A Distributed Algorithm for Minimum-Weight Spanning Trees

by

R. G. Gallager, P.A. Humblet, and P. M. Spira

ACM, Transactions on Programming Language and systems,1983

presented by
Hanan Shpungin

Outline

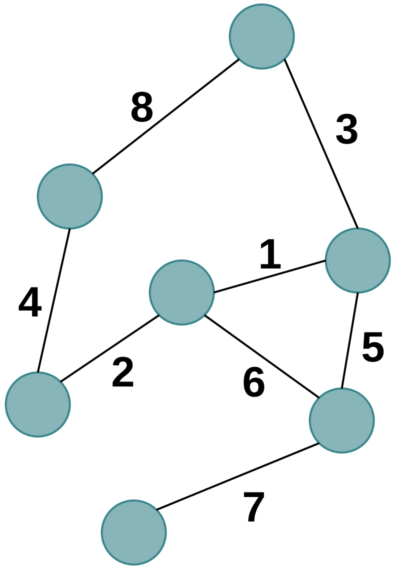
- Introduction
- The idea of Distributed MST
- The algorithm

Outline

- **Introduction**
- The idea of Distributed MST
- The algorithm

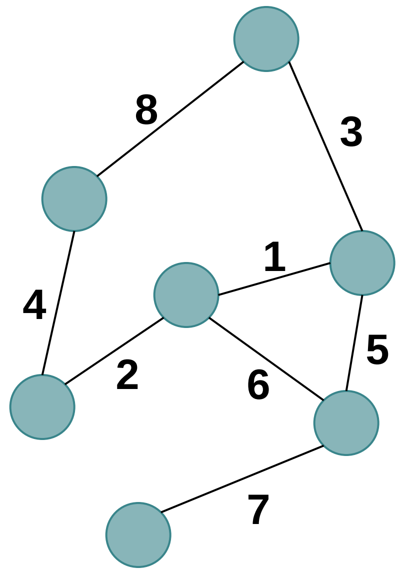
Introduction

- Problem
  - A graph $G = (V, E)$
  - Every edge has a weight $\forall e \in E, w(e) \in \mathbb{R}$



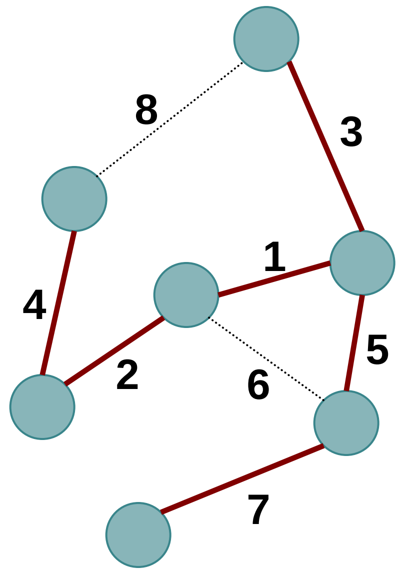
Introduction

- Solution
  - A spanning tree $T = (V, E')$
  - So that the sum $\sum_{e \in E'} w(e)$ is minimized



Introduction

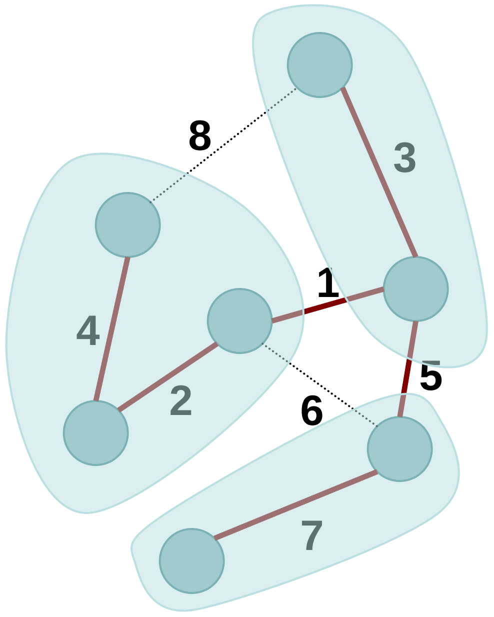
- Solution
  - A spanning tree $T = (V, E')$
  - So that the sum $\sum_{e \in E'} w(e)$ is minimized



Introduction

- Definition – MST *fragment*
  - A connected sub-tree of MST

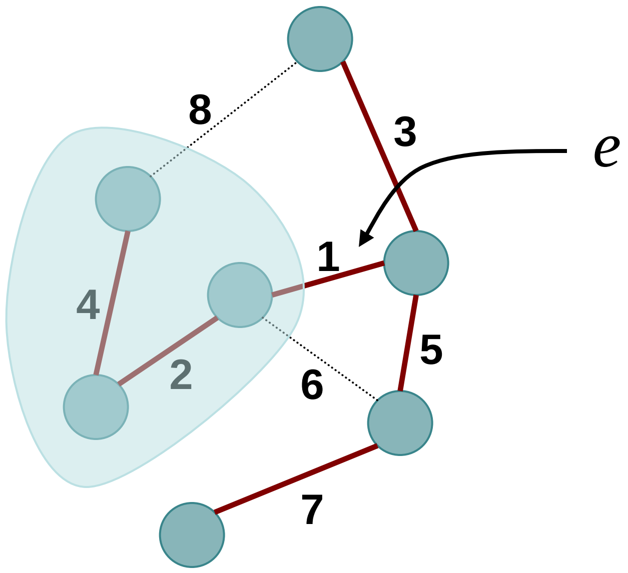
    Example of possible fragments:



Introduction

- MST Property 1

  Given a fragment of an MST, let e be a minimum-weight outgoing edge of the fragment. Then joining e and its adjacent non-fragment node to the fragment yields another fragment of an MST.

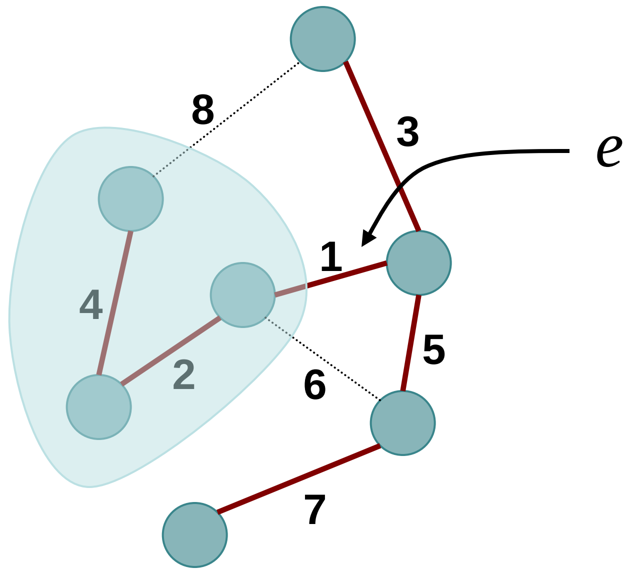


Introduction

- MST Property 1

  Given a fragment of an MST, let e be a minimum-weight outgoing edge of the fragment. Then joining e and its adjacent non-fragment node to the fragment yields another fragment of an MST.

- *Proof:*



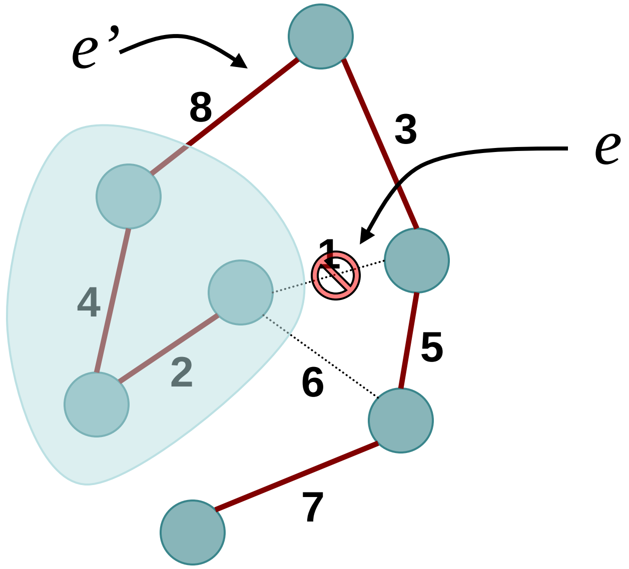
Introduction

- MST Property 1

  Given a fragment of an MST, let e be a minimum-weight outgoing edge of the fragment. Then joining e and its adjacent non-fragment node to the fragment yields another fragment of an MST.

- *Proof:*

  Suppose e is not in MST, but some e' instead.



Introduction

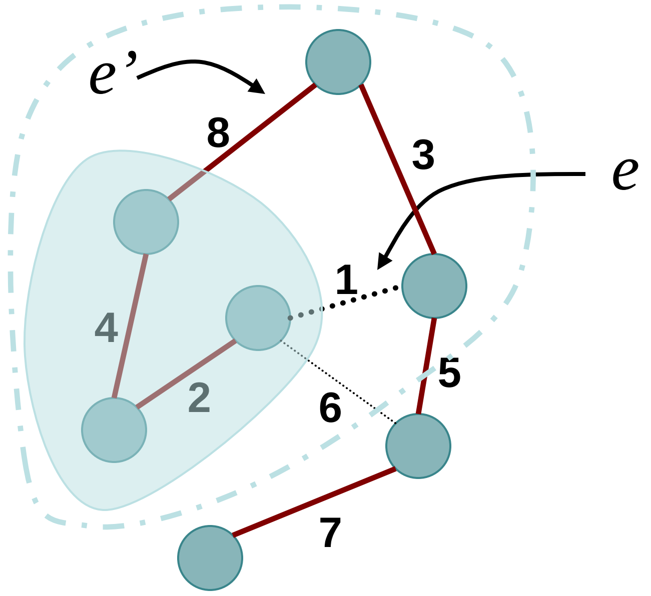
- MST Property 1

  Given a fragment of an MST, let e be a minimum-weight outgoing edge of the fragment. Then joining e and its adjacent non-fragment node to the fragment yields another fragment of an MST.

- *Proof:*

  Suppose e is not in MST, but some e' instead.

  MST with e forms a cycle.



Introduction

- MST Property 1

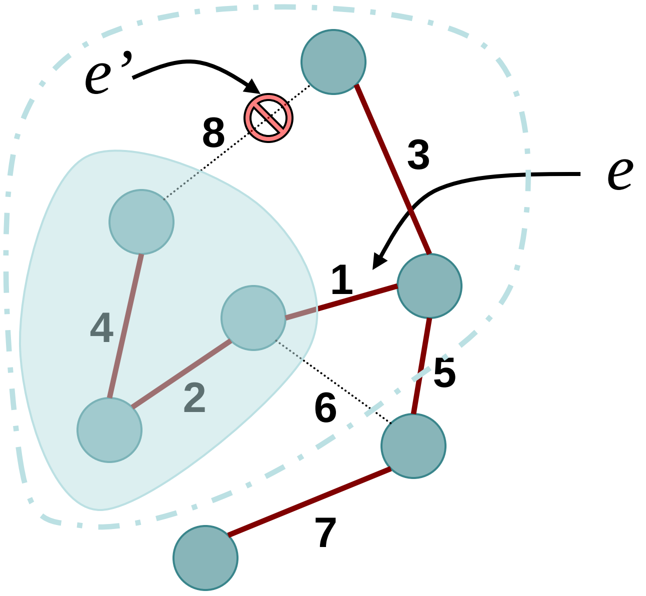
  Given a fragment of an MST, let e be a minimum-weight outgoing edge of the fragment. Then joining e and its adjacent non-fragment node to the fragment yields another fragment of an MST.

- *Proof:*

  Suppose e is not in MST, but some e' instead.

  MST with e forms a cycle.

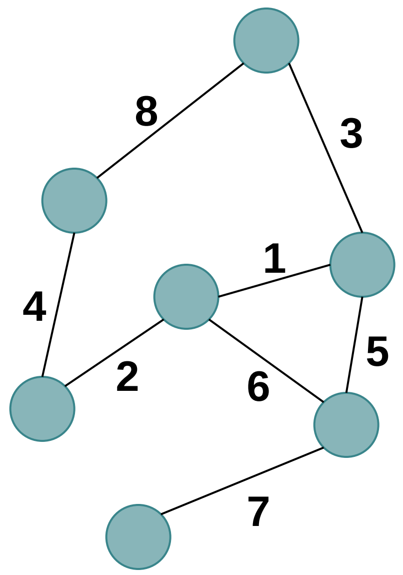
  We obtain a cheaper MST with e instead of e'.



Introduction

- MST Property 2

  If all the edges of a connected graph have different weights, then the MST is unique.

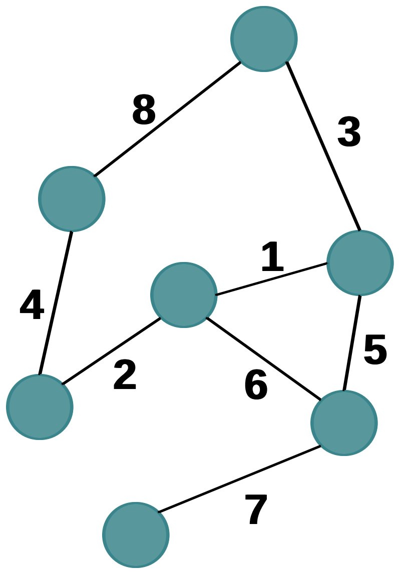


Introduction

- MST Property 2

  If all the edges of a connected graph have different weights, then the MST is unique.

- *Proof:*



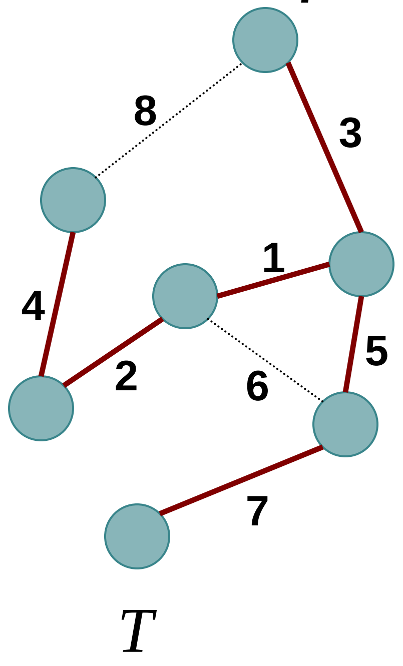
Introduction

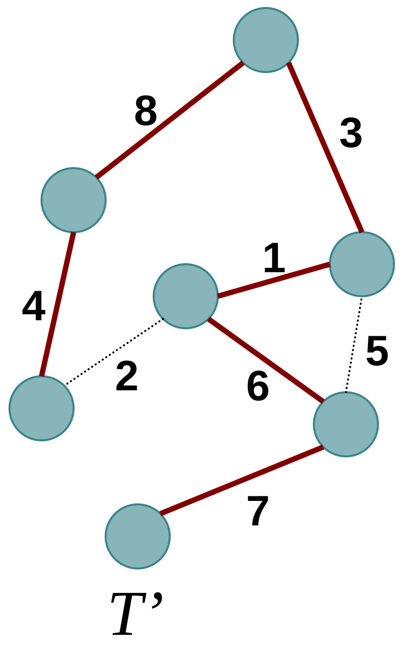
- MST Property 2

  If all the edges of a connected graph have different weights, then the MST is unique.

- *Proof:*

  Suppose existence of two MSTs.



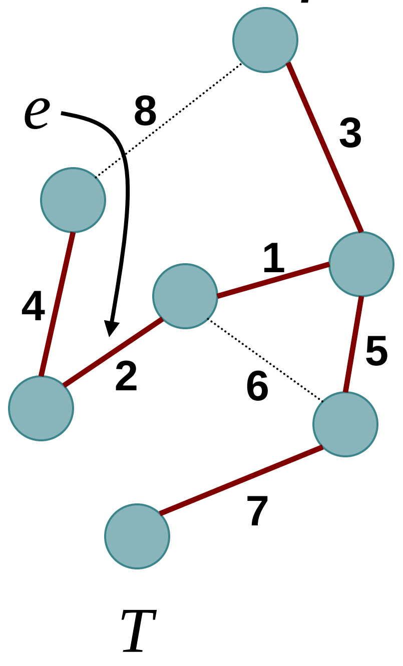


Introduction

- MST Property 2

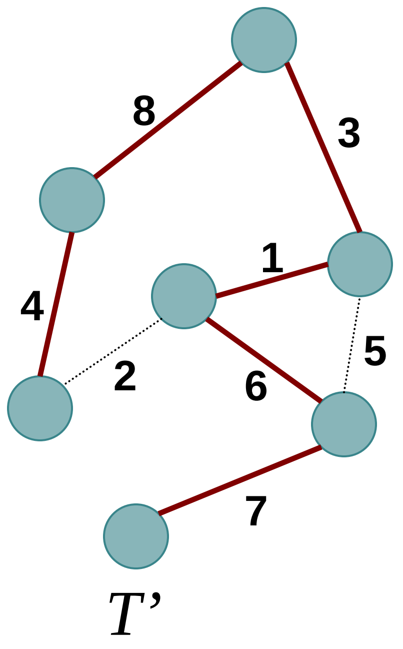
  If all the edges of a connected graph have different weights, then the MST is unique.

- *Proof:*



Suppose existence of two MSTs.

Let e be the minimal-weight edge not in both MSTs (wlog e in T).

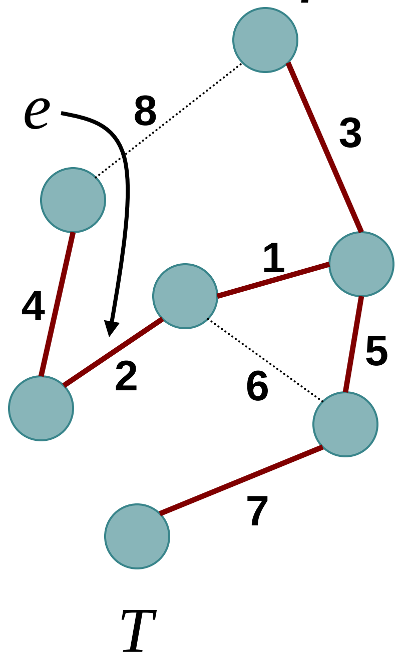


Introduction

- MST Property 2

  If all the edges of a connected graph have different weights, then the MST is unique.

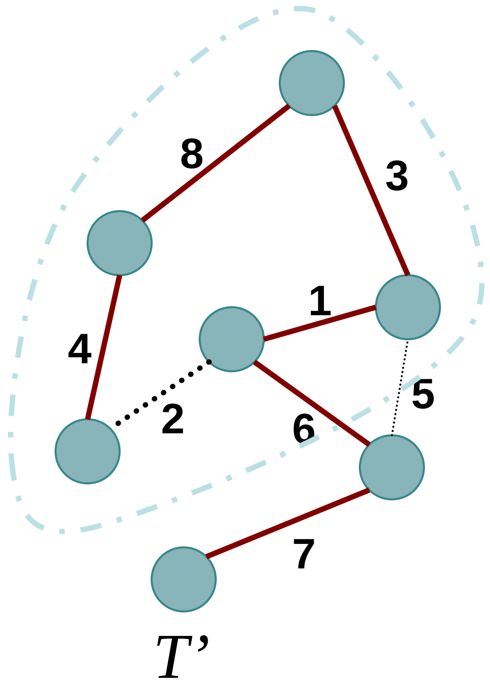
- *Proof:*



Suppose existence of two MSTs.

Let e be the minimal-weight edge not in both MSTs (wlog e in T).

T' with e has a cycle

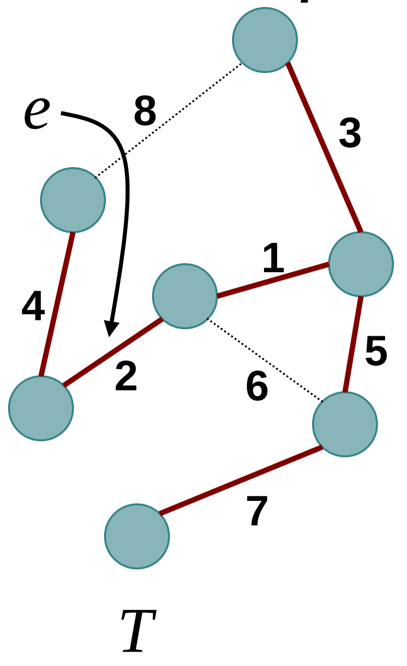


Introduction

- MST Property 2

  If all the edges of a connected graph have different weights, then the MST is unique.

- *Proof:*

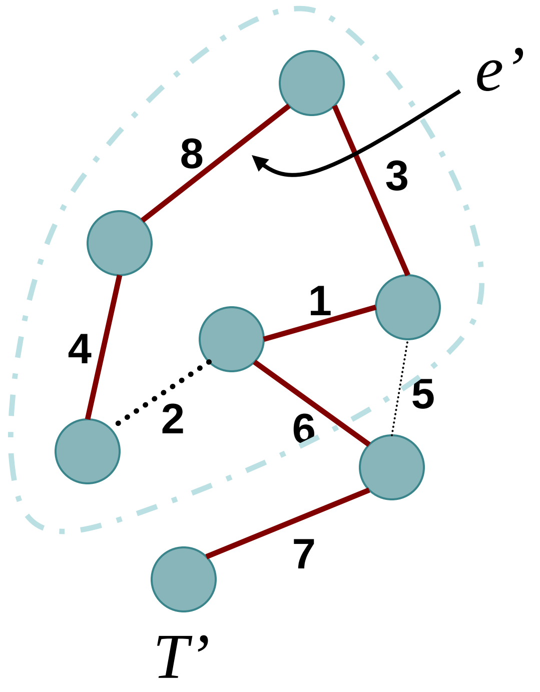


Suppose existence of two MSTs.

Let e be the minimal-weight edge not in both MSTs (wlog e in T).

T' with e has a cycle

At least cycle edge e' is not in T.

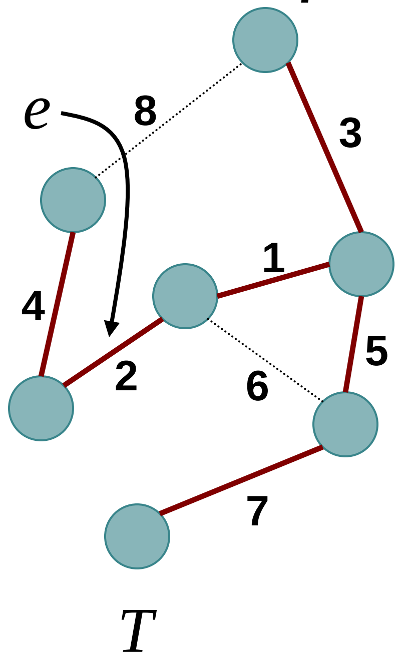


Introduction

- MST Property 2

  If all the edges of a connected graph have different weights, then the MST is unique.

- *Proof:*



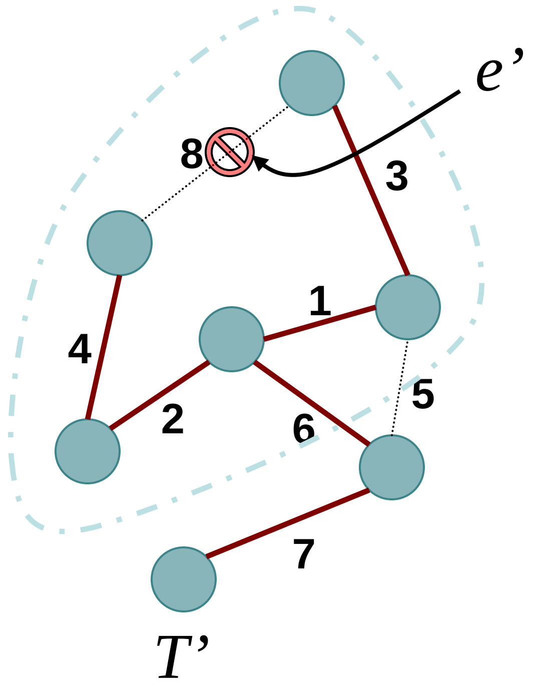
Suppose existence of two MSTs.

Let e be the minimal-weight edge not in both MSTs (wlog e in T).

T' with e has a cycle

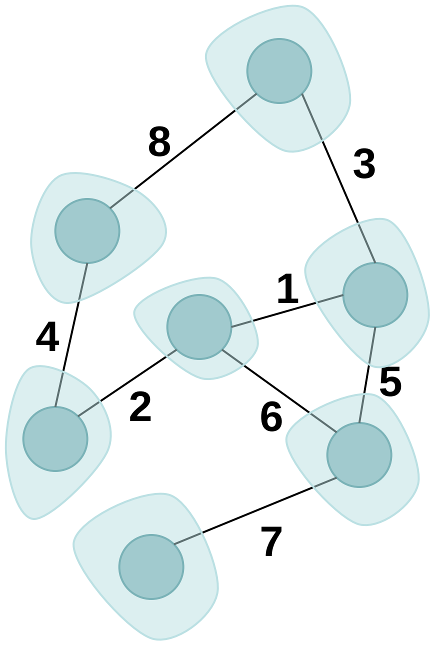
At least cycle edge e' is not in T.

Since w(e) < w(e') we conclude that T' with e and without e' is a smaller MST than T'.



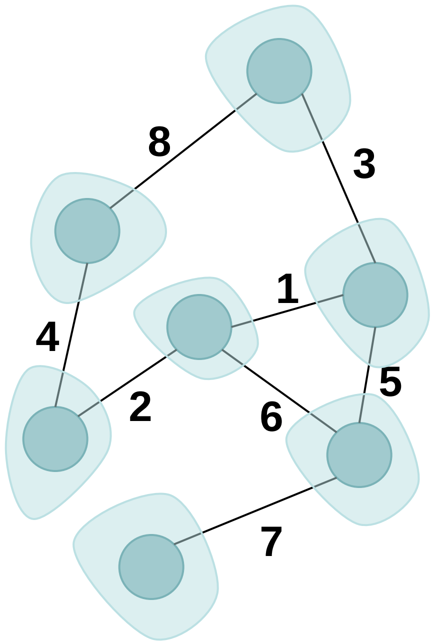
Introduction

- Idea of MST based on properties 1 & 2
  - Start with fragments of one node.



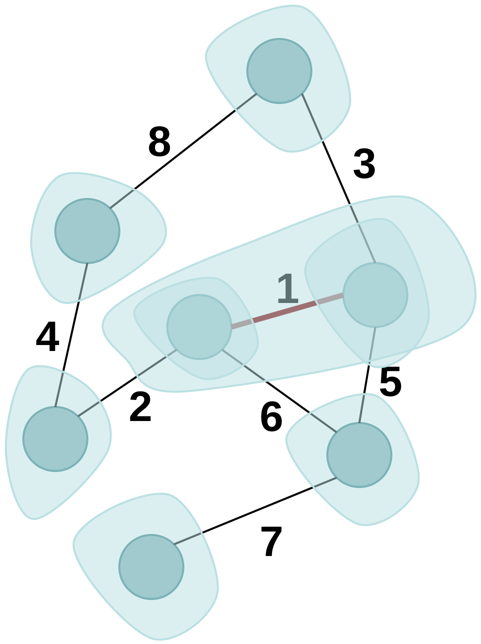
Introduction

- Idea of MST based on properties 1 & 2
  - Enlarge fragments in any order (property 1)
  - Combine fragments with a common node (property 2)



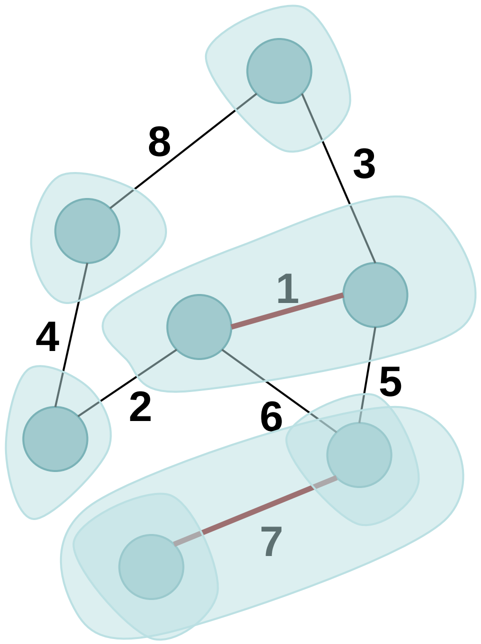
Introduction

- Idea of MST based on properties 1 & 2
  - Enlarge fragments in any order (property 1)
  - Combine fragments with a common node (property 2)



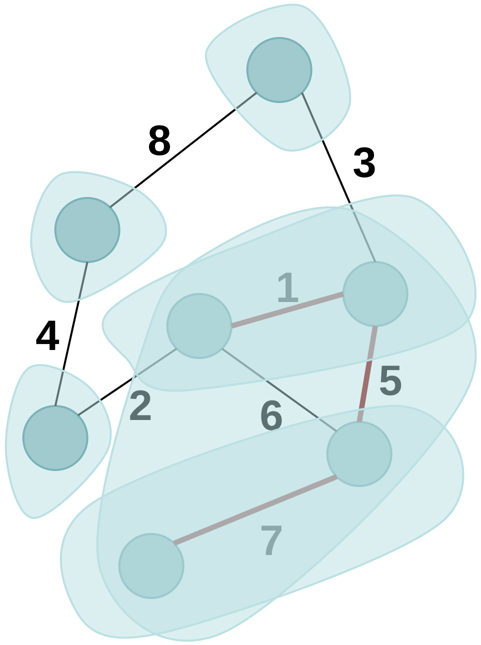
Introduction

- Idea of MST based on properties 1 & 2
  - Enlarge fragments in any order (property 1)
  - Combine fragments with a common node (property 2)



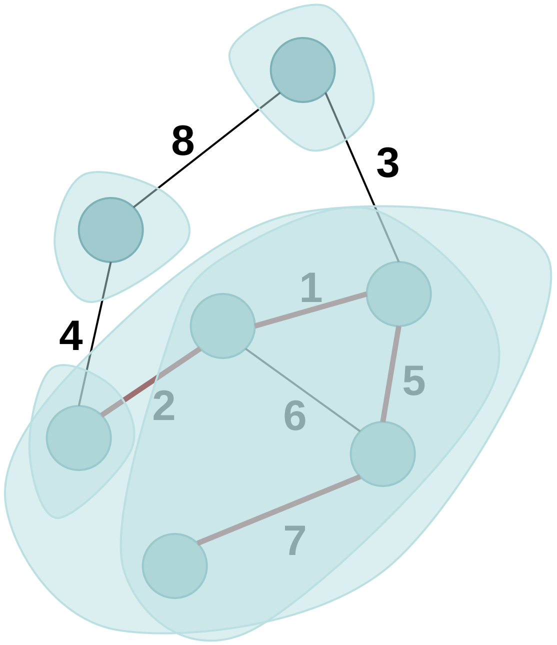
Introduction

- Idea of MST based on properties 1 & 2
  - Enlarge fragments in any order (property 1)
  - Combine fragments with a common node (property 2)



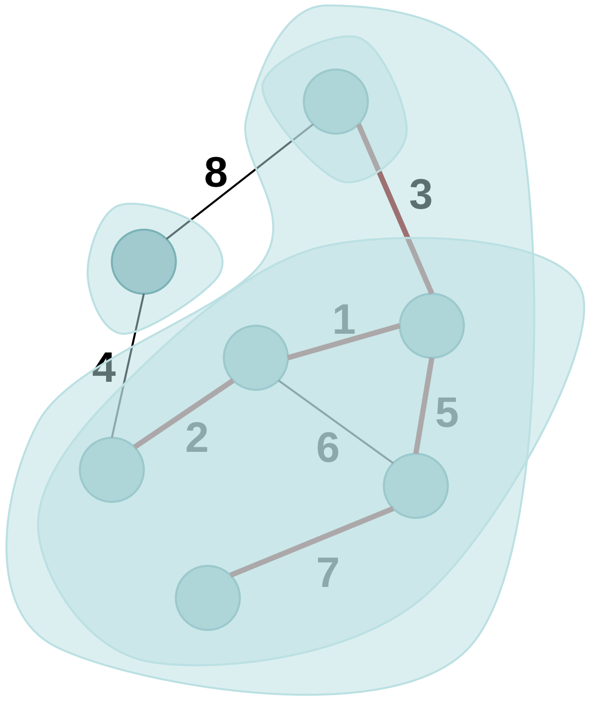
Introduction

- Idea of MST based on properties 1 & 2
  - Enlarge fragments in any order (property 1)
  - Combine fragments with a common node (property 2)



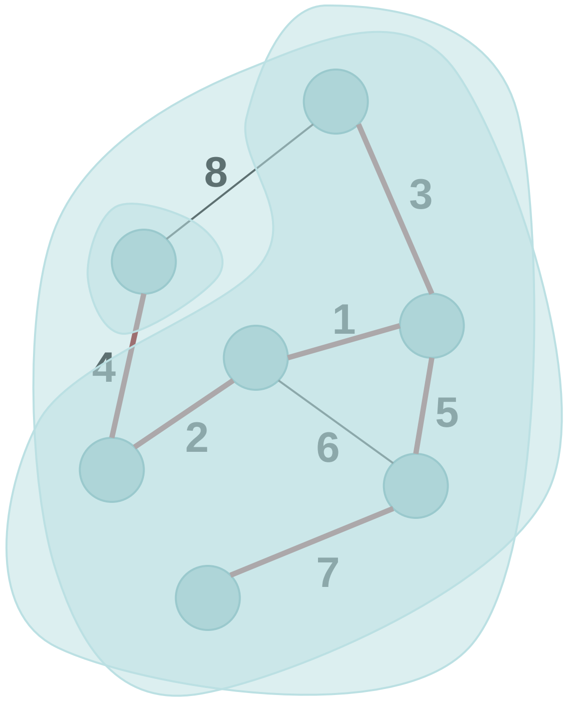
Introduction

- Idea of MST based on properties 1 & 2
  - Enlarge fragments in any order (property 1)
  - Combine fragments with a common node (property 2)



Introduction

- Idea of MST based on properties 1 & 2
  - Enlarge fragments in any order (property 1)
  - Combine fragments with a common node (property 2)

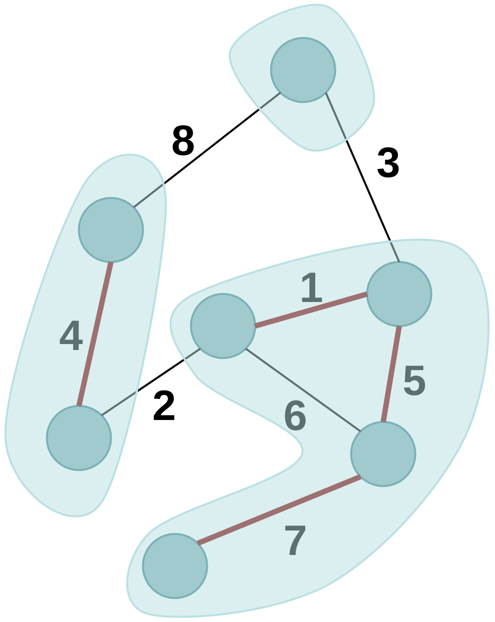


Outline

- Introduction
- **The idea of Distributed MST**
- The algorithm

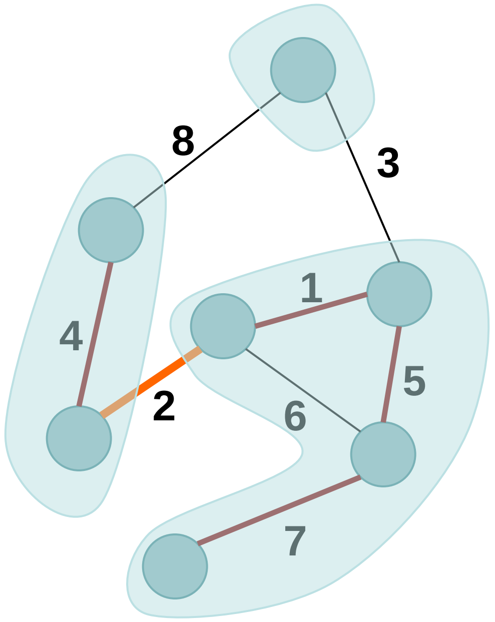
The idea of Distributed MST

- Fragments
  - Every node starts as a single fragment.



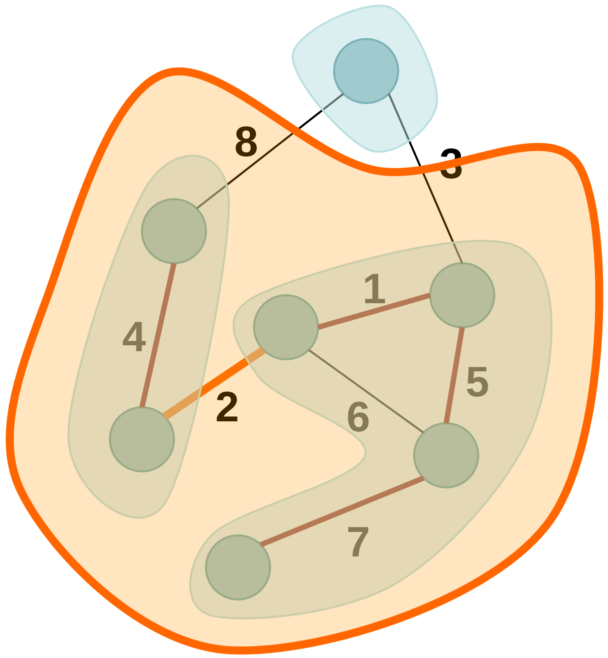
The idea of Distributed MST

- Fragments
  - Each fragment finds its minimum outgoing edge.



The idea of Distributed MST

- Fragments
  - Each fragment finds its minimum outgoing edge.
  - Then it tries to combine with the adjacent fragment.

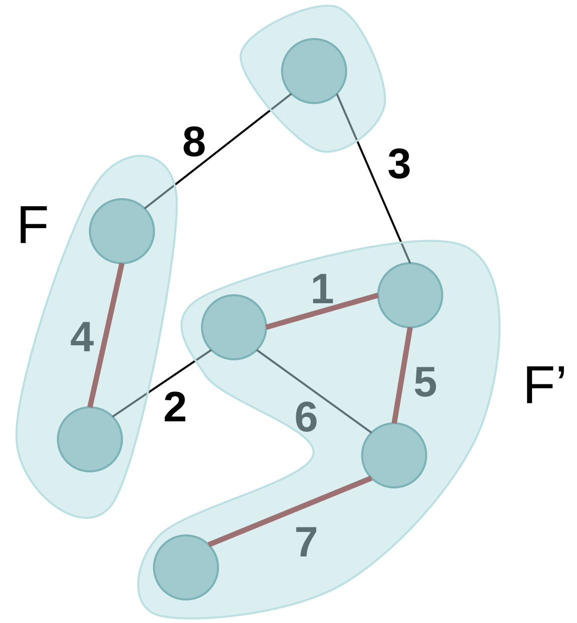


The idea of Distributed MST

- Levels
  - Every fragment has an associated level that has impact on combining fragments.
  - A fragment with a single node is defined to to be at level 0.

The idea of Distributed MST

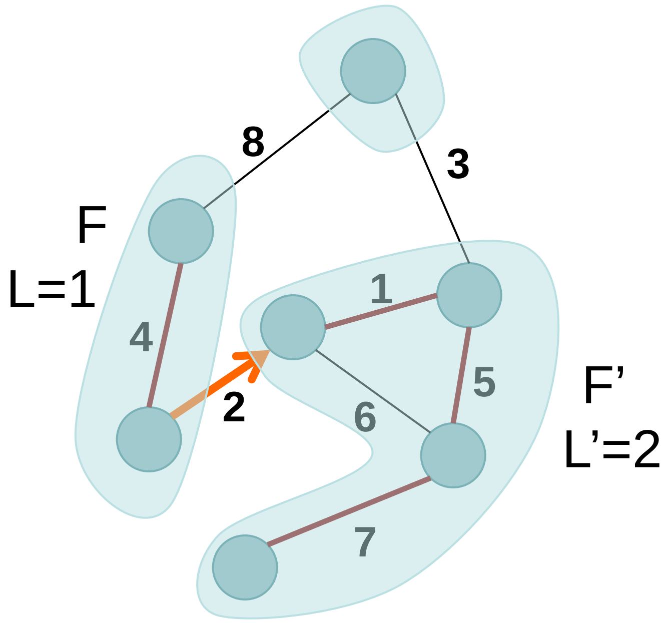
- Levels
  - The combination of two fragments depends on the levels of fragments.



The idea of Distributed MST

- Levels
  - The combination of two fragments depends on the levels of fragments.

If a fragment F wishes to connect to a fragment F' and L < L' then:

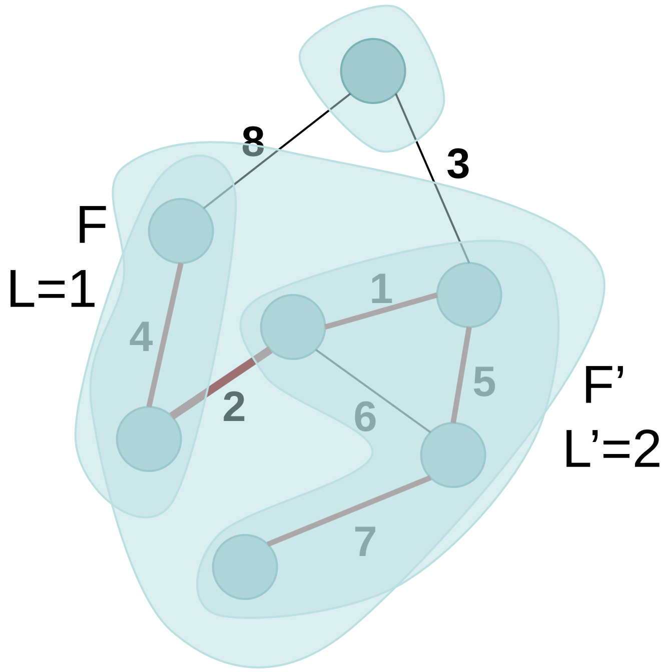


The idea of Distributed MST

- Levels
  - The combination of two fragments depends on the levels of fragments.

If a fragment F wishes to connect to a fragment F' and L < L' then:

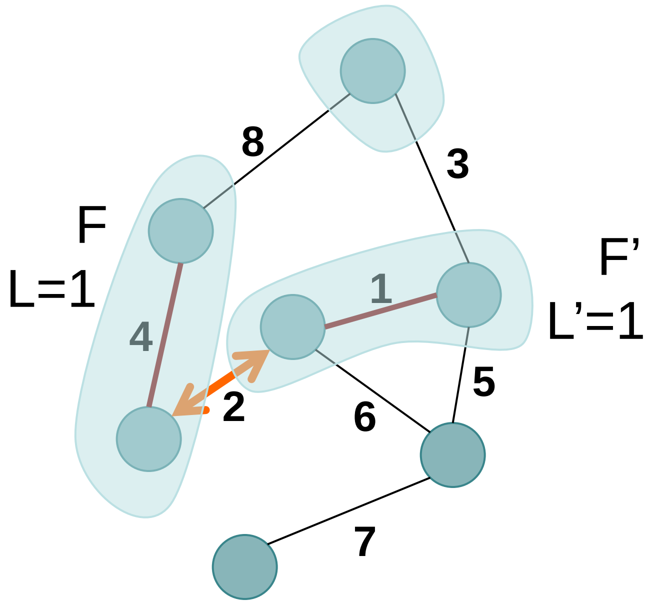
F is absorbed in F' and the resulting fragment is at level L'.



The idea of Distributed MST

- Levels
  - The combination of two fragments depends on the levels of fragments.

If fragments F and F' have the same minimum outgoing edge and L = L' then:

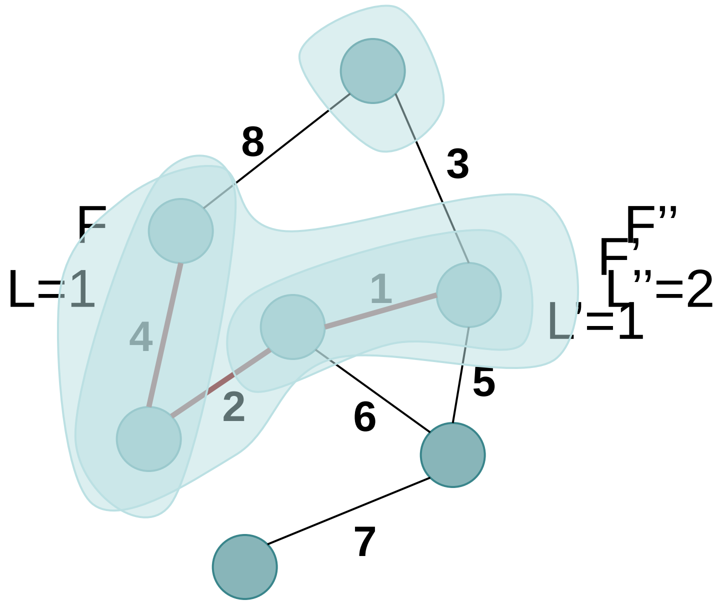


The idea of Distributed MST

- Levels
  - The combination of two fragments depends on the levels of fragments.

If fragments F and F' have the same minimum outgoing edge and L = L' then:

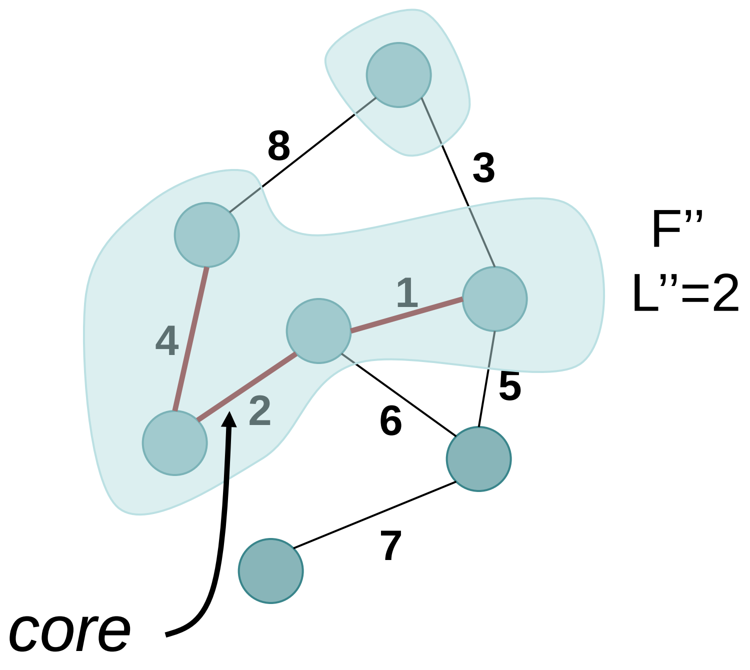
The fragments combine into a new fragment F" at level L" = L+1.



The idea of Distributed MST

- Levels
  - The identity of a fragment is the weight of its *core*.

If fragments F and F' with same level were combined, the combining edge is called the core of the new segment.



The idea of Distributed MST

- State
  - Each node has a state

    Sleeping - initial state

    Find - during fragment’s search for a minimal outgoing edge

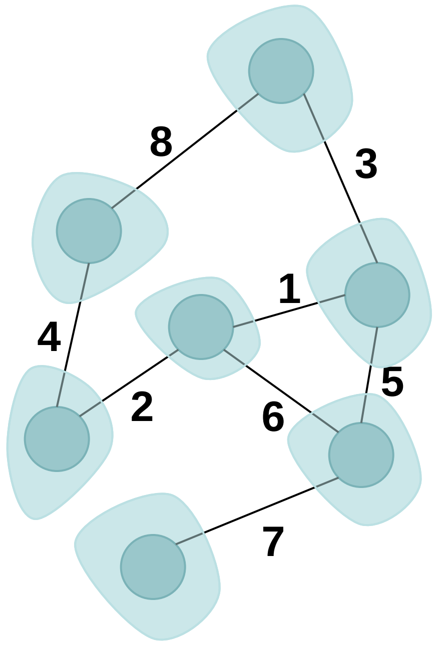
    Found - otherwise (when a minimal outgoing edge was found

Outline

- Introduction
- The idea of Distributed MST
- **The algorithm**

The Algorithm

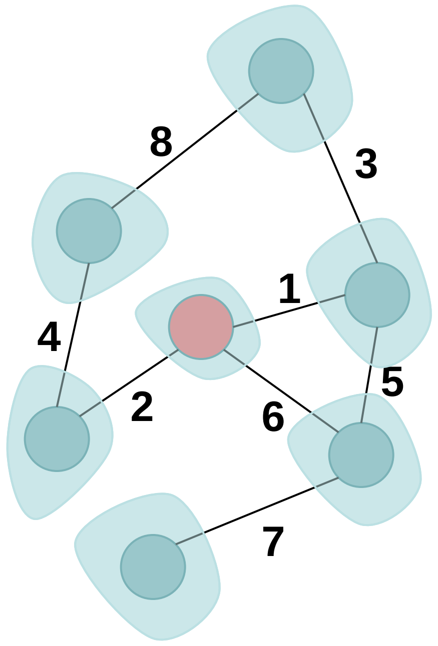
- Fragment minimum outgoing edge discovery
  - Special case of zero-level fragment (***Sleeping***).



The Algorithm

- Fragment minimum outgoing edge discovery
  - Special case of zero-level fragment (***Sleeping***).

*When a node awakes from the state **Sleeping**, it finds a minimum edge connected.*

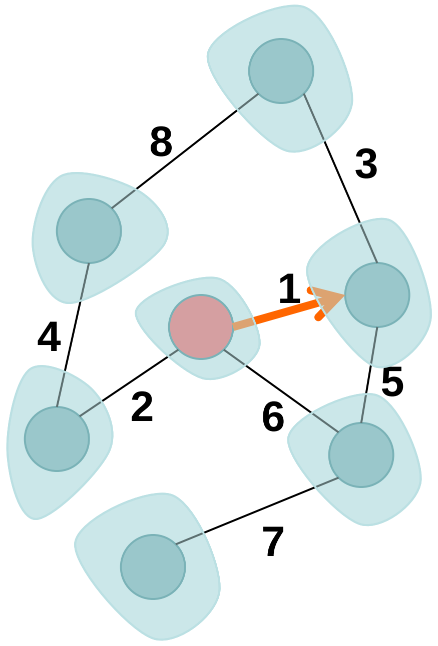


The Algorithm

- Minimum outgoing edge discovery
  - Special case of zero-level fragment (***Sleeping***).

When a node awakes from the state ***Sleeping****, it finds a minimum edge connected.*

Marks it as a ***branch*** *of MST and sends a* ***Connect*** *message over this edge.*



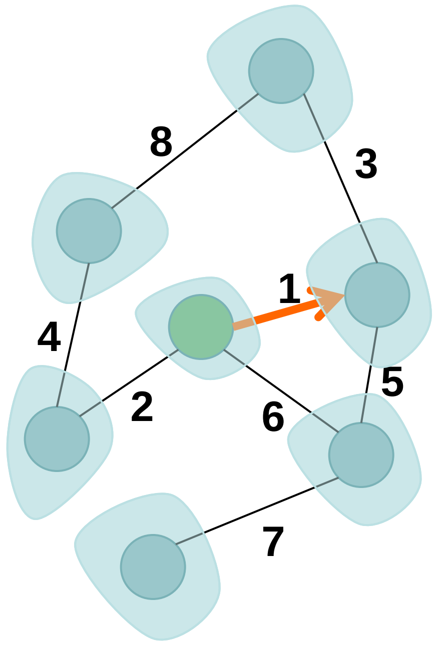
The Algorithm

- Minimum outgoing edge discovery
  - Special case of zero-level fragment (***Sleeping***).

When a node awakes from a state ***Sleeping****, it finds a minimum edge connected.*

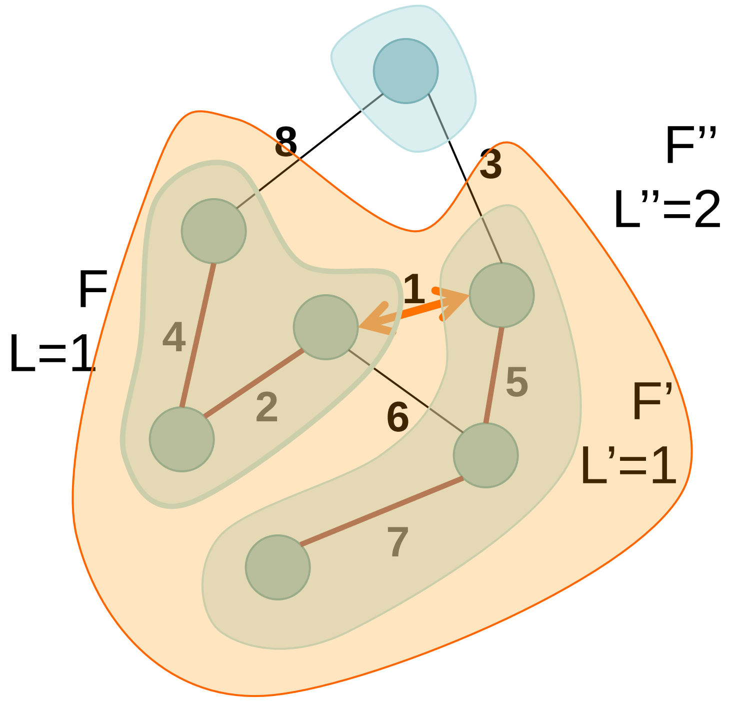
Marks it as a ***branch*** *of MST and sends a* ***Connect*** *message over this edge.*

Goes into a ***Found*** *state.*



The Algorithm

- Minimum outgoing edge discovery
  - Take a fragment at level L that was just combined out of two level L-1 fragments.

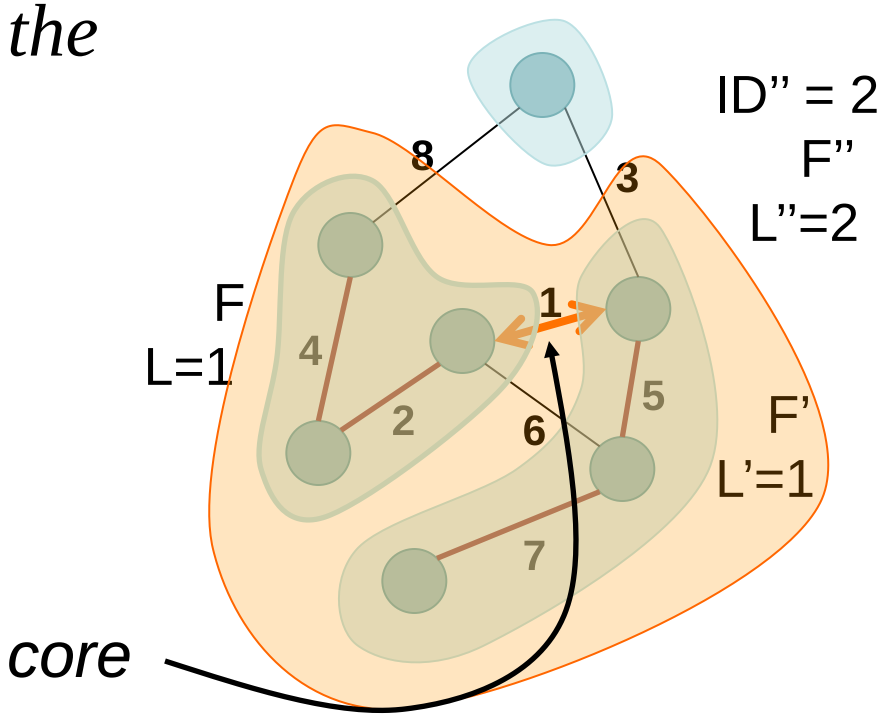


The Algorithm

- Minimum outgoing edge discovery
  - Take a fragment at level L that was just combined out of two level L-1 fragments.

The weight of the core is the identity of the fragment.

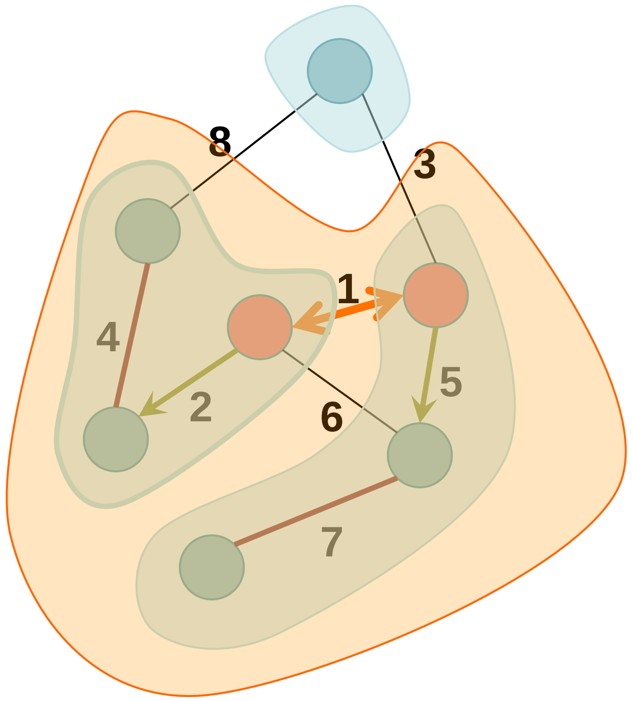
It acts as a root of a fragment tree.



The Algorithm

- Minimum outgoing edge discovery
  - Take a fragment at level L that was just combined out of two level L-1 fragments.

*Nodes adjacent to core send an **Initiate** message to the borders.*

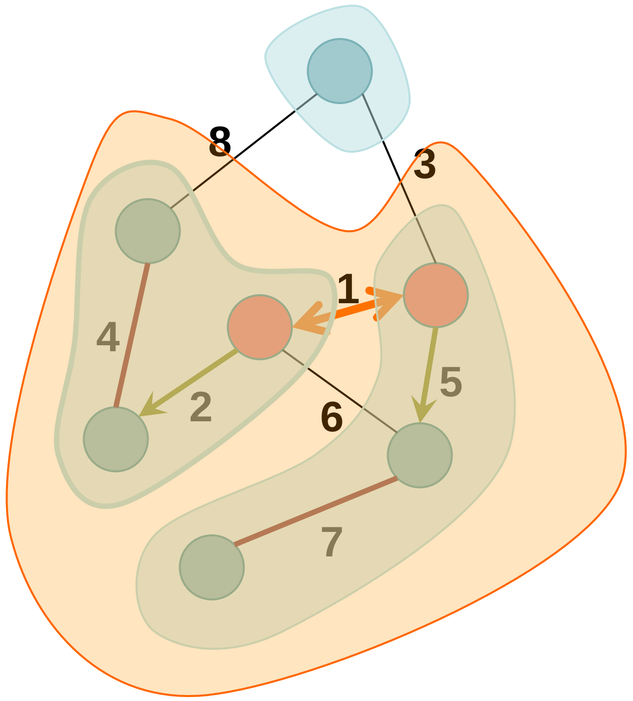


The Algorithm

- Minimum outgoing edge discovery
  - Take a fragment at level L that was just combined out of two level L-1 fragments.

Nodes adjacent to core send an ***Initiate*** *message to the borders.*

Relayed by the intermediate nodes in the fragment.



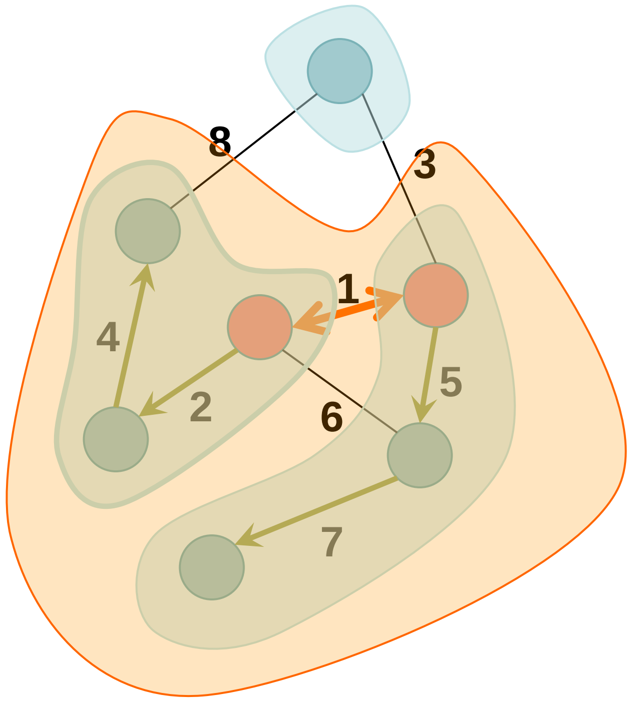
The Algorithm

- Minimum outgoing edge discovery
  - Take a fragment at level L that was just combined out of two level L-1 fragments.

*Nodes adjacent to core send an **Initiate** message to the borders.*

Relayed by the intermediate nodes in the fragment.

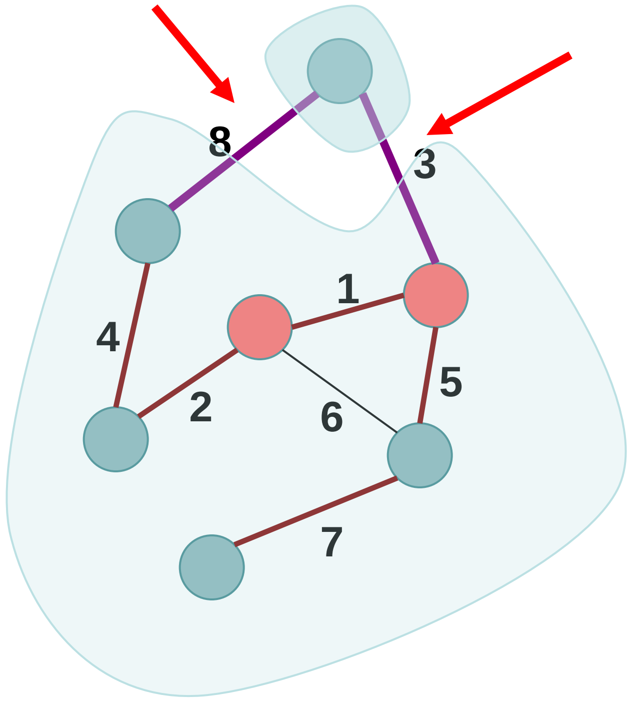
*Puts the nodes in the **Find** state.*



The Algorithm

- Minimum outgoing edge discovery
  - Edge classification/

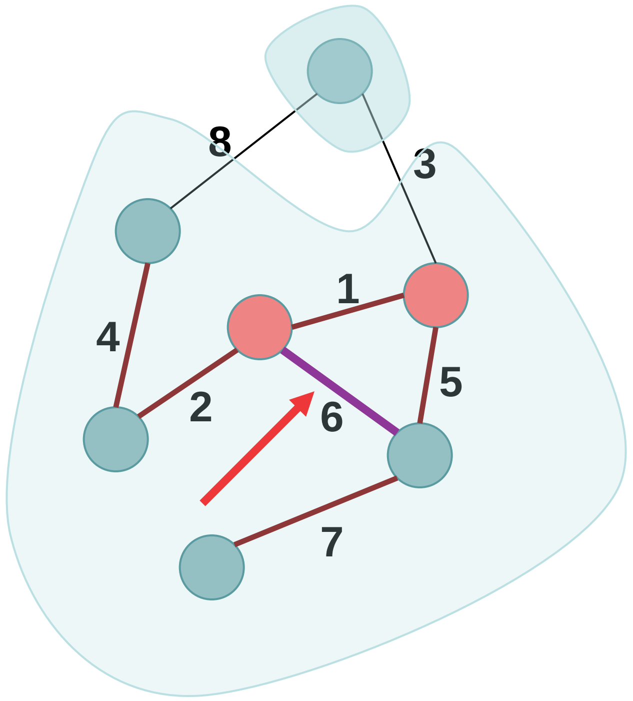
Basic - yet to be classified, can be inside fragment or outgoing edges



The Algorithm

- Minimum outgoing edge discovery
  - Edge classification.

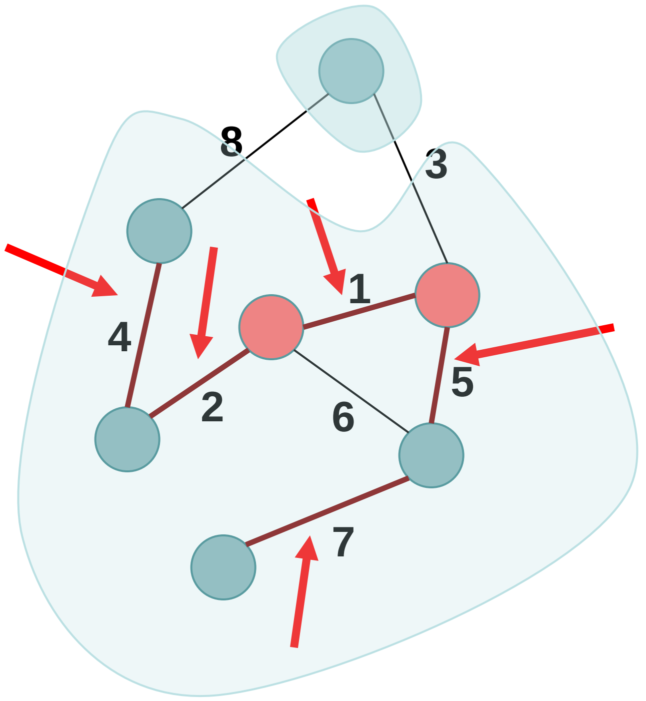
Rejected – an inside fragment edge



The Algorithm

- Minimum outgoing edge discovery
  - Edge classification.

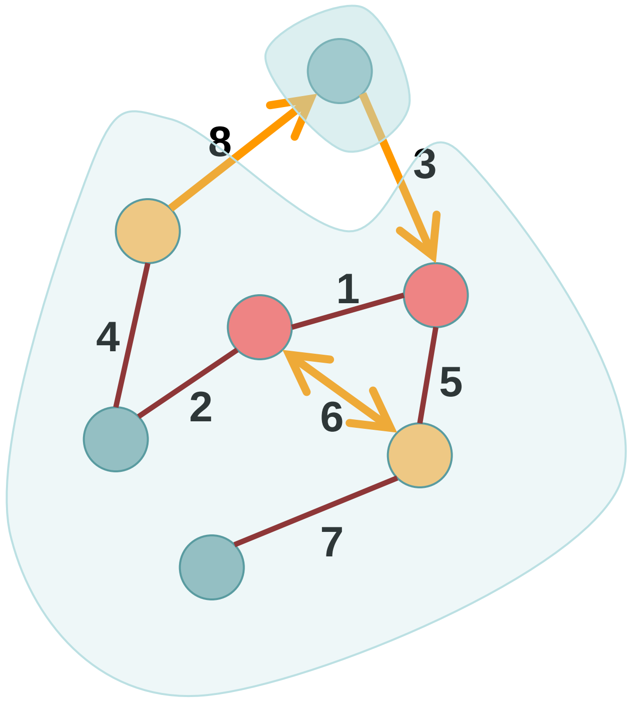
Branch – an MST edge



The Algorithm

- Minimum outgoing edge discovery
  - On receiving the ***Initiate*** message a node tries to find a minimum outgoing edge.

Sends a ***Test*** *message on* ***Basic*** *edges (minimal first)*



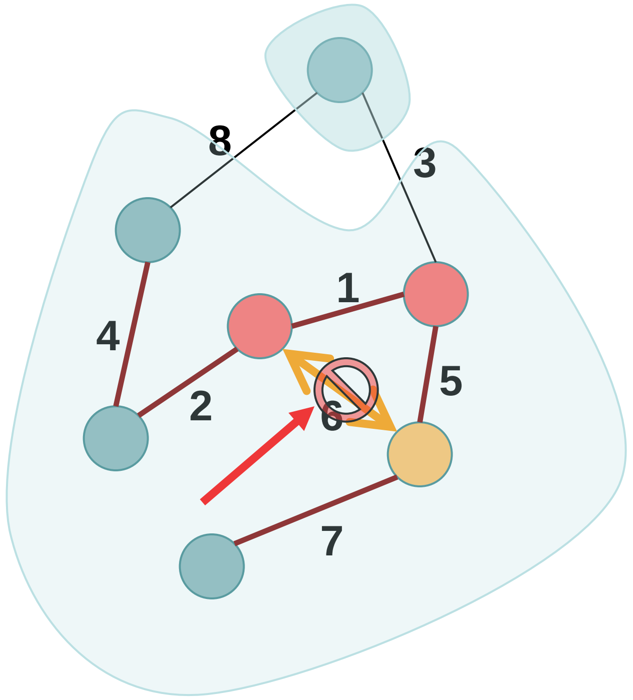
The Algorithm

- Minimum outgoing edge discovery
  - On receiving the ***Test*** message.

In case of same identity:

*send a **Reject** message, the edge is **Rejected**.*

*In case **Test** was sent in both directions, the edge is **Rejected** automatically without a **Reject** message.*

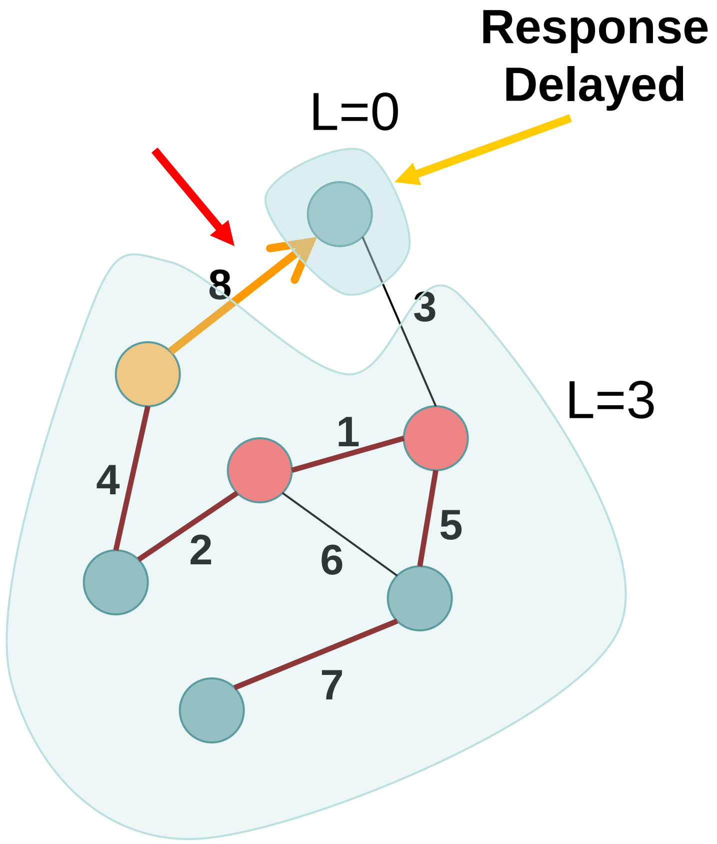


The Algorithm

- Minimum outgoing edge discovery
  - On receiving the ***Test*** message.

In case of a self lower level:

Delay the response until the identity rises sufficiently.



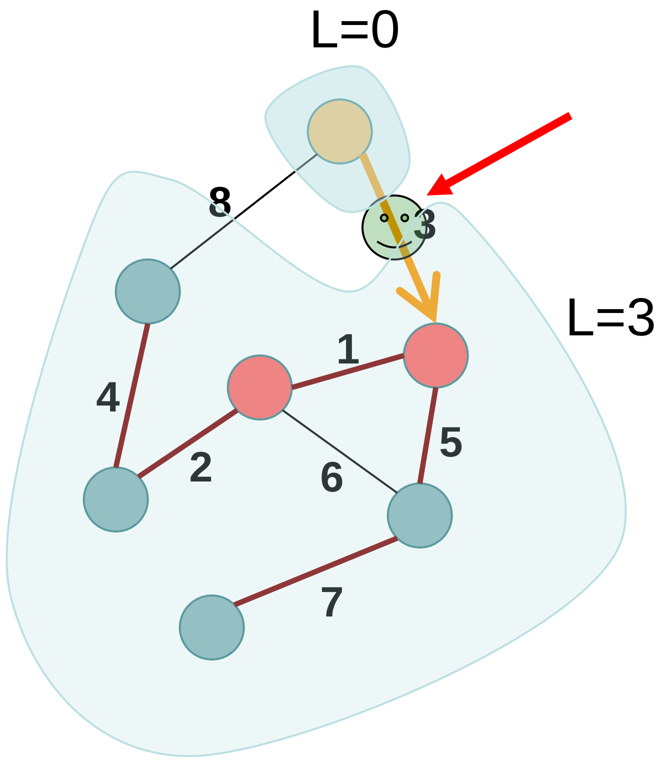
The Algorithm

- Minimum outgoing edge discovery
  - On receiving the ***Test*** message.

In case of a self higher level:

Send an ***Accept*** *message*

The edge is accepted as a candidate.

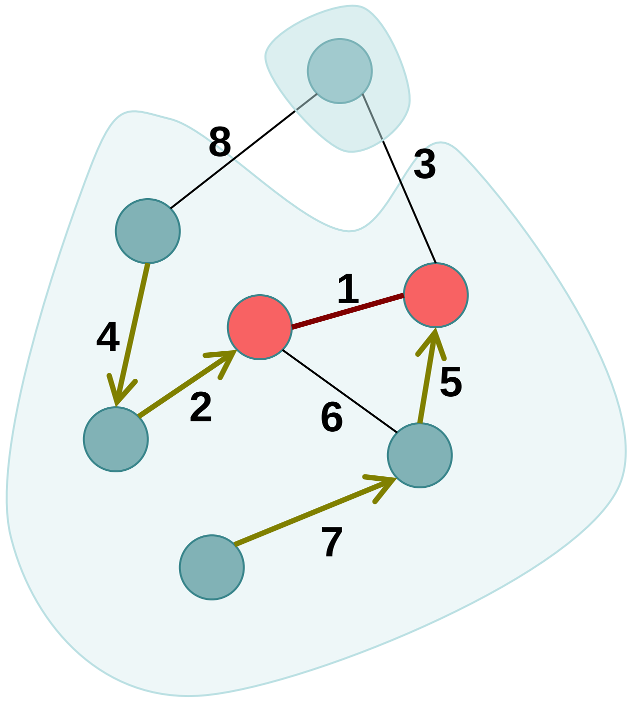


The Algorithm

- Minimum outgoing edge discovery
  - Agreeing on the minimal outgoing edge.

*Nodes send **Report** messages along the branches of the MST.*

If no outgoing edge was found the algorithm is complete.



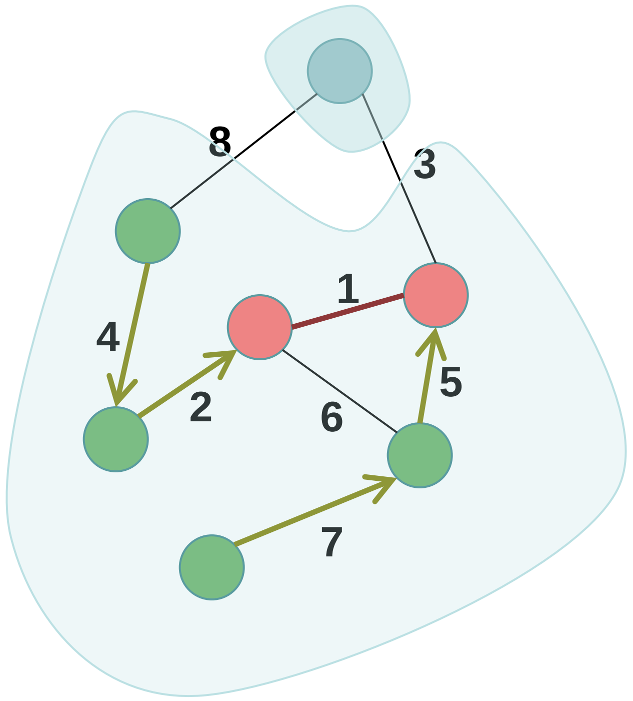
The Algorithm

- Minimum outgoing edge discovery
  - Agreeing on the minimal outgoing edge.

*Nodes send **Report** messages along the branches of the MST.*

If no outgoing edge was found the algorithm is complete.

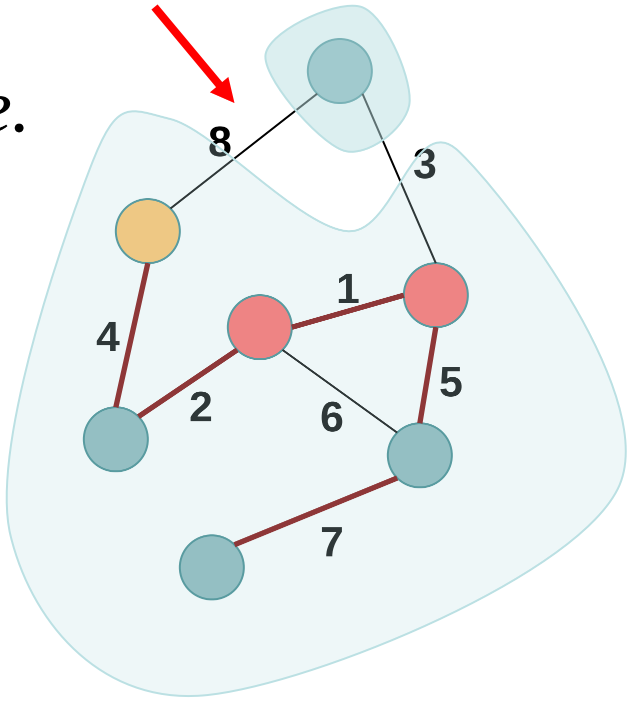
*After sending they go into **Found** mode.*



The Algorithm

- Minimum outgoing edge discovery
  - Agreeing on the minimal outgoing edge.

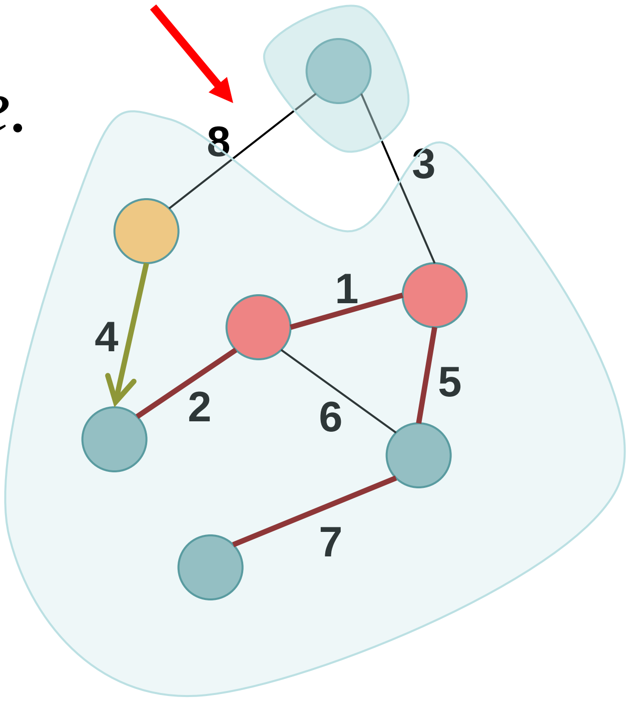
*Every leaf sends the **Report***
when resolved its outgoing edge.



The Algorithm

- Minimum outgoing edge discovery
  - Agreeing on the minimal outgoing edge.

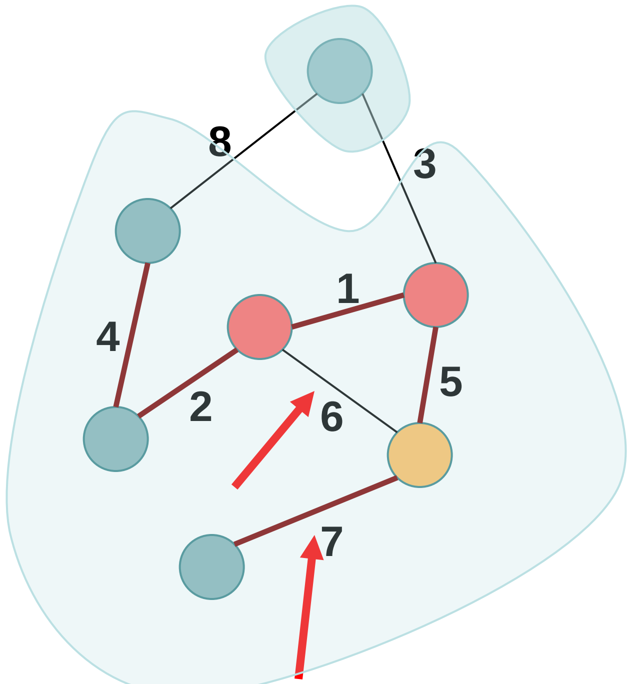
Every leaf sends the ***Report*** *when resolved its outgoing edge.*



The Algorithm

- Minimum outgoing edge discovery
  - Agreeing on the minimal outgoing edge.

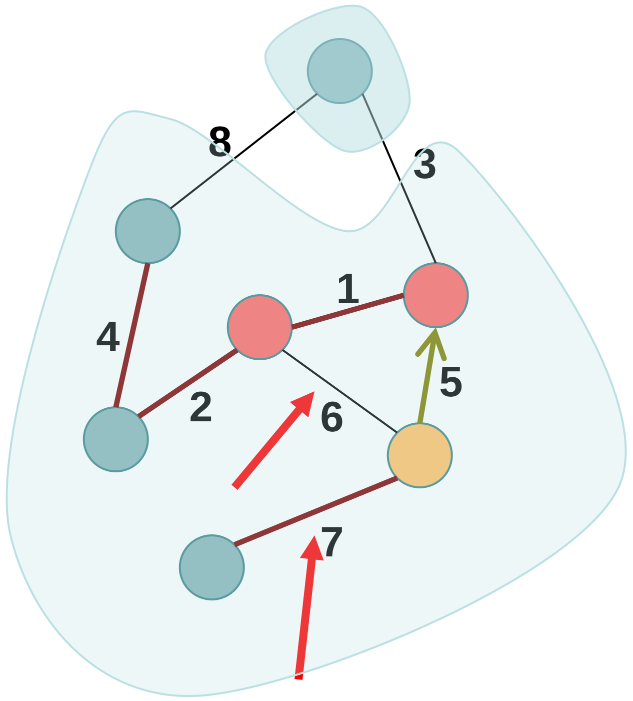
Every interior sends the ***Report*** *when resolved its outgoing and all its children sent theirs.*



The Algorithm

- Minimum outgoing edge discovery
  - Agreeing on the minimal outgoing edge.

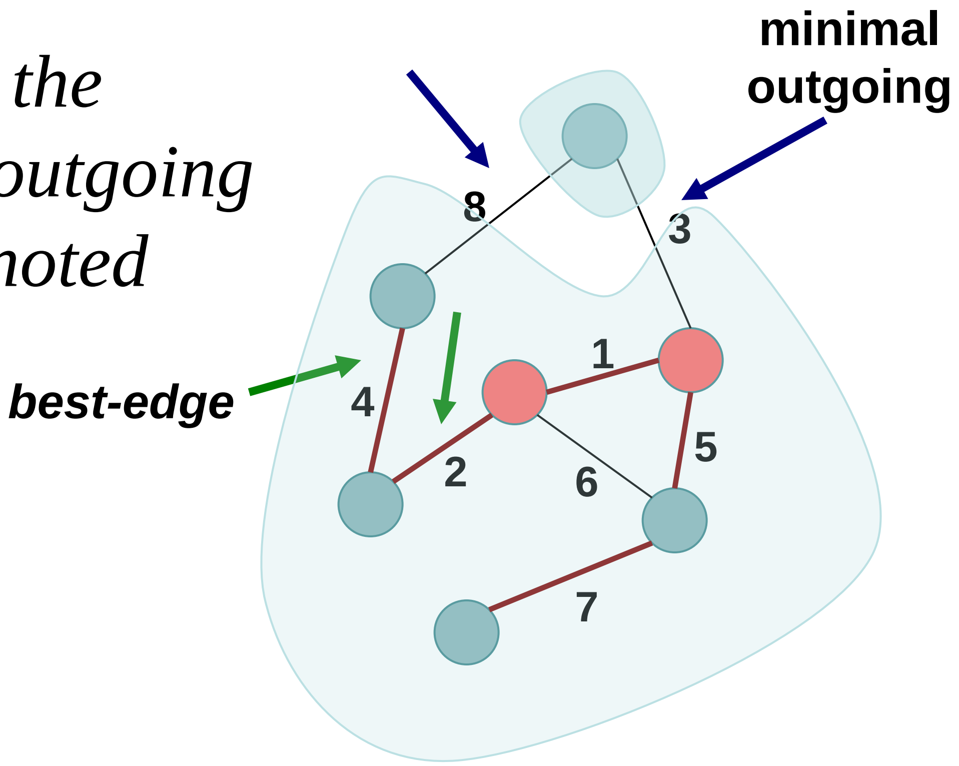
*Every interior sends the **Report** when resolved its outgoing and all its children sent theirs.*



The Algorithm

- Minimum outgoing edge discovery
  - Agreeing on the minimal outgoing edge.

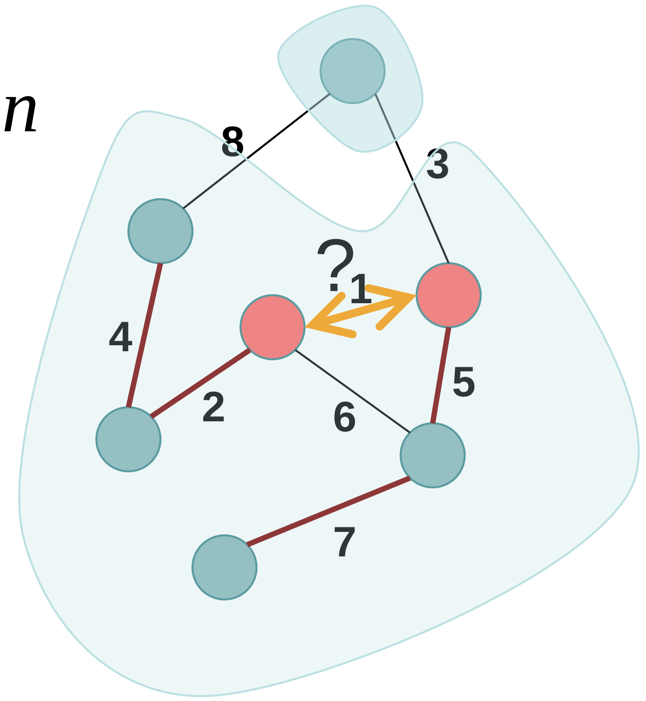
Every node remembers the branch to the minimal outgoing edge of its sub-tree, denoted ***best-edge****.*



The Algorithm

- Minimum outgoing edge discovery
  - Agreeing on the minimal outgoing edge.

The core adjacent nodes exchange ***Reports*** *and decide on the minimal outgoing edge.*

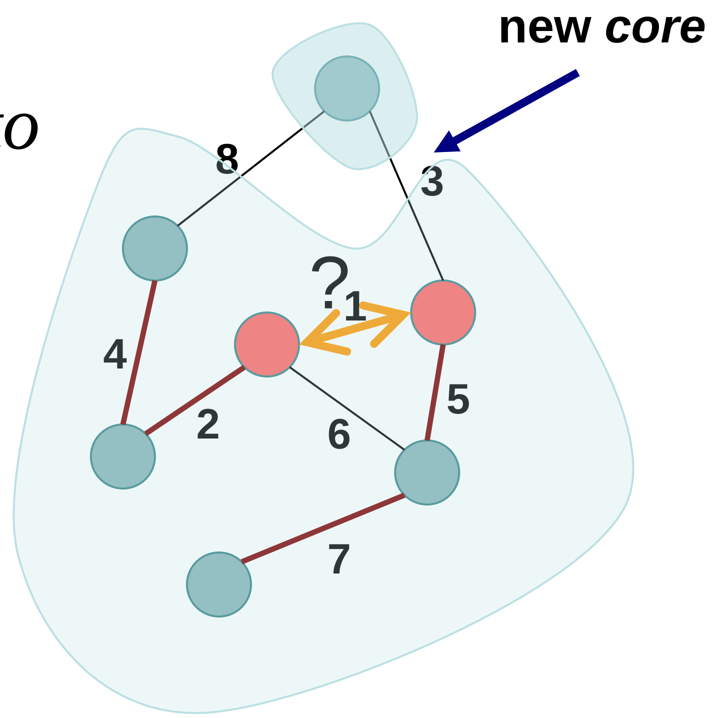


The Algorithm

- Combining segments
  - Changing core.

When decided a ***Change-core*** *message is sent over branches to the minimal outgoing edge.*

The tree ***branches*** *point to the new core.*



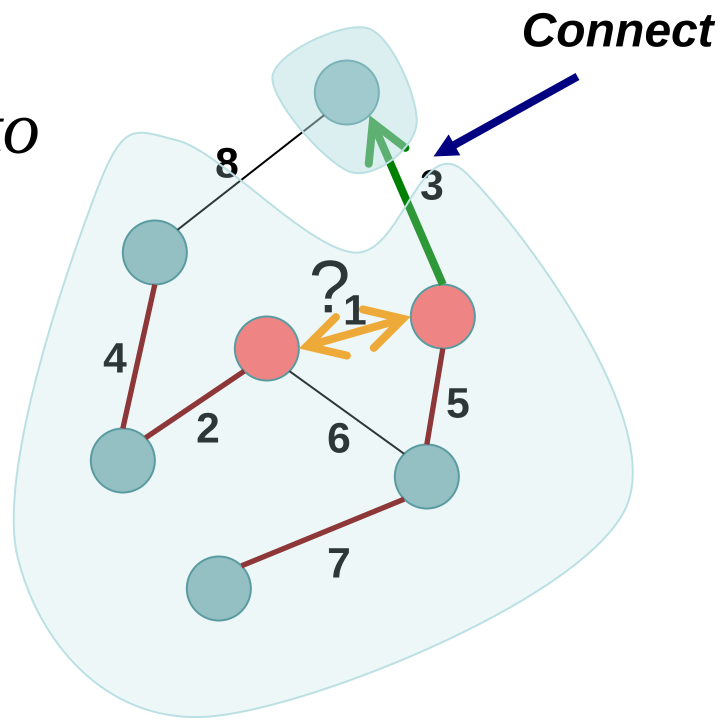
The Algorithm

- Combining segments
  - Changing core

*When decided a **Change-core** message is sent over branches to the minimal outgoing edge.*

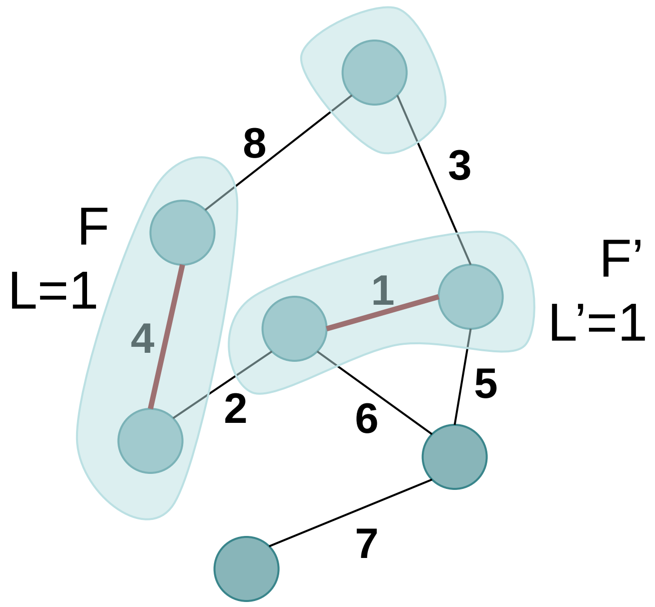
*The tree **branches** point to the new core.*

*Finally a **Connect** message is sent over the minimal edge.*



The Algorithm

- Final notes
  - Connecting same level fragments.

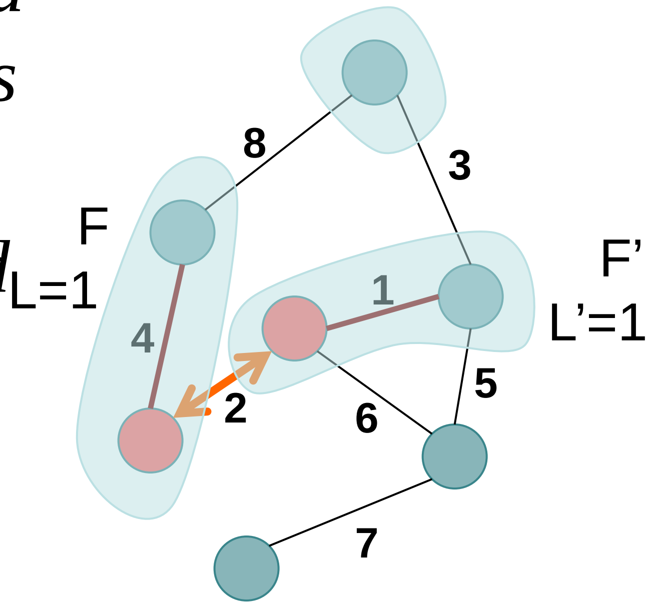


The Algorithm

- Final notes
  - Connecting same level fragments.

Both core adjacent nodes send a ***Connect*** *message, which causes the level to be increased.*

As a result, core is changed and new ***Initiate*** *messages are sent.*



The Algorithm

- Final notes
  - Connecting lower level fragments.

When lower level fragment F' at node n' joins some fragment F at node n before n sent its ***Report.***

We can send n' an ***Initiate*** *message with the* ***Find*** *state so it joins the search.*

The Algorithm

- Final notes
  - Connecting lower level fragments.

When lower level fragment F' at node n' joins some fragment F at node n after n sent its ***Report.***

It means that n already found a lower edge and therefore we can send n' an ***Initiate*** *message with the* ***Found*** *state so it <u>doesn't</u> join the search.*

The Algorithm

- Final notes
  - Forwarding the ***Initiate*** *message at level L.*

When forwarding an ***Initiate*** *message to the leafs, it is also forwarded to any pending fragments at level L-1, as they might be delayed with response.*

The Algorithm

- Final notes
  - Upper bound on fragment levels.

Level L+1 contains at least 2 fragments at level L.

Level L contains at least 2^L *nodes.*

$\log_2 N$ *is an upper bound on fragment levels.*

The Algorithm

- Proof outline
  - Correct MST build-up

The ***Connect*** *message is sent on the minimal outgoing edge only.*

As a result of properties 1 & 2 we should obtain an MST.

The Algorithm

- Proof outline
  - No deadlocks

Assume there is a deadlock.

Choose a fragment from the lowest level set
and with minimal outgoing edge in that set.
Its ***Test/Connect*** *message surely will be replied.*

There will always be a working fragment.

The Algorithm

- Complexity
  - Communication

At most E ***Reject*** *messages (with corresponding E* ***Test*** *messages, because each edge can be rejected only once.*

The Algorithm

- Complexity
  - Communication

At every but the zero or last levels, each node can accept up to 1 ***Initiate, Accept*** *messages. It can transmit up to 1* ***Test (successful), Report, ChangeRoot, Connect****. Since the number of levels is bounded by* $\log_2 N$ *number of such messages is at most* $5N(\log_2 N - 1)$.

The Algorithm

- Complexity
  - Communication

At level zero, each node receives at most one ***Initiate*** *and transmits at most one* **Connect**. *At the last level a node can send at most one* ***Report*** *message, as a result at most 3N such messages.*

The Algorithm

- Complexity
  - Communication

As a result, the upper bound is: $5N\log_2 N + 2E$.

The Algorithm

- Complexity
  - Time (under assumption of initial awakening it is $5N\log_2 N$)

We prove by induction that one needs 5lN - 3N time units for every node to reach level l.

The Algorithm

- Complexity
  - Time (under assumption of initial awakening it is $5N\log_2 N$)

l=1 → *Each node is awakened and sends a* ***Connect*** *message. By time 2N all nodes should be at level 1.*

The Algorithm

- Complexity
  - Time (under assumption of initial awakening it is $5N\log_2 N$)

Assume l → At level l, each node can send at most N ***Test*** *messages which will be answered before time 51N - N . The propagation of the* ***Report*** *messages,* ***ChangeRoot***, ***Connect***, *and* ***Initiate*** *messages can take at most 3N units, so that by time 5 (l + 1)N - 3N all nodes are at level l + 1.*

The Algorithm

- Complexity
  - Time (under assumption of initial awakening it is $5N\log_2 N$)

At the highest level only ***Test, Reject,*** *and* ***Report*** *messages are used.*

The Algorithm

- Complexity
  - Time (under assumption of initial awakening it is $5N\log_2 N$)

As a result we have the algorithm complete under $5N\log_2 N$ *time units.*