APM Seminar

# FP-Growth

Josefine Harzmann, Sebastian Oergel

# Agenda

- Why Pattern Mining?
- FP-Growth at a Glance
  - Use Case
  - Phase 1
  - Phase 2
- Experiment Environment
- Algorithm Evaluation
  - Problems
  - Experiment Results
- Future Work

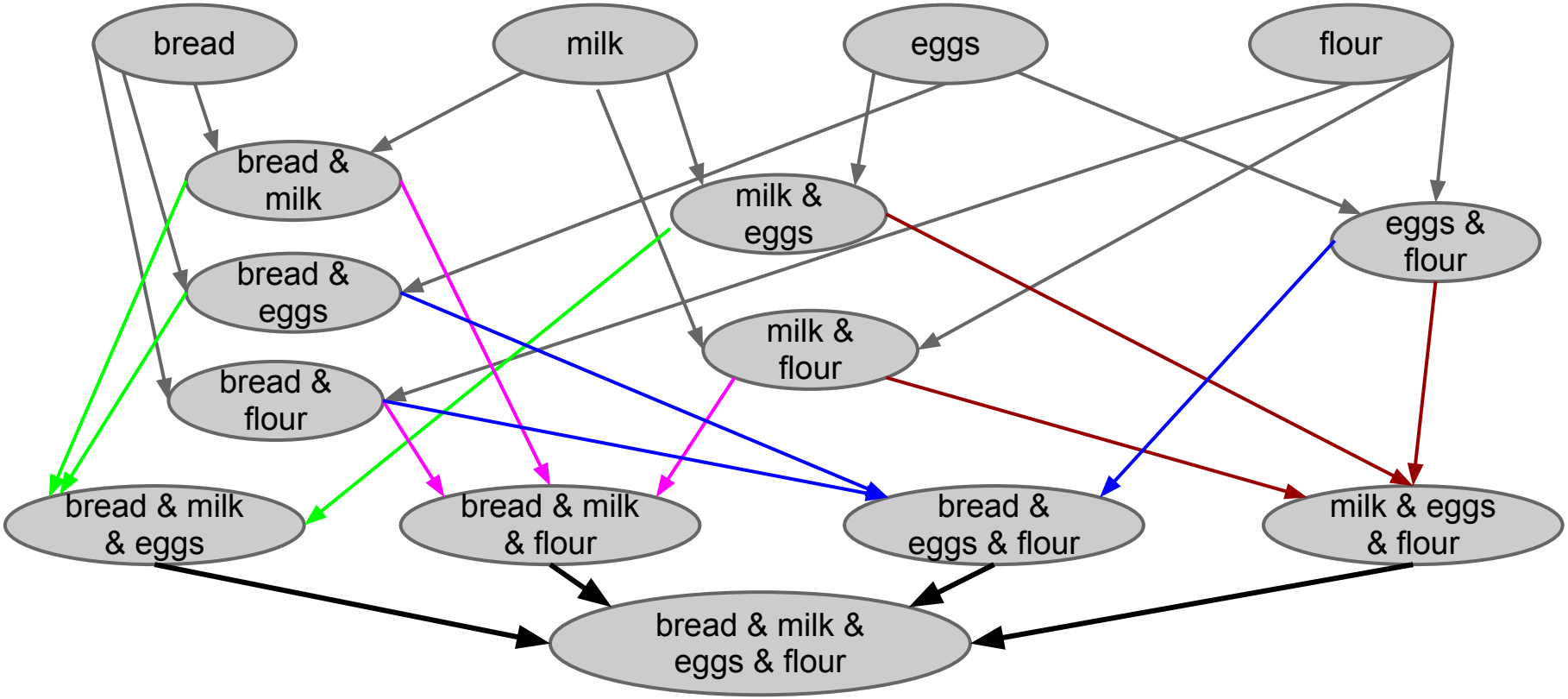
# Why Pattern Mining?

- Find frequent Itemsets
- Use cases:
  - Analysis of Shopping Behavior
    - Optimization of product placement
  - Proposals / Recommendation
    - Search
    - Keywords, Tags
    - Adaptive Music Streams
    - Interesting Web Pages
  - Person "Classification"
  - ...

# Drawbacks of A-priori

- Candidate generation yields a large set of items
  - Huge number of item combinations

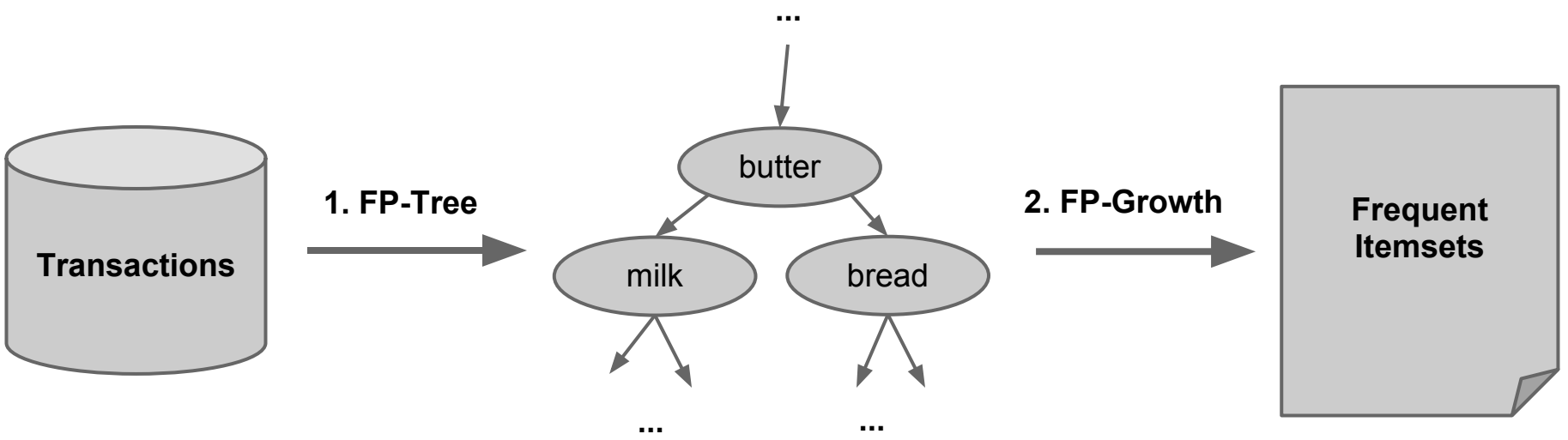
- Multiple DB scans required
  - For each candidate: determine number of occurrences
  - Check for Minimum Support/Frequency Threshold



# FP-Growth at a Glance

- Algorithm was first introduced at SIGMOD conference 2000 by Han, Pei & Yin

- Use compact data structure to find frequent itemsets
- 2 DB scans required
- 2 Phases
  - 1. FP-Tree: build the data structure to work on
  - 2. FP-Growth: inspect the data structure to find all frequent itemsets



# Phase 1
## FP-Tree

# Use Case

- **Twitter** data may contain hashtags (#xyz)
  - Keyword for references
- Sometimes several hashtags are used together
- Recommend those hashtag combinations
  - Example: "... #car ..." => recommend to use #auto, too
- Find (unexpected) popular relationships among subjects

# Introductory Example

- **Twitter** data may contain hashtags (#xyz)
  - Keyword for references
- Sometimes several hashtags are used together
- Recommend those hashtag combinations
  - Example: "... #car ..." => recommend to use #auto, too
- Find (unexpected) popular relationships among subjects

| | |
|---|---|
| 1 | {#justin, #britney, #jlo, #rammstein, #metallica, #ozzy_o, #shaki, #us} |
| 2 | {#britney, #nice, #jlo, #justin, #good_music, #shakira, #commercial} |
| 3 | {#nice, #justin, #will_i, #bored, #commercial} |
| 4 | {#nice, #jlo, #enrique, #aerzte, #us} |
| 5 | {#britney, #justin, #jlo, #iron_m, #good_music, #us, #shakira, #fergie} |
| 6 | {#justin, #britney, #shakira, #us} |

# Phase 1 - Building an FP tree

- Find frequency of each individual item (requires 1 table scan)

| | |
|---|---|
| #justin | 5 |
| #britney | 4 |
| #jlo | 4 |
| #rammstein | 1 |
| #metallica | 1 |
| #ozzy_o | 1 |
| #shaki | 4 |
| #us | 4 |
| #nice | 3 |

| | |
|---|---|
| #good_music | 2 |
| #commercial | 2 |
| #enrique | 1 |
| #aerzte | 1 |
| #iron_m | 1 |
| #fergie | 1 |
| #will_i | 1 |
| #bored | 1 |

# Phase 1 - Building an FP tree

- Sort items in descending order of their frequency and discarding infrequent items
- Minimum Frequency Threshold: 3

| | |
|---|---|
| #justin | 5 |
| #britney | 4 |
| #jlo | 4 |
| #shaki | 4 |
| #us | 4 |
| #nice | 3 |
| ~~#good_music~~ | ~~2~~ |
| ~~#commercial~~ | ~~2~~ |
| ~~#ozzy_o~~ | ~~1~~ |

| | |
|---|---|
| ~~#rammstein~~ | ~~1~~ |
| ~~#metallica~~ | ~~1~~ |
| ~~#enrique~~ | ~~1~~ |
| ~~#aerzte~~ | ~~1~~ |
| ~~#iron_m~~ | ~~1~~ |
| ~~#fergie~~ | ~~1~~ |
| ~~#will_i~~ | ~~1~~ |
| ~~#bored~~ | ~~1~~ |

# Phase 1 - Building an FP tree

- Filter for frequent items & order items according to sorting (requires 1 table scan)

| | |
|---|---|
| 1 | {#justin, #britney, #jlo, #shaki, #us} |
| 2 | {#justin, #britney, #jlo, #shaki, #nice} |
| 3 | {#justin, #nice} |
| 4 | {#jlo, #us, #nice} |
| 5 | {#justin, #britney, #jlo, #shaki, #us} |
| 6 | {#justin, #britney, #shaki, #us} |

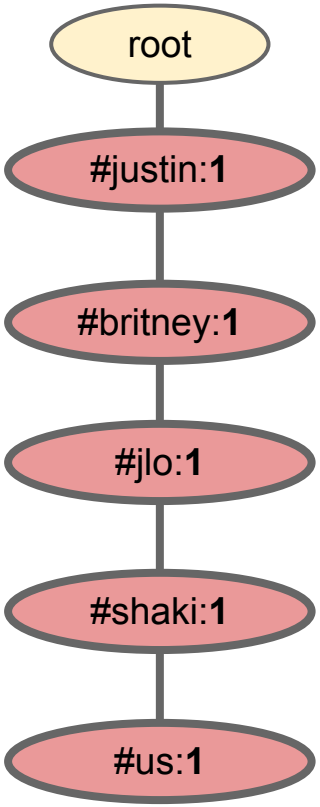
# Phase 1 - Building an FP tree

Build a tree

# Phase 1 - Building an FP tree

Build a tree and insert transactions

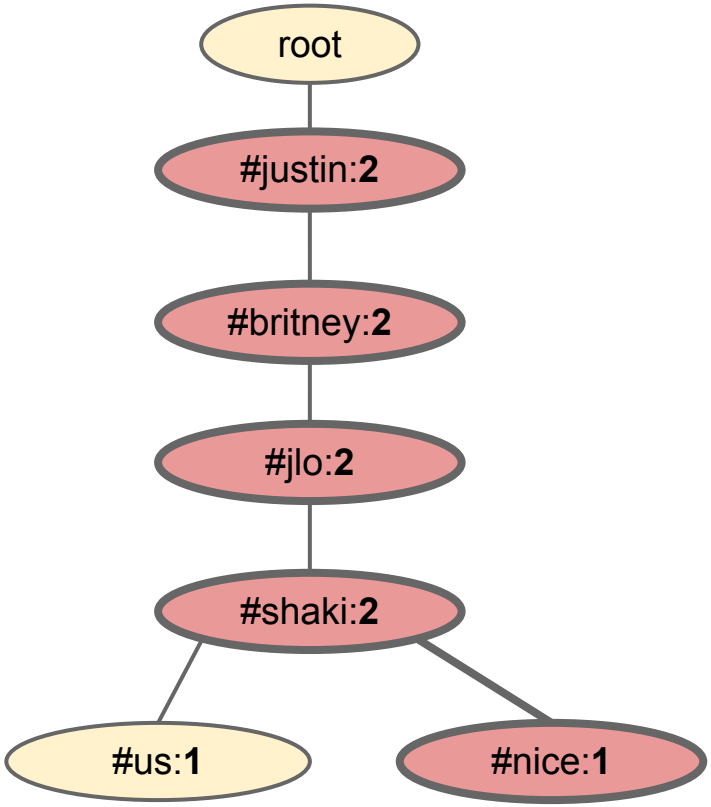
| | |
|---|---|
| 1 | {#justin, #britney, #jlo, #shaki, #us} |



# Phase 1 - Building an FP tree

Build a tree and insert transactions

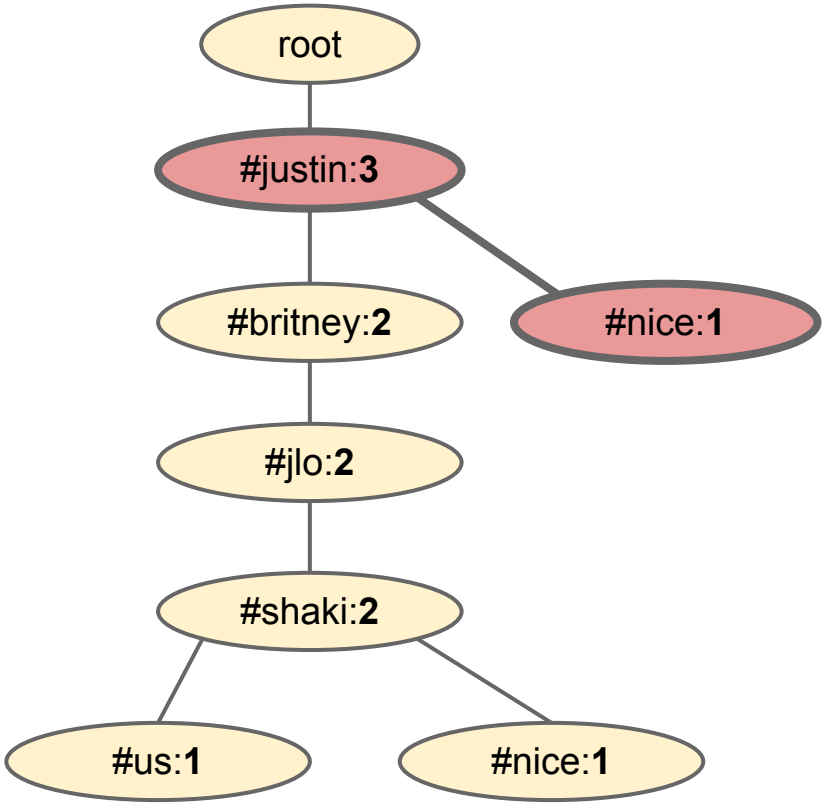
| | |
|---|---|
| 2 | {#justin, #britney, #jlo, #shaki, #nice} |



# Phase 1 - Building an FP tree

Build a tree and insert transactions

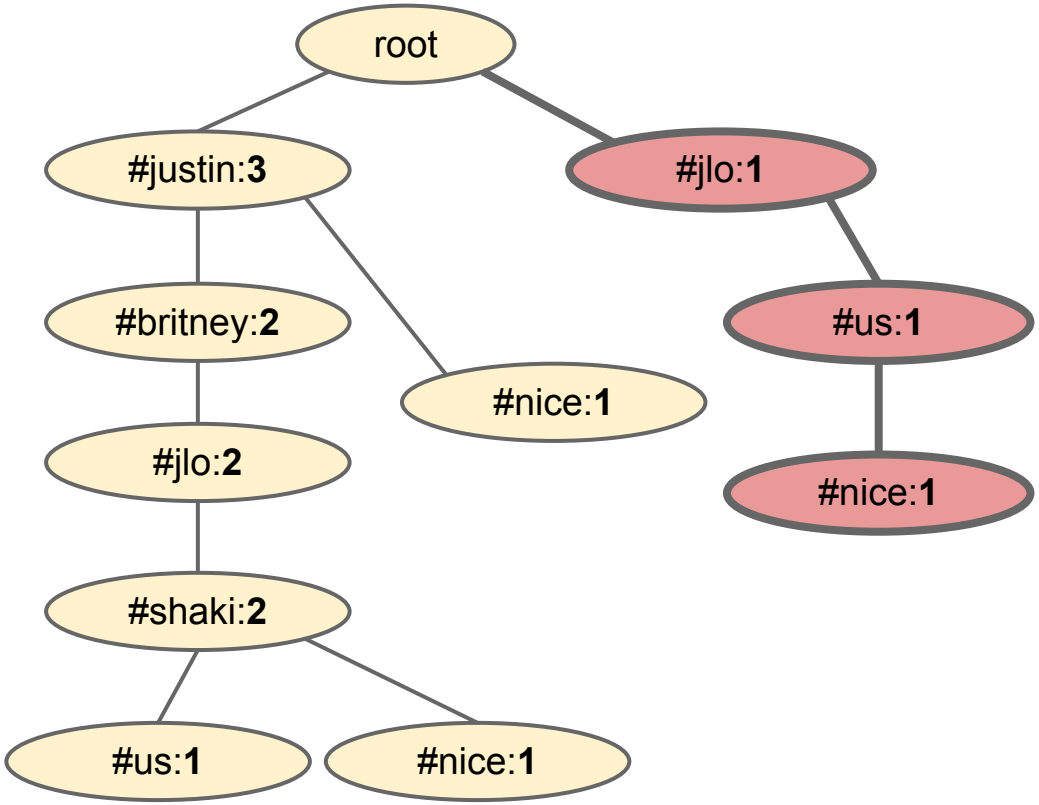
| | |
|---|---|
| 3 | {#justin, #nice} |



# Phase 1 - Building an FP tree

Build a tree and insert transactions

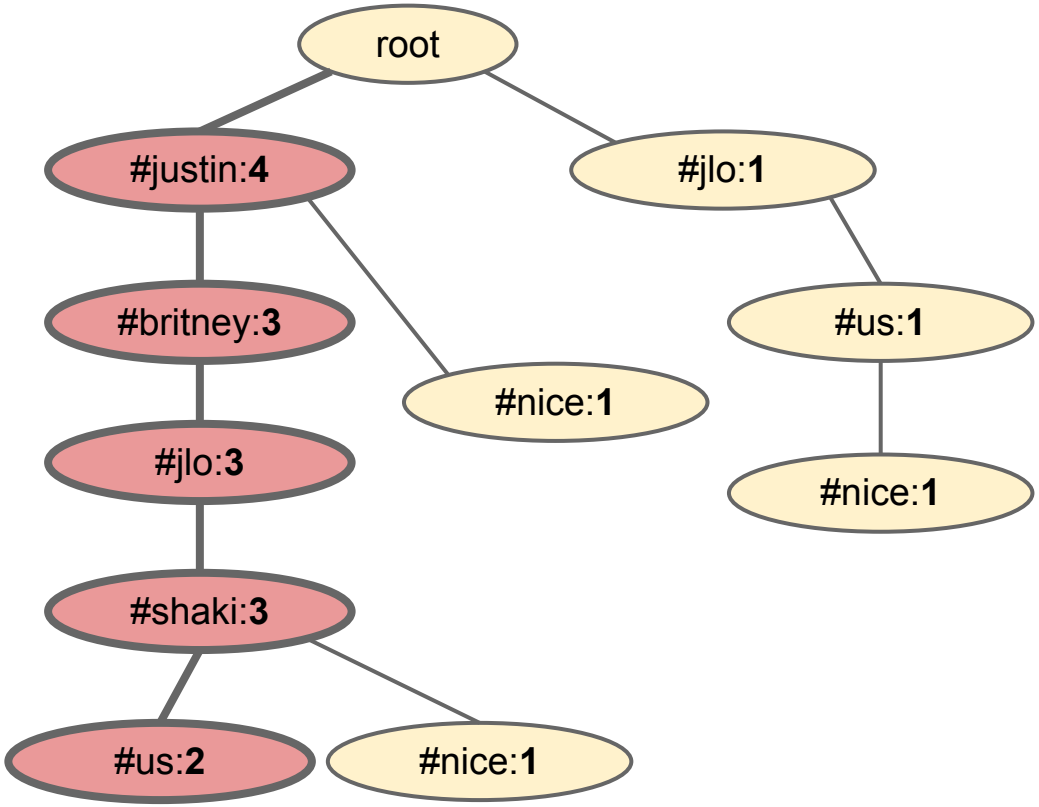
| | |
|---|---|
| 4 | {#jlo, #us, #nice} |



# Phase 1 - Building an FP tree

Build a tree and insert transactions

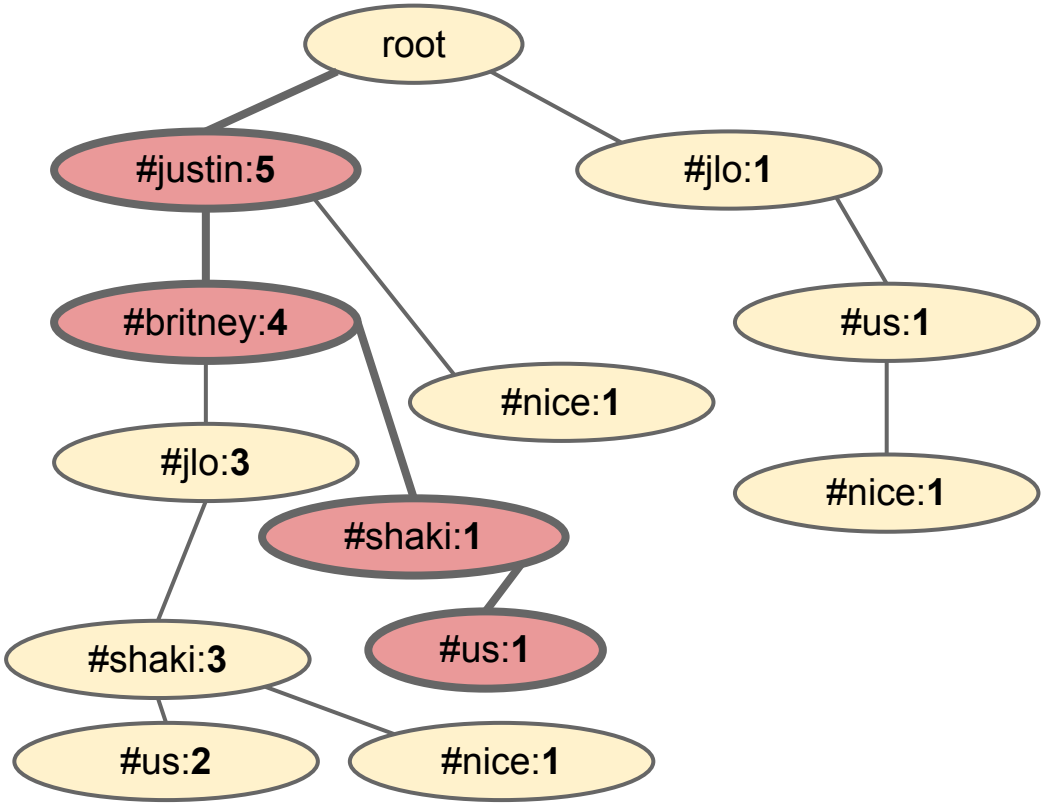
| | |
|---|---|
| 5 | {#justin, #britney, #jlo, #shaki, #us} |



# Phase 1 - Building an FP tree

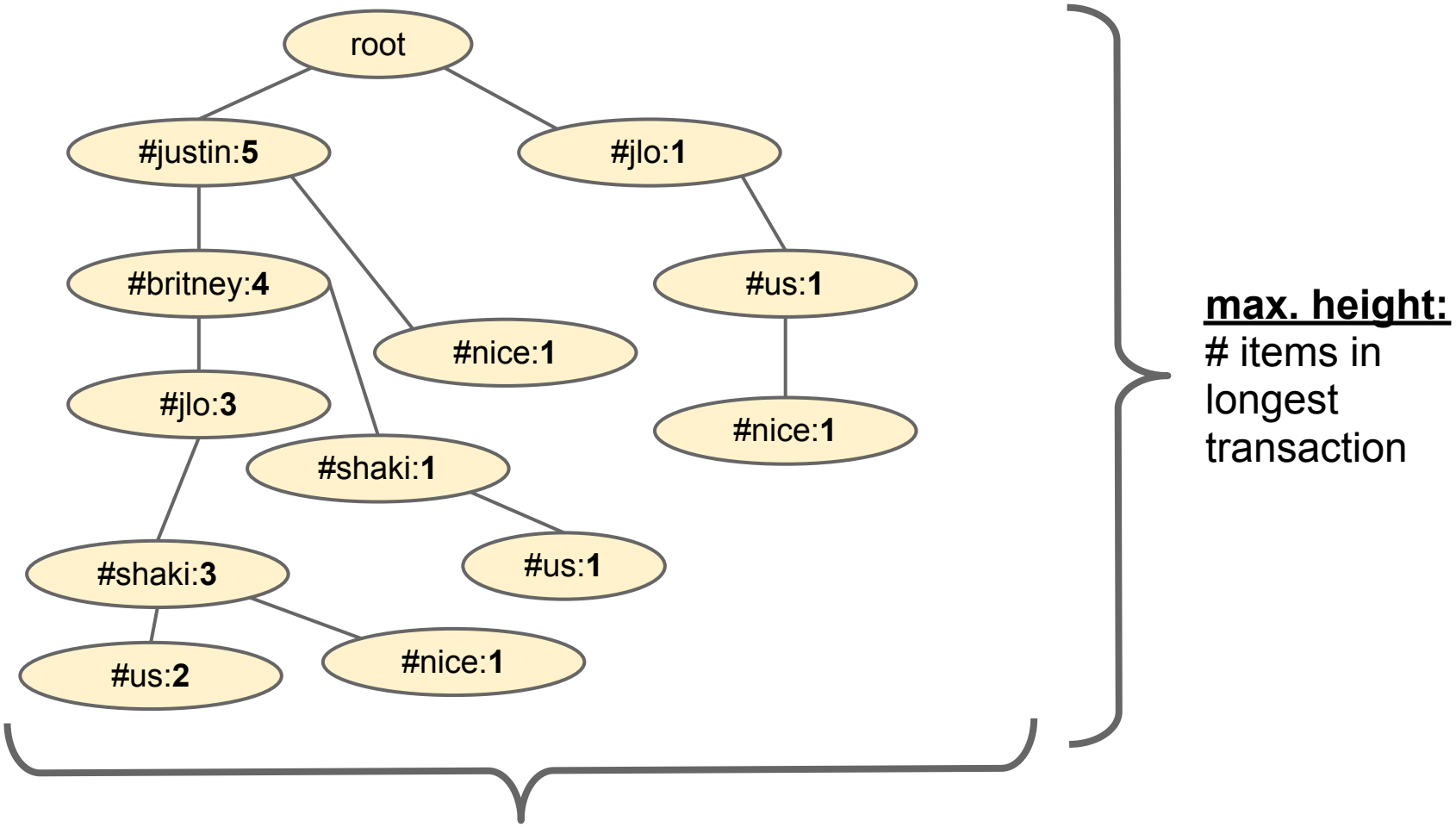
Build a tree and insert transactions

| | |
|---|---|
| 6 | {#justin, #britney, #shaki, #us} |



# Phase 1 - Building an FP tree

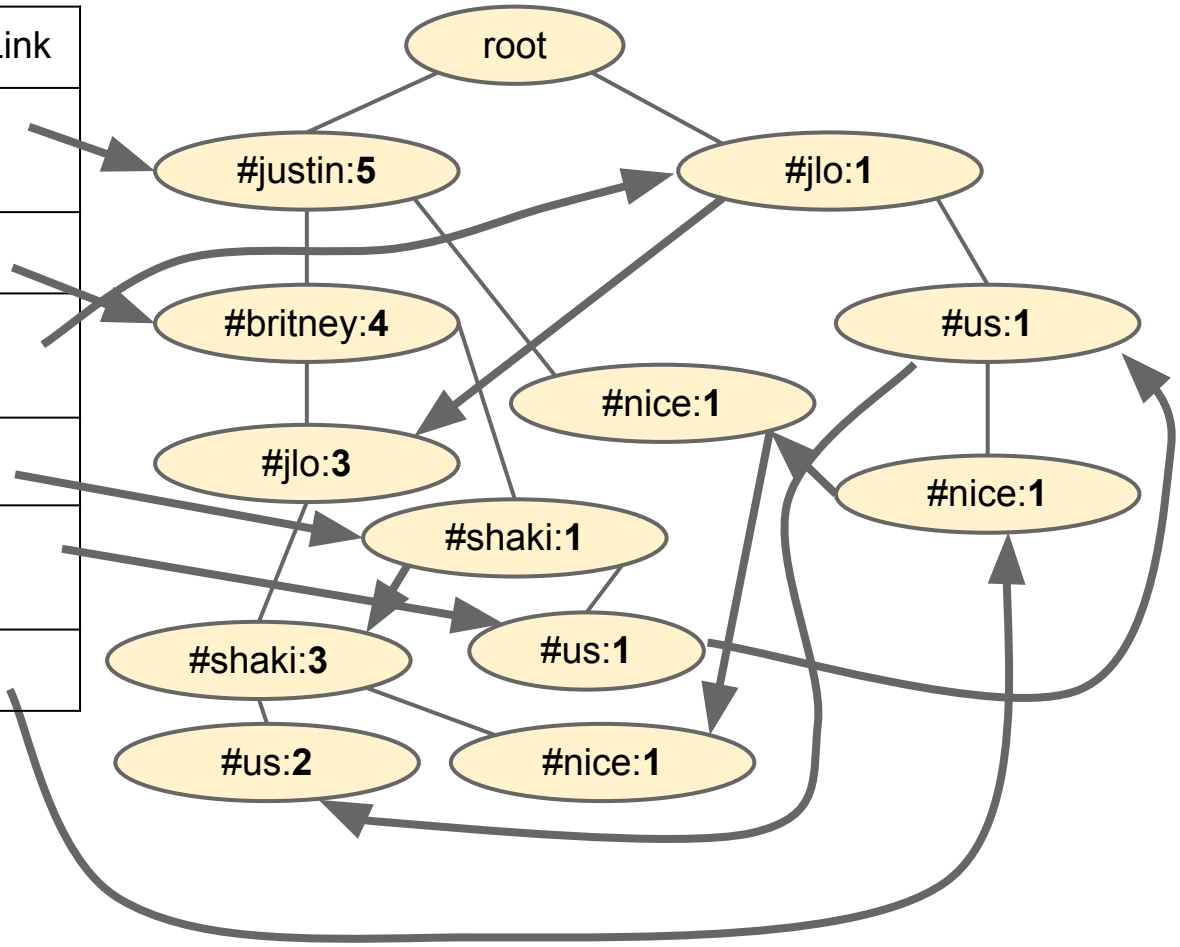
**Compact structure:** each transaction is represented



# Phase 2
## FP-Growth

# Phase 2 - Finding Frequent Itemsets

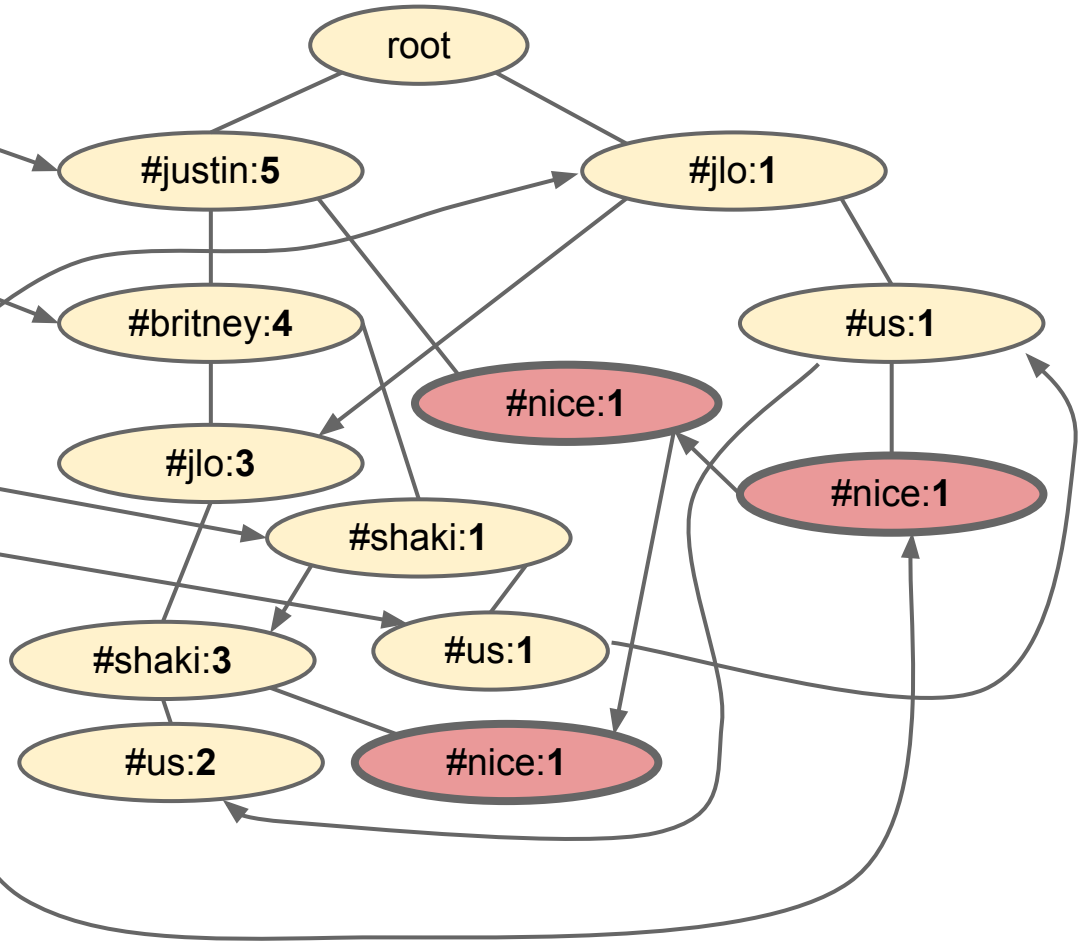
| Item | Link |
|---|---|
| #justin | |
| #britney | |
| #jlo | |
| #shaki | |
| #us | |
| #nice | |




- connect nodes with same label using header table (created while building up the tree)
- connected nodes' counters sum up to their total frequency

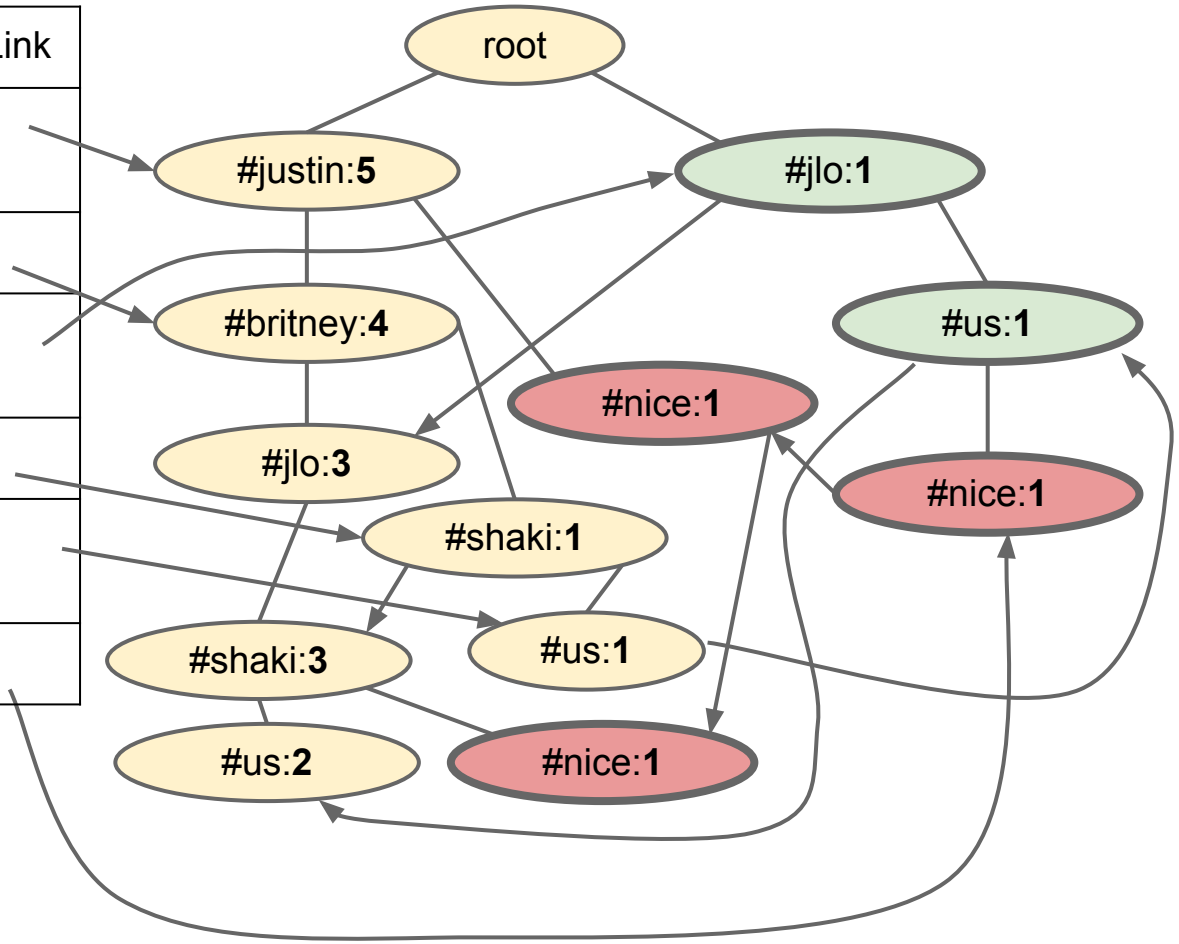
# Phase 2 - Finding Frequent Itemsets

| Item | Link |
|---|---|
| #justin | |
| #britney | |
| #jlo | |
| #shaki | |
| #us | |
| #nice | |



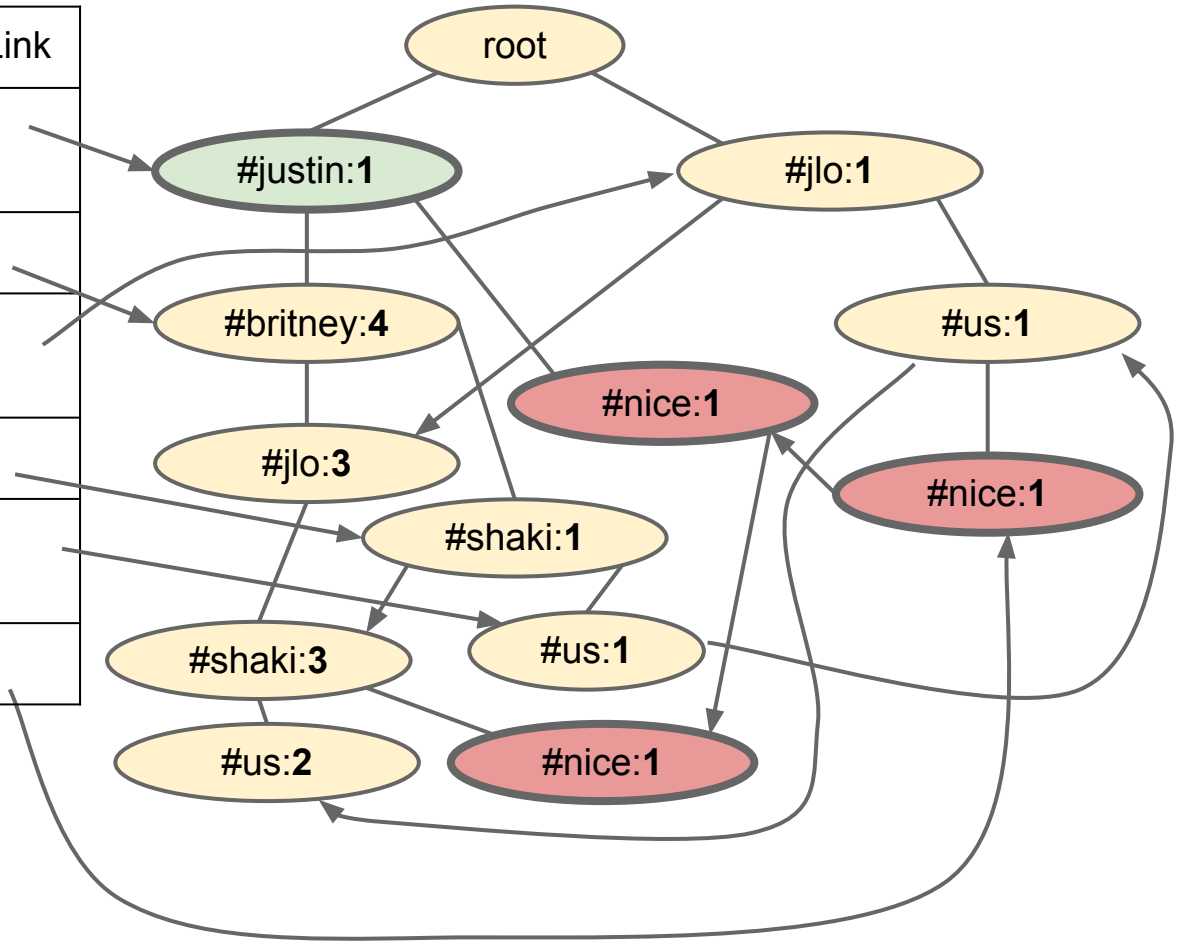
# Phase 2 - Finding Frequent Itemsets

| Item | Link |
|---|---|
| #justin | |
| #britney | |
| #jlo | |
| #shaki | |
| #us | |
| #nice | |



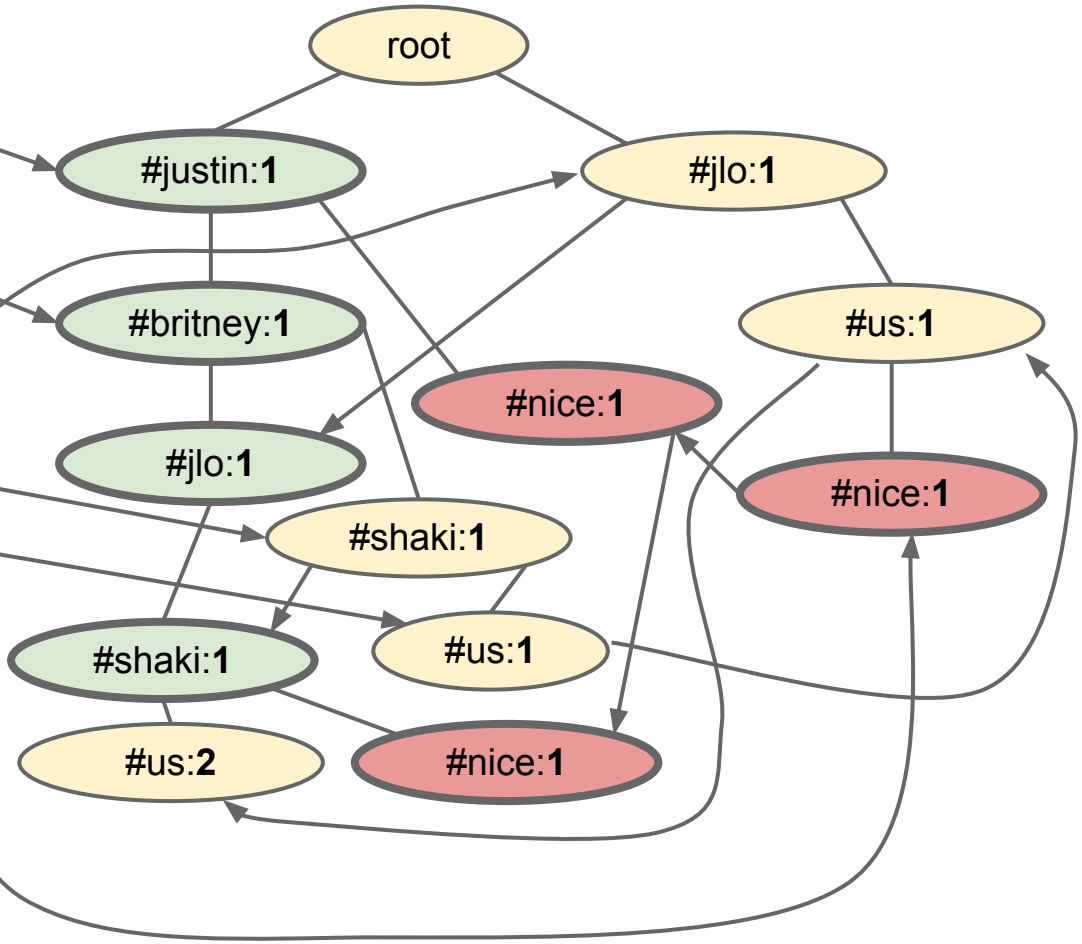
# Phase 2 - Finding Frequent Itemsets

| Item | Link |
|---|---|
| #justin | |
| #britney | |
| #jlo | |
| #shaki | |
| #us | |
| #nice | |



# Phase 2 - Finding Frequent Itemsets

| Item | Link |
|---|---|
| #justin | |
| #britney | |
| #jlo | |
| #shaki | |
| #us | |
| #nice | |



## Conditional Pattern Base for "#nice:3"

| | |
|---|---|
| 1 | {#jlo, #us} |
| 2 | {#justin} |
| 3 | {#justin, #britney, #jlo, #shaki} |

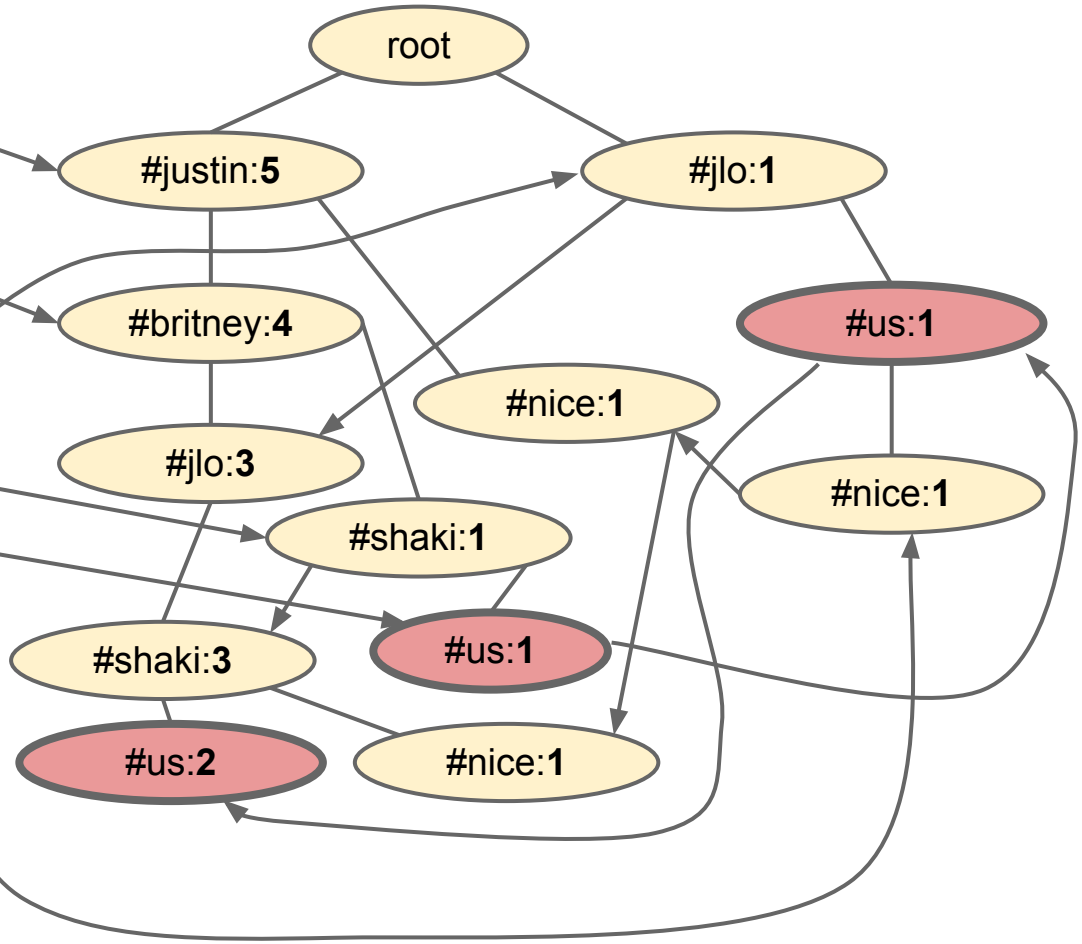
## Conditional Pattern Base for "#nice:3"

| | |
|---|---|
| 1 | {#jlo, #us} |
| 2 | {#justin} |
| 3 | {#justin, #britney, #jlo, #shaki} |

- Build (conditional) FP-tree & do FP-growth recursively
- Frequencies:
  - #jlo: 2
  - #us: 1
  - #justin: 2
  - #britney: 1
  - #shaki: 1
- no item is frequent enough (over Minimum Frequency Threshold of 3)

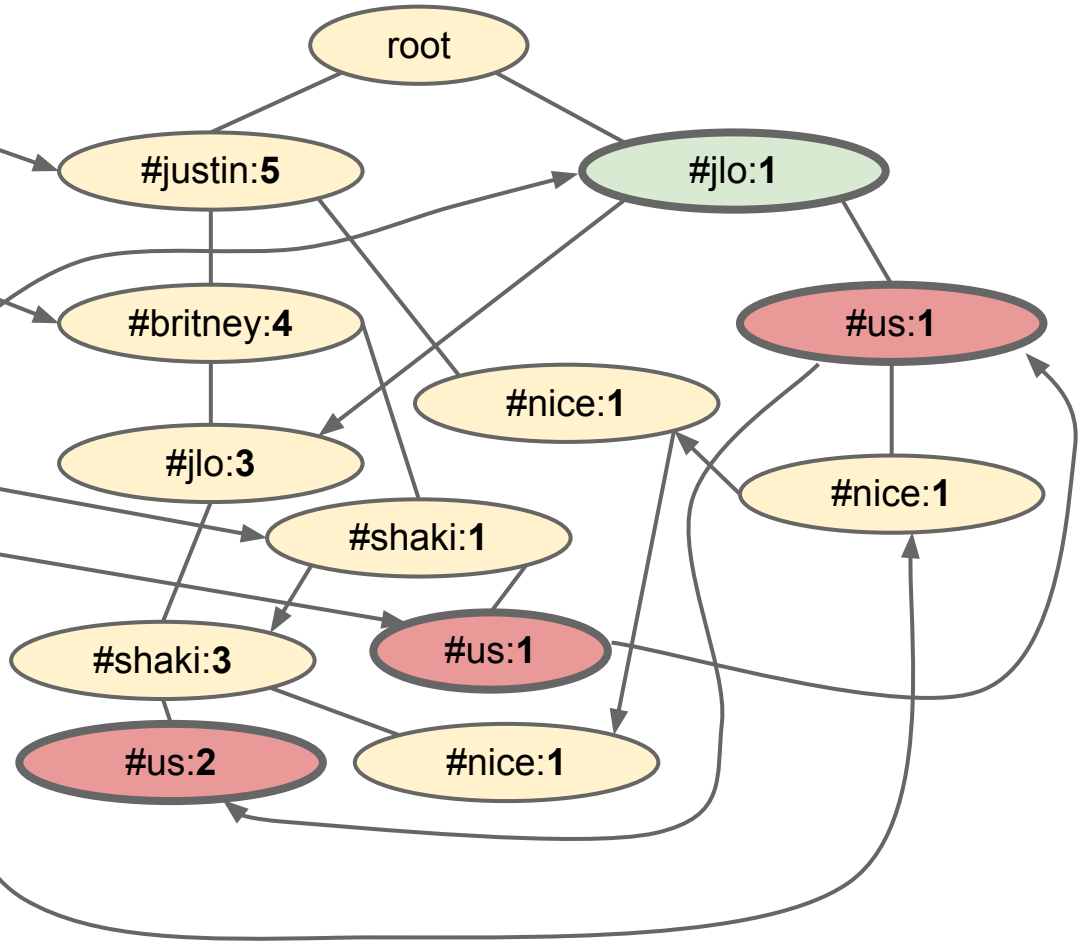
# Phase 2 - Finding Frequent Itemsets

| Item | Link |
|---|---|
| #justin | |
| #britney | |
| #jlo | |
| #shaki | |
| #us | |
| #nice | |



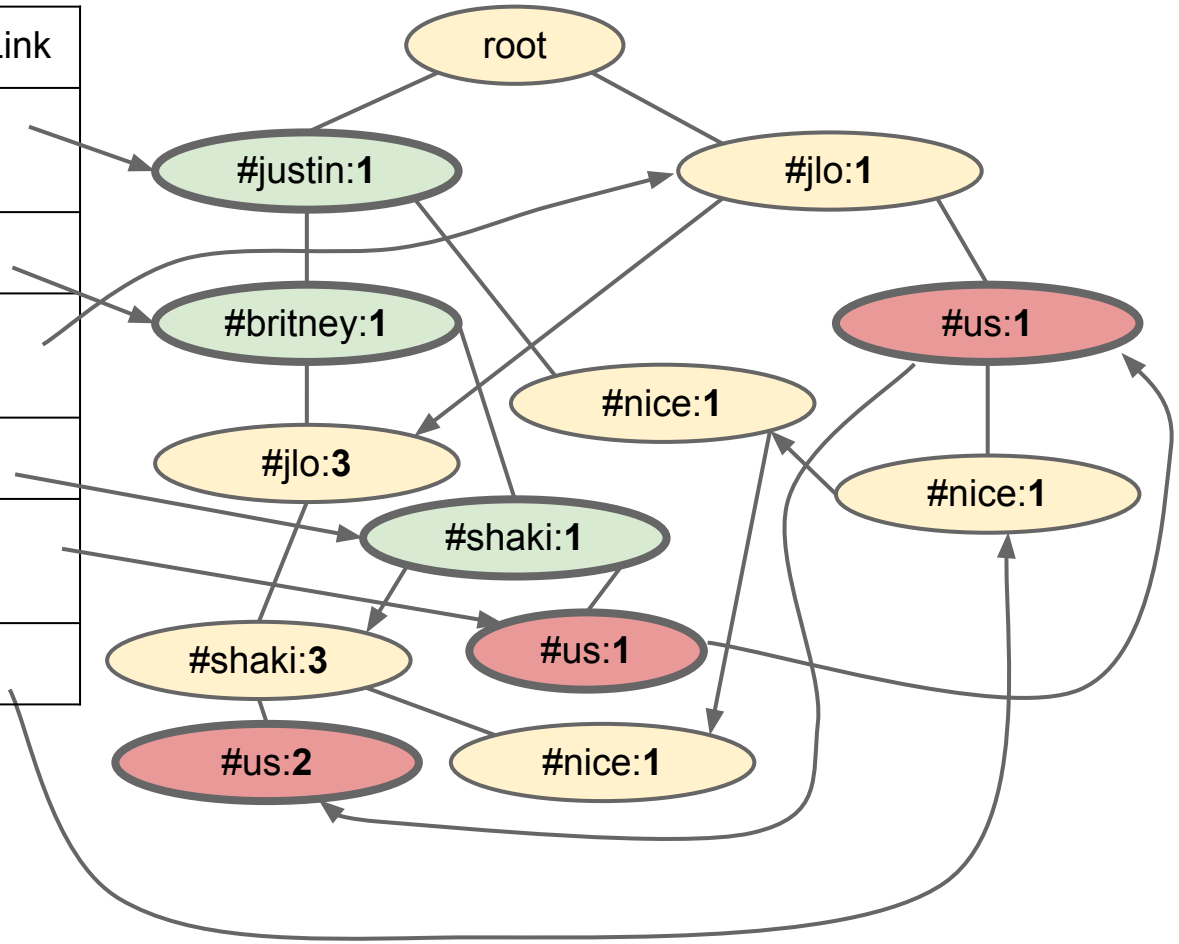
# Phase 2 - Finding Frequent Itemsets

| Item | Link |
|---|---|
| #justin | |
| #britney | |
| #jlo | |
| #shaki | |
| #us | |
| #nice | |



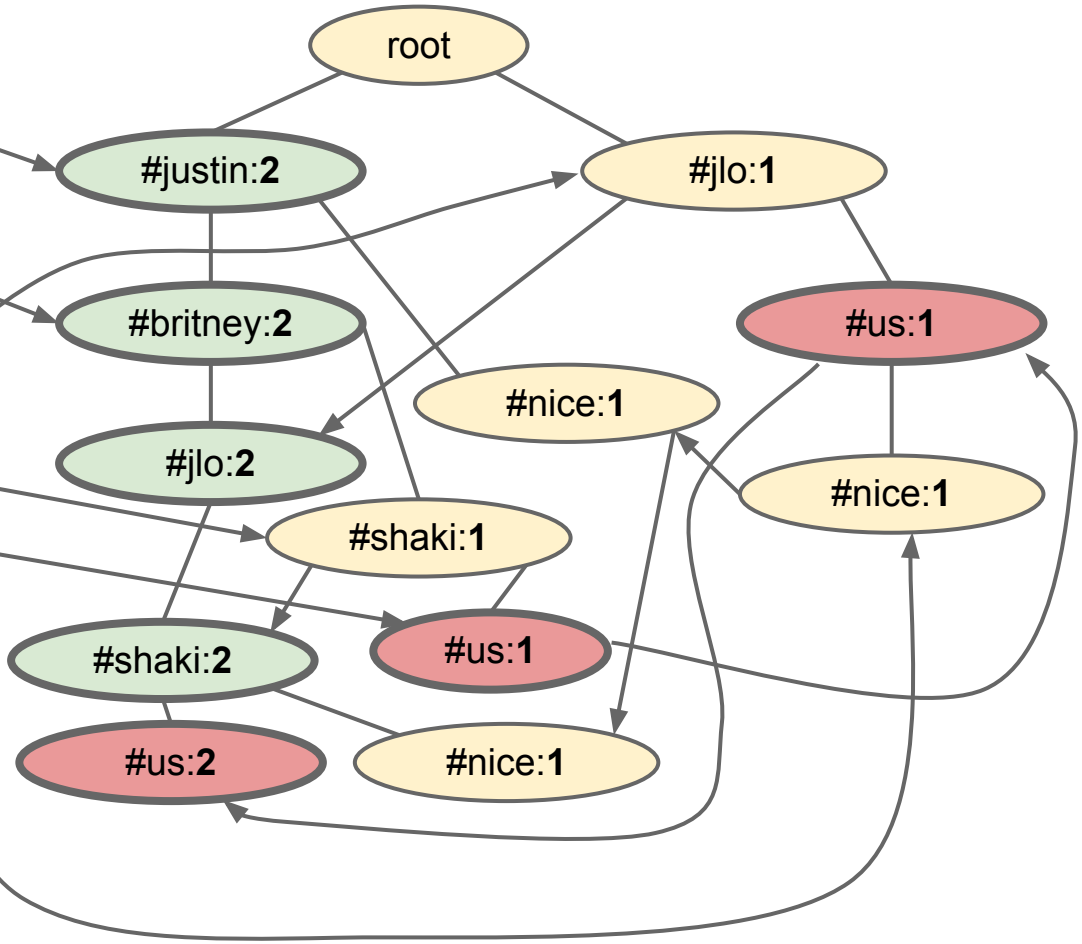
# Phase 2 - Finding Frequent Itemsets

| Item | Link |
|---|---|
| #justin | |
| #britney | |
| #jlo | |
| #shaki | |
| #us | |
| #nice | |



# Phase 2 - Finding Frequent Itemsets

| Item | Link |
|---|---|
| #justin | |
| #britney | |
| #jlo | |
| #shaki | |
| #us | |
| #nice | |



## Conditional Pattern Base for "#us:4"

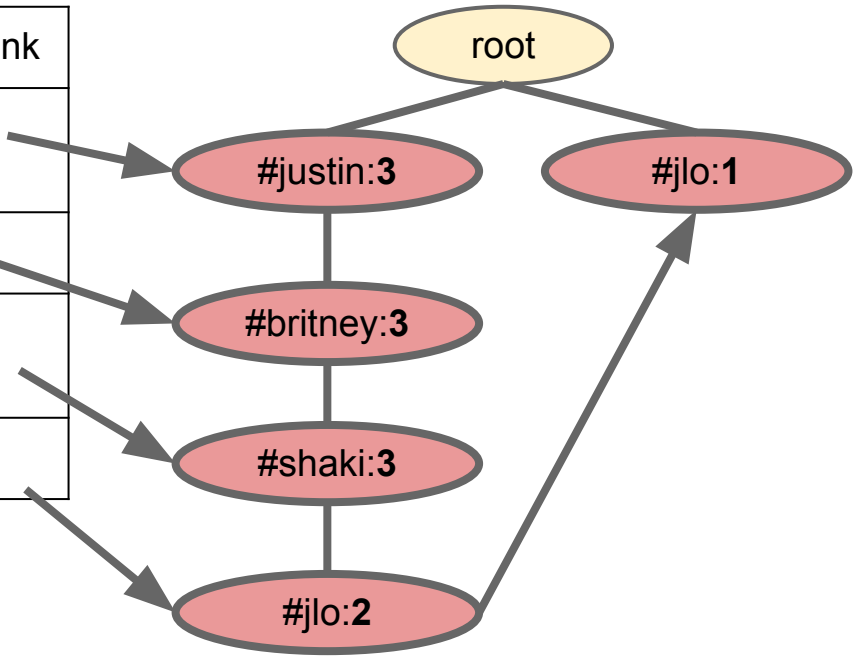
| | |
|---|---|
| 1 | {#jlo} |
| 2 | {#justin, #britney, #shaki} |
| 3 | {#justin, #britney, #shaki, #jlo} |
| 4 | {#justin, #britney, #shaki, #jlo} |

- Each item has a frequency of 3

## Conditional Pattern Base for "#us:4"

| | |
|---|---|
| 1 | {#jlo} |
| 2 | {#justin, #britney, #shaki} |
| 3 | {#justin, #britney, #shaki, #jlo} |
| 4 | {#justin, #britney, #shaki, #jlo} |

| Item | Link |
|---|---|
| #justin | |
| #britney | |
| #shaki | |
| #jlo | |

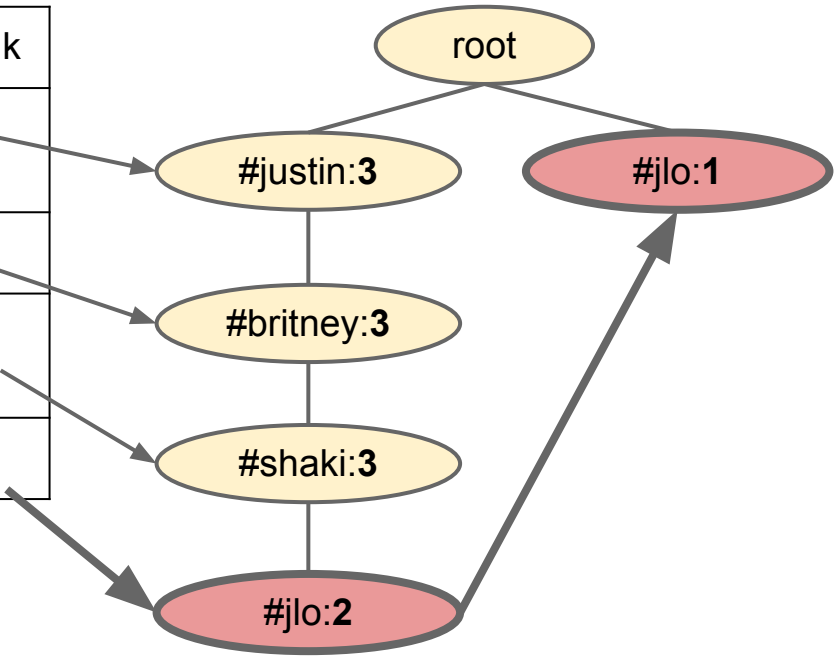



- Each item has a frequency of 3
- Apply FP-growth on the conditional pattern tree (recursively)
- "#us" must always be added

## Conditional Pattern Base for "#us:4"

| | |
|---|---|
| 1 | {#jlo} |
| 2 | {#justin, #britney, #shaki} |
| 3 | {#justin, #britney, #shaki, #jlo} |
| 4 | {#justin, #britney, #shaki, #jlo} |

| Item | Link |
|---|---|
| #justin | |
| #britney | |
| #shaki | |
| #jlo | |

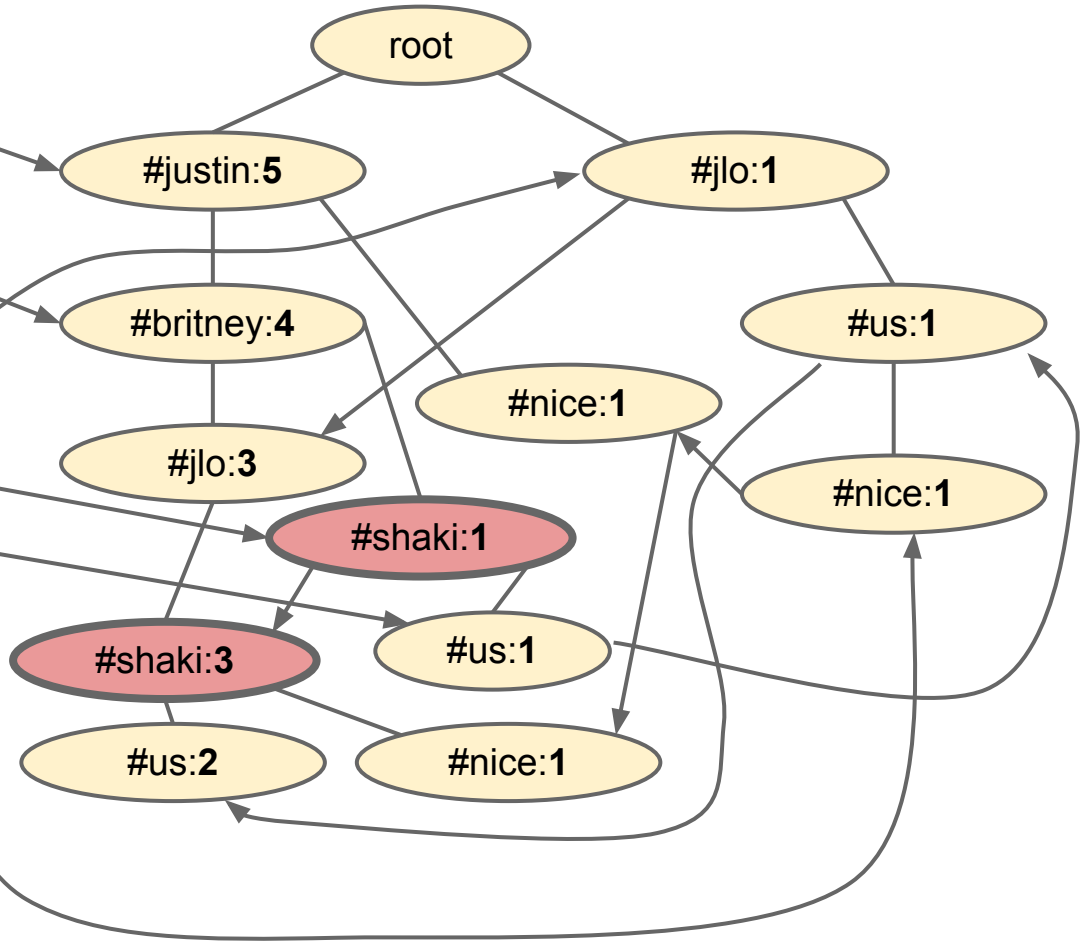



- Each item has a frequency of 3
- Apply FP-growth on the conditional pattern tree (recursively)
- "#us" must always be added

**This results in {#us, #jlo} : 3**

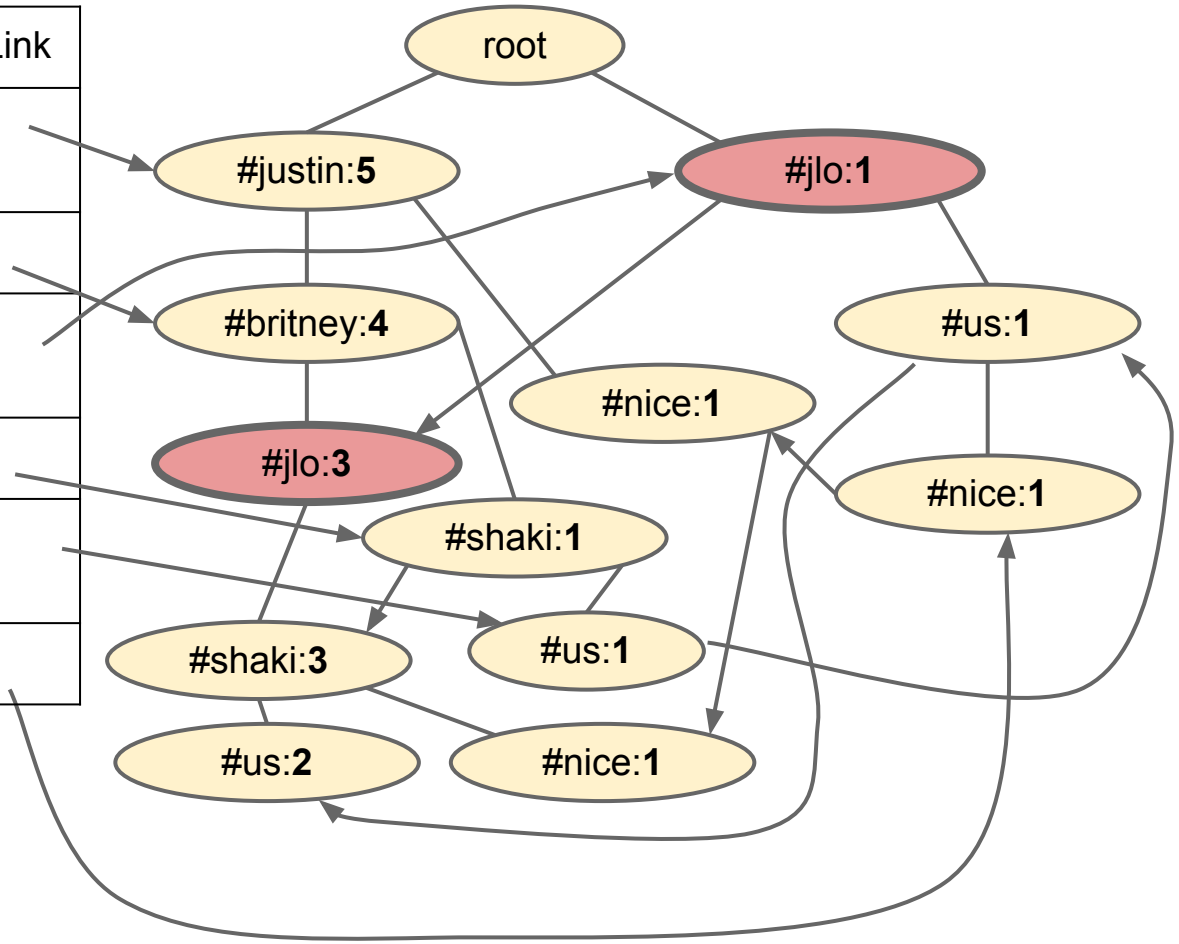
# Phase 2 - Finding Frequent Itemsets

| Item | Link |
|---|---|
| #justin | |
| #britney | |
| #jlo | |
| #shaki | |
| #us | |
| #nice | |



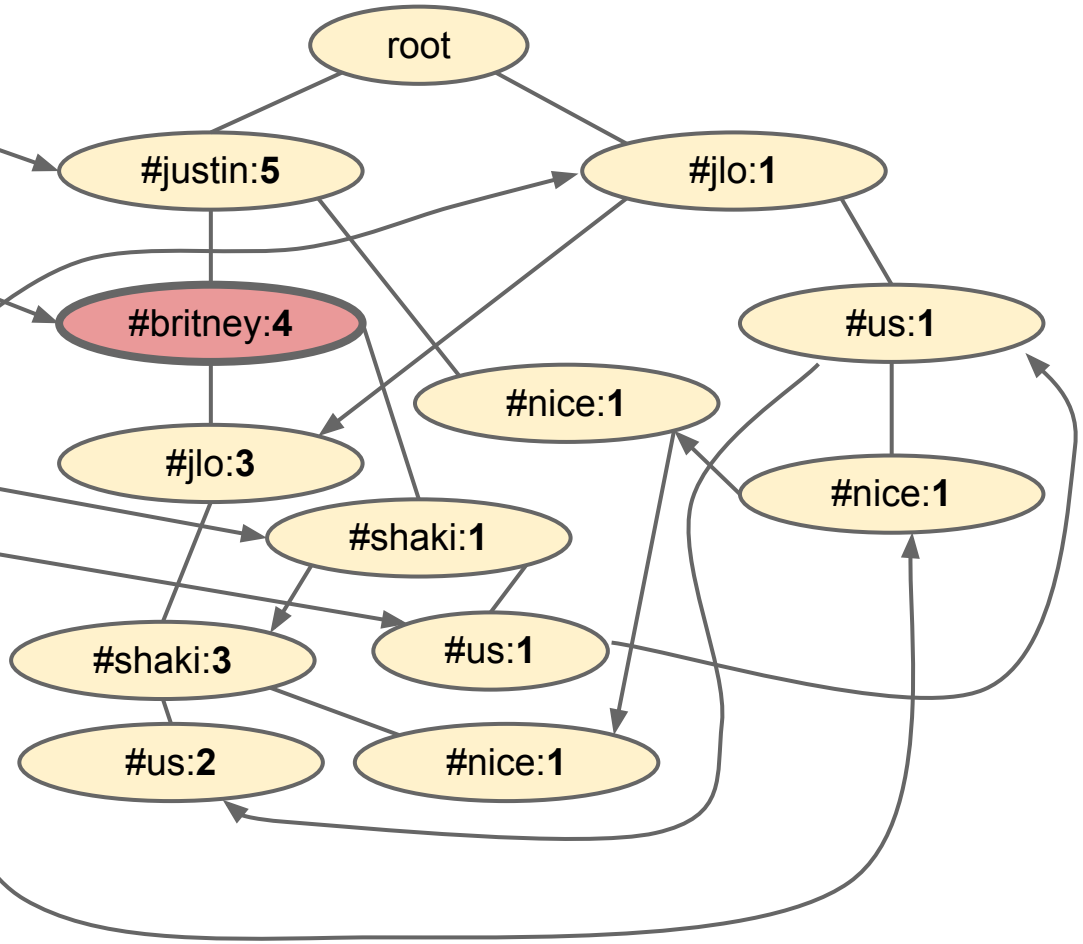
# Phase 2 - Finding Frequent Itemsets

| Item | Link |
|---|---|
| #justin | |
| #britney | |
| #jlo | |
| #shaki | |
| #us | |
| #nice | |



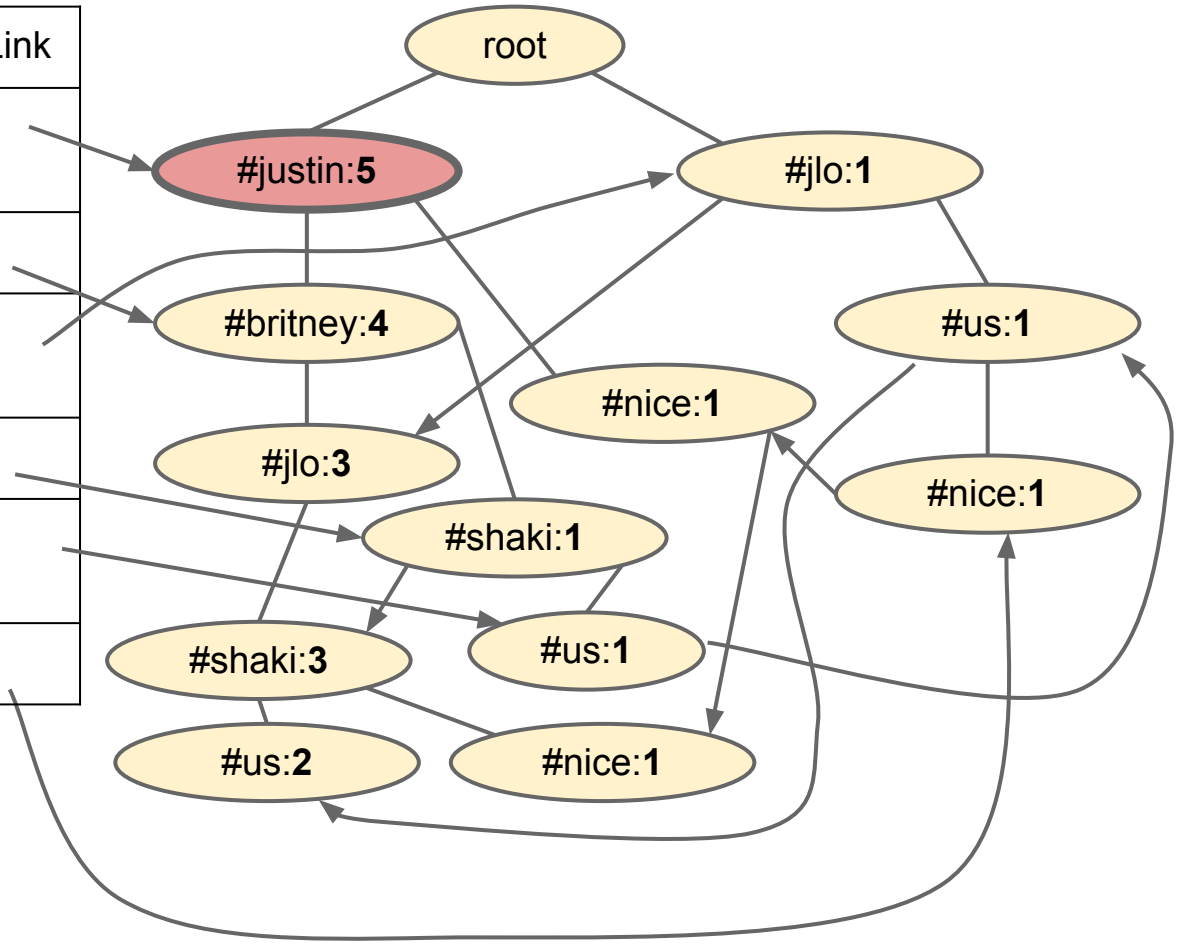
# Phase 2 - Finding Frequent Itemsets

| Item | Link |
|---|---|
| #justin | |
| #britney | |
| #jlo | |
| #shaki | |
| #us | |
| #nice | |



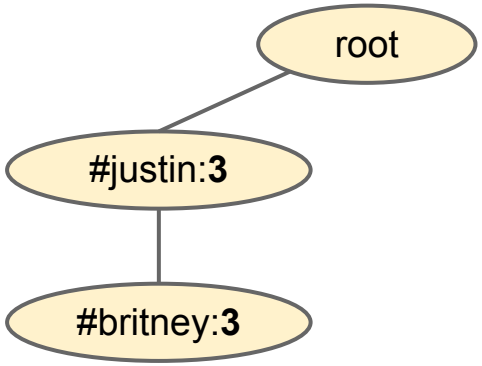
# Phase 2 - Finding Frequent Itemsets

| Item | Link |
|---|---|
| #justin | |
| #britney | |
| #jlo | |
| #shaki | |
| #us | |
| #nice | |



- If conditional FP tree consist of only one path:
  all combinations are frequent patterns

## Conditional pattern tree for #jlo: 4




results in:

- {#jlo} : 4
- {#jlo, #justin} : 3
- {#jlo, #britney} : 3
- {#jlo, #justin, #britney} : 3

# Experiment Environment

- Scala Implementation (JVM)
  - Specification tests for algorithm
  - Runner for different measurement / logging settings
  - different possible data readers (here: CSV)
- Measurements on laptop with SSD

- **Data:**
  - Hashtags of Twitter tweets (#xyz)
  - 2 different datasets:
    - 389,895 tweets (Maximilian Jenders)
      - Partly randomly selected set
      - Filtered out obvious spam
      - Used for measurement
      - Average transaction length: 3.4076
    - 138,120 tweets (Matthias Kohnen; "extra" data for comparison)
      - "All" tweets coming in through the Twitter Streaming API
      - Not explicitly filtered
      - Average transaction length: 3.7981
    - (Only worked with tweets with 2 or more hashtags)

# Some Experiment Results (Max)

- p2, tcot (4033)
- humanrights, news (3279)
- auto, car (2018)
- instantfollowback, followback (1985)
- tea, chai (1679)
- fb, news (1362)
- punchlines, listen2me (1504)
- 2futures, 4jobs (745)
- pregnancy, health (351)
- animals, pets (341)
- disease, medical, diabetes, illness (282)
- joke, pastor, tshirt, christian (107)
- unity, harmony, consciousness (69)
- environment, sustainability, green (62)

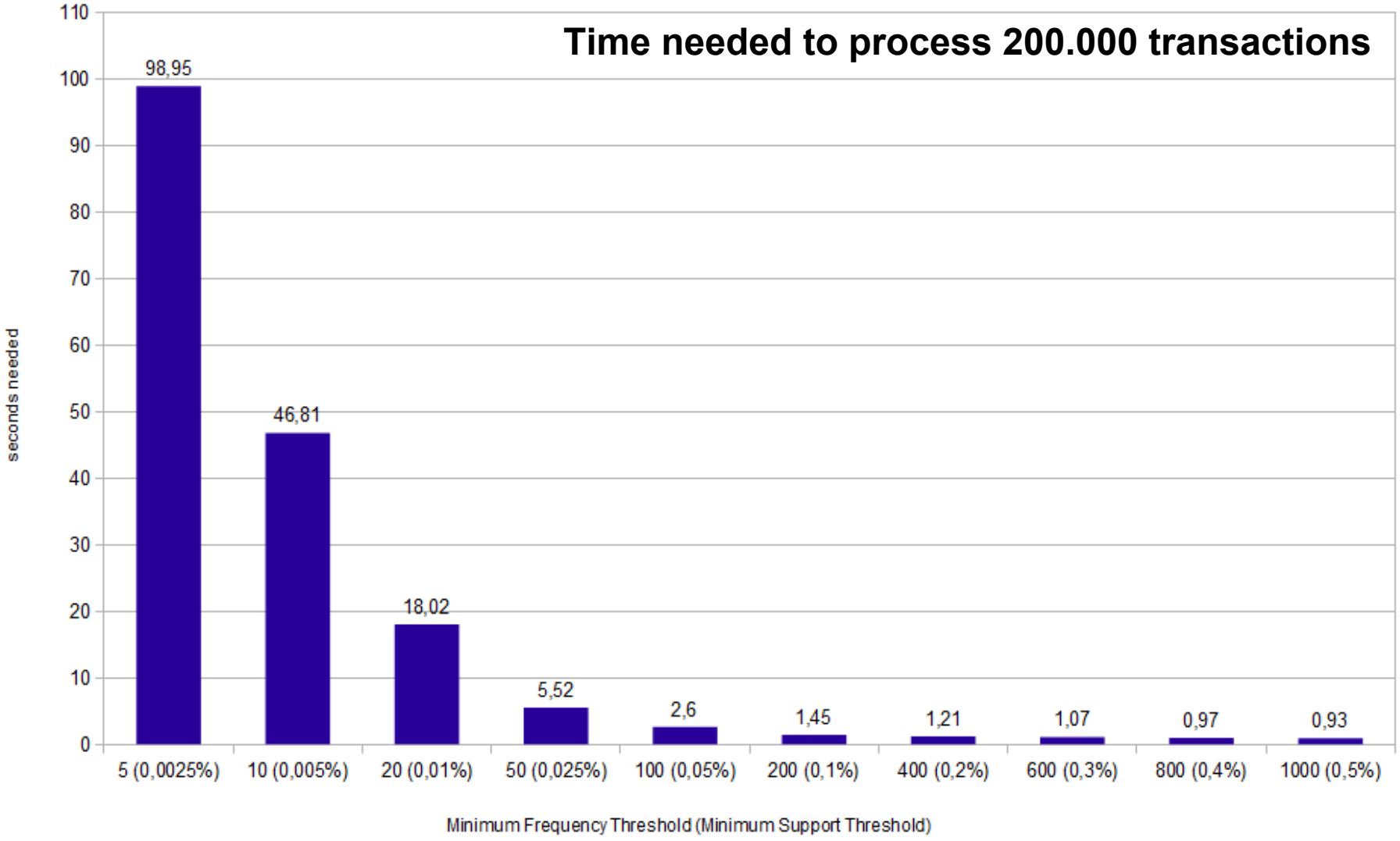
for more information, see: **http://tagdef.com/**

# Some Experiment Results (Matthias; unfiltered!)

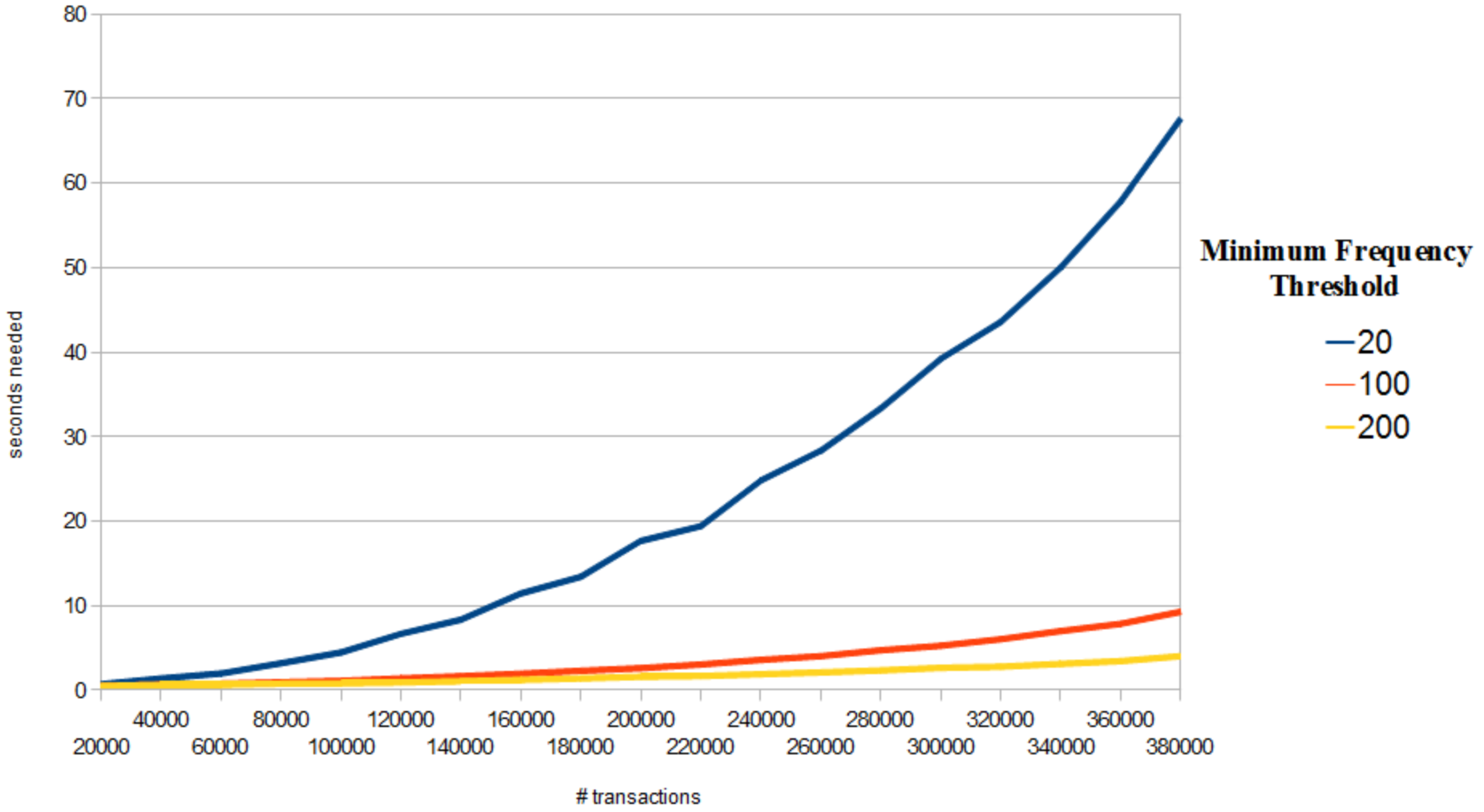
- porn, sex (1696)
- followmejp, sougofollow (948)
- xxx, porn (930)
- sougofollow, followme (898)
- followmejp, followme (740)
- followme, teamfollowback (678)
- follow, followme (673)
- xxx, sex (662)
- sougofollow, followmejp, followme (654)
- TFB, TeamFollowBack (648)
- followback, followme (619)
- Android, Androidgames (609)
- follow, sougofollow (593)

for more information, see: **http://tagdef.com/**

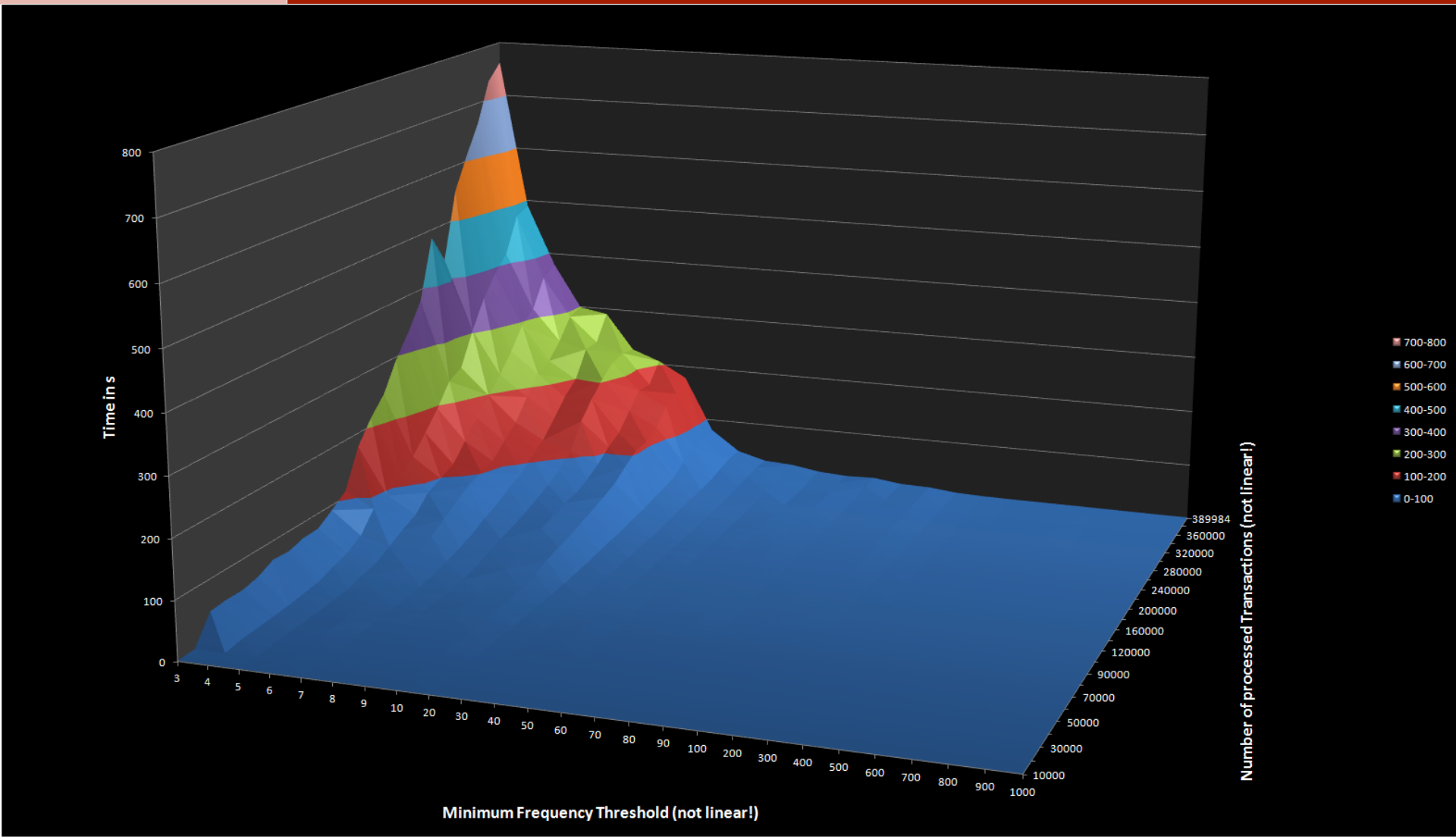
# Algorithm Evaluation



# Algorithm Evaluation



# Algorithm Evaluation



# Problems

- Implementation:
  - Separate data structures for each recursion step
  - Don't let the recursion steps destroy foreign data!

- Runtime:
  - Too low Minimum Frequency Threshold (MFT) causes Heap Space Error
    - Even compact tree structure can become large
  - For ~400.000 transactions with an MFT of 3, we needed to increase Java's reserved memory to 3GB
  - For a MFT of 2, even this isn't enough

# Future Work

- Possible Extensions:
  - Parallelization
    - General intent: increase performance
    - Map/Reduce approach already described in a successor paper
    - Our idea: parallelization for single computers: usage of Scala actors system
      - Wouldn't solve main memory issue
      - Comparable to the already collected results
  - Make algorithm incremental
    - New tweets are coming in frequently
    - Transactions have expiration date
    - Discovering trends in certain time frames easily

# References


- Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data* (SIGMOD '00). ACM, New York, NY, USA, 1-12.

- Haoyuan Li, Yi Wang, Dong Zhang, Ming Zhang, and Edward Y. Chang. 2008. Pfp: parallel fp-growth for query recommendation. In *Proceedings of the 2008 ACM conference on Recommender systems* (RecSys '08). ACM, New York, NY, USA, 107-114.

- Xiu-Li Ma, Yun-Hai Tong, Shi-Wei Tang, and Dong-Qing Yang. 2004. Efficient incremental maintenance of frequent patterns with FP-tree. *J. Comput. Sci. Technol.* 19, 6 (November 2004), 876-884.