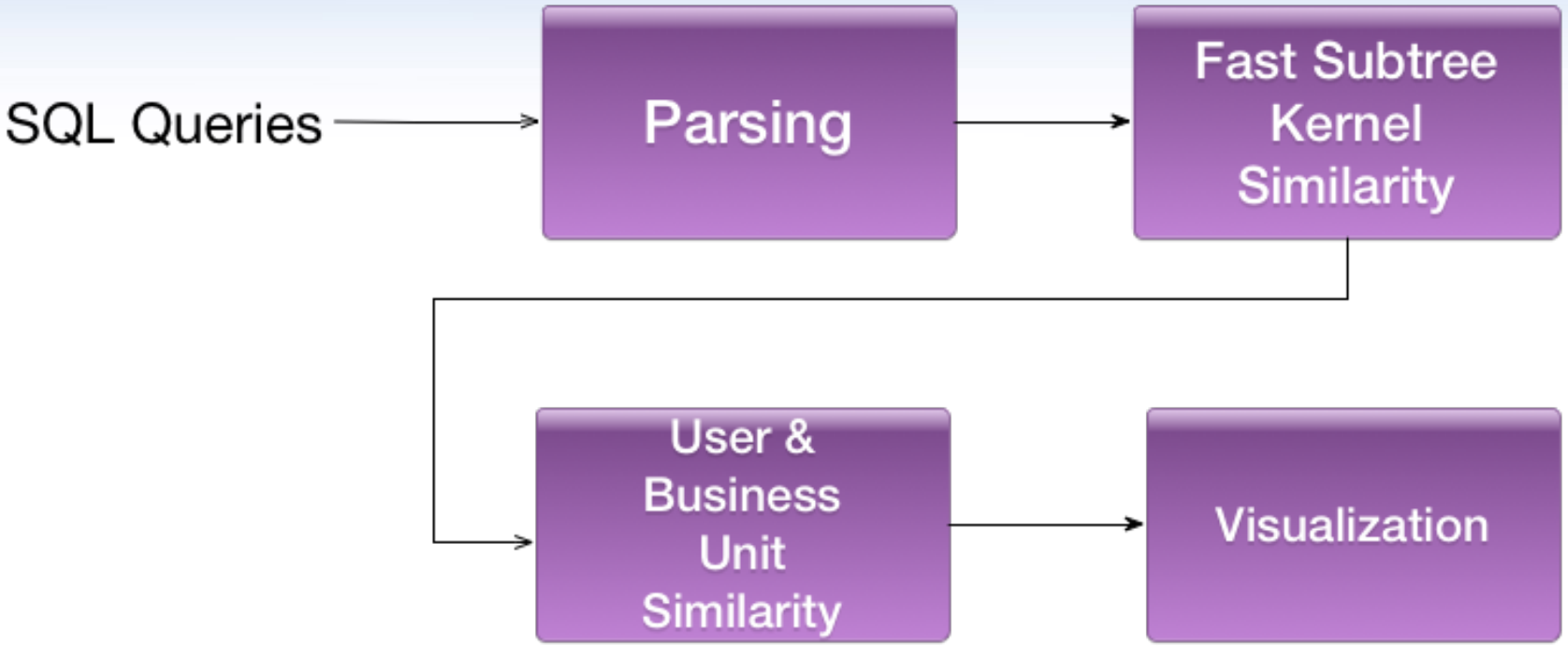
- Create similarity matrix
- Symmetrical
- Visualize

# Betweenness Centrality

- How central a node is in a graph/network

- Tells us which users are similar to many other users
  - Versatile employee
  - Common underlying element

# Workflow


SQL Queries
Parsing
Fast Subtree Kernel Similarity
User & Business Unit Similarity
Visualization

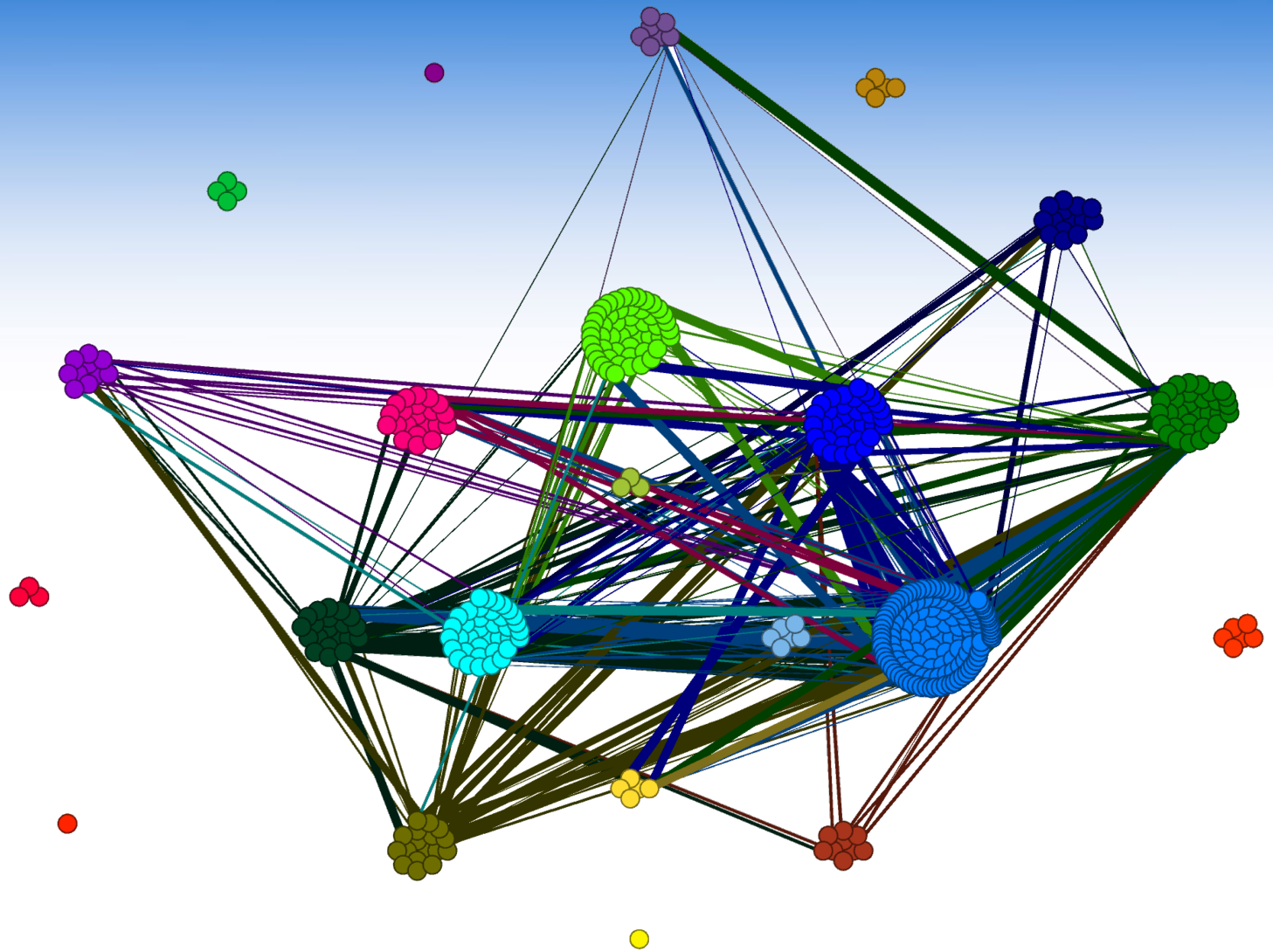
# Experimental Results

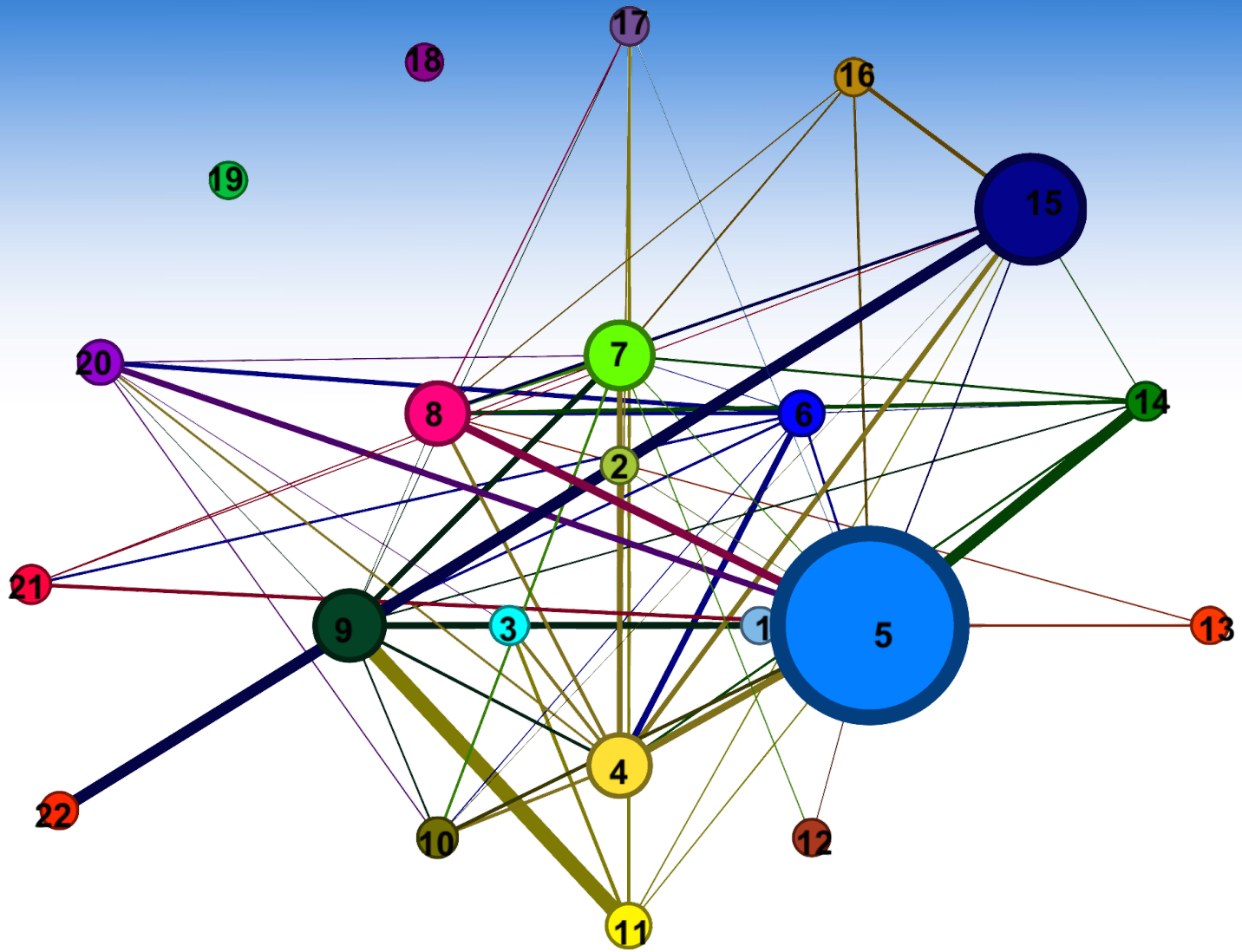
- 2 datasets (provided by JP Morgan Chase & Co.)
- Anonymized SQL queries
- Visualization
  - User similarity
  - Betweenness centrality
  - Heat Map

UNIVERSITY OF DELAWARE

# Dataset 1

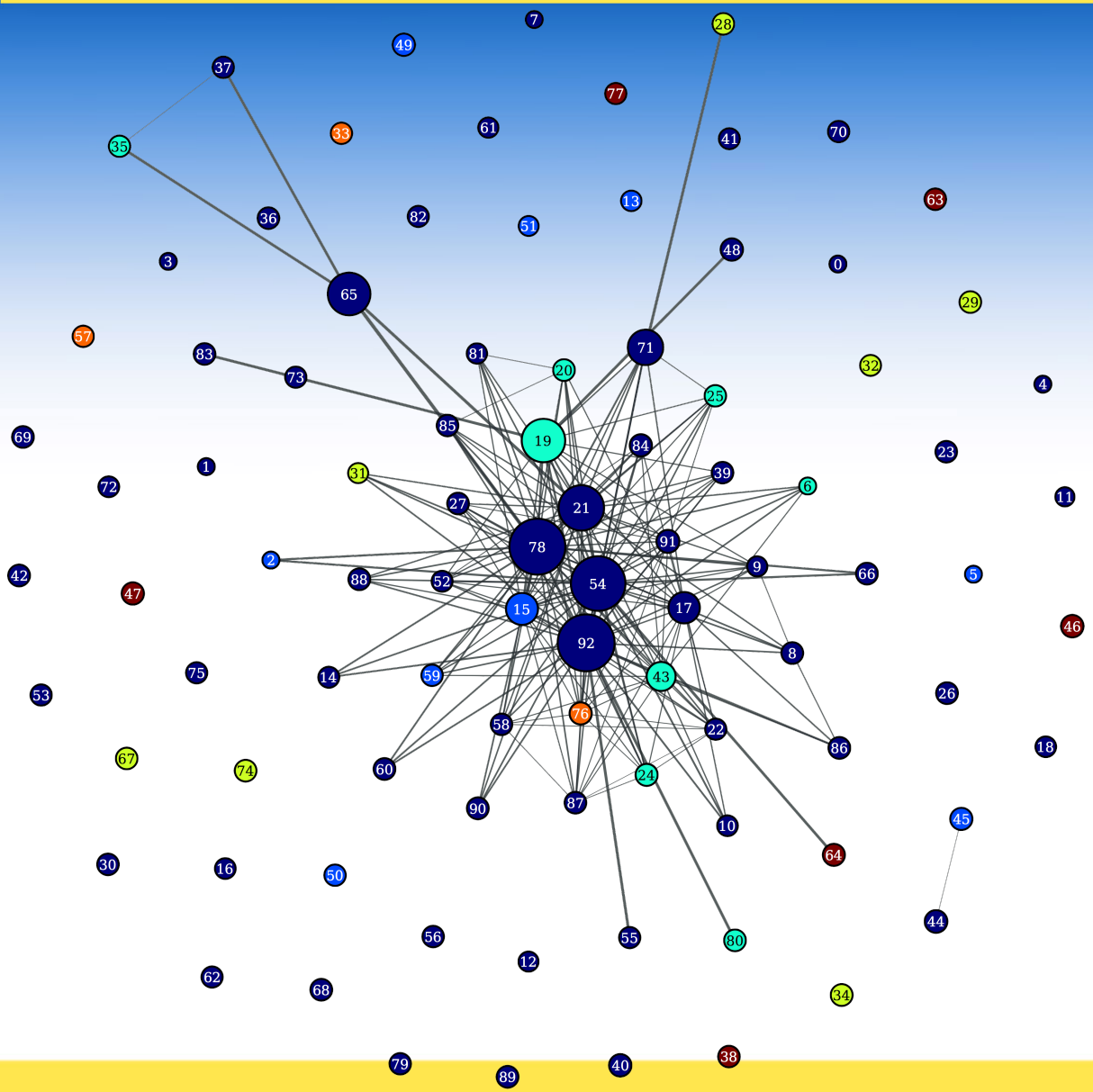
- 49,655 queries (provided by JP Morgan Chase & Co.)
- 427 users
- Organized into 22 business units




17
18
16
19
15
20
7
14
8
6
2
21
9
3
1
5
13
4
22
10
12
11

# Dataset 2

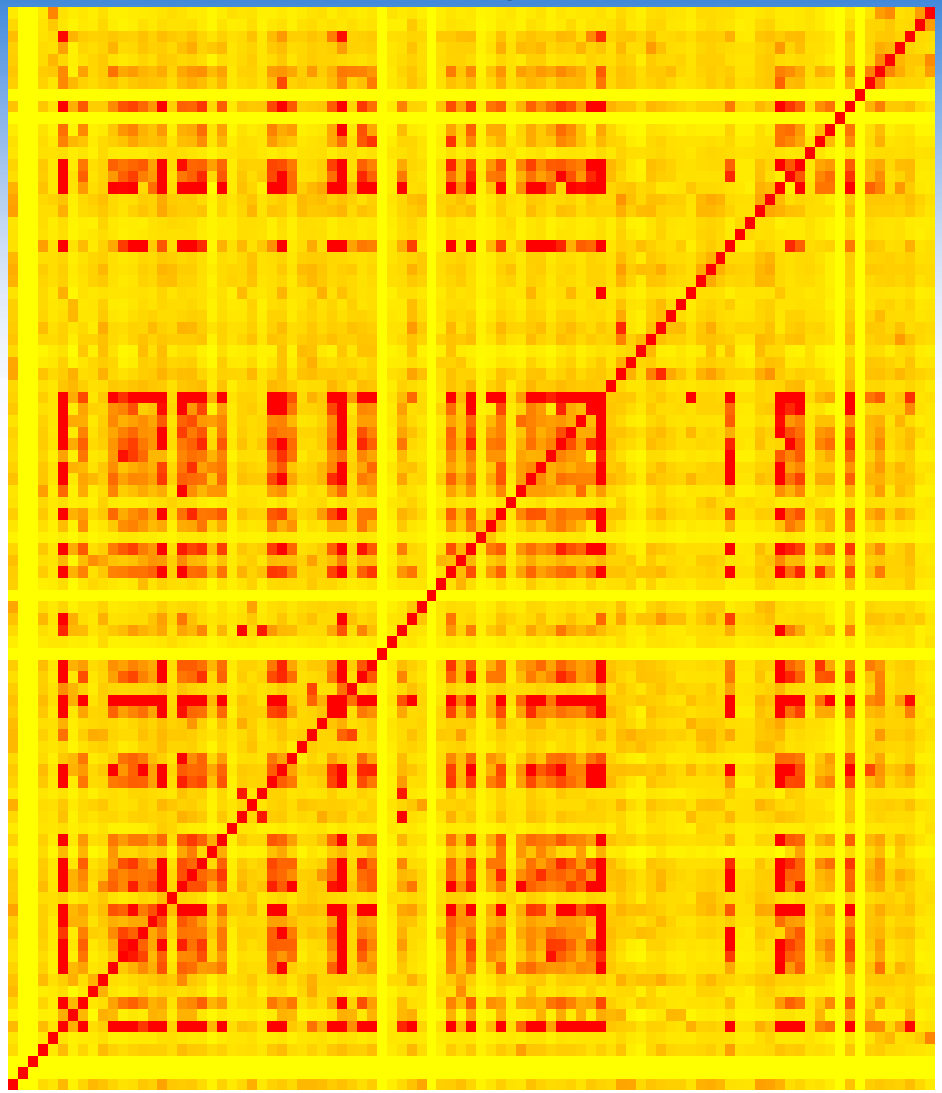
- 787,732 queries (provided by JP Morgan Chase & Co.)
- 92 users
- Organized into 6 business units


7
28
49
37
77
33
61
41
70
35
13
63
36
82
51
48
3
0
65
29
57
71
83
81
20
32
73
25
4
85
19
84
69
23
1
31
39
72
6
11
27
21
78
91
2
9
66
5
42
88
52
54
47
15
17
46
92
8
75
14
59
43
53
26
76
58
22
67
86
18
74
60
24
90
87
10
45
64
30
16
50
44
56
55
80
12
62
68
34
79
89
40
38

# Heat Map

- Shows all the similarities uncovered in dataset 2
- Users sorted according to business unit
- Red = similar, Yellow = not similar

## User Similarity Heatmap



# Challenges

- Volume of data
  - 49,655 x 49,655 = 10GB approx.
  - 787,732 x 787,732 = 2.5TB approx.

- Computational cost
  - User similarity is quadratic in the number of queries.
  - Calculating user similarity matrix is $n^4$

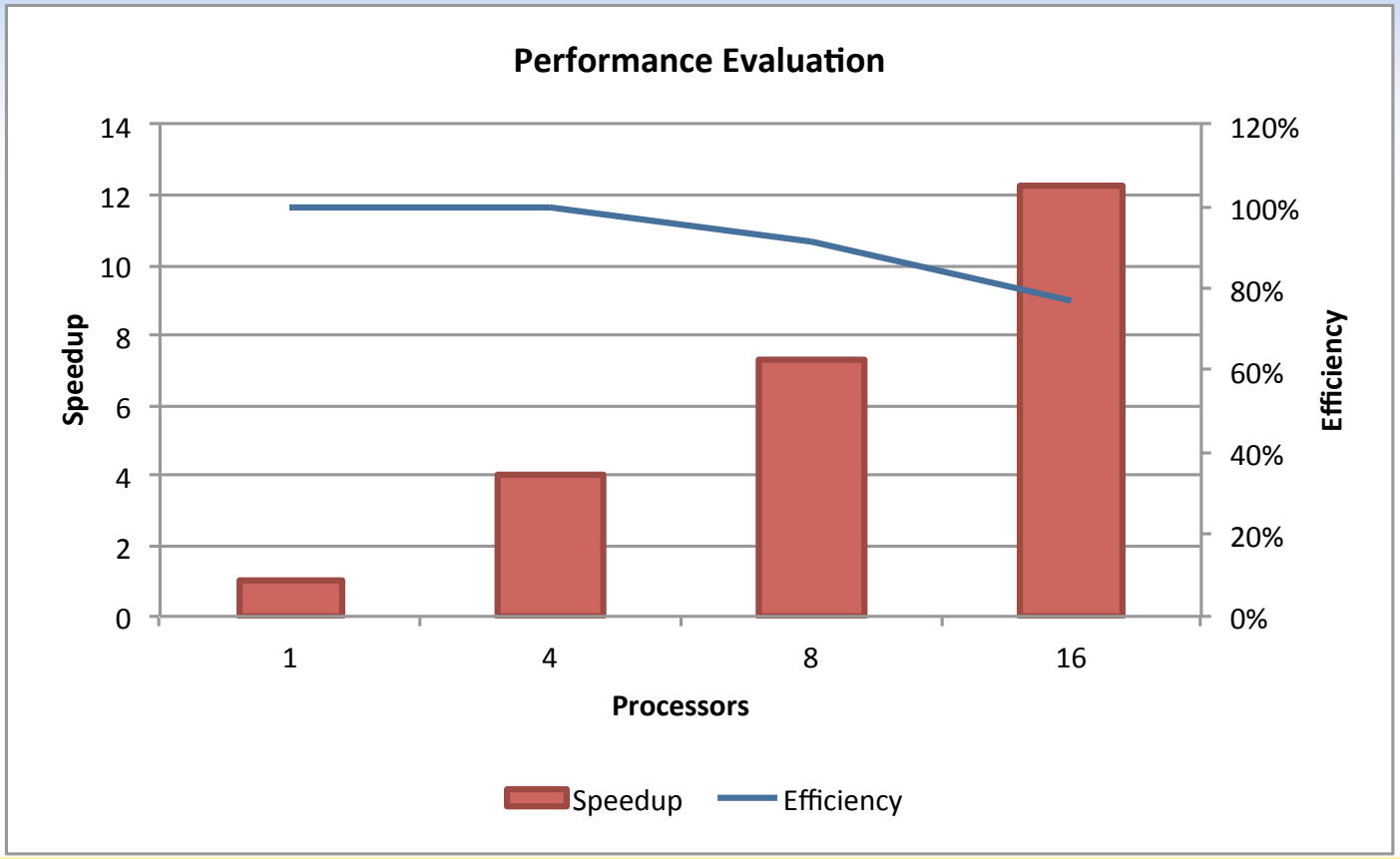
# Acceleration

- Compute user similarity in parallel
  - No data dependencies
- If using accelerator, compute tree similarities in parallel
- OpenMP

# Experimental Setup

- Machine: 16 GB of memory and 2x AMD Opteron 6320 CPUs, 8 cores per CPU clocked at 1.4 GHz
- Dataset 1: 49,655 queries
- Dataset 2: 787,732 queries

# Acceleration



# Contributions

- Created an algorithm that was used to measure the similarity between employees in a workforce (JP Morgan)
- Designed a scalable, encoding-based algorithm to measure the similarity between SQL queries
- Used these algorithms to create a visual representation of employee similarity across a workforce

UNIVERSITY OF DELAWARE

# Conclusion

- Developed a framework for calculating similarity between users/analysts
- Discovered similarities in the data
- Found similar users in different parts of the company

UNIVERSITY OF DELAWARE

# Acknowledgement


- Special thanks to JP Morgan Chase & Co.
- Collaboration on real-world problem
- Provided anonymized data