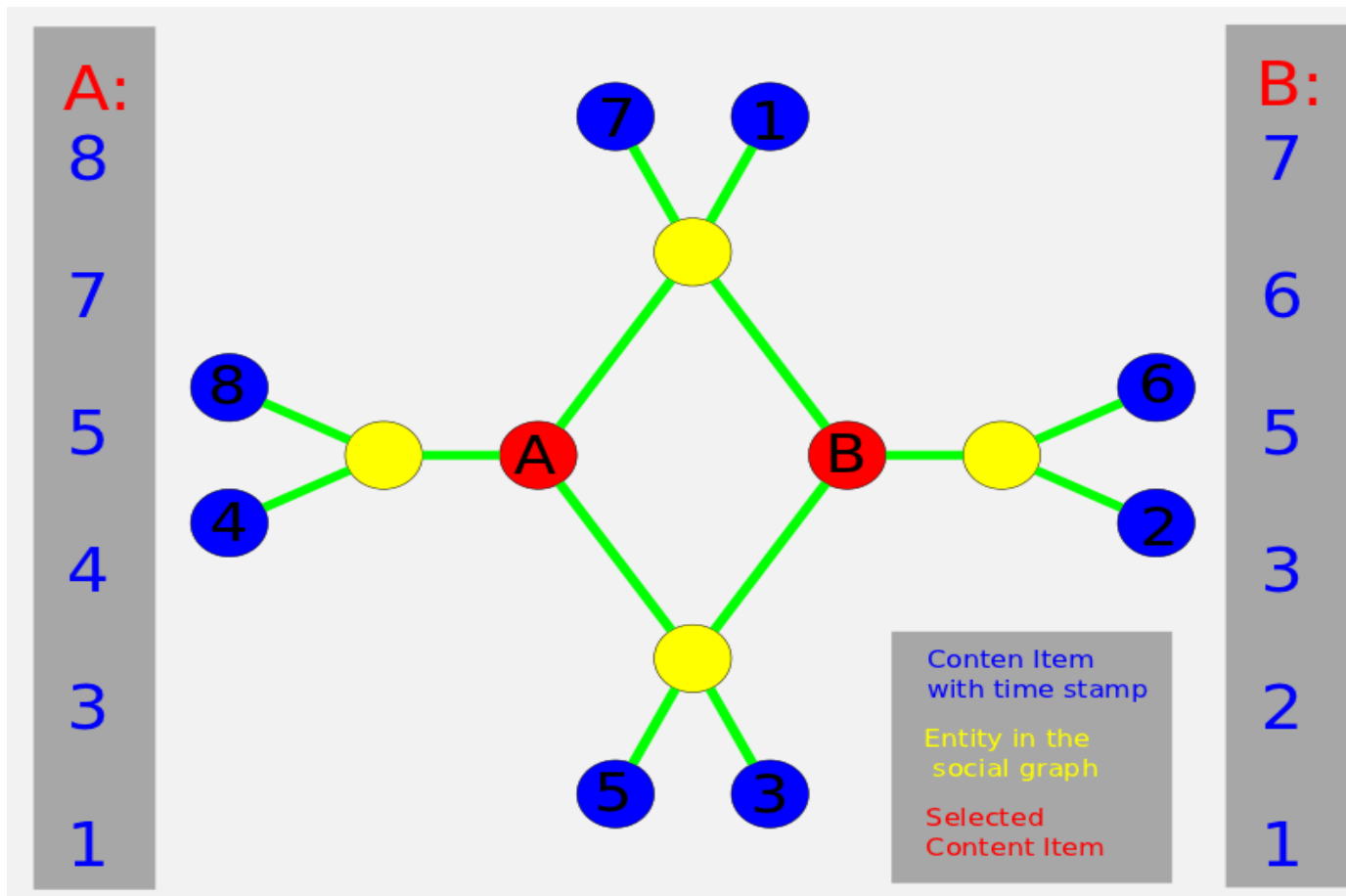


Social news streams and time indices on graphs for social networks

René Pickhardt



- Motivation
- Introduction to the problem
- Best practices
- my own solution
 - failed solution
 - successfull solution

The strength of weak ties:

"Stress is laid on the cohesive power of weak ties, thus confining their applicability to small, well defined groups. Emphasis on weak ties lends itself to discussion of relations between groups and to analysis of segments of social structure no easily defined in terms of primary groups" [1]

A user wants to receive the most relevant news from his

- social circle &
- topics of interest

[1] Mark S. Granovetter - American journal of sociology, 1973 - JSTOR

Introduction to the problem

Input:

- Given a (social) network graph that contains
 - entities and relations between them as well as
 - content items that are created or owned by these entities and linked to them.
 - content items have some kind of timestamp

Task:

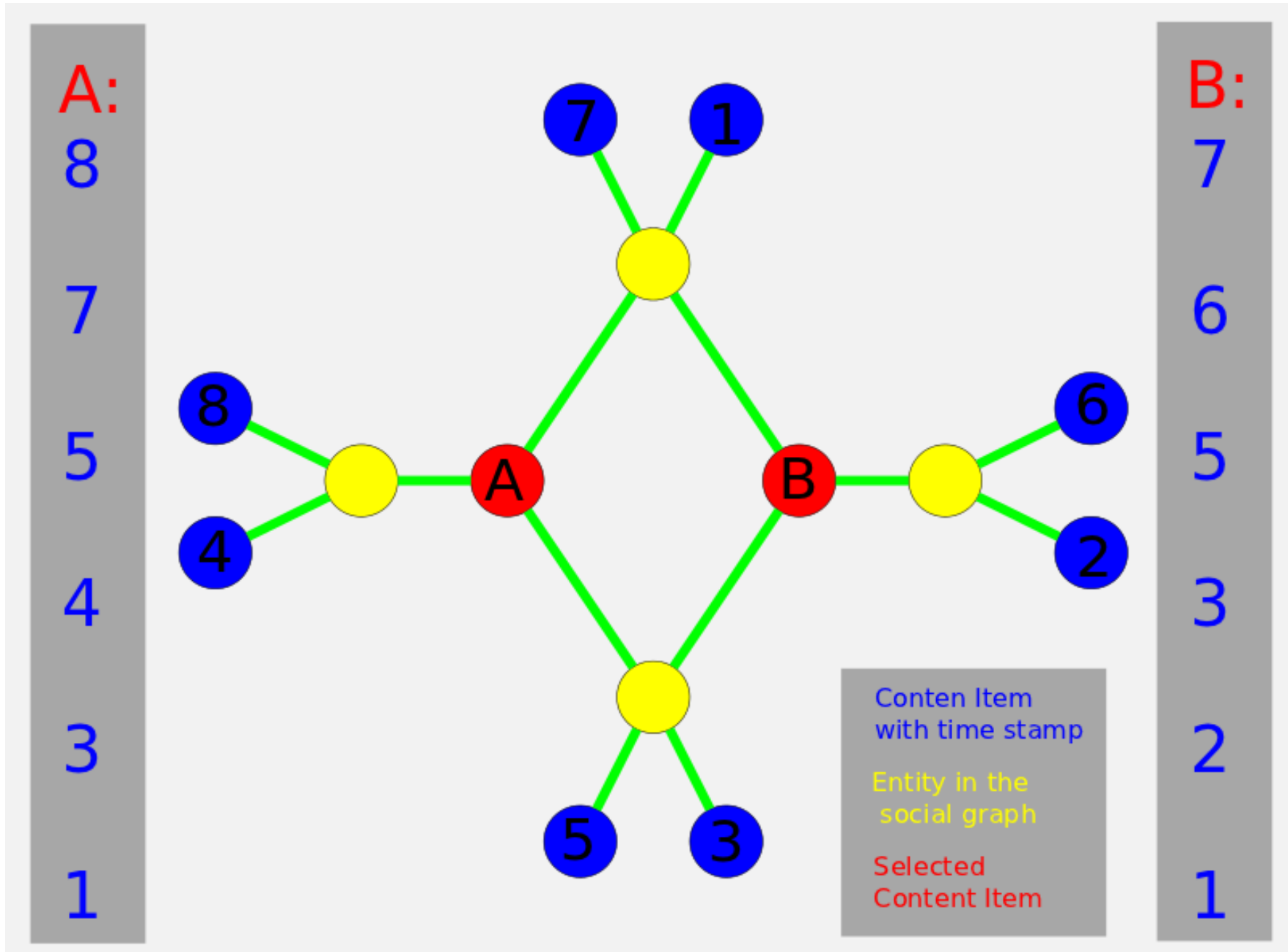
- Choose an arbitrary entity X .
- Retrieve content items from X 's (extended) egonet network that are
 - most recent &
 - most relevant (!) (i.e. filtering!)

Output:

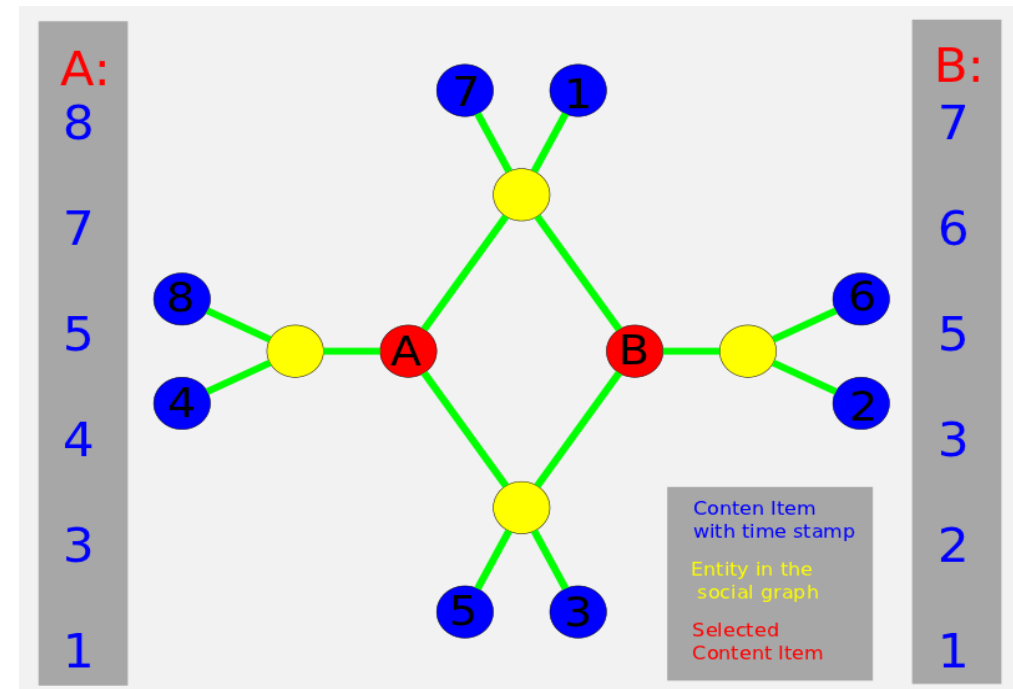
- a list of content items from X 's ego network sorted by time.

	Web search	News stream
Ranking	by relevance	by time
Filtering	no filtering	by relevance

- Information retrieval with
 - very view (?) relevance measures.
 - a strong time dependent component
- The problem seems to be local.
 - just the ego network of a user seems to be important
- Graphs seem to
 - be the natural modelling
 - have problems with time dependence



- create news stream with breadth first search
- sorting the content items after BFS is too time consuming
- Even if no sorting was required for BFS we would still have to traverse too many nodes
- we don't know in which direction to start BFS
- Timestamps are useless at
 - edges
 - entities



1. very fast reading

- independent of network size
- independent of node degree

1. scaling

- low storage (?)
- ...

2. flexibility

- news stream should be sensitive to changes of the graph
- not to be "precalculated"

Best Practices

- **Twitter created FlockDB [1]**
 - index for MySQL
 - traversing adjacency lists
 - able to handle sparse relations
 - low storage (?)
- **pro:**
 1. very dynamic
- **contra:**
 1. joins are expensive
 2. joins don't really scale



[1] <http://engineering.twitter.com/2010/05/introducing-flockdb.html>

- **Flat files**

- for every User exists a flat file with his news stream
- new content ==> update of all relevant files
 - Lady Gaga: "hi" ==> 42 mio updates!

- **pro:**

1. very fast reading
2. scales perfectly

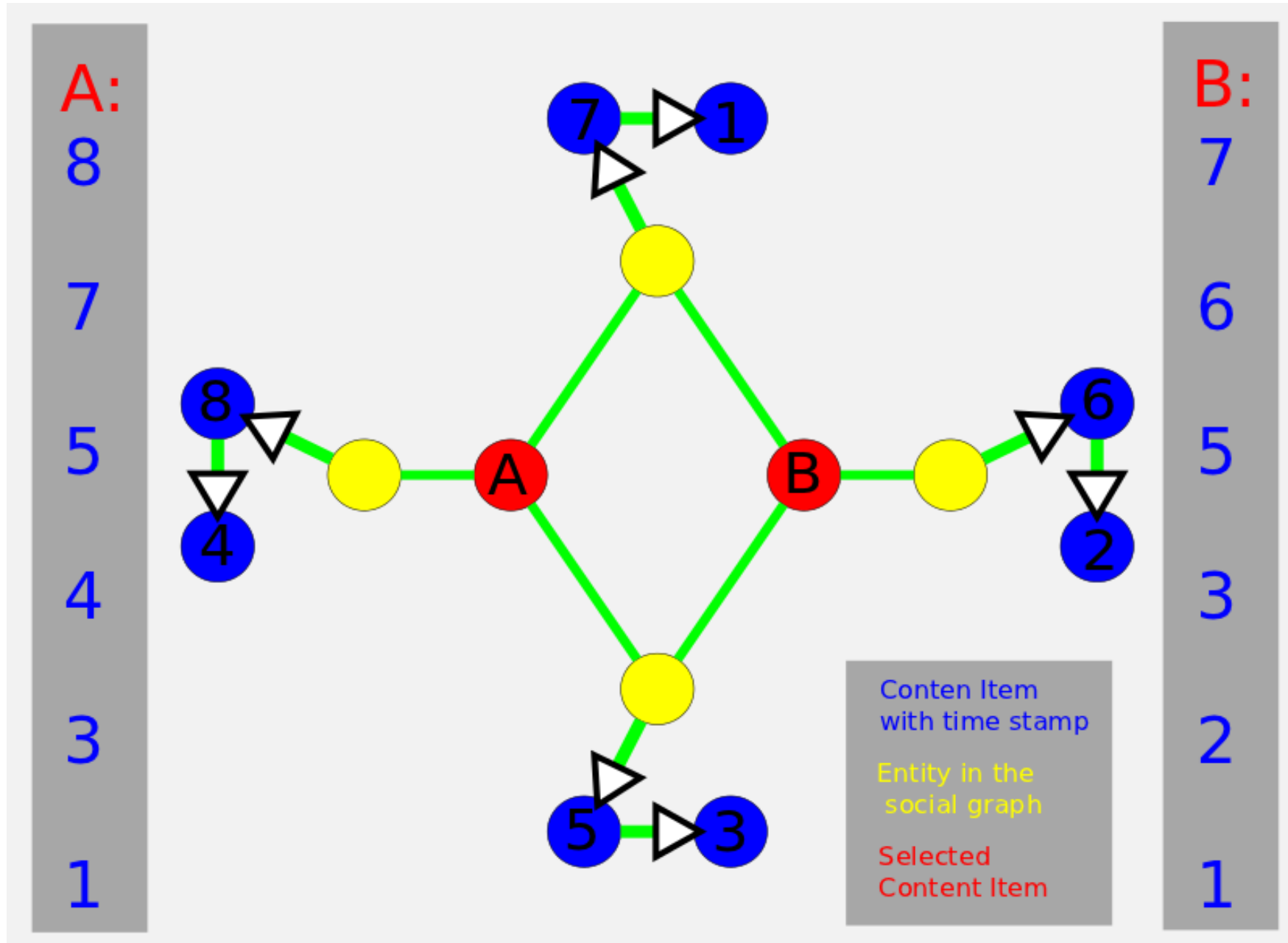
- **contra:**

1. uses a lot of storage (redundancy)
2. not very flexible

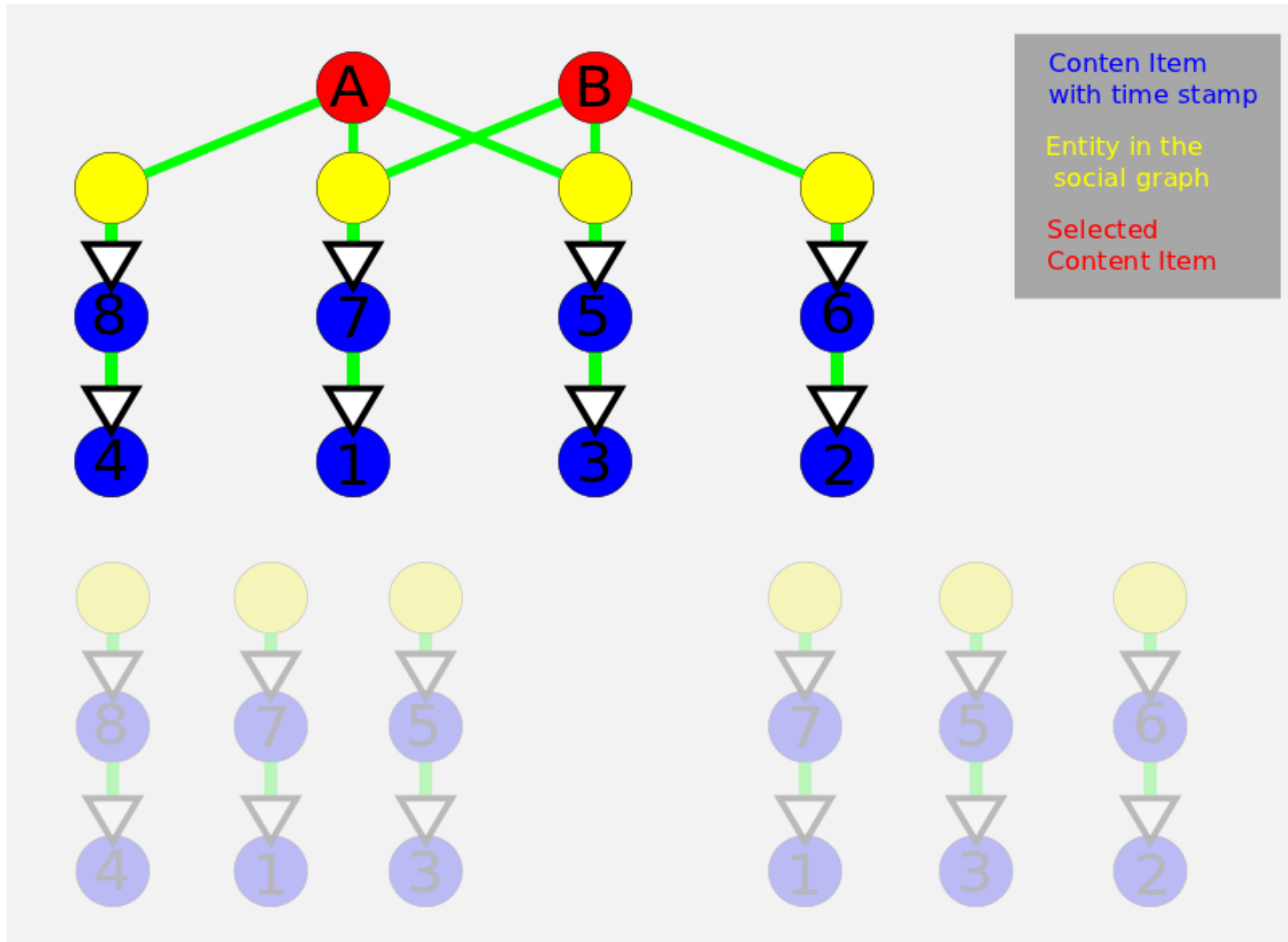
[?] Can't find resource any more ! (still a reasonable approach)

Graph Solution (failed)

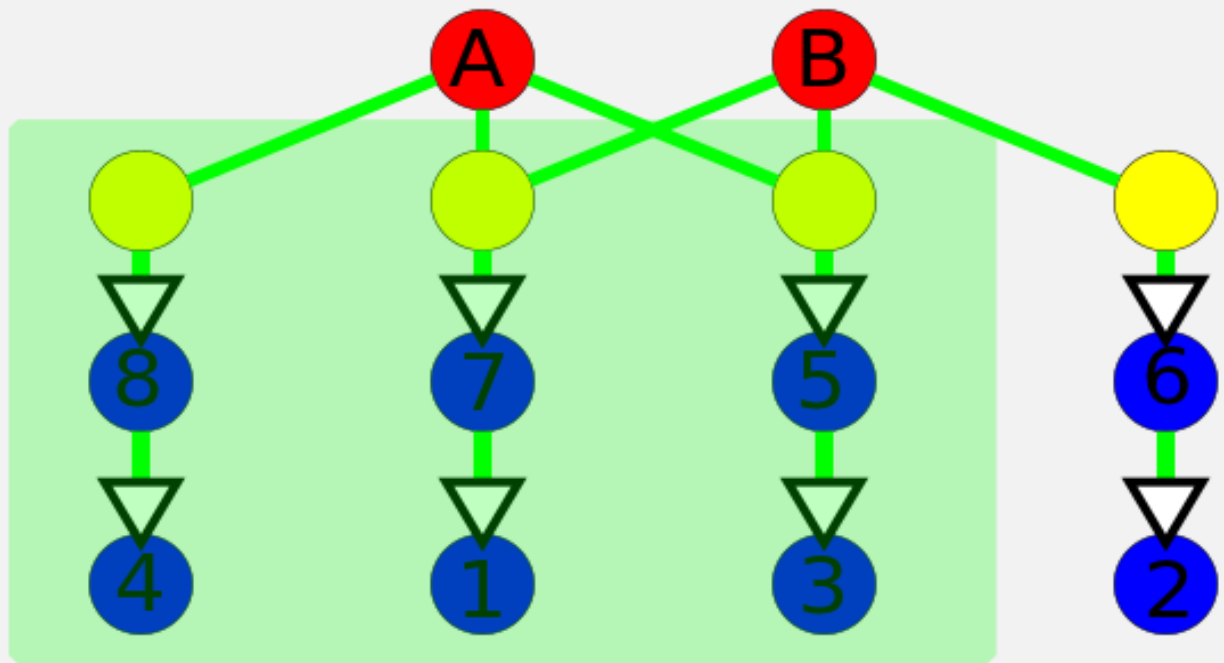
Will linking content items by time help?



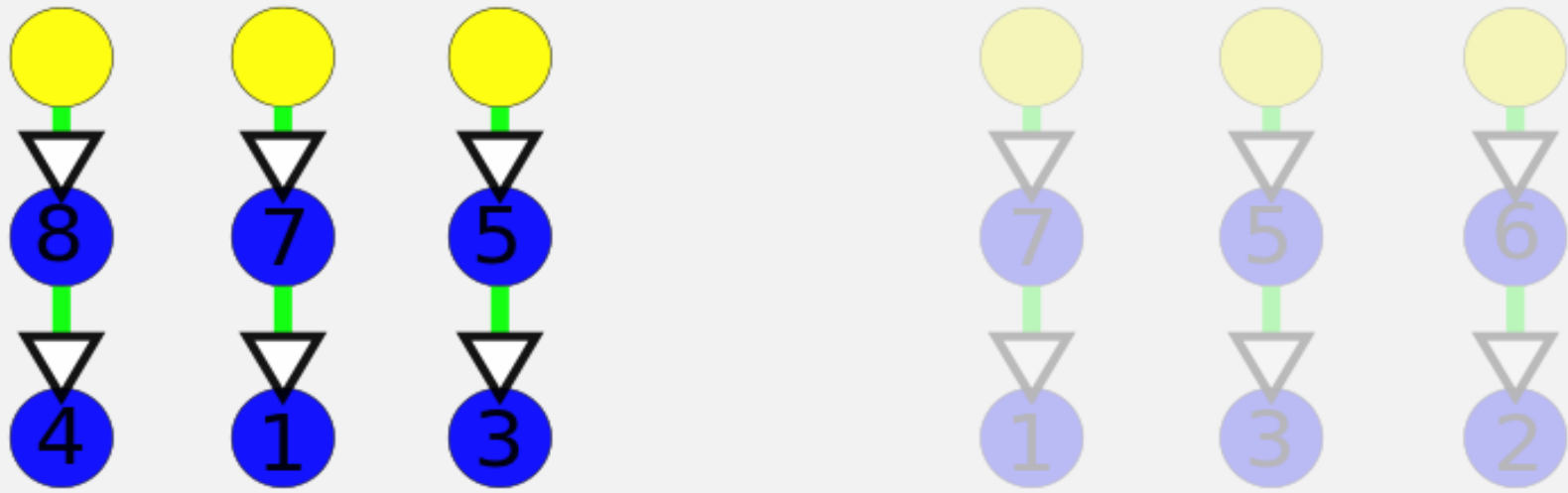
Will linking content items by time help?



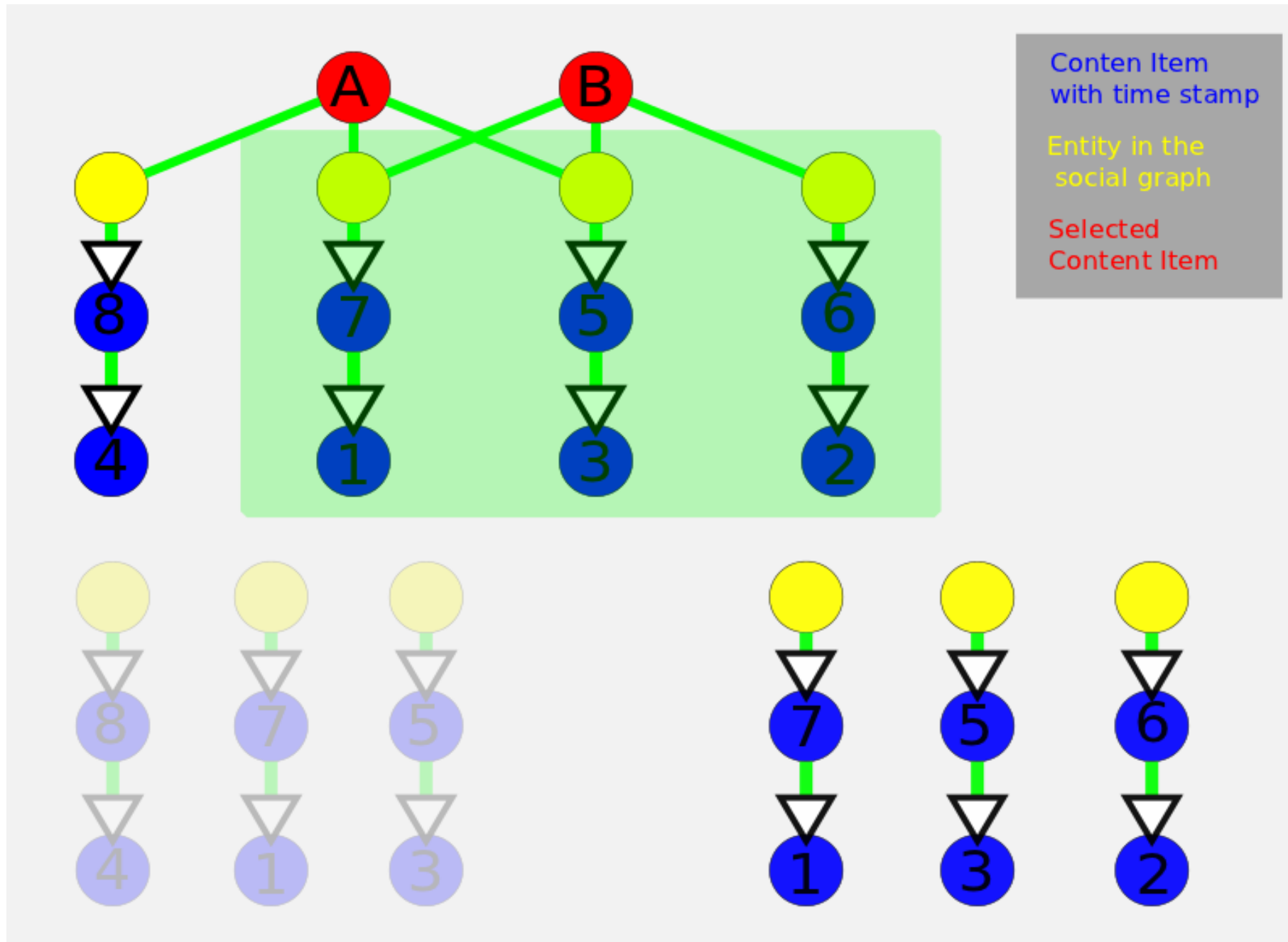
Will linking content items by time help?



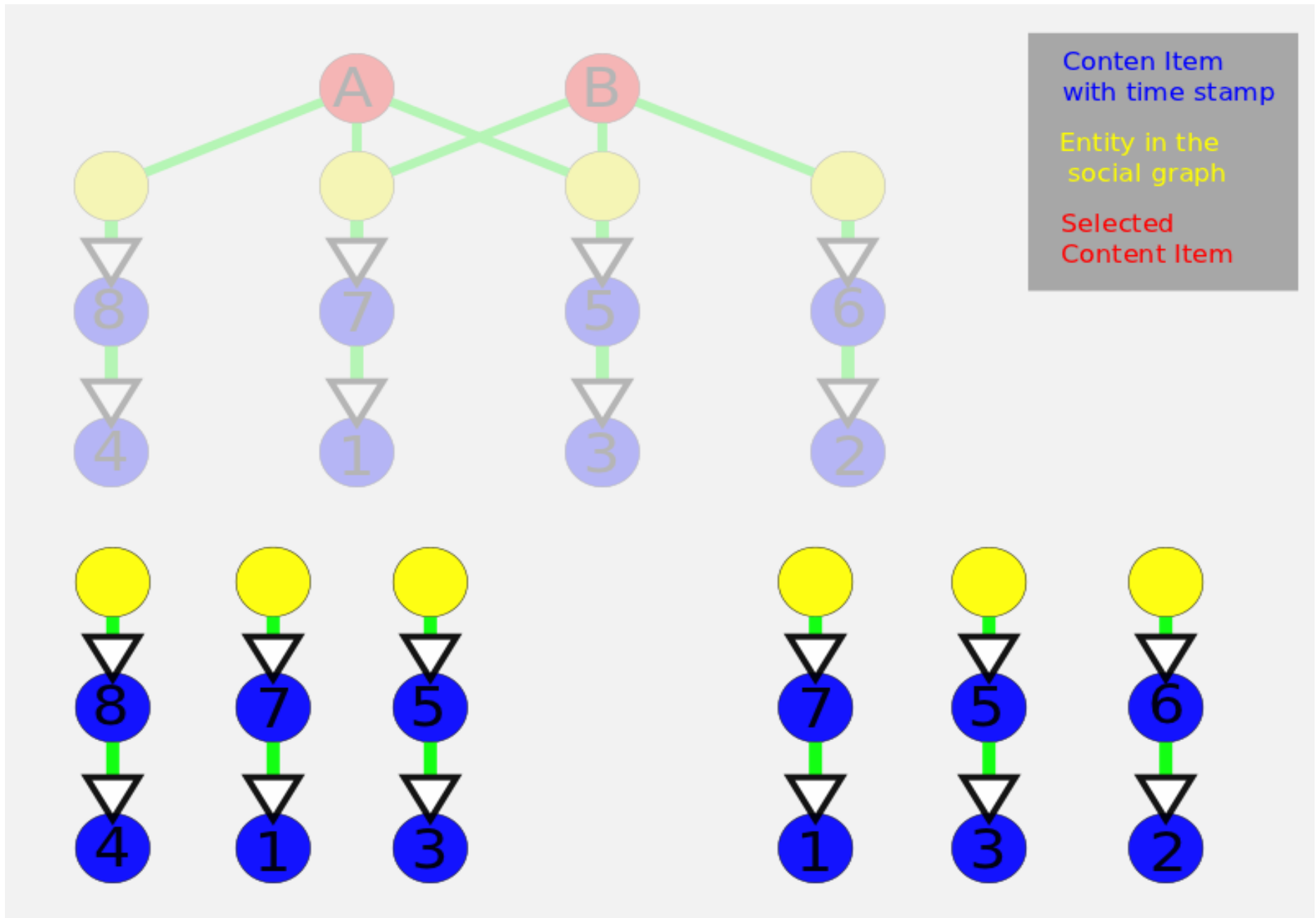
Content Item with time stamp
Entity in the social graph
Selected Content Item

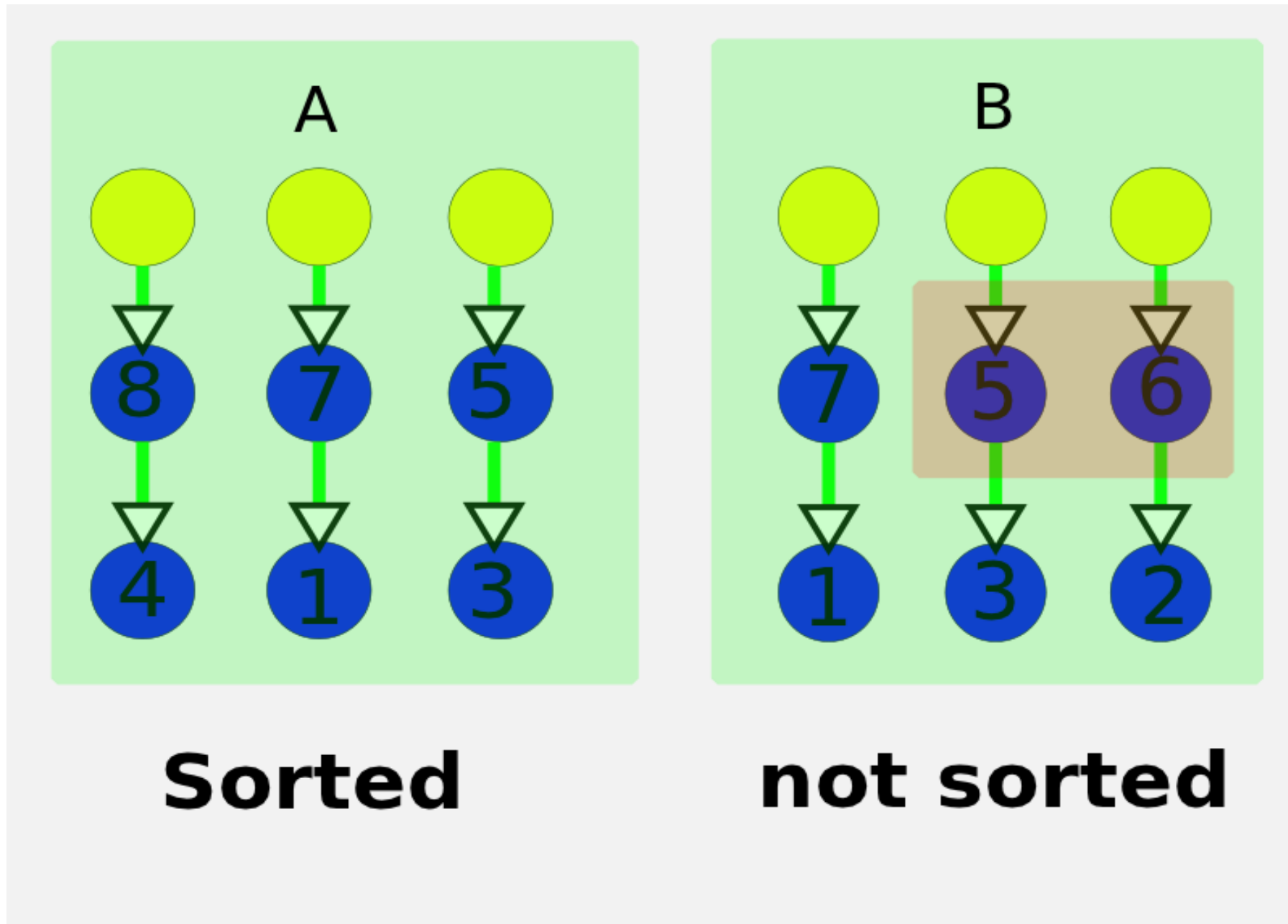


Will linking content items by time help?



Will linking content items by time help?



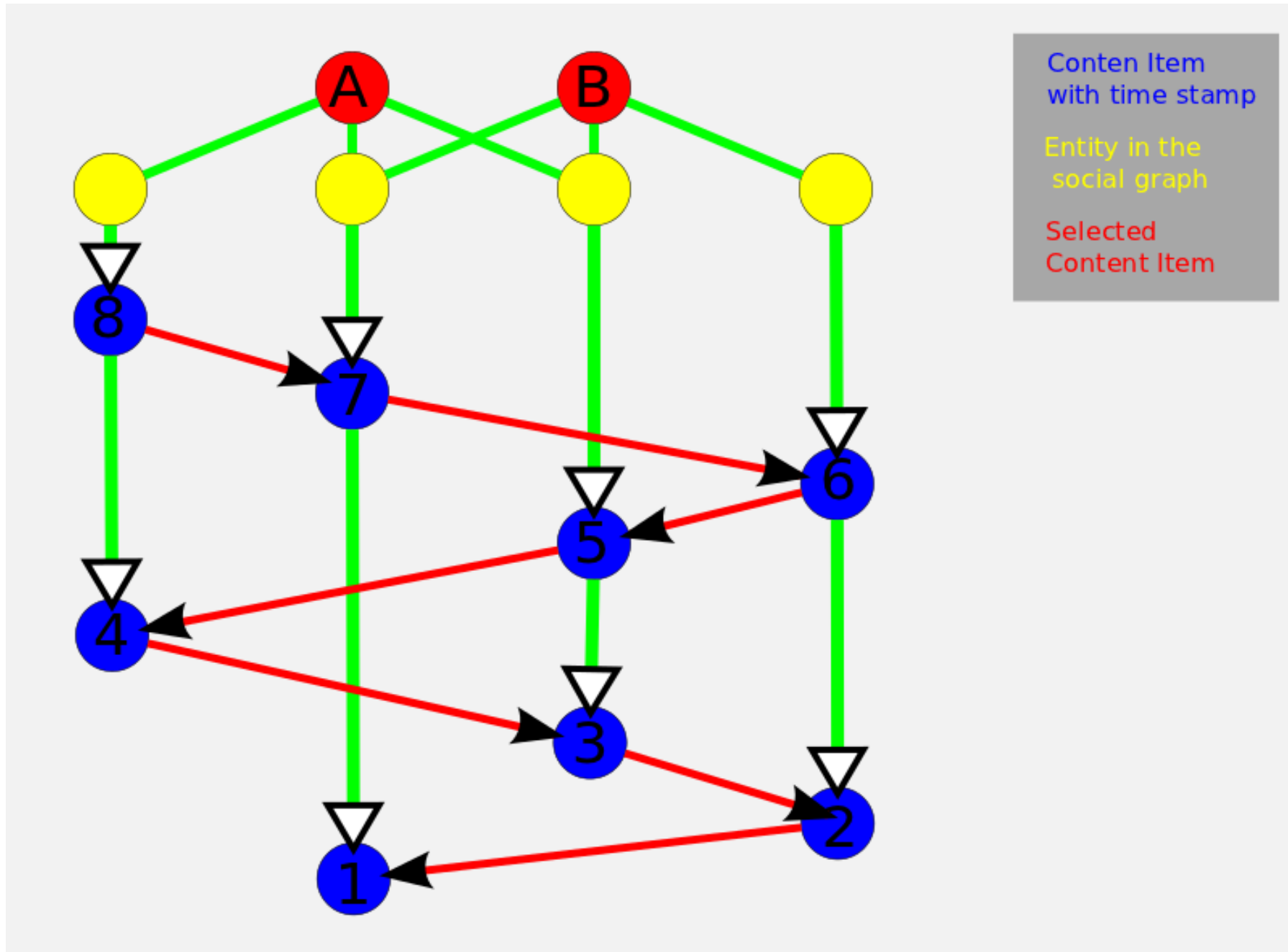


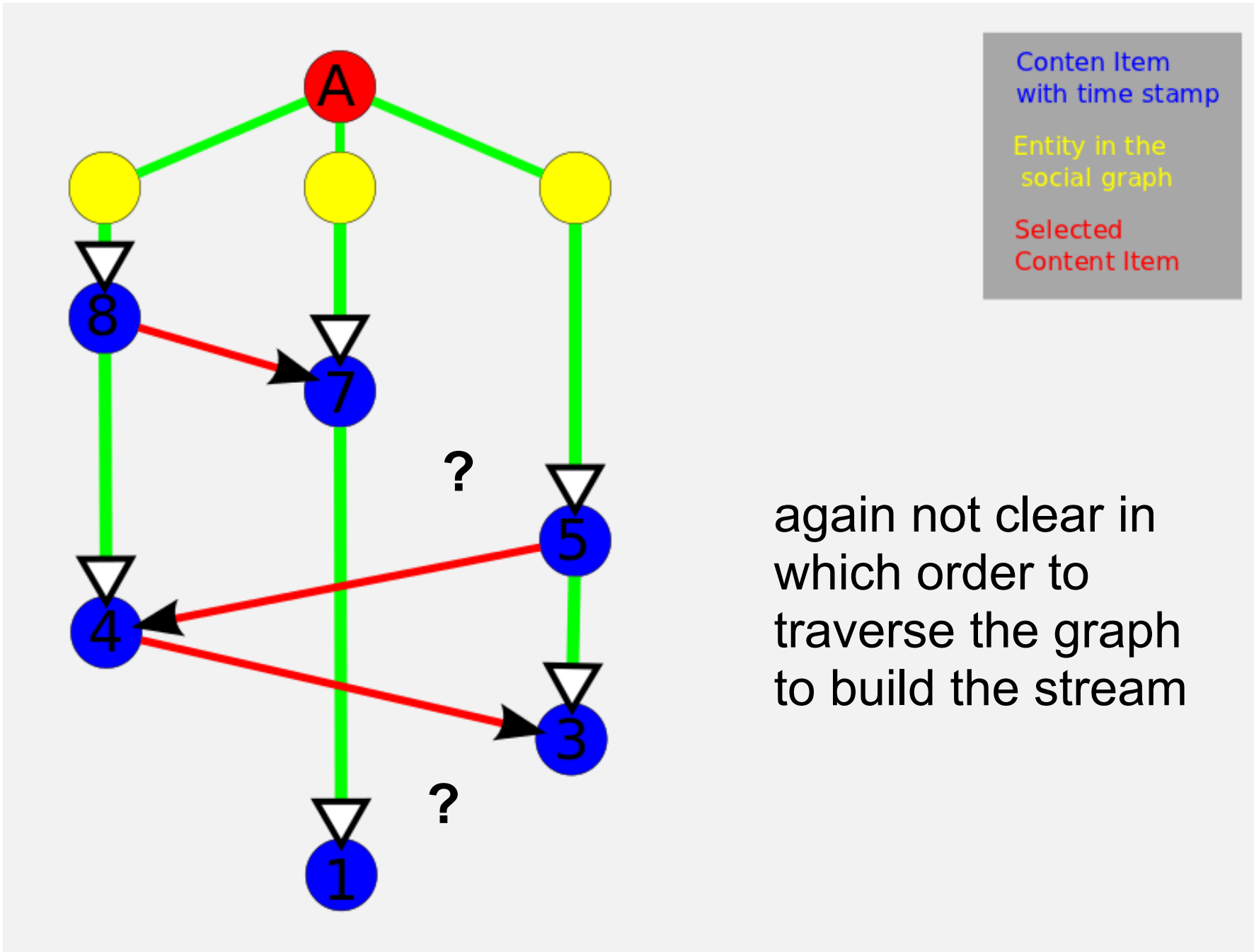
Results & observations

- lists of content items are sorted
- it is not clear with which list to start building the stream
- building the stream still depends on the node degree
 - usually more than 100 entities within the ego network
 - newsstream will only consist of 15 items
 - we don't want to look at all lists in a nodes ego network
 - especially we don't want to sort those lists

Is there a possibility to build a global time index on the entities or content items that will

- descent to the "randomly" choosen nodes in the ego network
- enable us to build the news stream independently of the node degree
- enable us to use some kind of smart graph traversal which always chooses the best (most recent) edges?





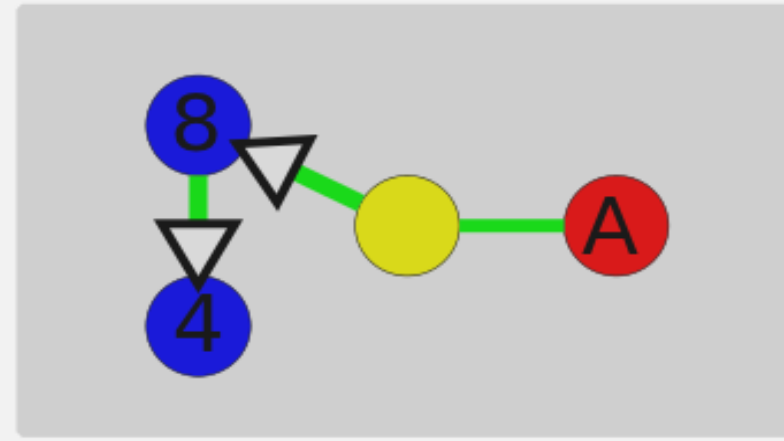
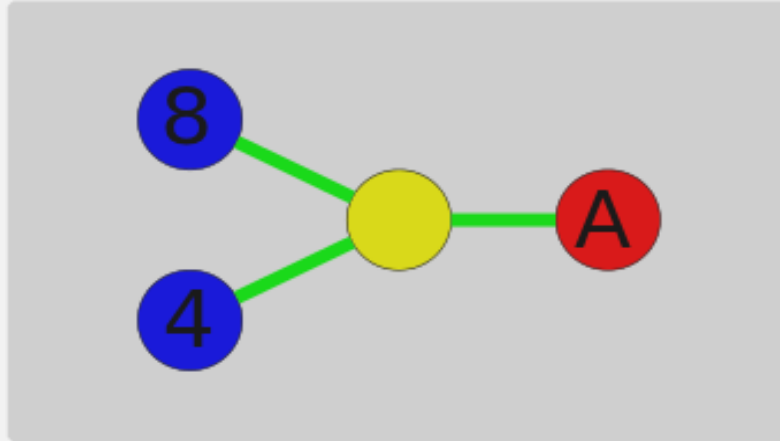
- create and maintain some data structure on the global time index
- the data structure has to descend to the chosen entities from the ego network
- possibilities
 - Hashtables
 - Skip lists (randomization)
 - tree structures

- Our solution is becoming too complicated
- even if such data structure existed there might be new problems such as:
 - access often only in log time (too slow!)
 - maintaining it in an often changing graph
 - it might be tailored to our specific setting and not be general

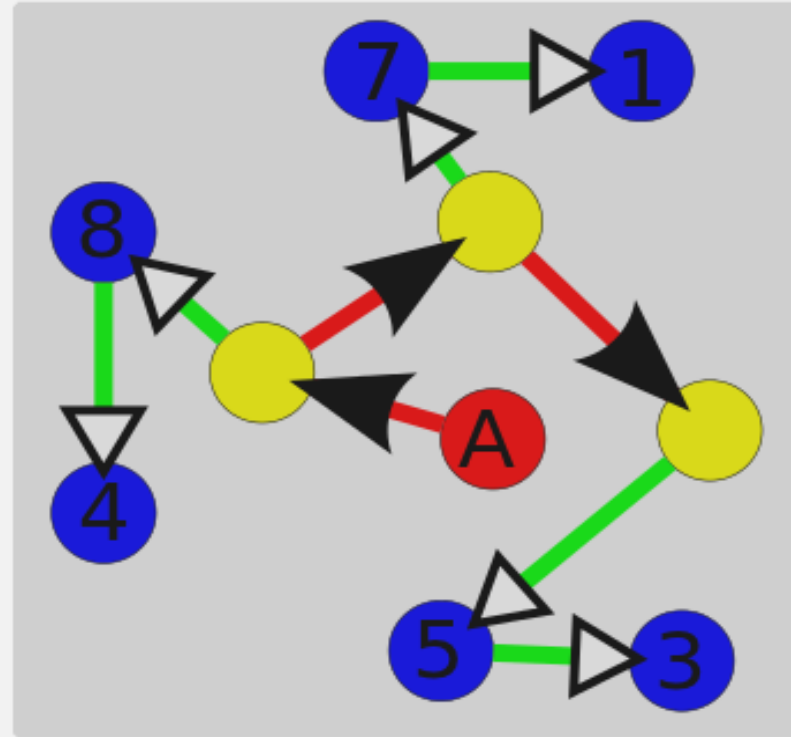
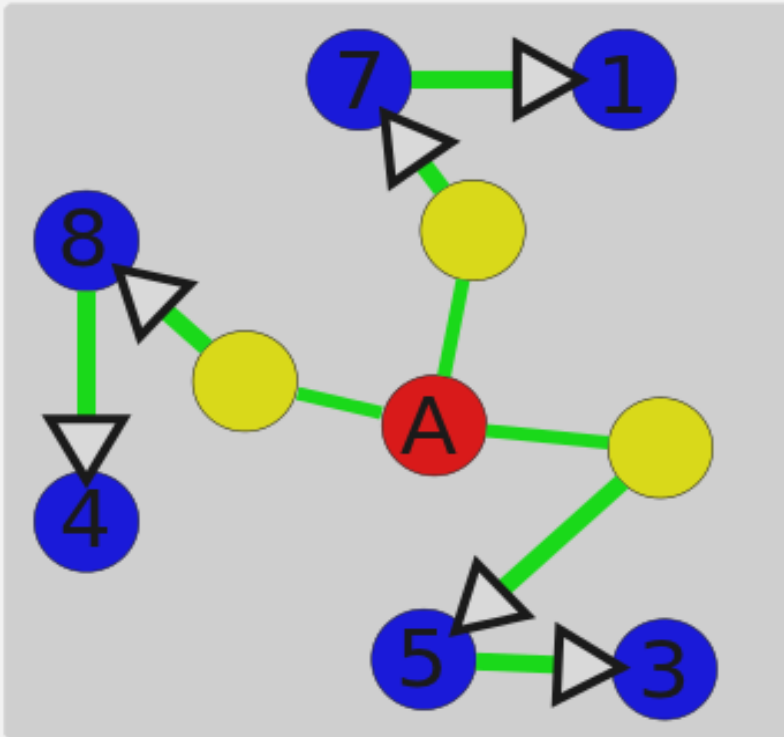
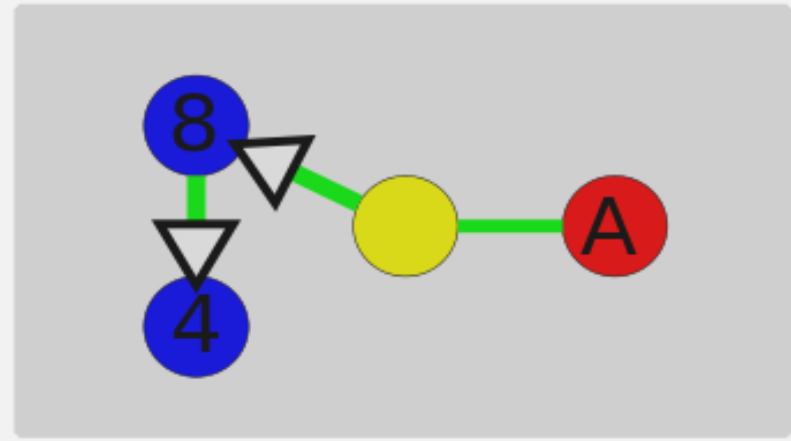
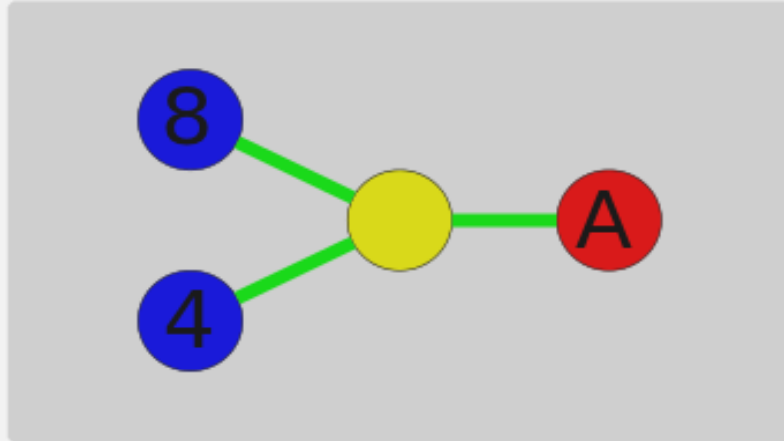
Result:

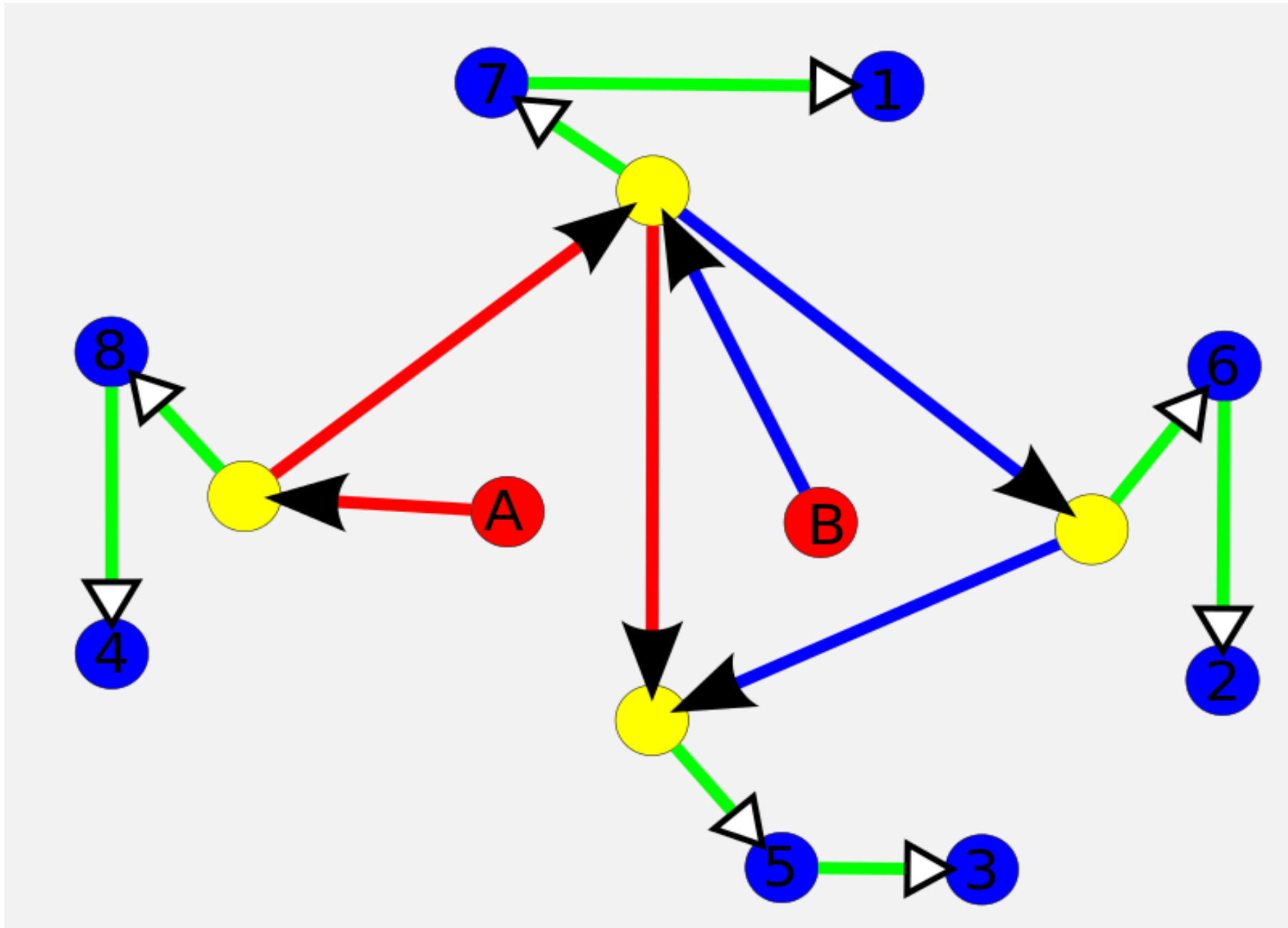
- We are now looking for a different approach.

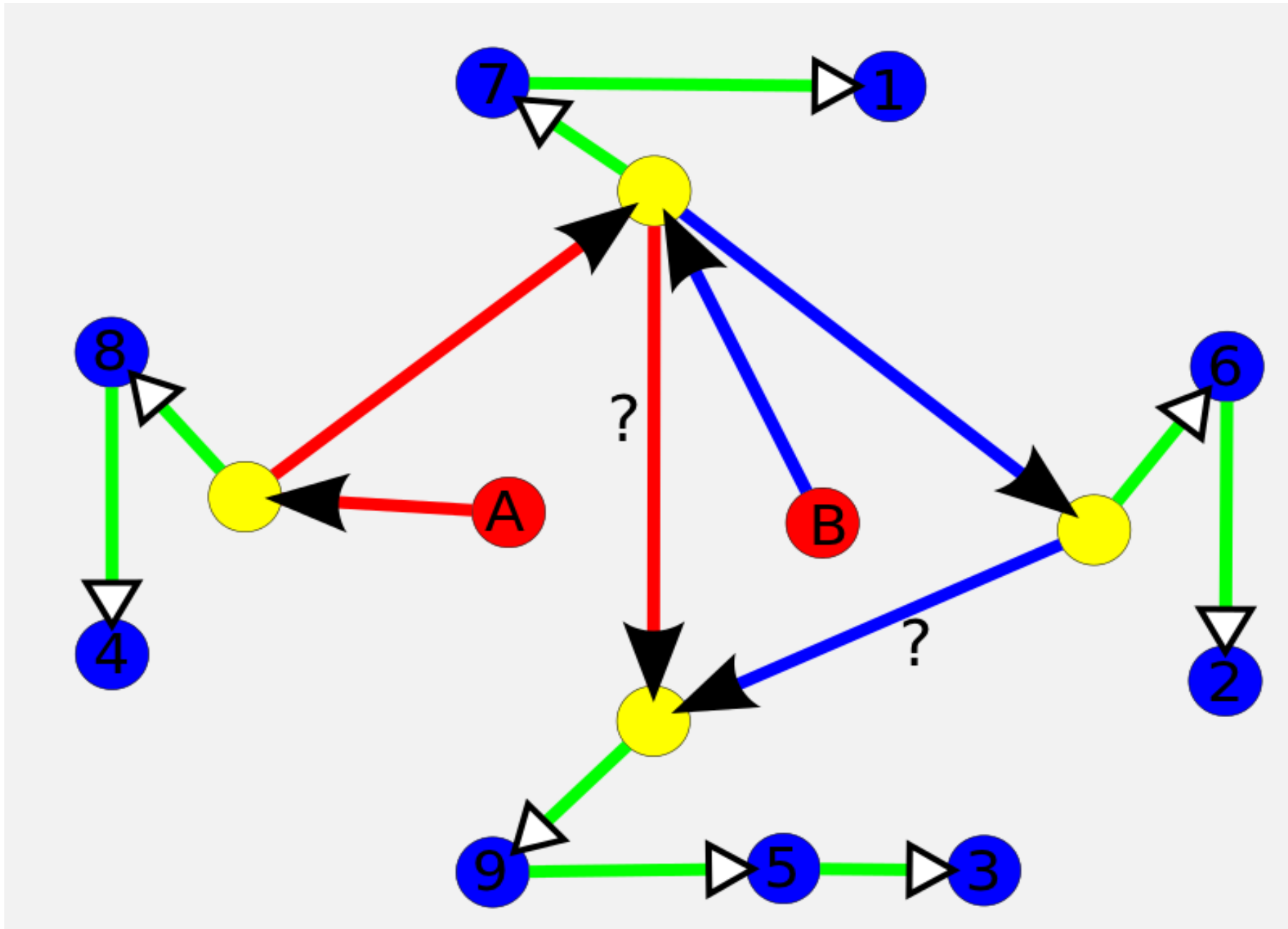
a better graph solution

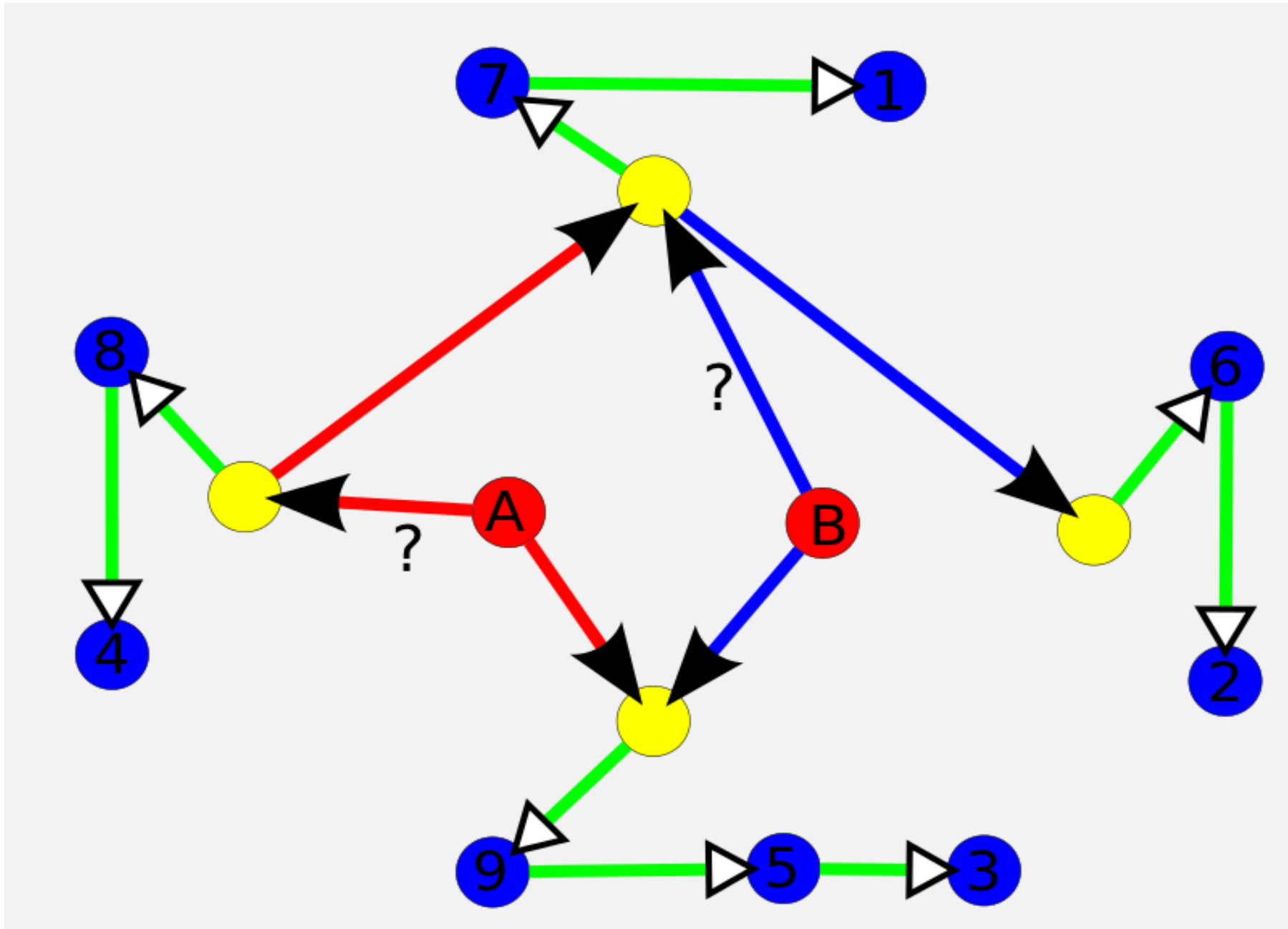


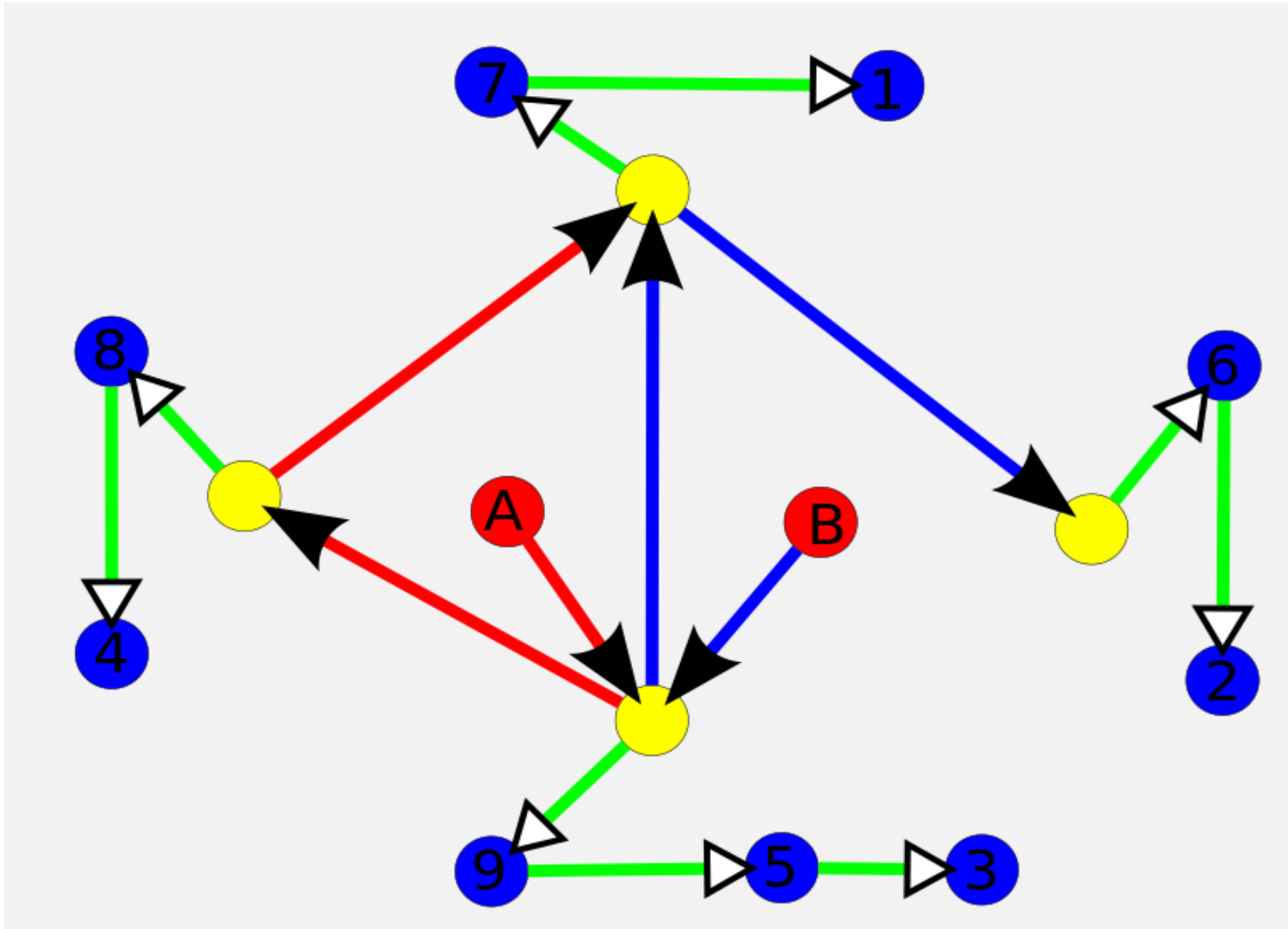
- for content items we went from star topology to an ordered list
- what happens if we do the same with entities inheriting the timestamps from content items?











1. □ Link Content item to the relevant entity X
2. use the old star topology to get X's ego network
 - BFS (depth 1)
1. for every node Y from X's ego network
 - look up edge type ET for Y's ego network
 - go back to X
 - with respect to ET
 - retrieve X's predecessor P and successor S ($P \rightarrow X \rightarrow S$)
 - Link P to S ($P \dashrightarrow S$)
 - Insert X to the beginning of the ego network list
2. Done

This is Linear in X's node degree!

reading - building the stream for any node will:

- take as much time as the length of the stream
- be independent of network size!
- be independent of node degree (in theory!)

writing - create a content item we need:

- link it to the correct entity and update the list of content items = $O(1)$
- update all ego networks in which the content items appears $O(\text{node degree})$
 - same as Facebook with updating all text files!

Storage:

- edges of the ego networks are saved twice for every entity.
- Storage doubles.
 - much more efficient than Facebook

We created a data structure for social news streams that

- is as efficient as a flat file system for
 - reading
 - writing
 - scaling
- additionally has several advantages
 - it is flexible to changes of the underlying graph
 - it uses less storage / redundancy
 - after reading the stream we can still very fast do local searches in order to filter and rank news by relevance

- number of edge types is dynamically
 - Is choosing an edge type for any given node while traversing really possible in $O(1)$?
 - if not at least $O(\log(\text{node degree}))$?
- Is rearranging the ego network lists really $O(1)$ for every list
 - is ego network trees maybe a better approach?
- Build the system with neo4j!
- Filtering by
 - relevance
 - or extend the system for G+ like human filtering

**Thank you
for your
comments!**

Backup Slides

- <http://www.rene-pickhardt.de/time-lines-and-news-streams-neo4j-is-377-times-faster-than-mysql/>
- 14'986 Content items (forum discussions)
- Belonging to 1391 Entities (Bands)
- 854 users contributed at least one content item
- 3142 users have at least one connection to one of these bands
- bipartite graph between users and bands
- now create the social news stream of forum discussion for all users using
 - mysql
 - flat files
 - neo4j

	Flat file full stream	Flat file first entries	MySQL + sorting	neo4j no sorting	neo4j + sorting
read	45 s	0,8 s	152 s	1,7 s	2,1 s
store	1,7 GB	1,7 GB	5,3 MB	24 MB	24 MB
write	hard ?	hard ?	fast ?	fast ?	? ?

Reading happens much more frequently than writing to the graph in general or the user's ego network in particular

	Flat files	normalized DB
reading	fast	slow (joins!)
writing	slow	rather fast
redundancy	high	no
consistency	hard	easy
Flexibility (graph updates)	hard	easy
Storage space	high	low
scalability	seems ok	hard (joins!)

We are seeking for a data structure to model social news streams that enables us the following:

- fast reading
 - sublinear in node degree!
 - flatfiles are $O(1)$
- fast writing
- low redundancy
- flexibility in reading
 - news streams should update if graph changes
- scalability
 - Remember our problem is local.
 - Data structure should be independent of the graph's size

- storing data in a graph data base seems a good idea
- creating news streams for an entity is equivalent to traversing the graph with BFS
- with a graph the following problems are trivial
 - scaling (as long as the graph db scales)
 - low redundancy
 - fast traversals & local searches with many "joins"
 - fast writing
 - flexibility
- but some problem are still there
 - creating (reading) the news stream for a particular entity
 - especially if the stream has to be sorted by time!

- MySQL is much slower than flat files and graph data base
- MySQL if normalized will not scale!
- flat files need huge amount of disc space since they are redundant
- processing the same amount of data with flat files and neo4j we see neo4j as a winner
- writing flat files will be slow since files have to be completely rewritten to hard disk
- files have some huge advantages
 - don't have to be completely processed (first rows suit)
 - easy to scale and distribute over several machines
 - order by time is naturally inside