Approximating the Degree-Bounded Minimum-Diameter Spanning Tree Problem

Jochen Könemann

Department of Combinatorics & Optimization

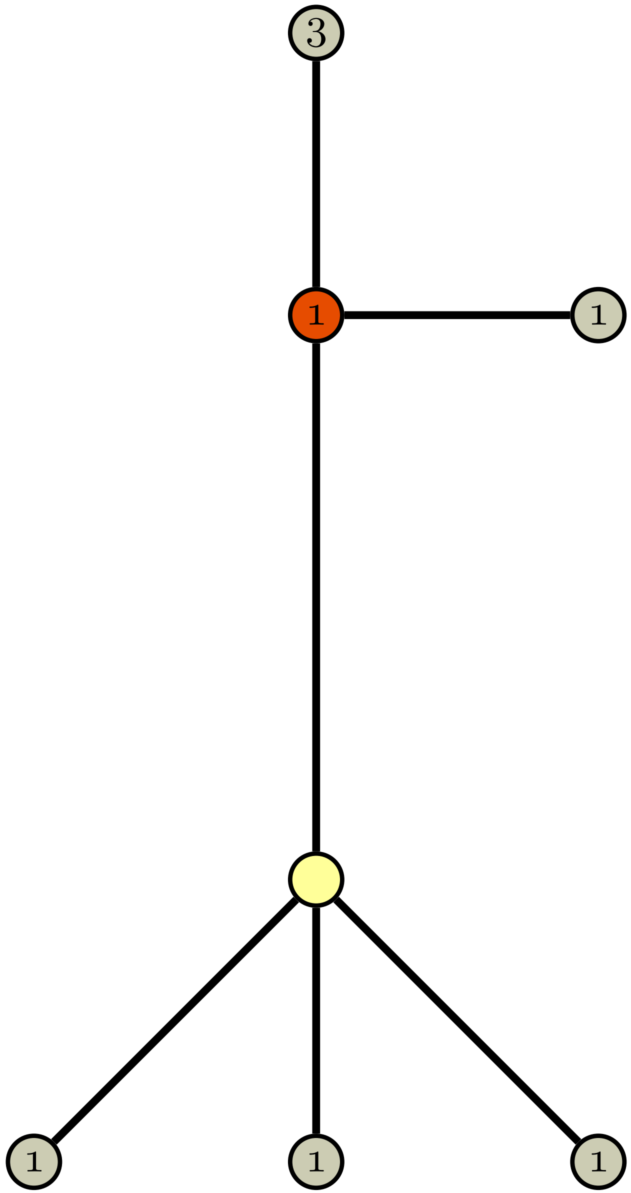
University of Waterloo

Joint work with

A. Levin, Tel Aviv University

A. Sinha, Carngie Mellon University

A real world swarm robotics example



k awake robots

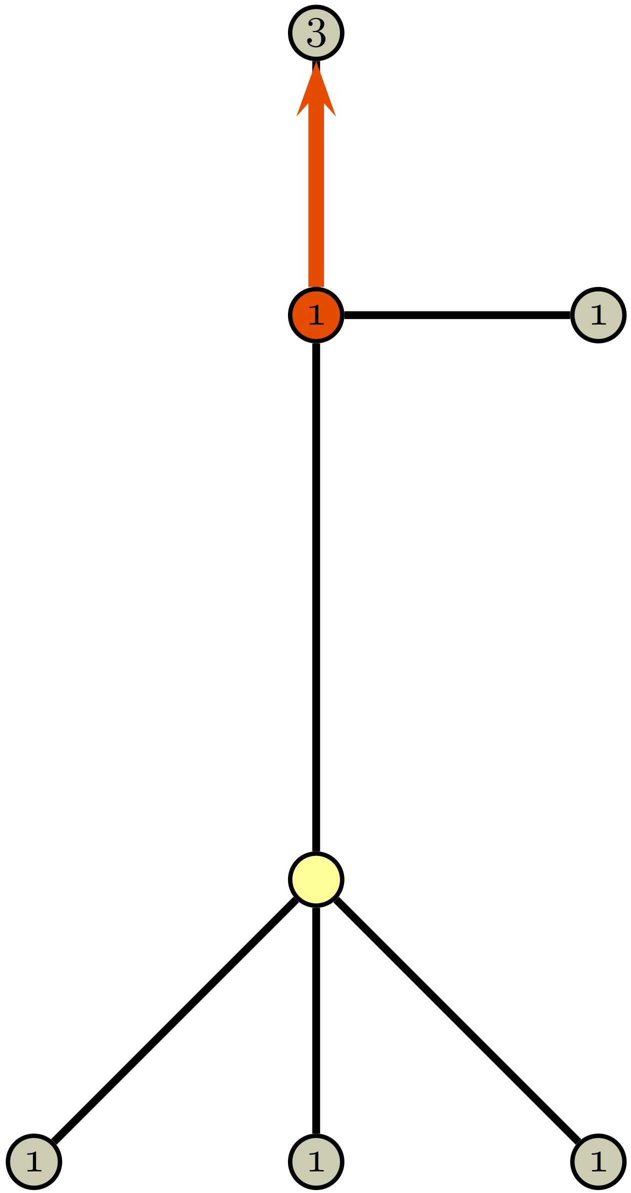
l asleep robots

Given graph with one awake and many asleep robots

Awake robots can travel unit distance per time unit

Wake up all robots as quickly as possible

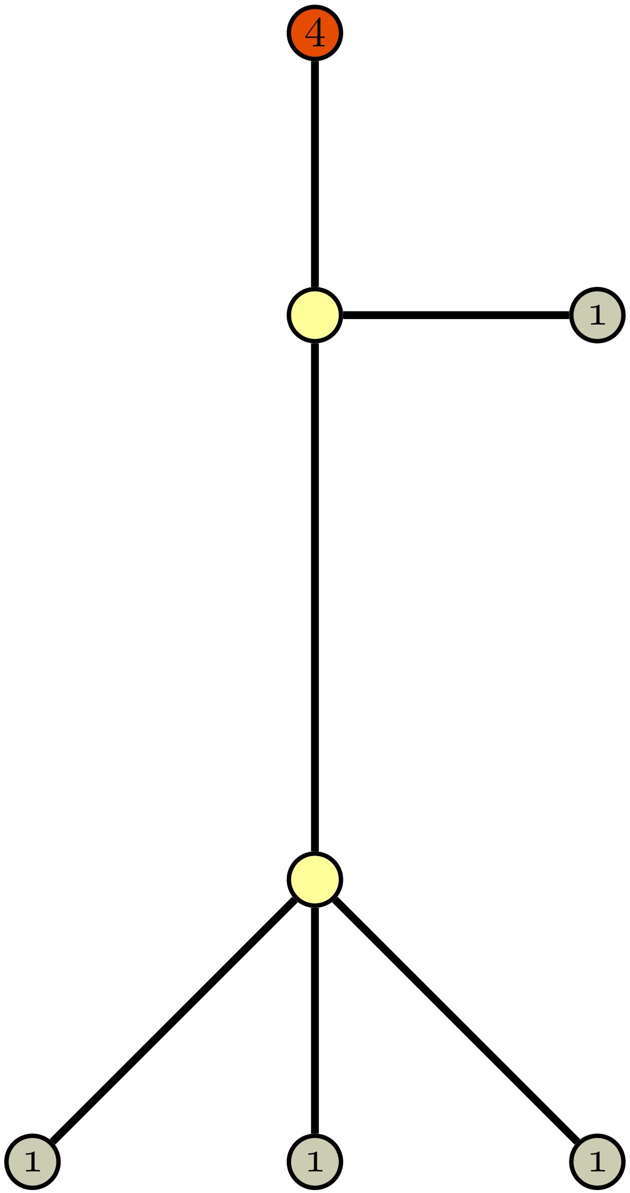
A real world swarm robotics example

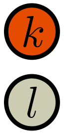


k awake robots

l asleep robots

A real world swarm robotics example

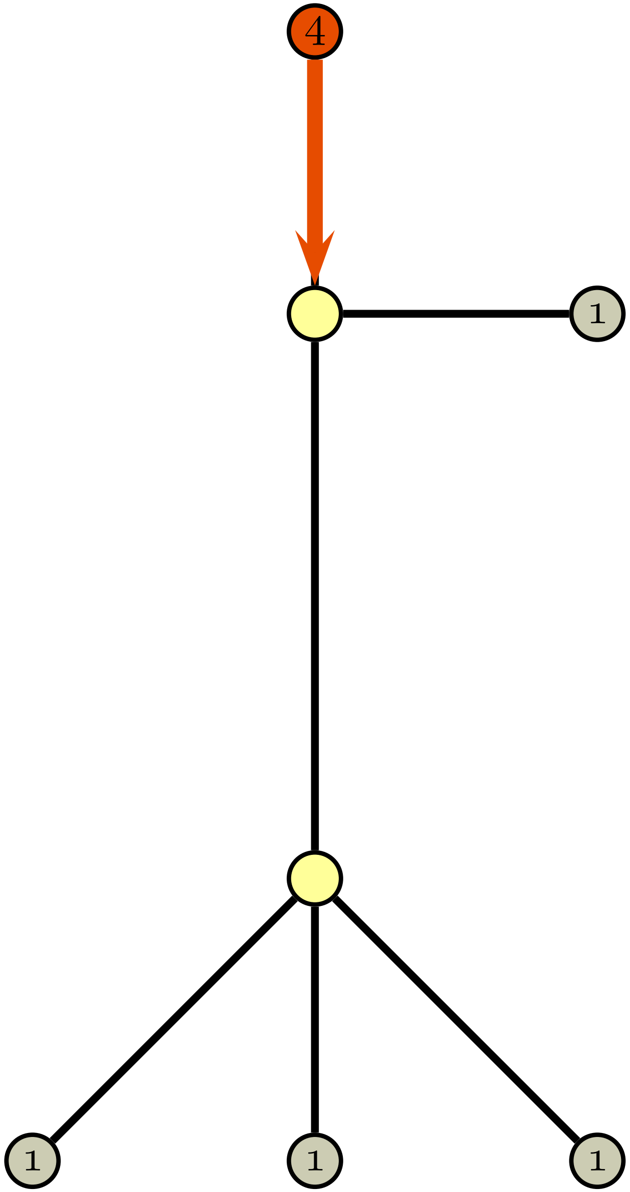


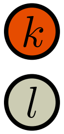


k awake robots

l asleep robots

A real world swarm robotics example

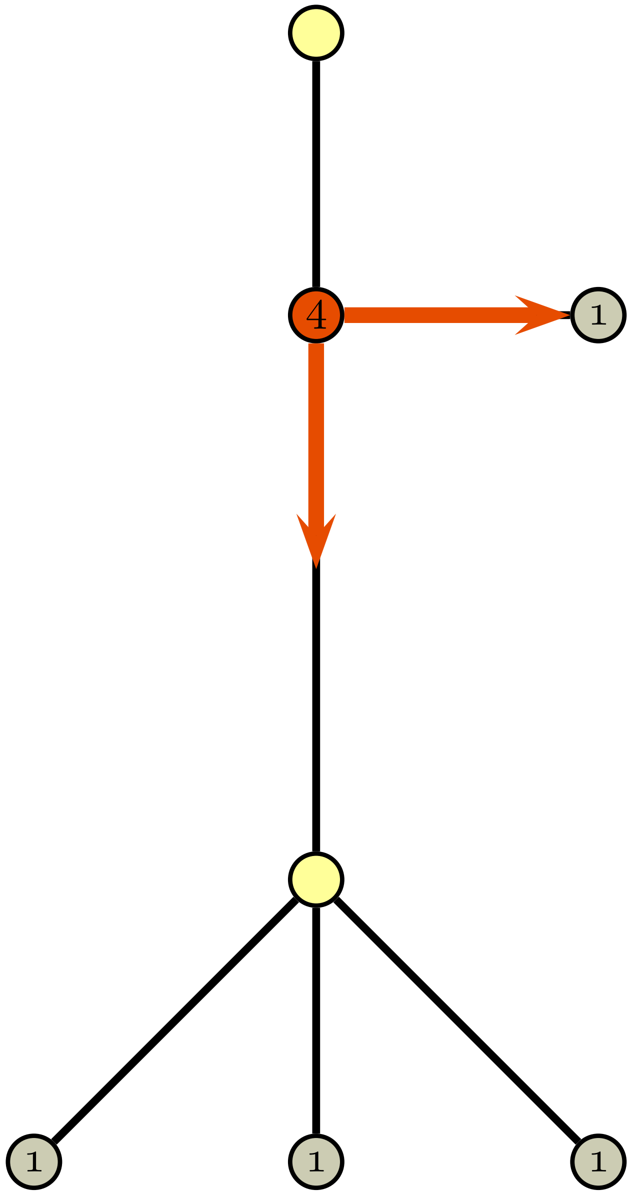




k awake robots

l asleep robots

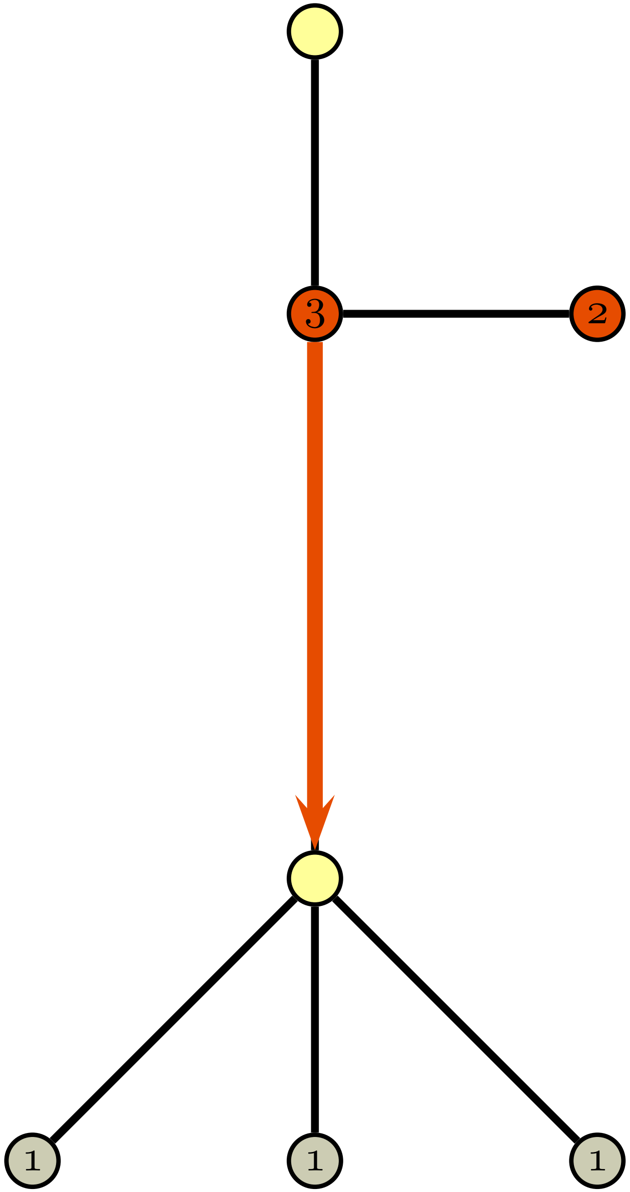
A real world swarm robotics example



k awake robots

l asleep robots

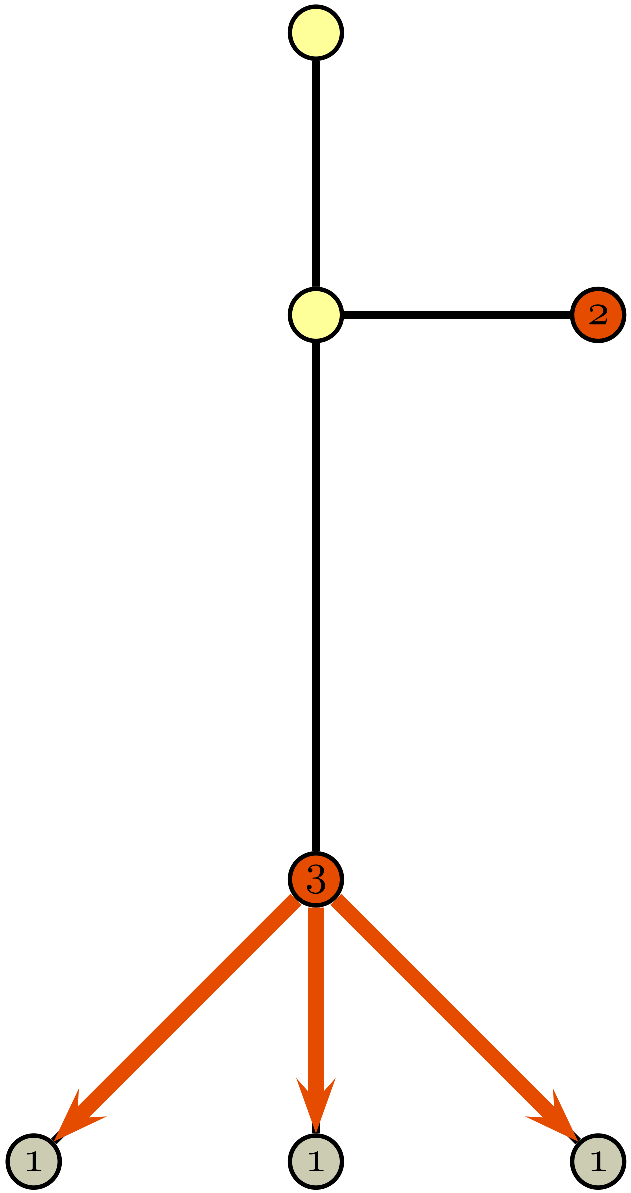
A real world swarm robotics example

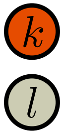


k awake robots

l asleep robots

A real world swarm robotics example

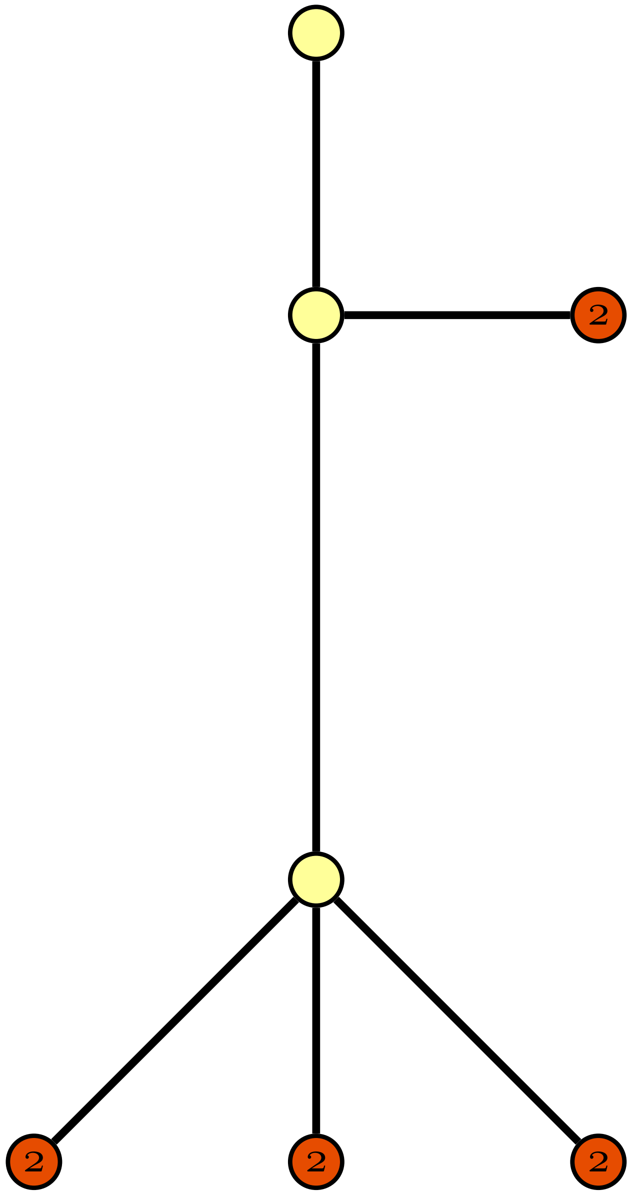


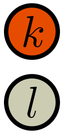


k awake robots

l asleep robots

A real world swarm robotics example





k awake robots

l asleep robots

[Arkin et al. '02] call this the Freeze-Tag problem. Achieve $O(\log n)$-competitive online algorithm for dense n-node graphs.

This paper: Improve competitive ratio to $O(\sqrt{\log n})$.

BDST: Degree-bounded min-diameter trees

Definition [BDST]

Given Undirected complete graph G on nodes V,
Metric length $\{l_{uv}\}_{u,v \in V}$, and
Degree-bound $B_v > 0$ for all $v \in V$.

Find minimum-diameter spanning tree T with node-degree at most B_v for all $v \in V$.

BDST: Degree-bounded min-diameter trees

Definition [BDST]

Given Undirected complete graph G on nodes V,
Metric length $\{l_{uv}\}_{u,v \in V}$, and
Degree-bound $B_v > 0$ for all $v \in V$.

Find minimum-diameter spanning tree T with node-degree at most B_v for all $v \in V$.

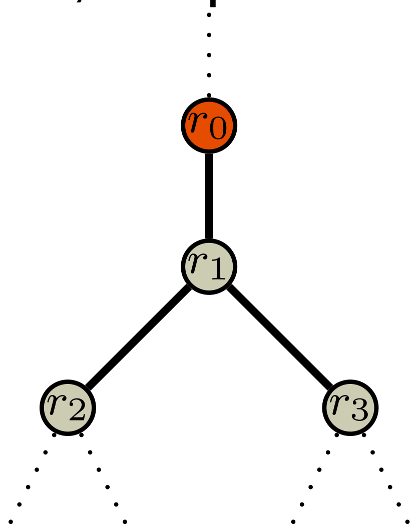
Quiz: Why does a $O(\sqrt{\log_B n})$-approx for BDST help to improve competitive ratio for Freeze-Tag?

Wake-up trees

- Define auxiliary complete graph G_R with one node for each robot.
  Distance between any two nodes u and v is distance in original graph.

Wake-up trees

- Define auxiliary complete graph G_R with one node for each robot.
  Distance between any two nodes u and v is distance in original graph.
- A solution to a Freeze-Tag instance with maximum wake-up time t corresponds to:
  A binary spanning tree T of G_R rooted at awake robot with longest root,leaf-path of length t
  Idea:



r_0 wakes up r_1 and then they both wake at most two other robots r_2 and r_3

Main Result

Theorem 1

Given:

1. Complete graph G on node-set V,
2. Metric $\{l_{uv}\}_{u,v \in V}$, and
3. Degree-bounds $\{B_v\}_{v \in V}$.

We show: Can compute spanning tree T with

1. Degree at most B_v at node v for all $v \in V$
2. Diameter of T is $O(\sqrt{\log_B n}) \cdot \Delta$
   (Δ: minimum diameter of any feasible solution
   $B = \max_v B_v$)

Main Result

Theorem 1

Given:

1. Complete graph G on node-set V,
2. Metric $\{l_{uv}\}_{u,v \in V}$, and
3. Degree-bounds $\{B_v\}_{v \in V}$.

We show: Can compute spanning tree T with

1. Degree at most B_v at node v for all $v \in V$
2. Diameter of T is $O(\sqrt{\log_B n}) \cdot \Delta$
   (Δ: minimum diameter of any feasible solution
   $B = \max_v B_v$)

Hardness [Arkin et al. '02]

Not approximable within $5/3 - \epsilon$ for any $\epsilon > 0$ unless P=NP.

Previous Work

[Ravi '94]

Approximation algorithm for broadcasting

Given: Graph $G(V, E)$, (non-metric) length on edges, degree-bounds $B_v > 0$ for all $v \in V$

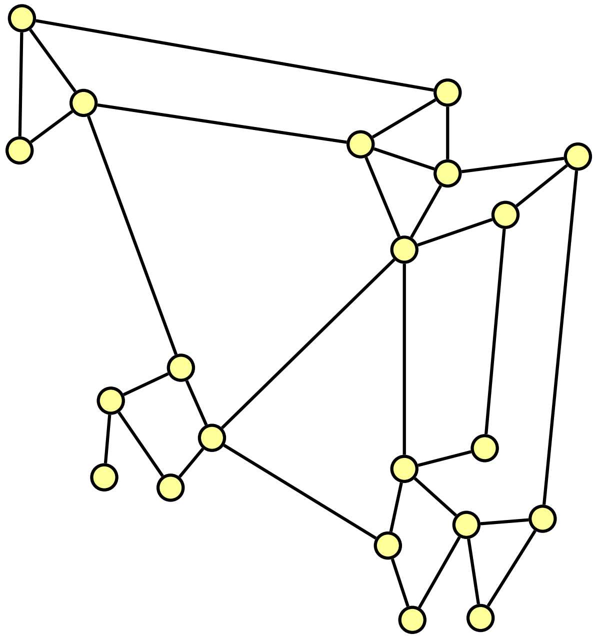
Computes: Tree T with degree $O(\log^2 n) \cdot B_v$ at node $v \in V$ and diameter $O(\log n) \cdot \Delta$

[Arkin et al. '02]
[Arkin et al. '03]

Approximation algorithms for Freeze-Tag in various topologies

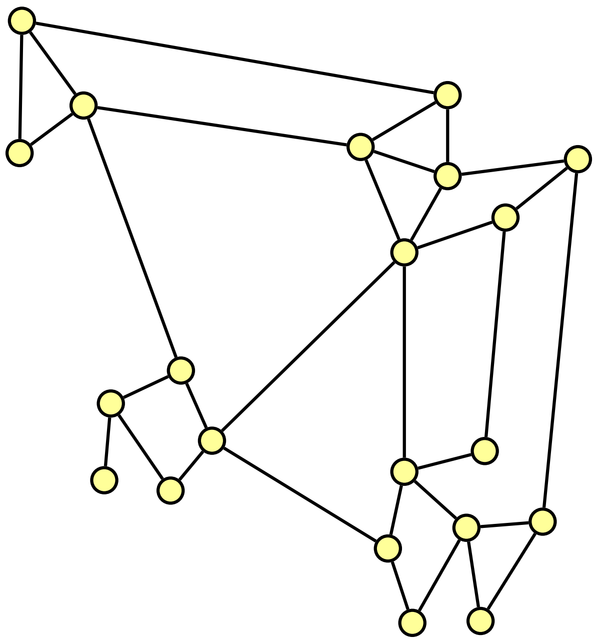
Obtain a $O(\log \Delta)$ approximation for general graphs with maximum degree Δ and metric lengths.

Algorithm: Intuition



Given: Input graph G and degree bound B.

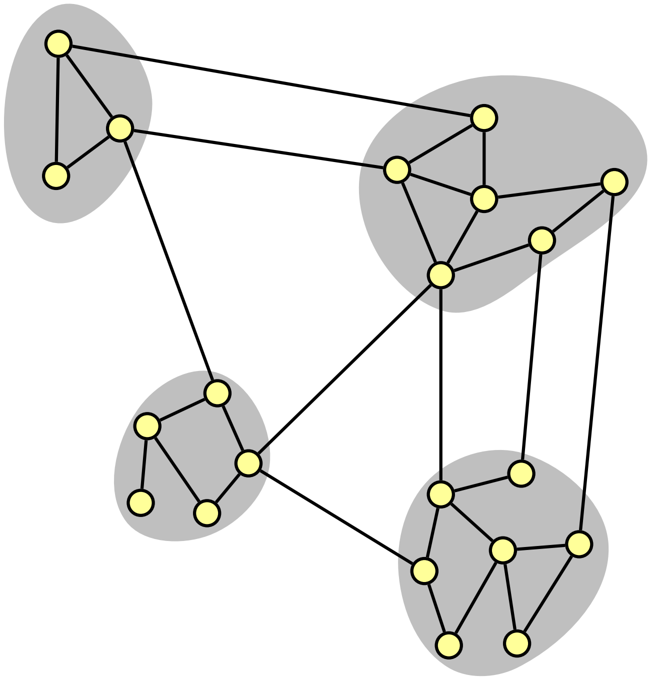
Algorithm: Intuition



Given: Input graph G and degree bound B.

1) Partition G into low-diameter components

Algorithm: Intuition

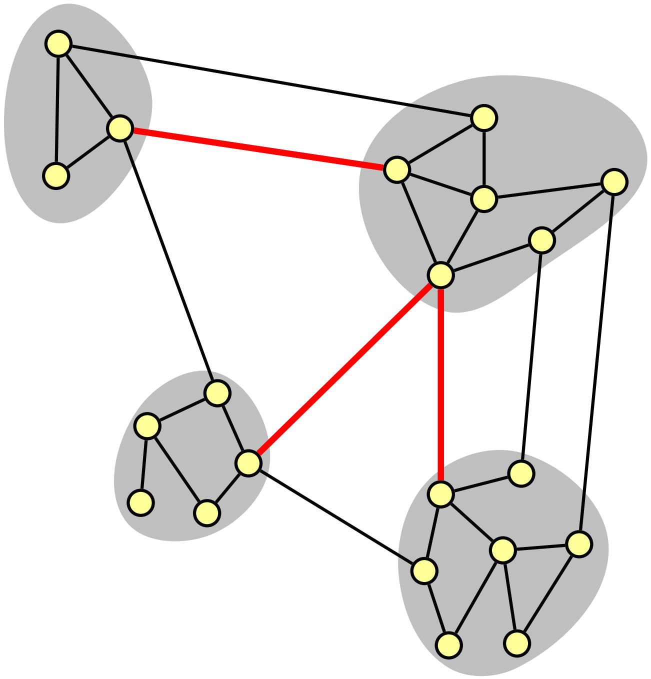


Given: Input graph G and degree bound B.

1) Partition G into low-diameter components

2) [Global Tree] Connect components with low-diameter tree

Algorithm: Intuition

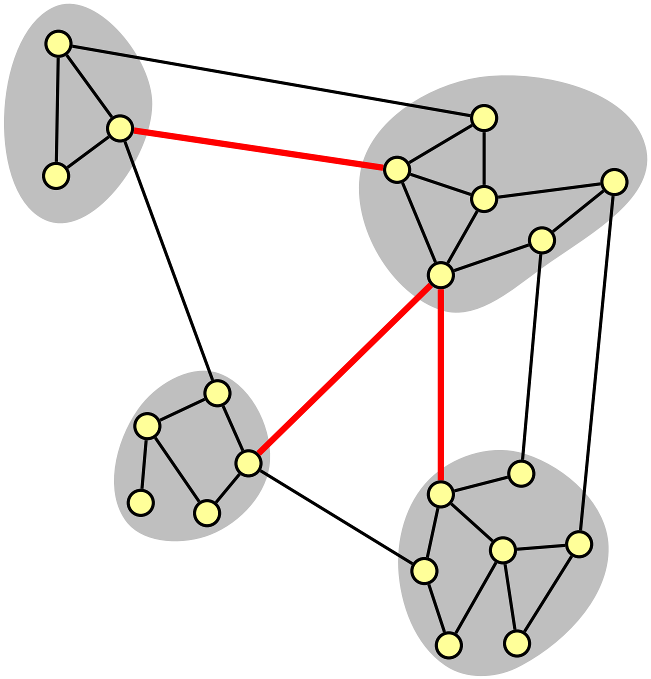


Given: Input graph G and degree bound B.

1) Partition G into low-diameter components

2) [Global Tree] Connect components with low-diameter tree

Algorithm: Intuition



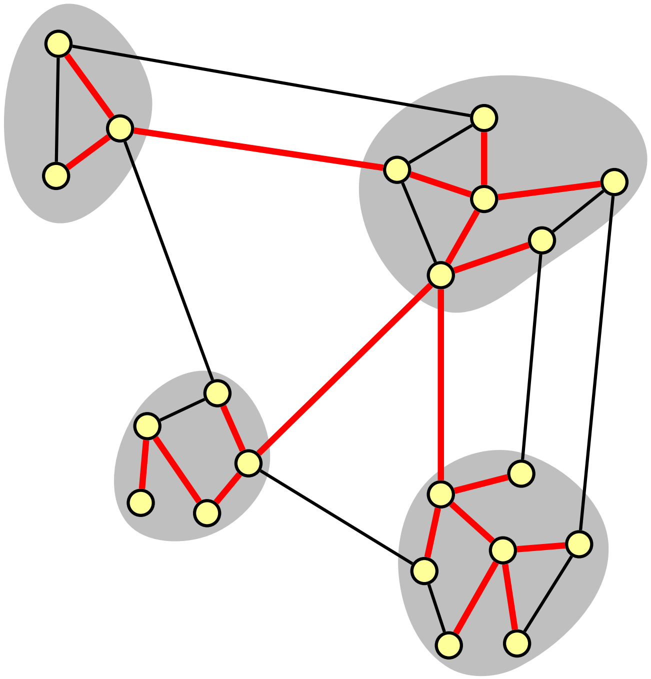
Given: Input graph G and degree bound B.

1) Partition G into low-diameter components

2) [Global Tree] Connect components with low-diameter tree

3) [Local Trees] For all components find low-diameter trees with max-degree B

Algorithm: Intuition



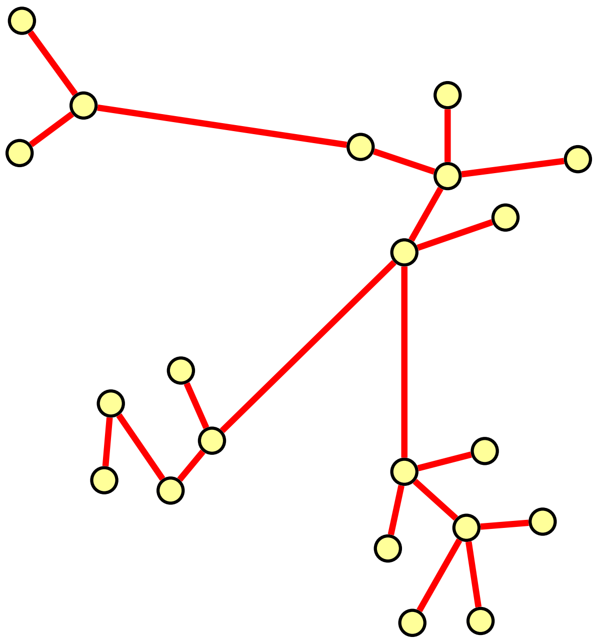
Given: Input graph G and degree bound B.

1) Partition G into low-diameter components

2) [Global Tree] Connect components with low-diameter tree

3) [Local Trees] For all components find low-diameter trees with max-degree B

Algorithm: Intuition



Given: Input graph G and degree bound B.

1) Partition G into low-diameter components

2) [Global Tree] Connect components with low-diameter tree

3) [Local Trees] For all components find low-diameter trees with max-degree B

Ensure by construction: Final tree has max-degree B.
Diameter proof bounds short and long edges independently.

Algorithm: Preliminaries

- Assume for rest of talk that optimum diameter Δ is known. Reasonable assumption since

$$\Delta \in [\max_{e \in E} l_e, n \cdot \max_{e \in E} l_e]$$

Can do binary search on this interval!

Algorithm: Preliminaries

- Assume for rest of talk that optimum diameter Δ is known. Reasonable assumption since

$$\Delta \in [\max_{e \in E} l_e, n \cdot \max_{e \in E} l_e]$$

Can do binary search on this interval!

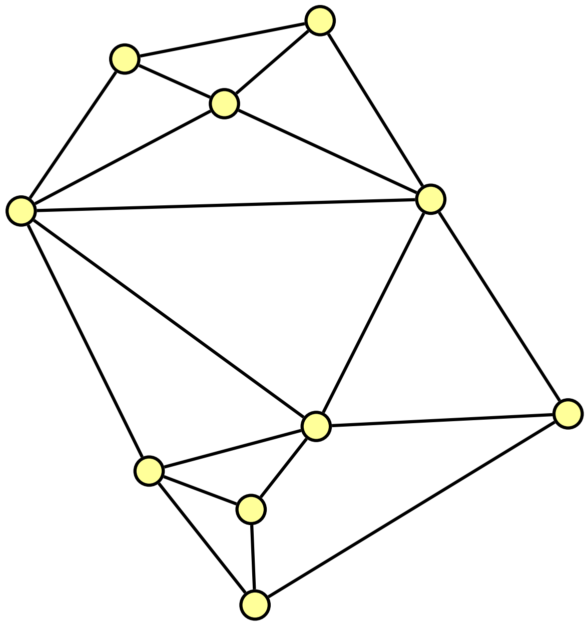
- Algorithm picks threshold α and computes (set,center) pairs

$$\{(V_1, v_1), \ldots, (V_l, v_l)\}$$

such that

1. $V = V_1 \cup \ldots \cup V_l$, and
2. For all i: $v_i \in V_i$ and $\texttt{dist}_l(v_i, u) \leq 3\alpha$ for all $u \in V_i$

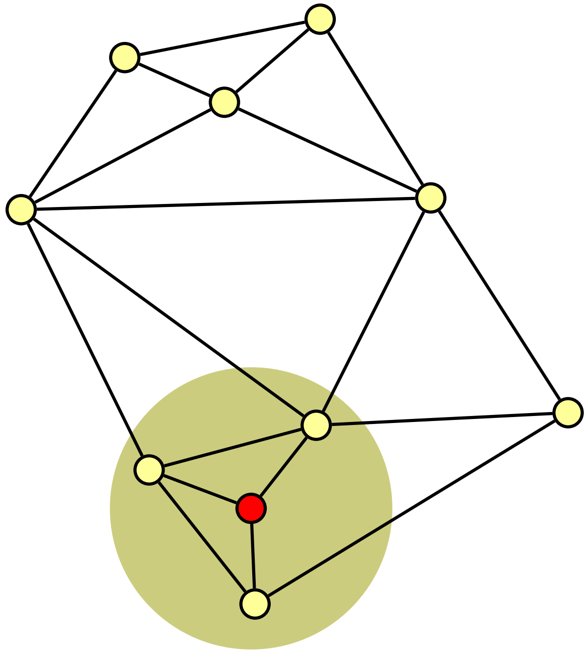
Algorithm: Partitioning



Algorithm picks centers iteratively.

covered   inactive   center

Algorithm: Partitioning

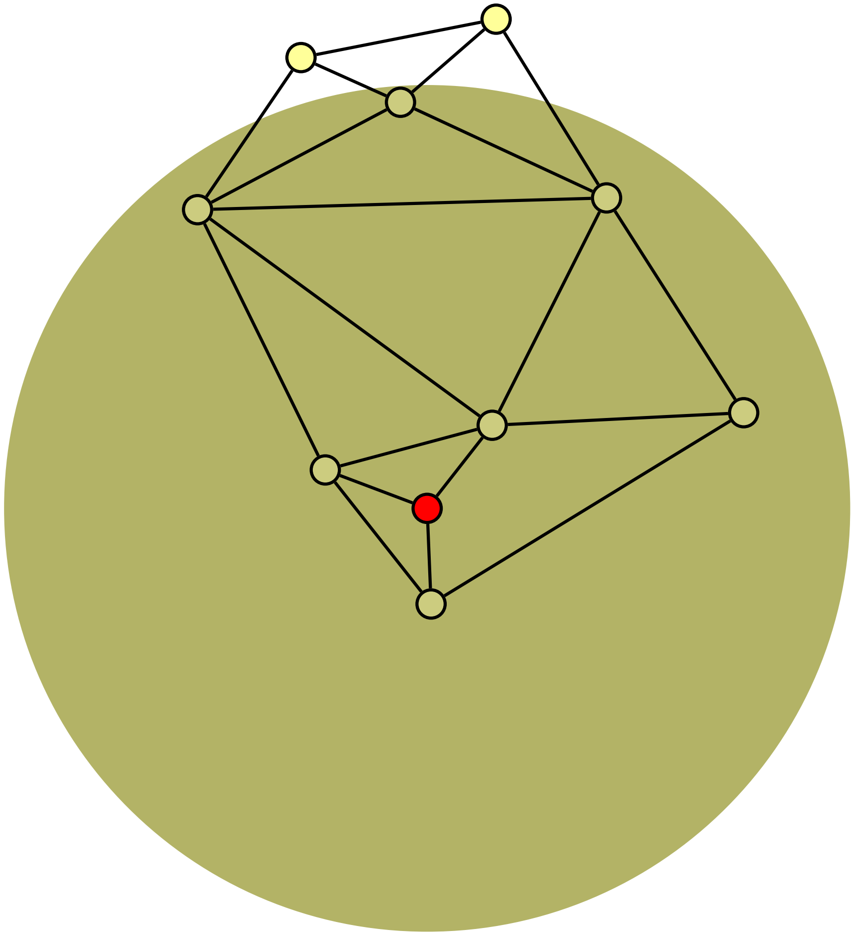


Algorithm picks centers iteratively.

Pick active node that has most active nodes within ball of radius α.

covered inactive center

Algorithm: Partitioning



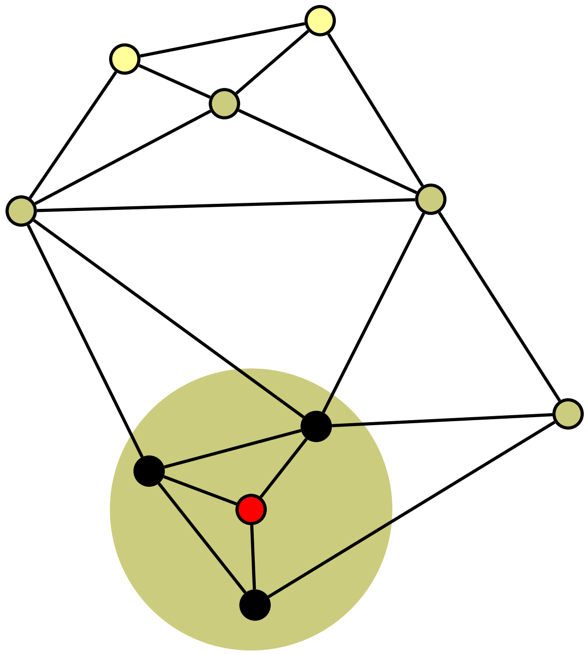
Algorithm picks centers iteratively.

Pick active node that has most active nodes within ball of radius α.

Mark all nodes within 3α of new center covered.

covered inactive center

Algorithm: Partitioning



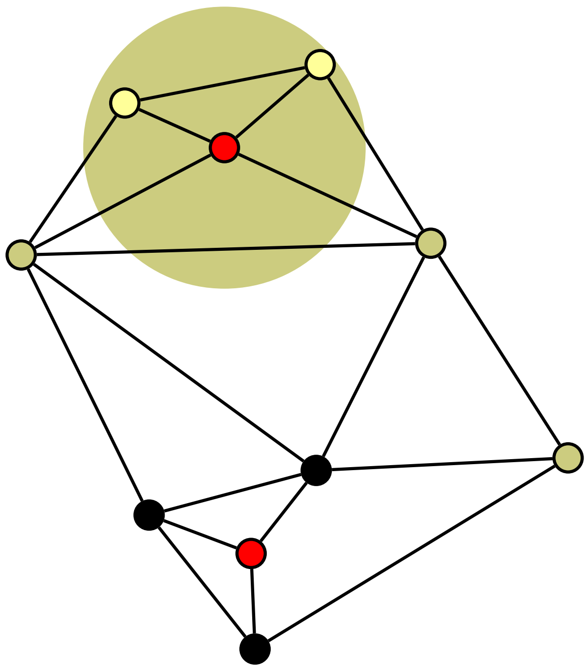
Algorithm picks centers iteratively.

Pick active node that has most active nodes within ball of radius α.

Mark all nodes within 3α of new center covered.

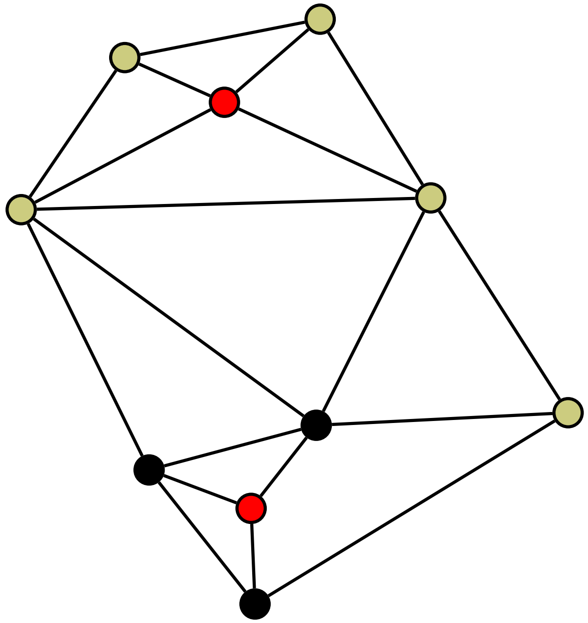
Mark all nodes within α of new center inactive.

Algorithm: Partitioning



- Algorithm picks centers iteratively.
- → Pick active node that has most active nodes within ball of radius α.
- Mark all nodes within 3α of new center covered.
- Mark all nodes within α of new center inactive.

Algorithm: Partitioning



Algorithm picks centers iteratively.

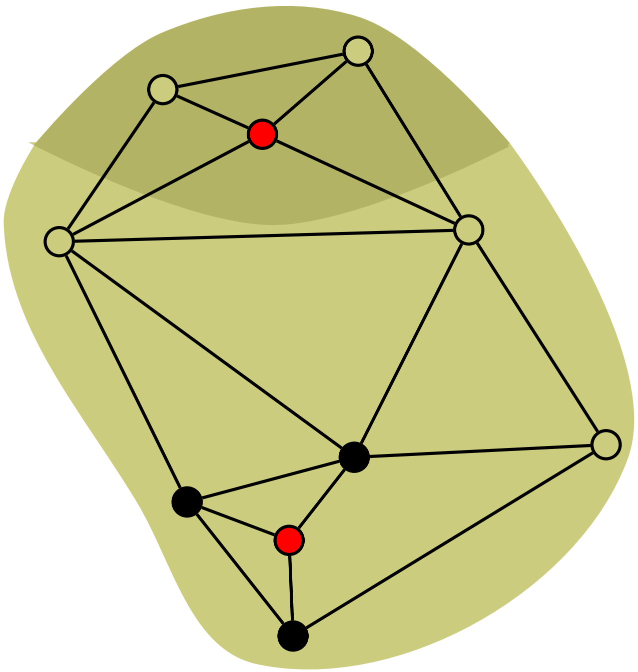
Pick active node that has most active nodes within ball of radius α.

→ Mark all nodes within 3α of new center covered.

Mark all nodes within α of new center inactive.

covered inactive center

Algorithm: Partitioning



Algorithm picks centers iteratively.

Pick active node that has most active nodes within ball of radius α.

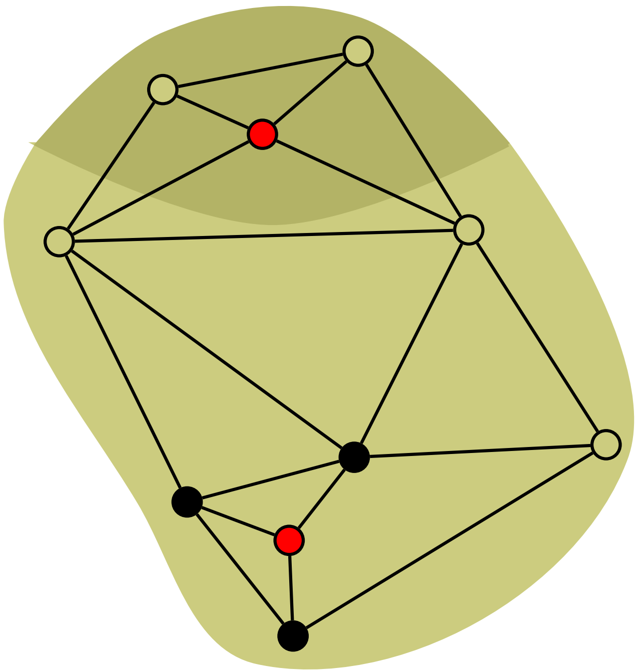
Mark all nodes within 3α of new center covered.

Mark all nodes within α of new center inactive.

Stop when all nodes are covered.

covered inactive center

Algorithm: Partitioning



Algorithm picks centers iteratively.

Pick active node that has most active nodes within ball of radius α.

Mark all nodes within 3α of new center covered.

Mark all nodes within α of new center inactive.

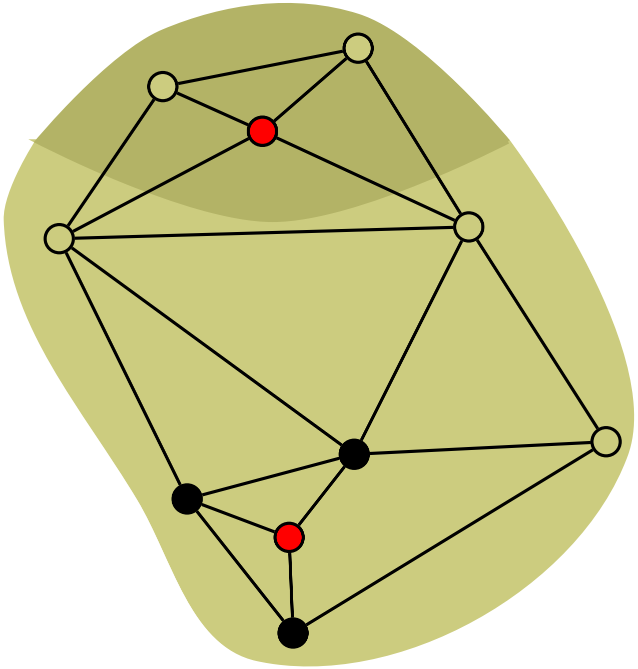
Stop when all nodes are covered.

Result:

Partition: $V_1, \ldots, V_l$

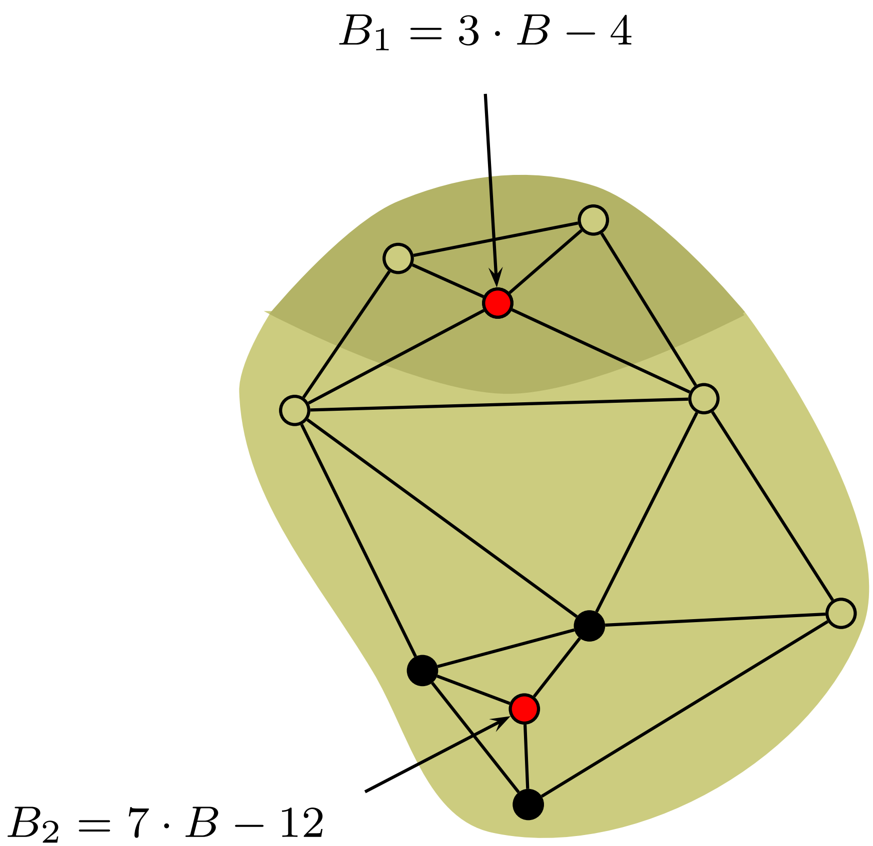
Centers: $v_1, \ldots, v_l$

Algorithm: Global tree



Goal: Set up global instance on center nodes.

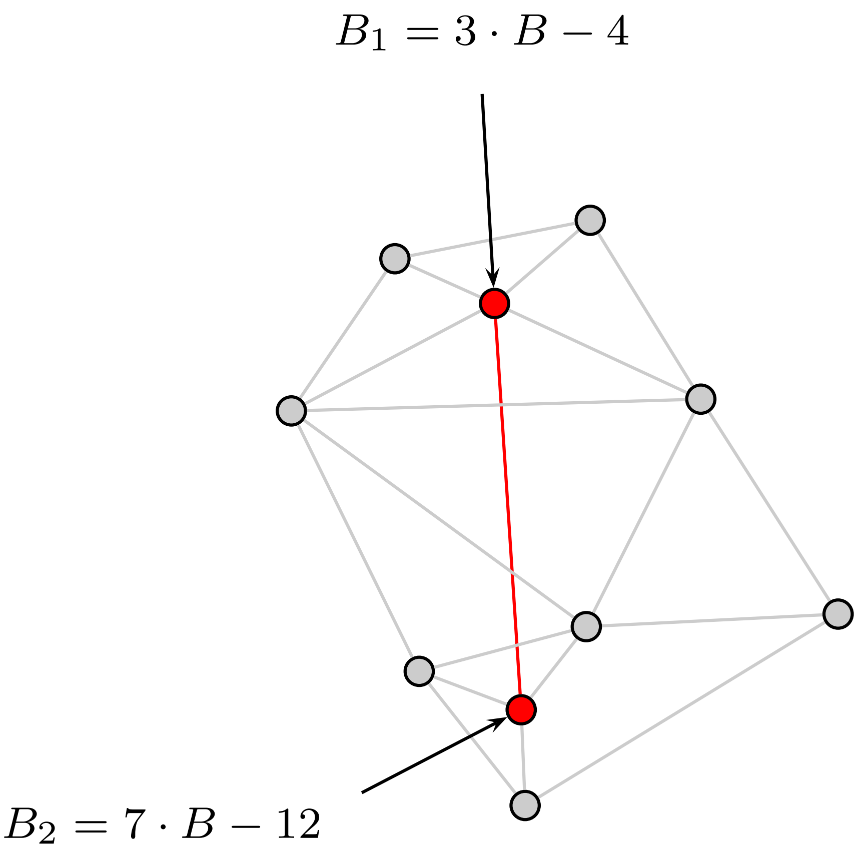
Algorithm: Global tree



Goal: Set up global instance on center nodes.

Set new degree-bounds on center nodes.

Algorithm: Global tree

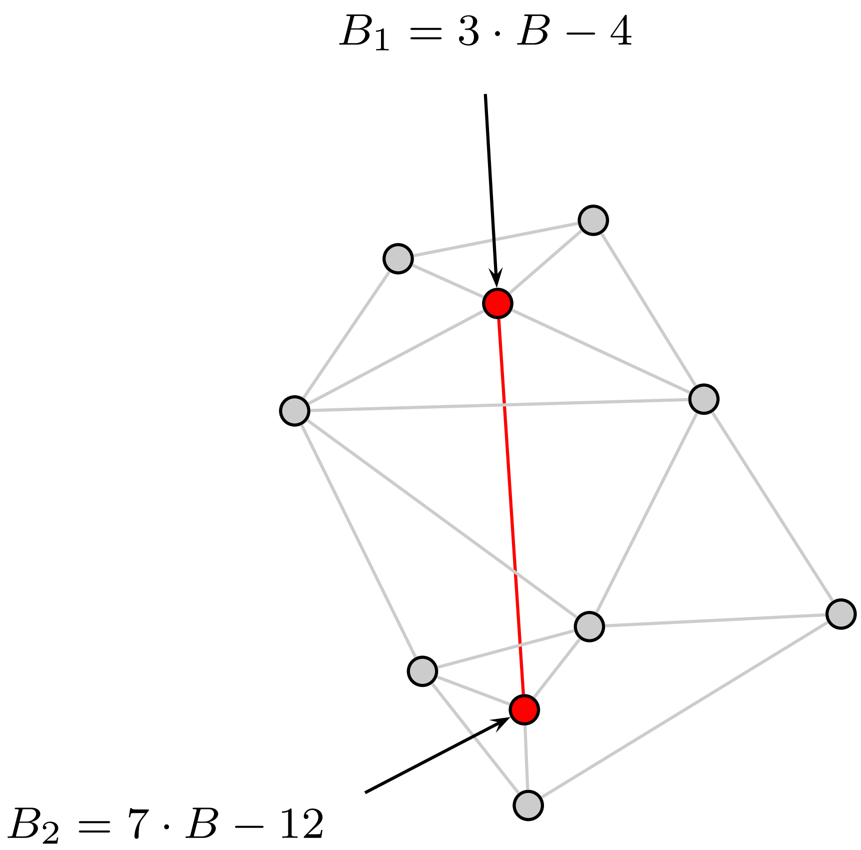


Goal: Set up global instance on center nodes.

Set new degree-bounds on center nodes.

Consider complete graph on center nodes.

Algorithm: Global tree



Goal: Set up global instance on center nodes.

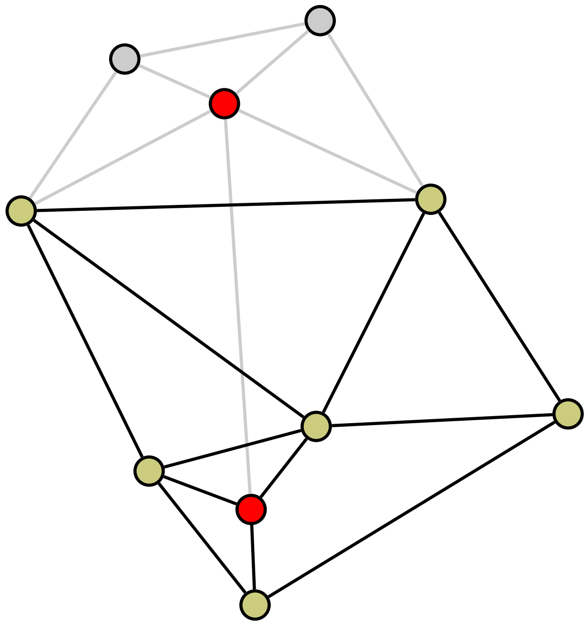
Set new degree-bounds on center nodes.

Consider complete graph on center nodes.

Global Tree algorithm:

1. Order center nodes by non-increasing degree-bounds $v_1, v_2, \ldots$
2. v_1 is root of global tree
3. Consider centers one by one in that order
4. Always connect next center to earliest node in list whose degree-bound is not yet exhausted

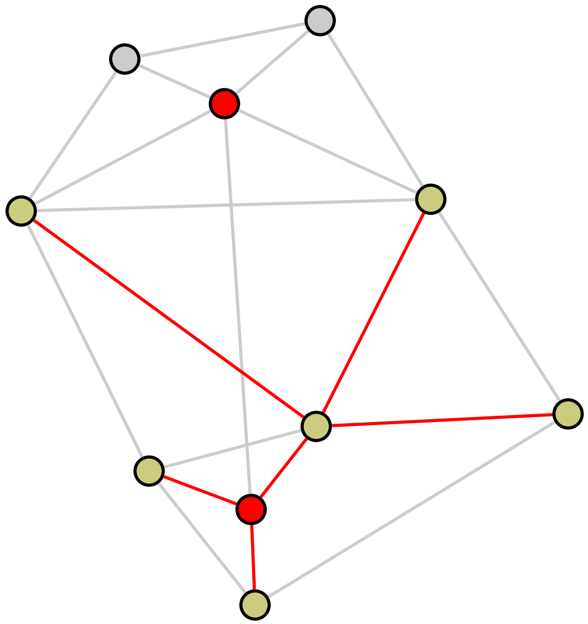
Algorithm: Local Trees



For each set V_i in partition:

1. Consider complete graph on V_i
2. Find complete $B-1$-ary tree on of $G[V_i]$ rooted at v_i

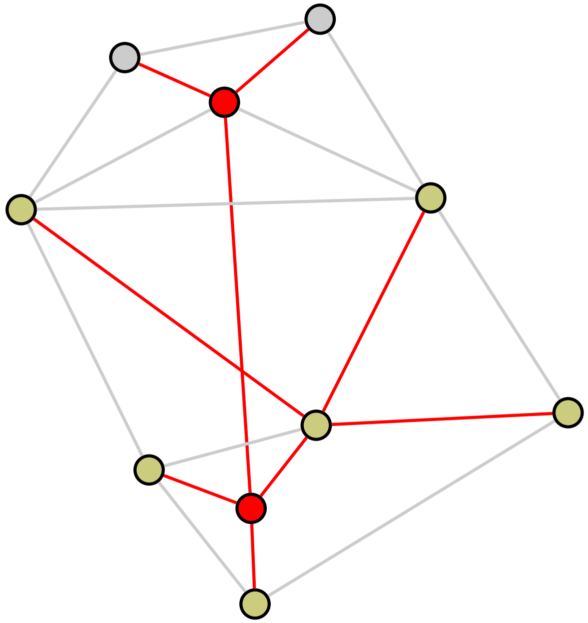
Algorithm: Local Trees



For each set V_i in partition:

1. Consider complete graph on V_i
2. Find complete $B-1$-ary tree on of $G[V_i]$ rooted at v_i

Algorithm: Local Trees

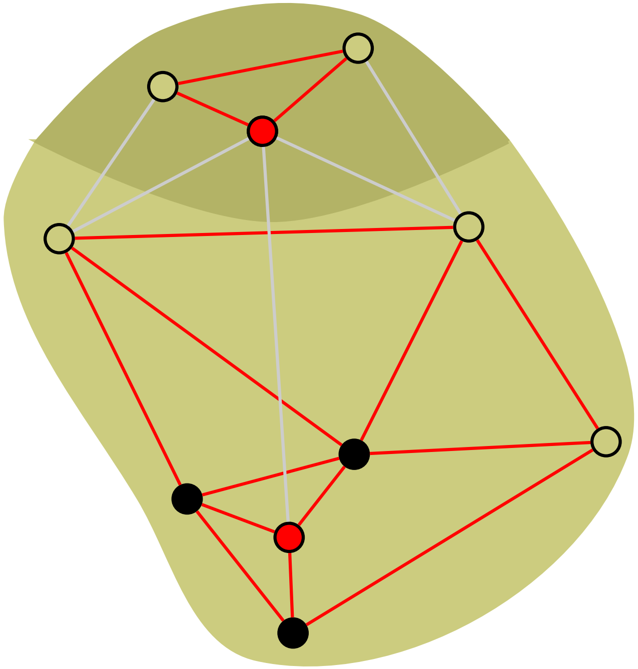


For each set V_i in partition:

1. Consider complete graph on V_i
2. Find complete $B-1$-ary tree on of $G[V_i]$ rooted at v_i

Merge local and global trees.

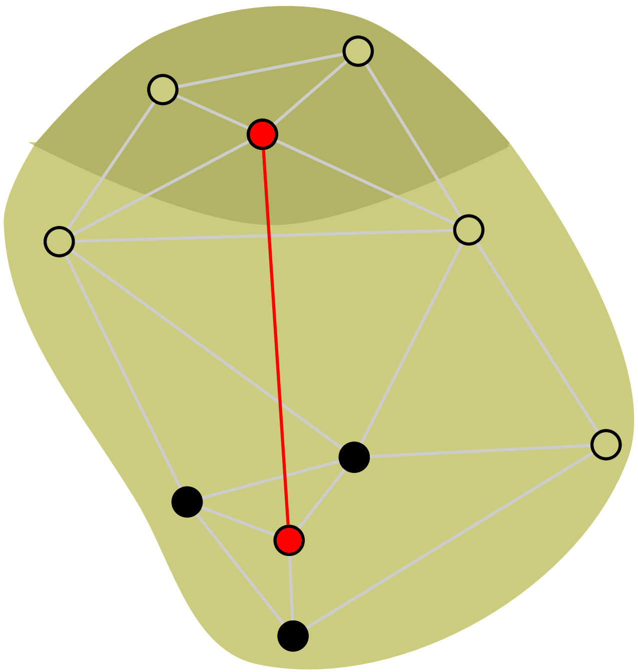
Analysis: Short and Long Edges



Short edges connect nodes in the same set of the partition.

Their length is at most 6α.

Analysis: Short and Long Edges



Short edges connect nodes in the same set of the partition.

Their length is at most 6α.

Long edges connect center nodes.

Their length is at most Δ

Analysis: Outline

- Show that any root,leaf path in our tree has
  1. $O(\sqrt{\log_B n})$ long edges, and
  2. $O(\log_B n)$ short edges.

Analysis: Outline

- Show that any root,leaf path in our tree has
  1. $O(\sqrt{\log_B n})$ long edges, and
  2. $O(\log_B n)$ short edges.
- This implies: Length of any root,leaf path is at most

$$O(\sqrt{\log_B n}) \cdot \Delta + O(\log_B n) \cdot \alpha$$

Analysis: Outline

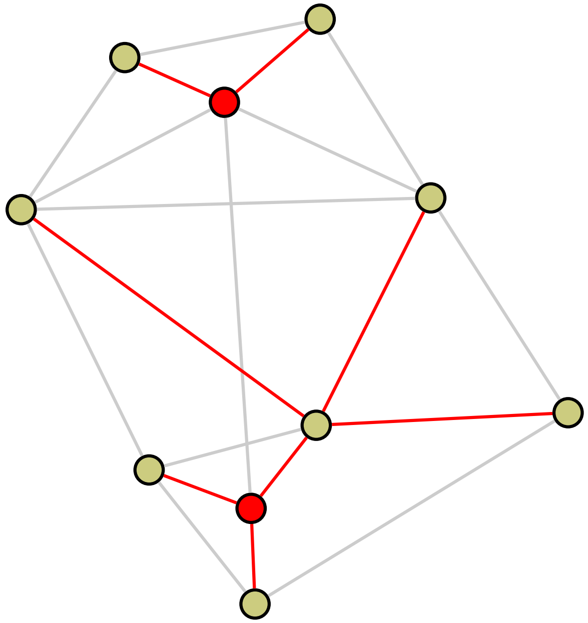
- Show that any root,leaf path in our tree has
  1. $O(\sqrt{\log_B n})$ long edges, and
  2. $O(\log_B n)$ short edges.
- This implies: Length of any root,leaf path is at most

$$O(\sqrt{\log_B n}) \cdot \Delta + O(\log_B n) \cdot \alpha$$

- With $\alpha = \Delta/\sqrt{\log_B n}$: Length of root,leaf path is bouded by

$$O(\sqrt{\log_B n}) \cdot \Delta$$

Analysis: Degree

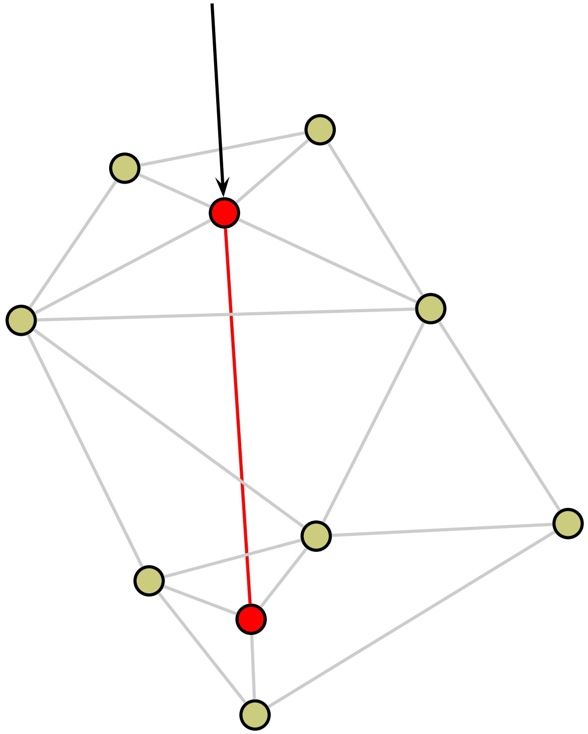


Short edges:

Each node has at most B of these incident to it.

Analysis: Degree

$B_1 = 3 \cdot B - 4$



Short edges:

Each node has at most B of these incident to it.

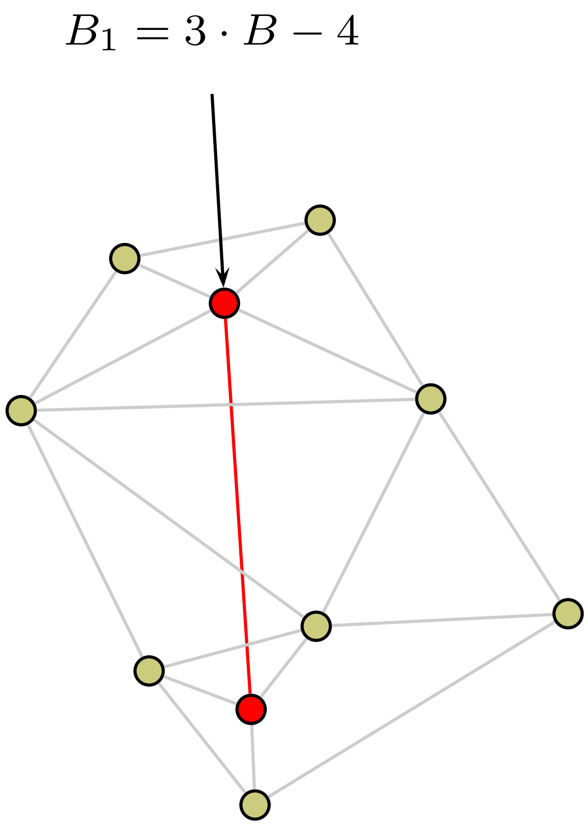
Long edges:

Center i has at most B_i incident to it.

Example: Total available degree in V_1 is $3 \cdot B$.

Short edges consume 4. Leaves us with B_1...

Analysis: Degree



Short edges:

Each node has at most B of these incident to it.

Long edges:

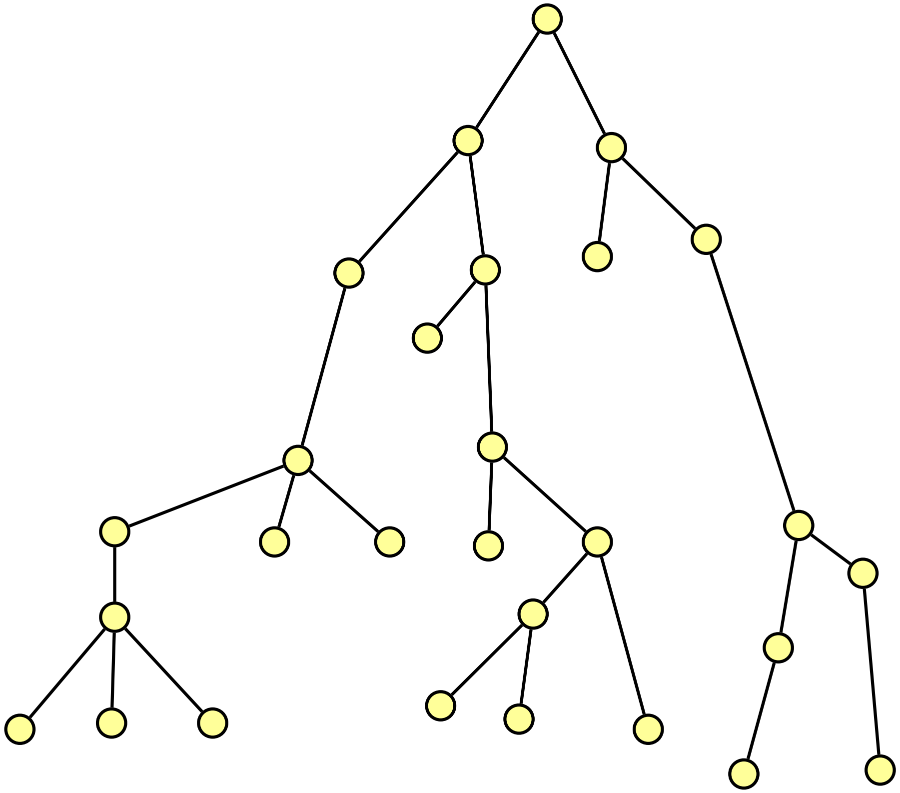
Center i has at most B_i incident to it.

Example: Total available degree in V_1 is $3 \cdot B$.

Short edges consume 4. Leaves us with B_1...

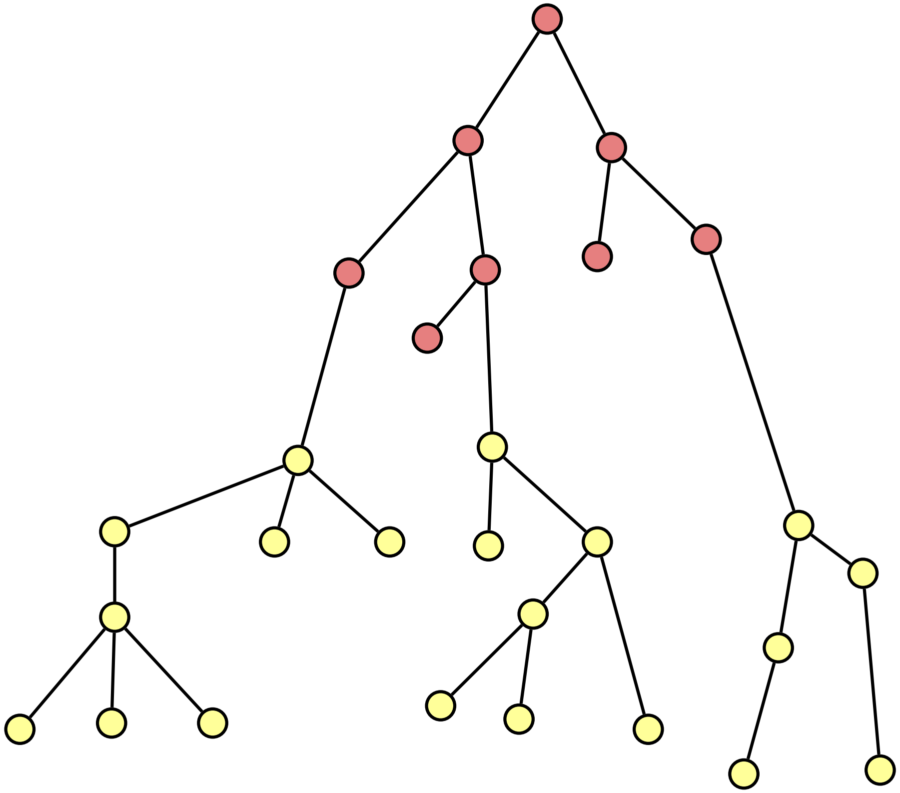
Redistribute long edges incident to v_i over V_i!

Analysis: Long Edges



Partition an optimum solution T^*:

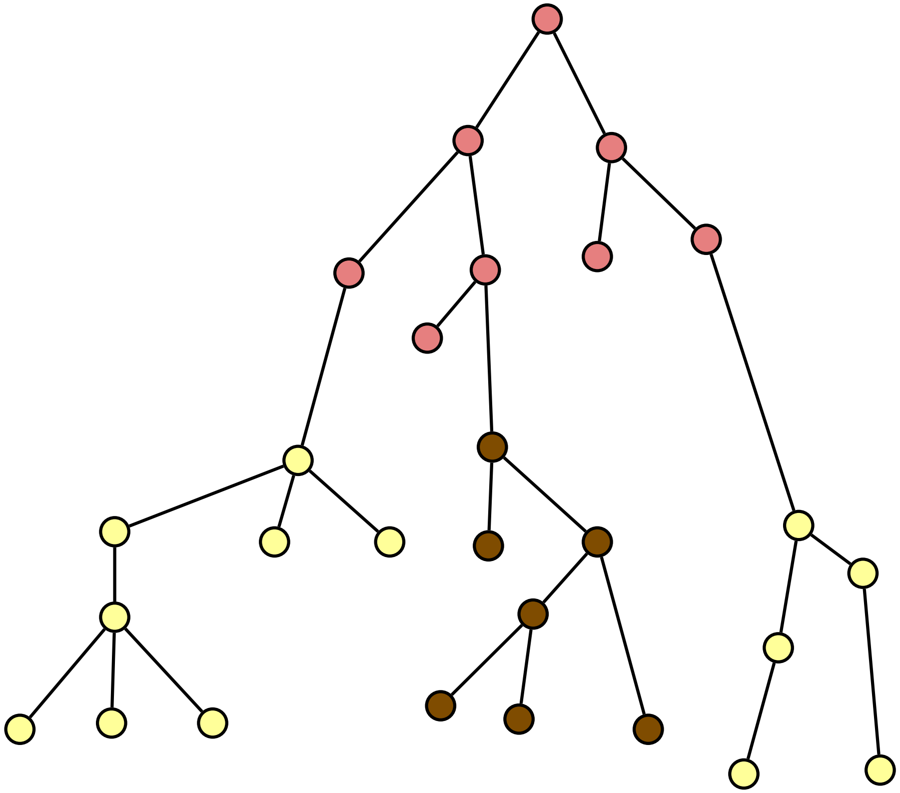
Analysis: Long Edges



Partition an optimum solution T^*:

Start with root node and cover all nodes at distance α in T^* from it.

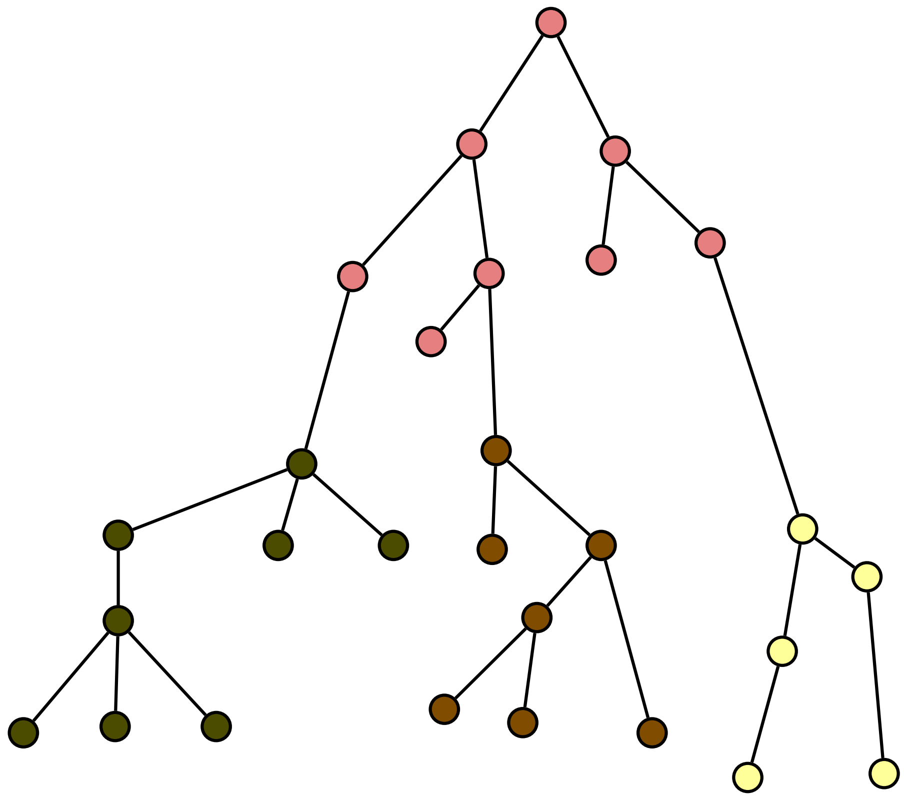
Analysis: Long Edges



Partition an optimum solution T^*:

Start with root node and cover all nodes at distance α in T^* from it.
Take highest uncovered node as next root and cover all nodes at distance α in T^* from it.

Analysis: Long Edges



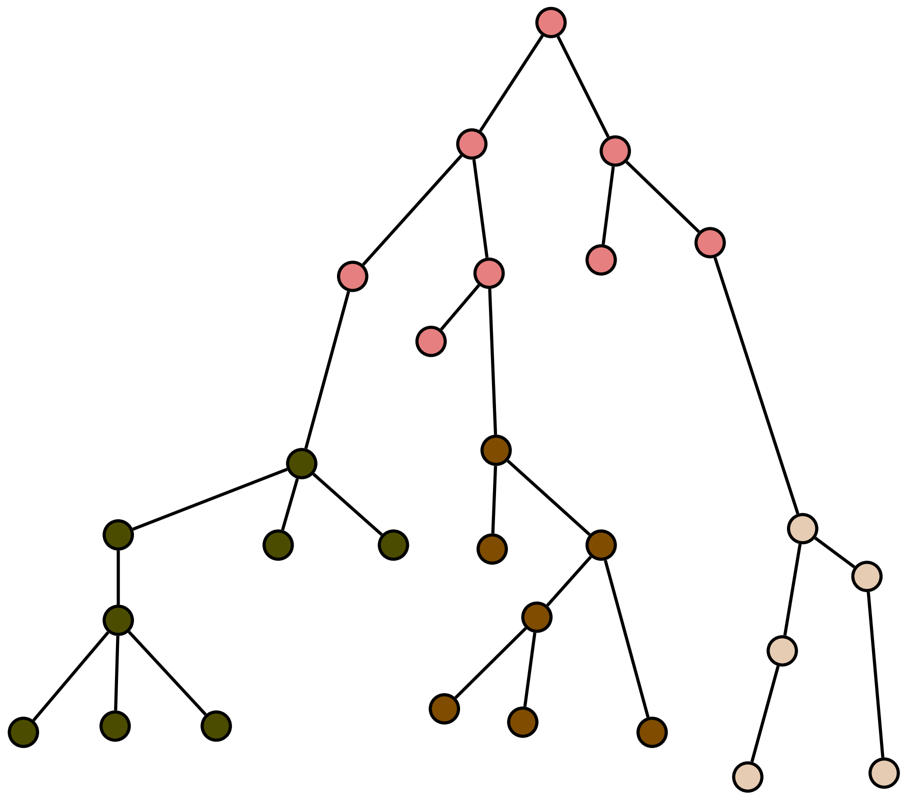
Partition an optimum solution T^*:

Start with root node and cover all nodes at distance α in T^* from it.

Take highest uncovered node as next root and cover all nodes at distance α in T^* from it.

Repeat!

Analysis: Long Edges



Partition an optimum solution T^*:

Start with root node and cover all nodes at distance α in T^* from it.

Take highest uncovered node as next root and cover all nodes at distance α in T^* from it.

Repeat!

Process leads to partition $V_1^*, \ldots, V_k^*$ with centers $v_1^*, \ldots, v_k^*$.

Analysis: Long Edges

Observation: $|V_i^*|$ induces connected piece of T^*.
Hence: V_i^* can have at most

$$B_i^* = |V_i^*| \cdot B - 2(|V_i^*| - 1)$$

children in T^*.

Analysis: Long Edges

Observation: $|V_i^*|$ induces connected piece of T^*.
Hence: V_i^* can have at most

$$B_i^* = |V_i^*| \cdot B - 2(|V_i^*| - 1)$$

children in T^*.

Have: Two partitions

$$\{V_i\}_{1 \leq i \leq l} \text{ and } \{V_i^*\}_{1 \leq i \leq k}$$

W.l.o.g.: $B_1 \geq \ldots \geq B_l$ and $B_1^* \geq \ldots \geq B_k^*$

Analysis: Long Edges

Have: Two partitions

$$\{V_i\}_{1\le i\le l} \text{ and } \{V_i^*\}_{1\le i\le k}$$

W.l.o.g.: $B_1 \geq \ldots \geq B_l$ and $B_1^* \geq \ldots \geq B_k^*$

Lemma: We must have $l \leq k$ and for all $1 \leq i \leq l$

$$\sum_{j=1}^{i} B_i^* \leq \sum_{j=1}^{i} B_i$$

Analysis: Long Edges

Have: Two partitions

$$\{V_i\}_{1\le i\le l} \text{ and } \{V_i^*\}_{1\le i\le k}$$

W.l.o.g.: $B_1 \ge \ldots \ge B_l$ and $B_1^* \ge \ldots \ge B_k^*$

Lemma: We must have $l \le k$ and for all $1 \le i \le l$

$$\sum_{j=1}^{i} B_i^* \le \sum_{j=1}^{i} B_i$$

Proof idea: Uses the existence of $\{V_i^*\}_{1\le i\le k}$ and the fact that the sets in $\{V_i\}_{1\le i\le l}$ have radius 3α.

Analysis: Long Edges

Have: Two partitions

$$\{V_i\}_{1\le i\le l} \text{ and } \{V_i^*\}_{1\le i\le k}$$

W.l.o.g.: $B_1 \geq \ldots \geq B_l$ and $B_1^* \geq \ldots \geq B_k^*$

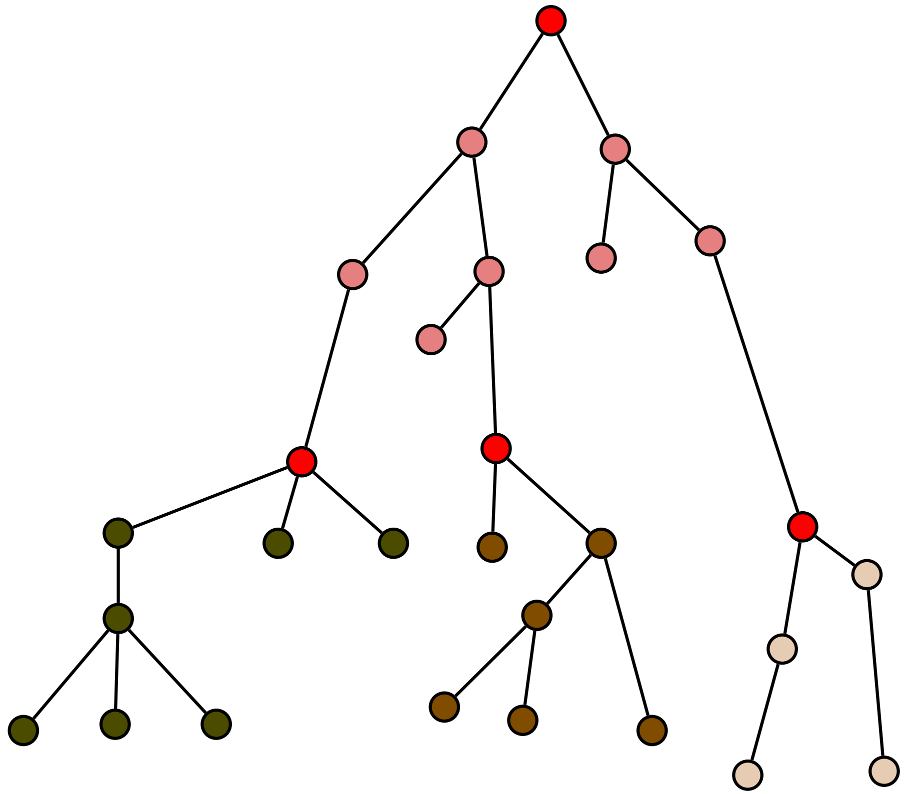
Lemma: We must have $l \leq k$ and for all $1 \leq i \leq l$

$$\sum_{j=1}^{i} B_i^* \leq \sum_{j=1}^{i} B_i$$

Proof idea: Uses the existence of $\{V_i^*\}_{1\le i\le k}$ and the fact that the sets in $\{V_i\}_{1\le i\le l}$ have radius 3α.

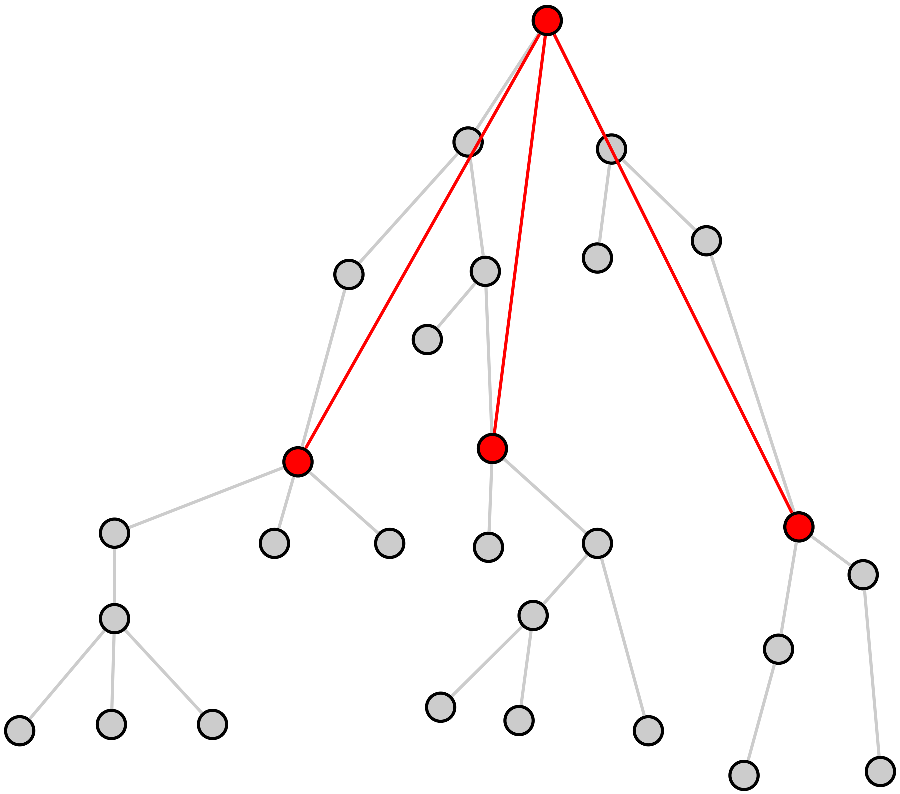
What does this imply?

Analysis: Long Edges



Recall partition of optimum solution T^*

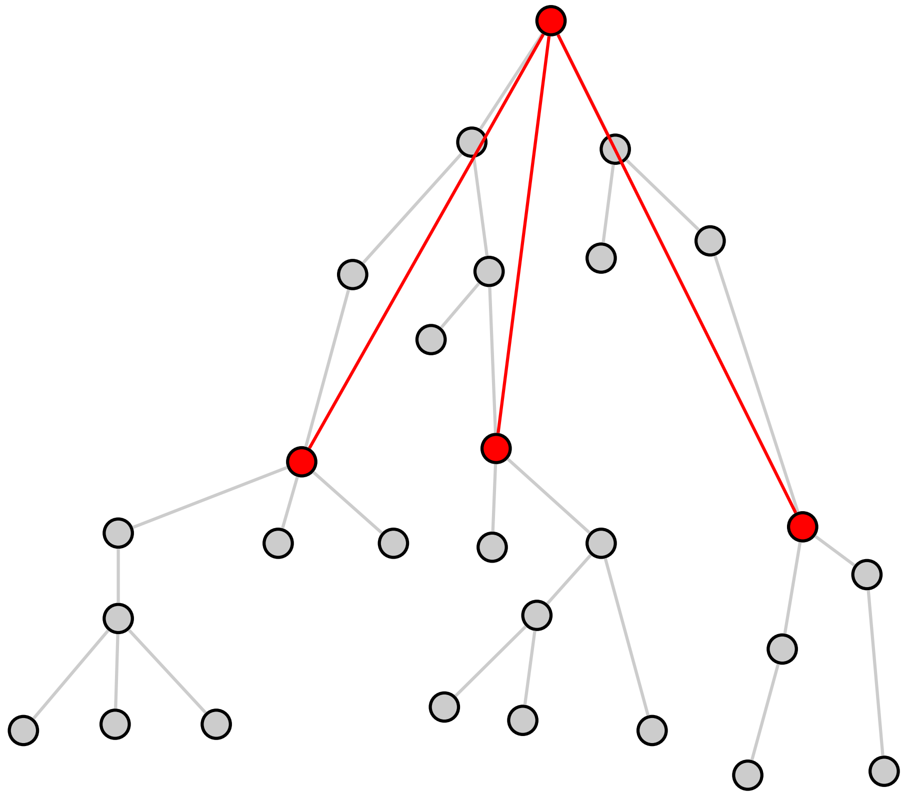
Analysis: Long Edges



Recall partition of optimum solution T^*

Consider tree T^g induced by center nodes in T^*

Analysis: Long Edges



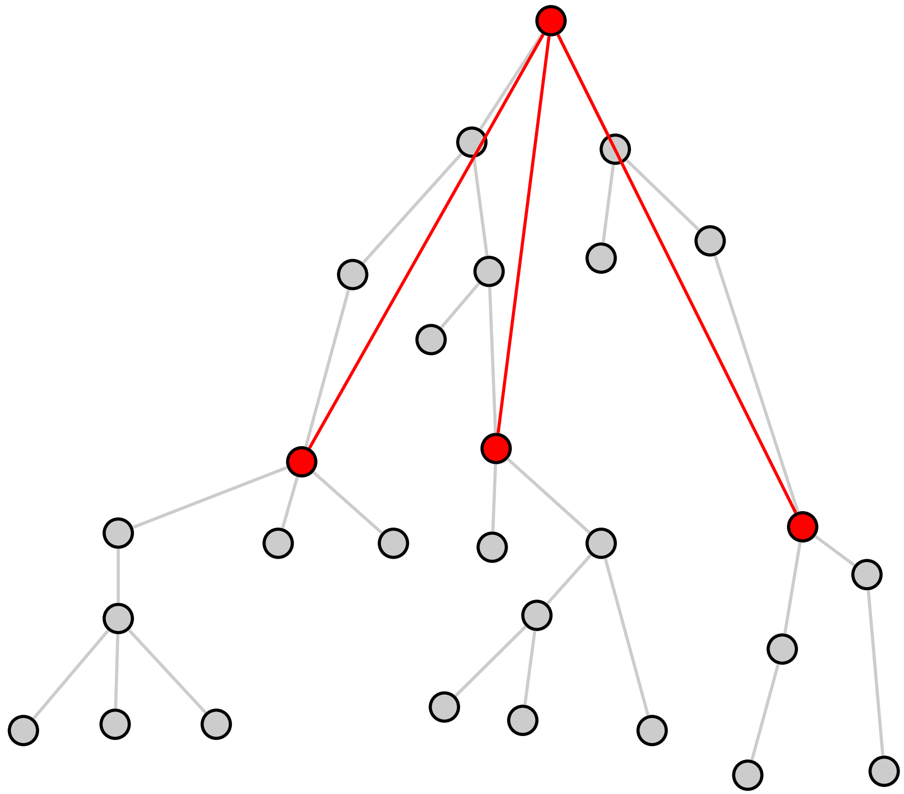
Recall partition of optimum solution T^*

Consider tree T^g induced by center nodes in T^*

Lemma: $\sum_i B_i^* \leq \sum_i B_i$

Implication: Global tree is at most as high as T^g

Analysis: Long Edges



Recall partition of optimum solution T^*

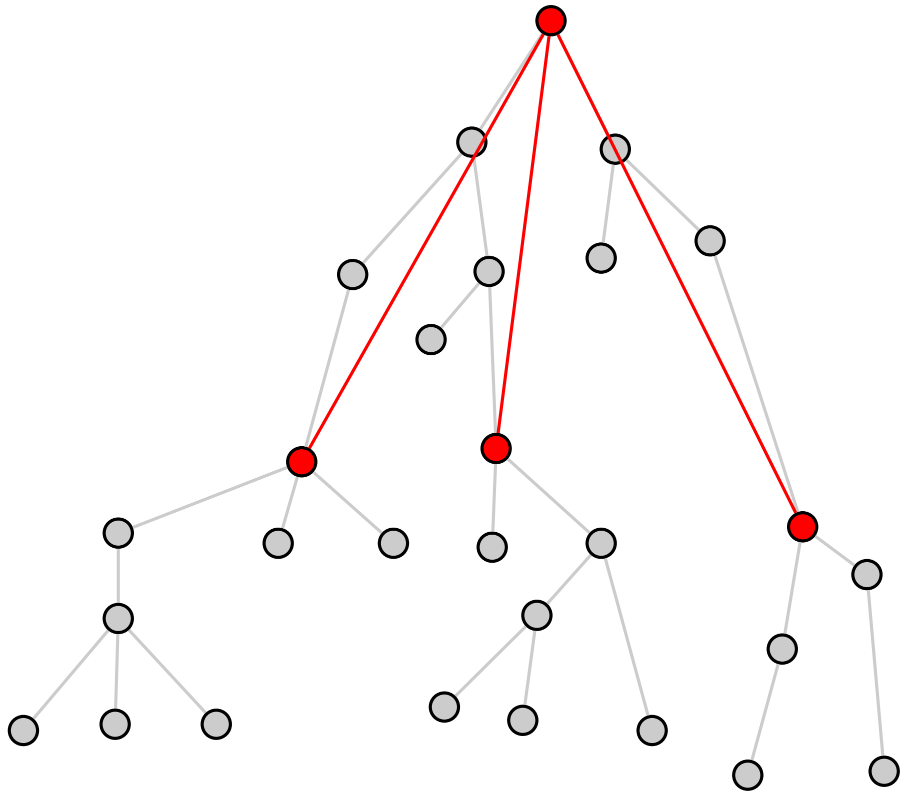
Consider tree T^g induced by center nodes in T^*

Lemma: $\sum_i B_i^* \leq \sum_i B_i$

Implication: Global tree is at most as high as T^g

Observation: Edges in T^g have length at least α!

Analysis: Long Edges



Recall partition of optimum solution T^*

Consider tree T^g induced by center nodes in T^*

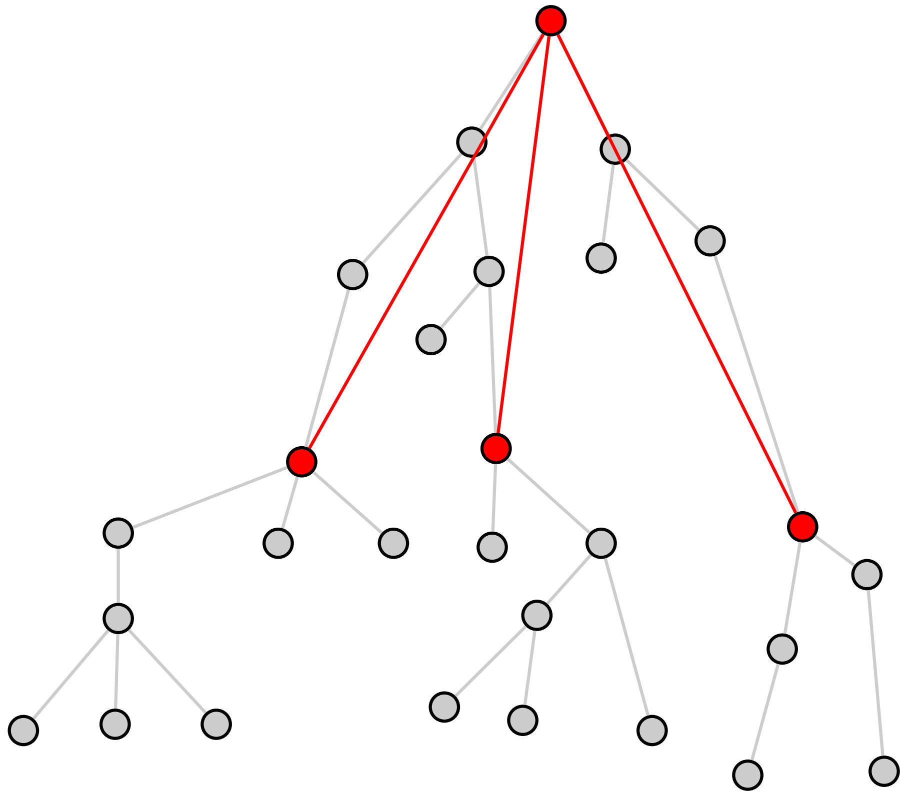
Lemma: $\sum_i B_i^* \le \sum_i B_i$

Implication: Global tree is at most as high as T^g

Observation: Edges in T^g have length at least α!

Hence: Height of T^g is at most Δ/α

Analysis: Long Edges



Recall partition of optimum solution T^*

Consider tree T^g induced by center nodes in T^*

Lemma: $\sum_i B_i^* \leq \sum_i B_i$

Implication: Global tree is at most as high as T^g

Observation: Edges in T^g have length at least α!

Hence: Height of T^g is at most Δ/α

Lemma: With $\alpha = \Delta/\sqrt{\log_B n}$ we must have that the our global tree has height $O(\sqrt{\log_B n})$.

Analysis: Outline

- Show that any root,leaf path in our tree has
  1. $O(\sqrt{\log_B n})$ long edges, and
  2. $O(\log_B n)$ short edges.
- This implies: Length of any root,leaf path is at most

$$O(\sqrt{\log_B n}) \cdot \Delta + O(\log_B n) \cdot \alpha$$

- With $\alpha = \Delta / \sqrt{\log_B n}$: Length of root,leaf path is bouded by

$$O(\sqrt{\log_B n}) \cdot \Delta$$

Analysis: Outline

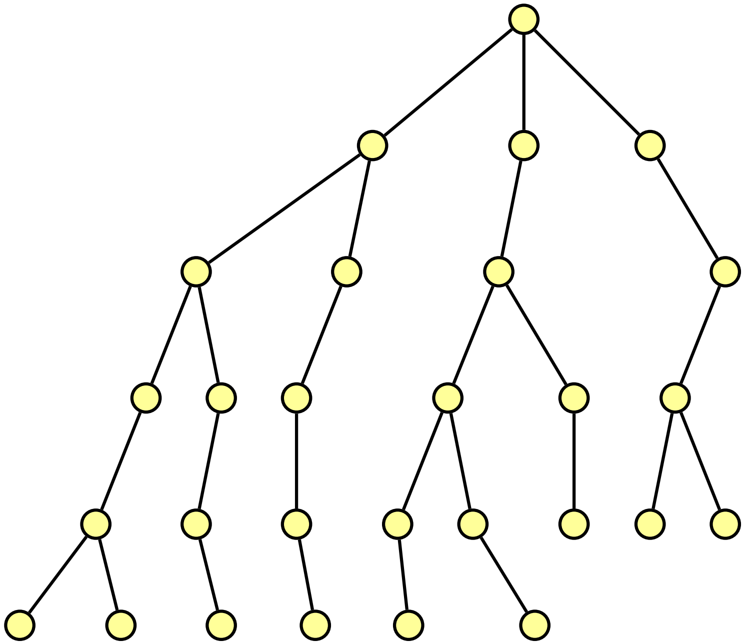
- Show that any root,leaf path in our tree has
  1. ~~$O(\sqrt{\log_B n})$ long edges, and~~
  2. $O(\log_B n)$ short edges.
- This implies: Length of any root,leaf path is at most

$$O(\sqrt{\log_B n}) \cdot \Delta + O(\log_B n) \cdot \alpha$$

- With $\alpha = \Delta/\sqrt{\log_B n}$: Length of root,leaf path is bouded by

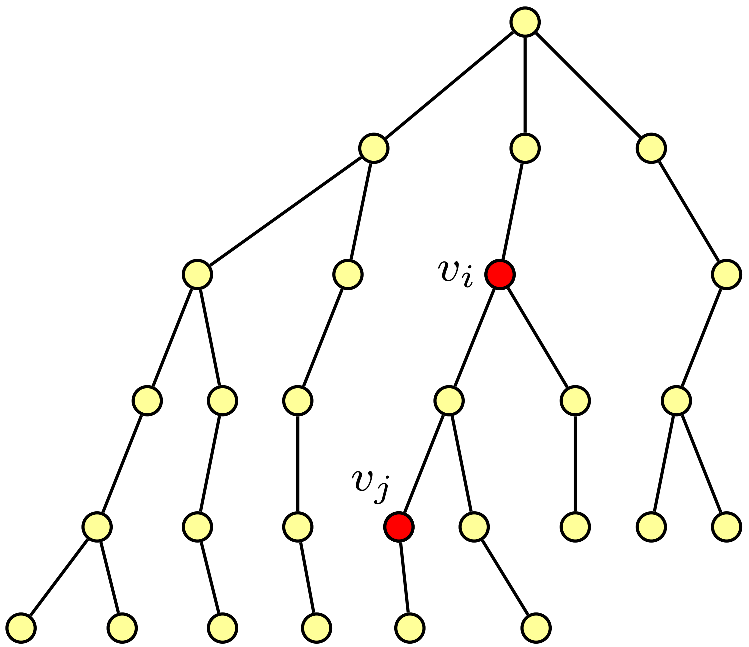
$$O(\sqrt{\log_B n}) \cdot \Delta$$

Analysis: Short Edges



Look at global tree on center nodes.

Analysis: Short Edges

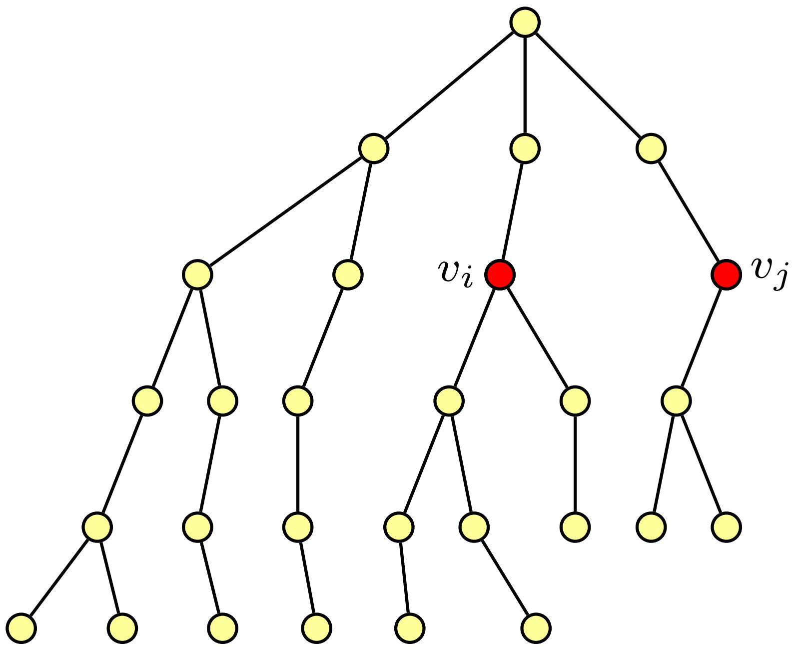


Look at global tree on center nodes.

Observe: Let v_i and v_j be to center nodes. Can organize global tree s.t.

1. $|V_i| > |V_j|$ if $\texttt{depth}(v_i) < \texttt{depth}(v_j)$

Analysis: Short Edges

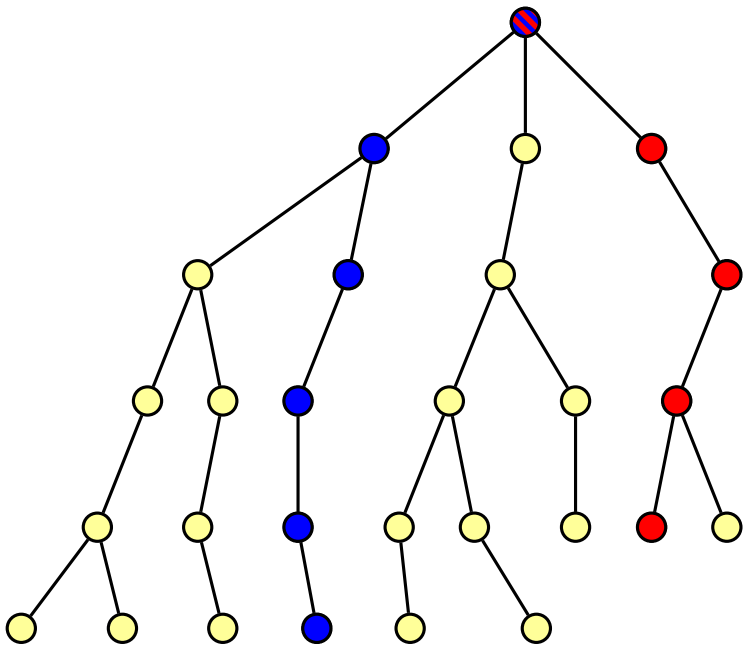


Look at global tree on center nodes.

Observe: Let v_i and v_j be to center nodes. Can organize global tree s.t.

1. $|V_i| > |V_j|$ if $\texttt{depth}(v_i) < \texttt{depth}(v_j)$
2. $|V_i| \geq |V_j|$ is $\texttt{depth}(v_i) = \texttt{depth}(v_j)$ and v_i is left of v_j

Analysis: Short Edges



Look at global tree on center nodes.

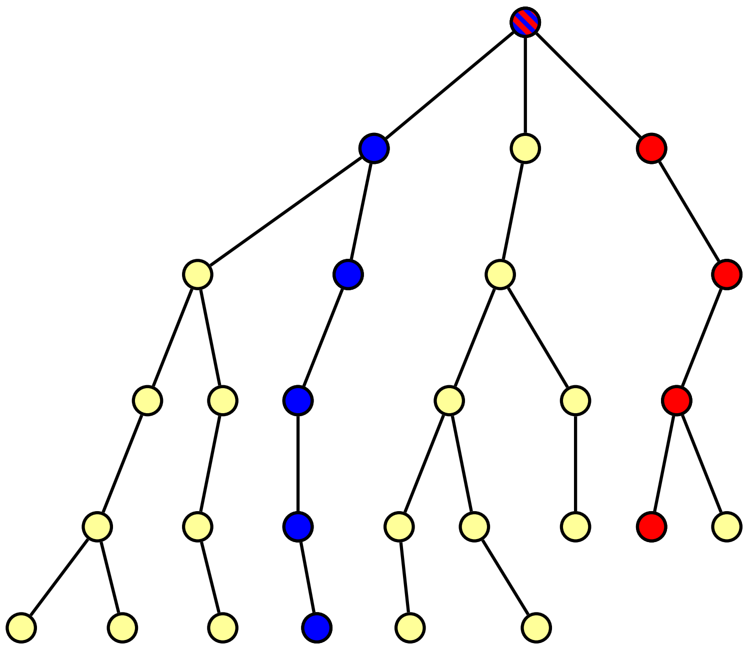
Observe: Let v_i and v_j be to center nodes. Can organize global tree s.t.

1. $|V_i| > |V_j|$ if $\texttt{depth}(v_i) < \texttt{depth}(v_j)$
2. $|V_i| \geq |V_j|$ is $\texttt{depth}(v_i) = \texttt{depth}(v_j)$ and v_i is left of v_j

Consider two root,leaf-paths

$$P_1 = \langle v_1^1, \ldots v_q^1 \rangle$$
$$P_2 = \langle v_1^2, \ldots v_r^2 \rangle$$

Analysis: Short Edges



Look at global tree on center nodes.

Observe: Let v_i and v_j be to center nodes. Can organize global tree s.t.

1. $|V_i| > |V_j|$ if $\mathtt{depth}(v_i) < \mathtt{depth}(v_j)$
2. $|V_i| \geq |V_j|$ is $\mathtt{depth}(v_i) = \mathtt{depth}(v_j)$ and v_i is left of v_j

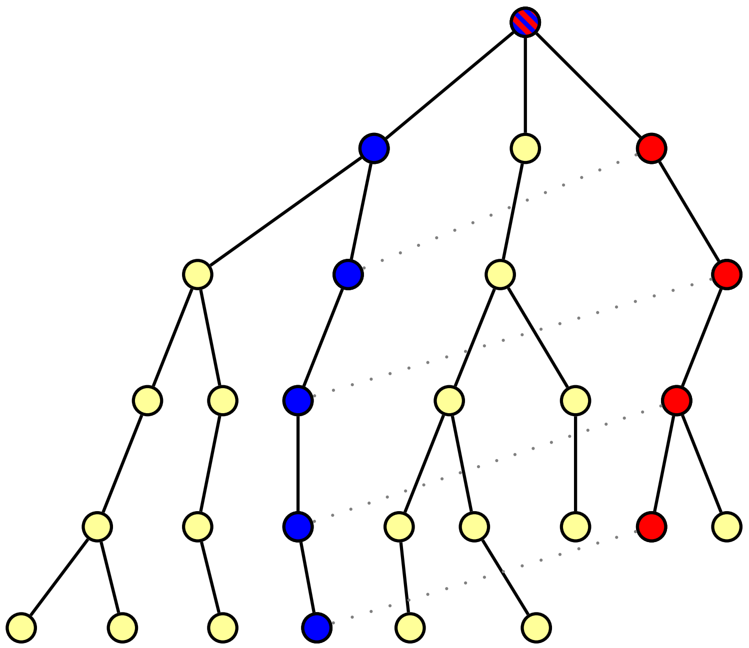
Consider two root,leaf-paths

$$P_1 = \langle v_1^1, \ldots v_q^1 \rangle$$
$$P_2 = \langle v_1^2, \ldots v_r^2 \rangle$$

Observations:

1. Leaves in global tree are on consecutive layers: $q \leq r + 1$

Analysis: Short Edges



Look at global tree on center nodes.

Observe: Let v_i and v_j be to center nodes. Can organize global tree s.t.

1. $|V_i| > |V_j|$ if $\texttt{depth}(v_i) < \texttt{depth}(v_j)$
2. $|V_i| \geq |V_j|$ is $\texttt{depth}(v_i) = \texttt{depth}(v_j)$ and v_i is left of v_j

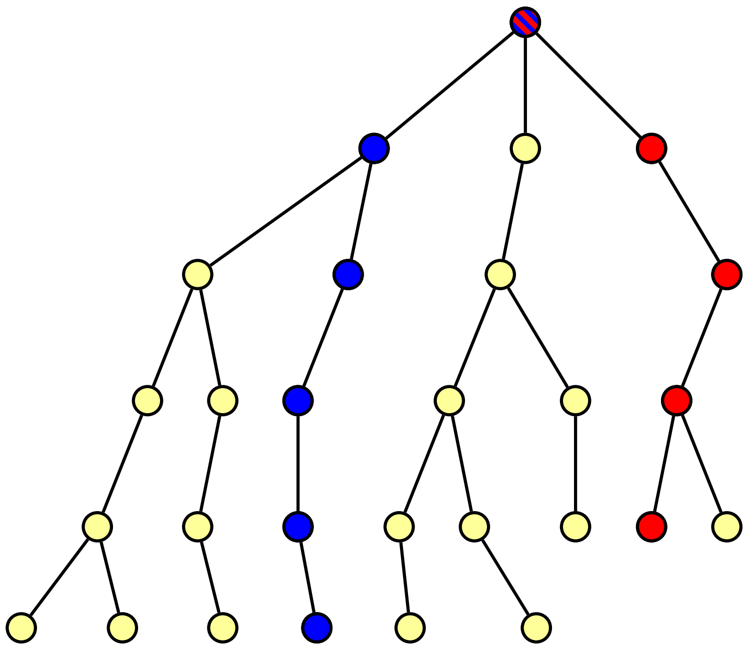
Consider two root,leaf-paths

$$P_1 = \langle v_1^1, \ldots v_q^1 \rangle$$
$$P_2 = \langle v_1^2, \ldots v_r^2 \rangle$$

Observations:

1. Leaves in global tree are on consecutive layers: $q \leq r + 1$
2. Know that $|V_i^1| \leq |V_{i-1}^2|$ for $2 \leq i \leq q$

Analysis: Short Edges



Consider two root,leaf-paths

$$P_1 = \langle v_1^1, \ldots v_q^1 \rangle$$
$$P_2 = \langle v_1^2, \ldots v_r^2 \rangle$$

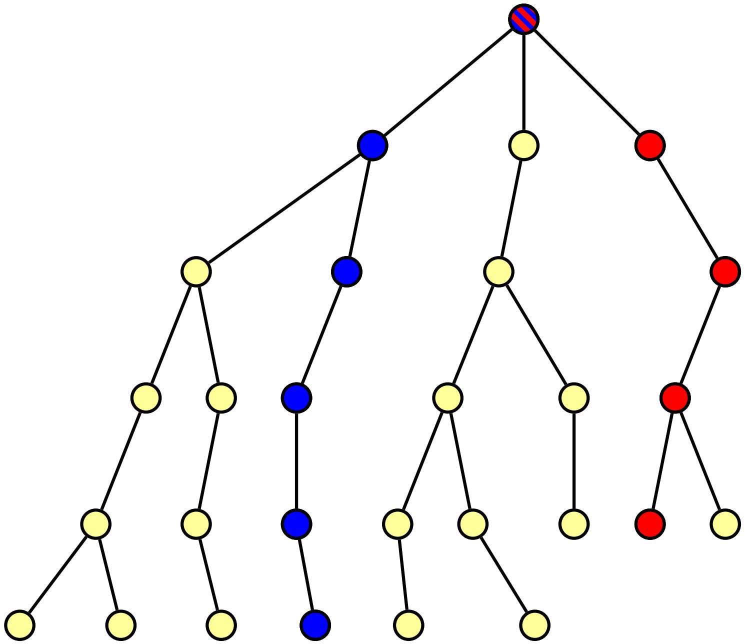
Observations:

1. Leaves in global tree are on consecutive layers: $q \leq r + 1$
2. Know that $|V_i^1| \leq |V_{i-1}^2|$ for $2 \leq i \leq q$

Hence:

$$|P_1|_s \leq |P_2|_s + q + O(\log_B |V_1|)$$
$$= |P_2|_s + 2 \log_B n$$

Analysis: Short Edges



Consider two root,leaf-paths

$$P_1 = \langle v_1^1, \ldots v_q^1 \rangle$$
$$P_2 = \langle v_1^2, \ldots v_r^2 \rangle$$

Observations:

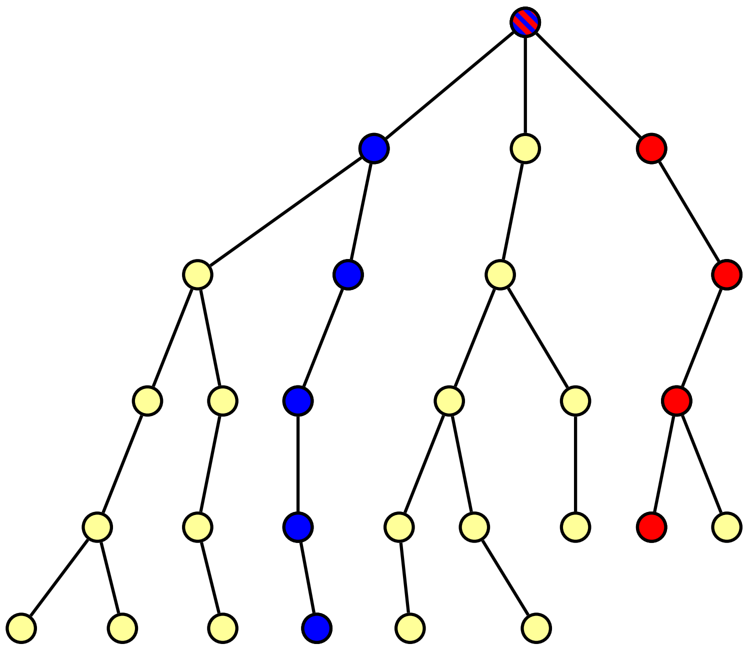
1. Leaves in global tree are on consecutive layers: $q \leq r + 1$
2. Know that $|V_i^1| \leq |V_{i-1}^2|$ for $2 \leq i \leq q$

Hence:

$$|P_1|_s \leq |P_2|_s + q + O(\log_B |V_1|)$$
$$= |P_2|_s + 2 \log_B n$$

Similar: $|P_2|_s \leq |P_1|_s + \log_B n$

Analysis: Short Edges



Hence:

$$|P_1|_s \leq |P_2|_s + q + O(\log_B |V_1|) = |P_2|_s + 2\log_B n$$

Similar: $|P_2|_s \leq |P_1|_s + \log_B n$

This means:

$|P|_s \leq \gamma + 2\log_B n$ for all root,leaf paths P

Lemma: $|P|_s = O(\log_B n)$ for all root,leaf paths in our tree.

Proof idea: All but $O(\log_B n)$ nodes on any root,leaf-path have degree B.

Analysis: Outline

- Show that any root,leaf path in our tree has
  1. ~~$O(\sqrt{\log_B n})$ long edges, and~~
  2. $O(\log_B n)$ short edges.
- This implies: Length of any root,leaf path is at most

$$O(\sqrt{\log_B n}) \cdot \Delta + O(\log_B n) \cdot \alpha$$

- With $\alpha = \Delta/\sqrt{\log_B n}$: Length of root,leaf path is bouded by

$$O(\sqrt{\log_B n}) \cdot \Delta$$

Analysis: Outline

- Show that any root,leaf path in our tree has
  1. ~~$O(\sqrt{\log_B n})$ long edges, and~~
  2. ~~$O(\log_B n)$ short edges.~~
- This implies: Length of any root,leaf path is at most

$$O(\sqrt{\log_B n}) \cdot \Delta + O(\log_B n) \cdot \alpha$$

- With $\alpha = \Delta / \sqrt{\log_B n}$: Length of root,leaf path is bouded by

$$O(\sqrt{\log_B n}) \cdot \Delta$$

Conclusion

This talk: We show how to compute a tree T with maximum degree B and diameter $O(\sqrt{\log_B n}) \cdot \Delta$ in complete metrics

This implies: $O(\sqrt{\log_B n})$-competitive algorithm for Freeze-Tag in general graphs.

Conclusion

This talk: We show how to compute a tree T with maximum degree B and diameter $O(\sqrt{\log_B n}) \cdot \Delta$ in complete metrics

This implies: $O(\sqrt{\log_B n})$-competitive algorithm for Freeze-Tag in general graphs.

Open questions:

1. Close gap between $5/3$-hardness and $O(\sqrt{\log_B n})$-approximation
2. We strongly use fact that input graph is complete. Best known for incomplete graphs is still [Ravi et al.]: Can compute tree with
   (a) Diameter $O(\log n)\Delta$, and
   (b) Maximum degree $O(\log^2 n) \cdot B$