Blank and Set

MLS IETF 103

Blank

- 1 rule:

  Everyone needs to agree on which nodes to blank

- **Upside**: it allows to evict someone from the group entirely

- **Downside**: puncturing the tree decreases efficiency

Set

- More rules:

  The node secret of that node needs to be KEMed to the resolution of the children

  The node must be blanked if the setter is evicted

- **Upside**: it allows to heal the tree without relying on others to do updates

- **Downside**: it introduces double joins (again)

Double Joins

- **Definition**:
  If a member knows the secret of a node that is not in their direct path, we say that member has double-joined that node

- **Downside**: it is more costly to evict a member, since all double-joined nodes now also have to blanked

Bookkeeping

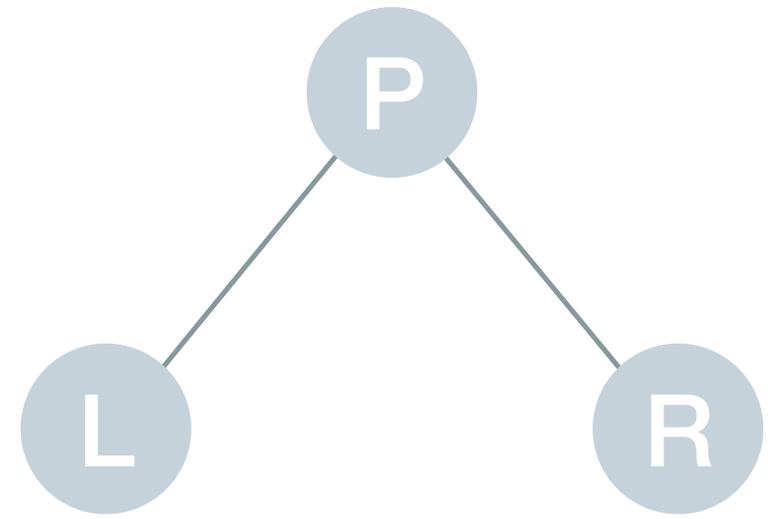
- A way to keep track of what member has double-joined what node
- The “**book**” is a data structure that contains entries for every double-joined node

  Each entry contains a list of “illicit owners”
- The book needs to be passed to new members/clients
- The book should be part of the group state

Bookkeeping

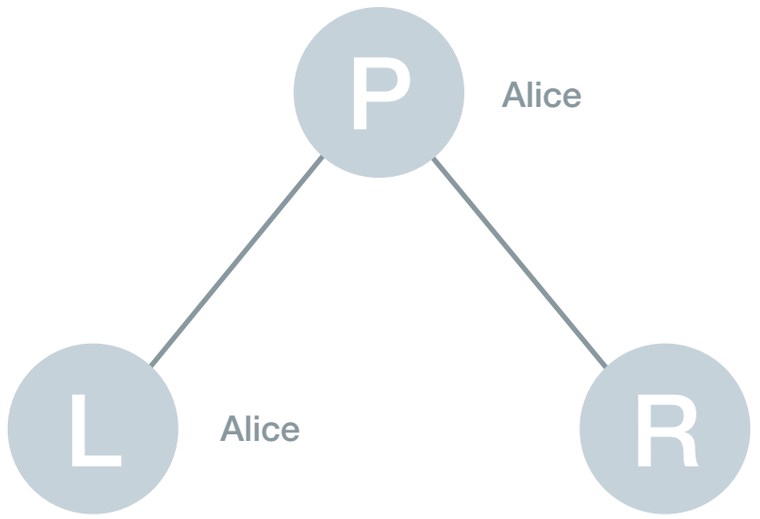
```
struct BookEntry {
    uint32 node;
    uint32 owners<0..2^32-1>
}

struct Book {
    BookEntry entries<0..2^32-1>
}
```

Setting a node



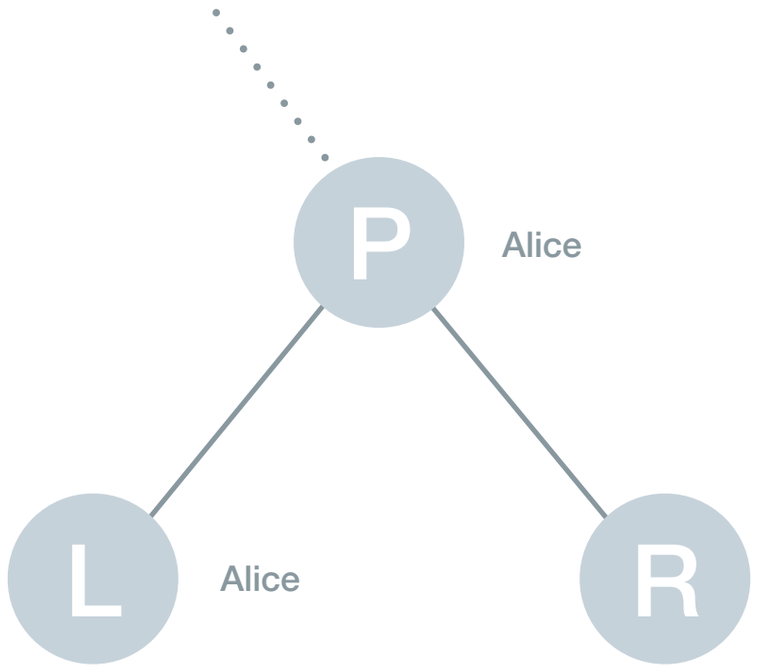
- Whenever a parent node secret is changed, its new secret is KEMed to its children (or their resolution)

Double join propagation



- **Set**: the children in the resolution of the set node learn about the secret of their parent. If any of the nodes in the resolution list were double-joined, the set node is now also double-joined with the same list of “illicit owners”

Double join propagation



- **Update**: the nodes in the copath learn the new secret of their parent. Because that secret is then hashed up along the direct path, a single double-join gets multiplied (worst factor is log N)

When does double join make sense?

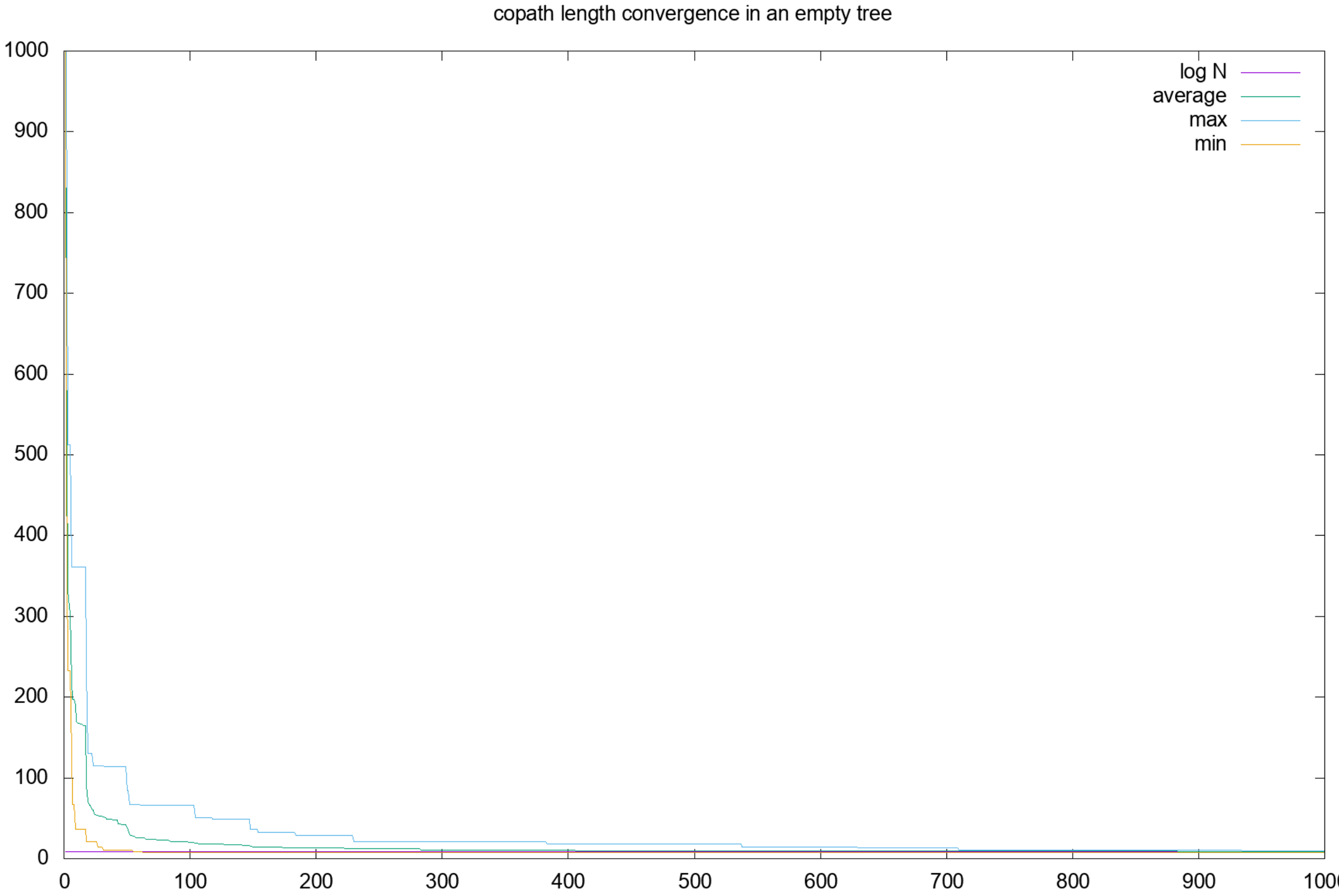
- **Invariant**:
  The more we heal a punctured tree with double-joins, the bigger the book grows and the faster group operations become.

  ```
  Book size + operation cost = const.
  ```

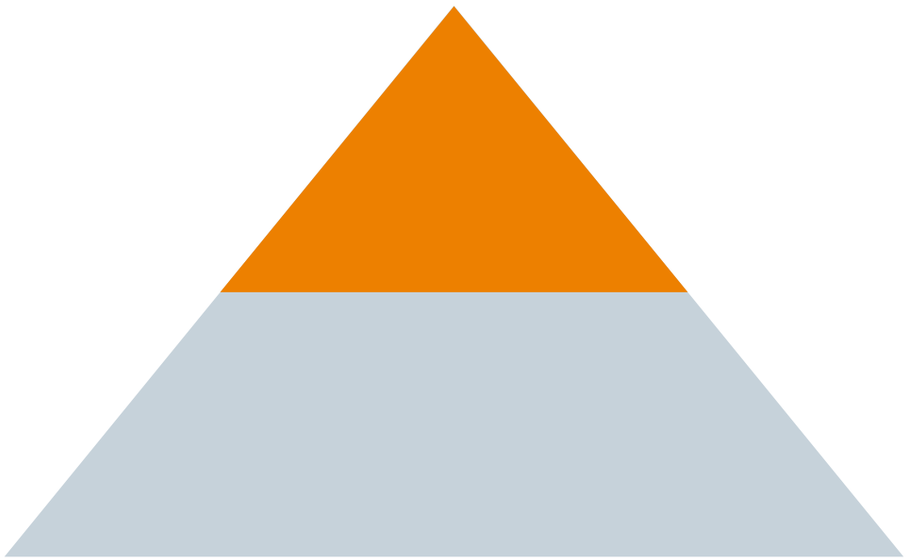
- A bigger book means bigger payload sizes. It also means there is more blanking to do when a member with double-joins is evicted. This makes the tree less efficient again.

Edge-case: group creation

- Large group creation: nodes are blank initially
- Cost to do updates is O(N)
- Converges quickly, still bad for early members


copath length convergence in an empty tree
log N
average
max
min
1000
900
800
700
600
500
400
300
200
100
0
0
100
200
300
400
500
600
700
800
900
1000

Introducing warm-up



- Pre-populate top of the tree
- Most effective at the top, halving cost with new level

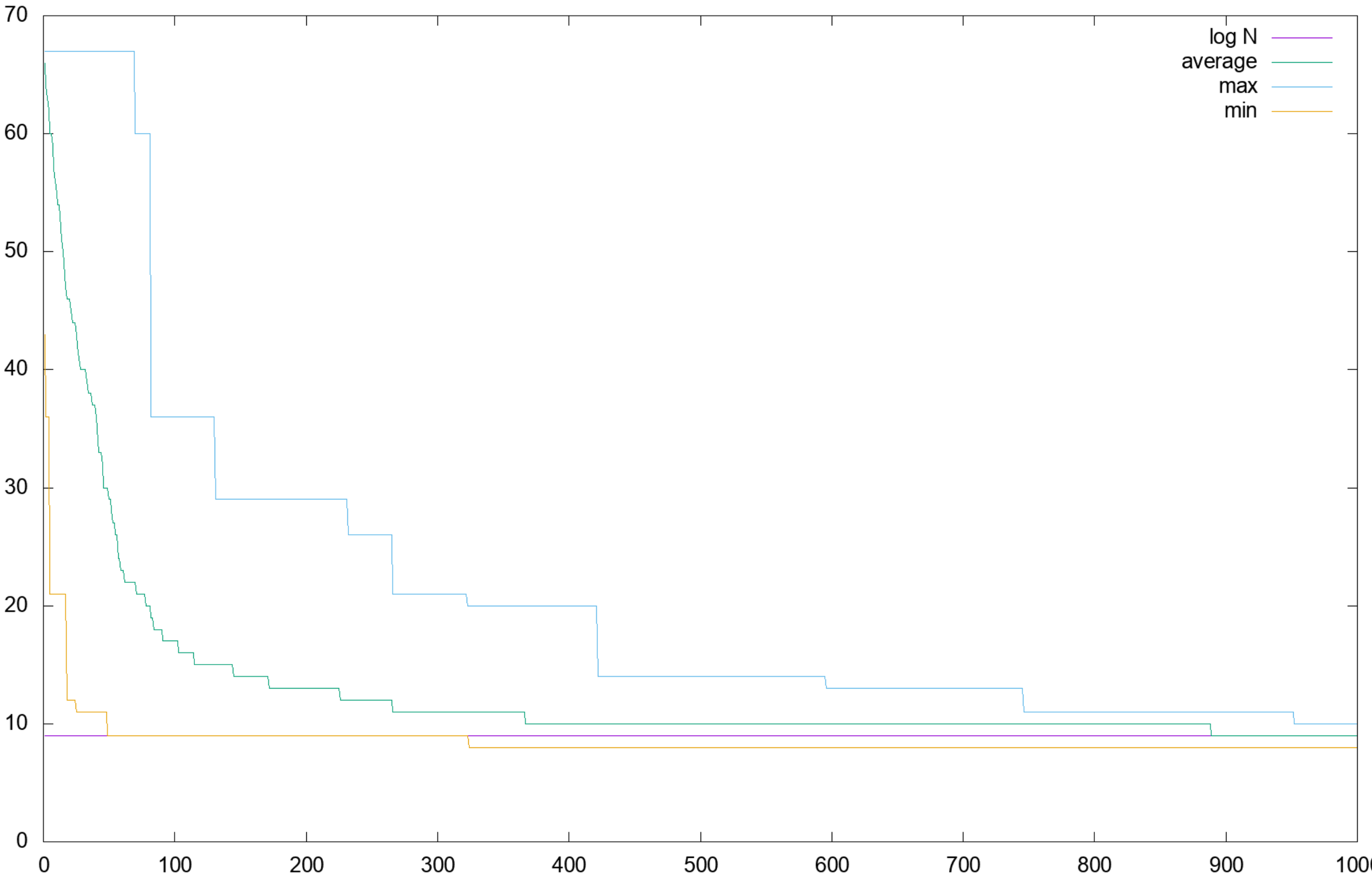
Introducing warm-up

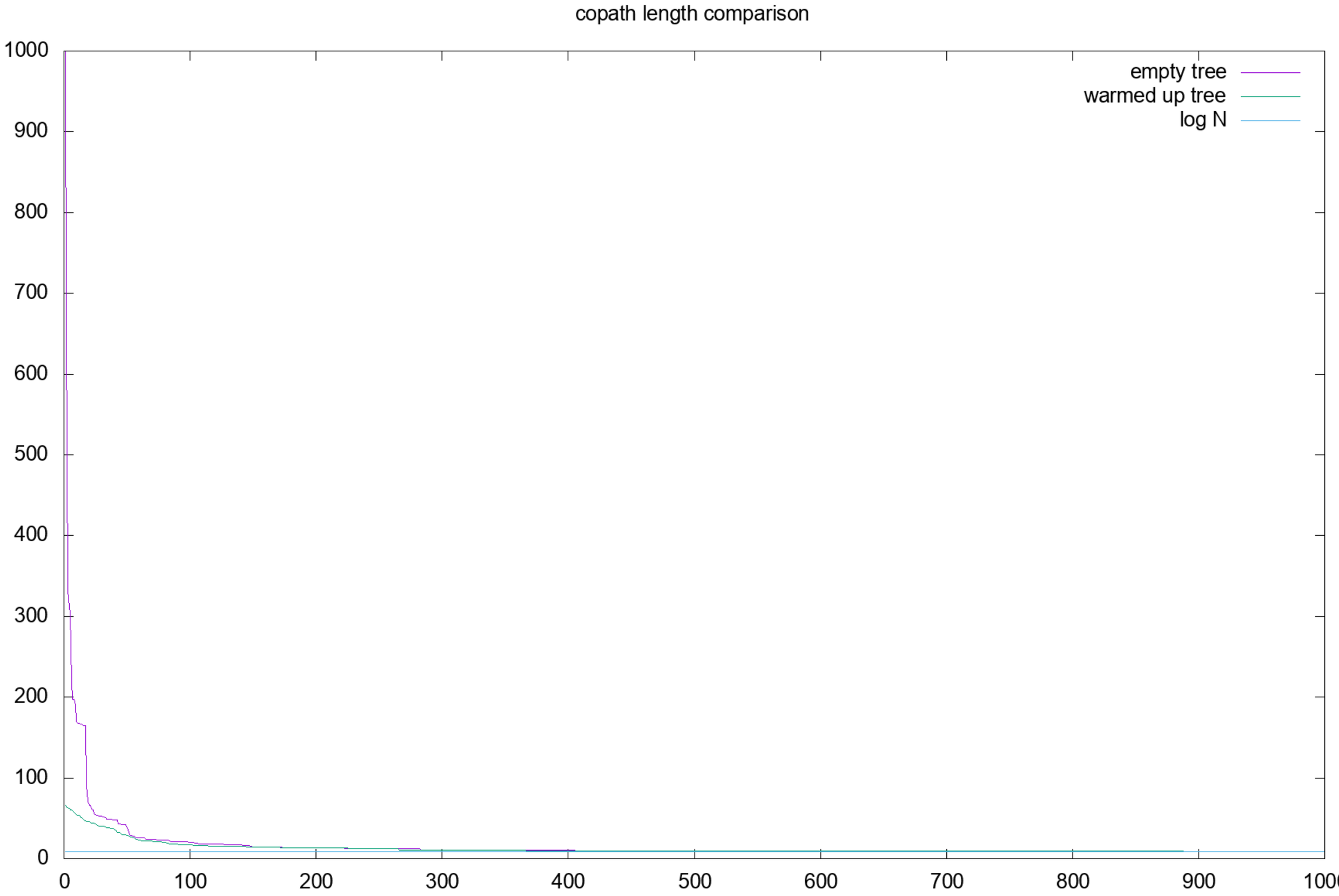
- Example in numbers:

  Creator creates a group of 1000 members.

  Creator populates the top 3% of the tree.

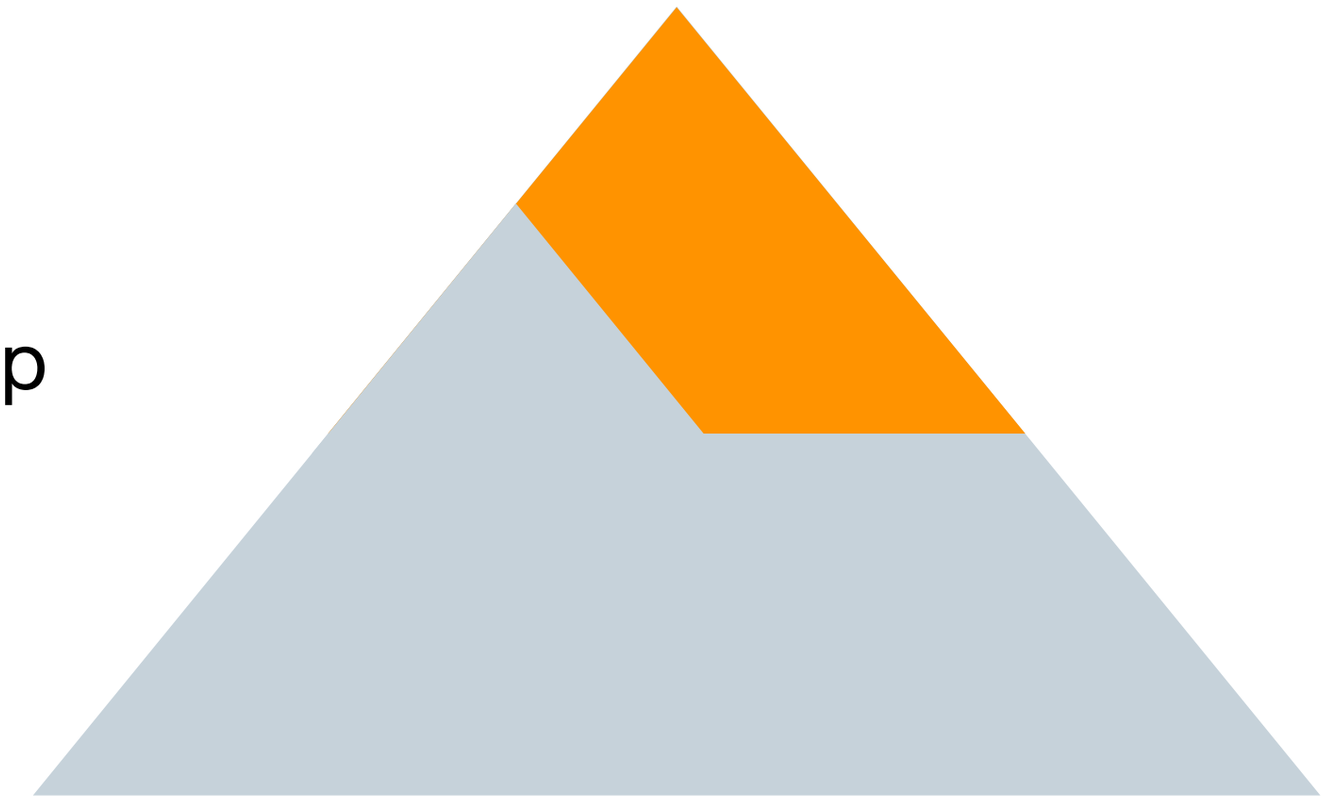
copath length convergence in a warmed up tree




copath length comparison
empty tree
warmed up tree
log N
1000
900
800
700
600
500
400
300
200
100
0
0
100
200
300
400
500
600
700
800
900
1000

Cleaning

- k levels warmed-up
- ~ 2^{k+2} updates



Summary

- Setting nodes is generally expensive
- Warming-up the top of the tree increases efficiency