Locating a Robber on a Tree.

Axel Brandt, Jennifer Diemunsch,
Catherine Erbes, Jordan LeGrand, Casey
Moffatt.
University of Colorado Denver

MAY 24, 2013

Original Cops and Robbers

Pursuit-Evasion on a graph introduced by Parsons (1976).

Cops and Robbers introduced independently by Nowakowski and Winkler (1983) and by Quillot (1977).

Original Cops and Robbers

Pursuit-Evasion on a graph introduced by Parsons (1976).

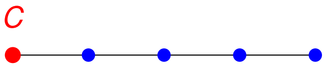
Cops and Robbers introduced independently by Nowakowski and Winkler (1983) and by Quillot (1977).

Game

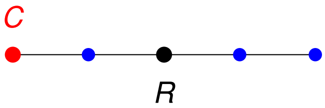
1. *Cop chooses vertex*
2. *Robber chooses vertex*
3. *Cop and Robber take turns moving to adjacent vertices (visible to both players)*
4. *Cop wins if he 'arrests' the Robber*

Examples

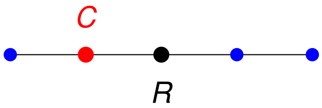
Examples



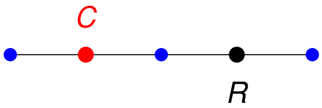
Examples



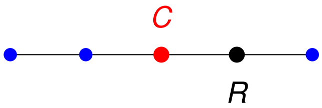
Examples



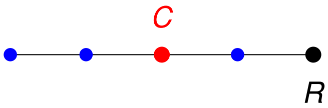
Examples



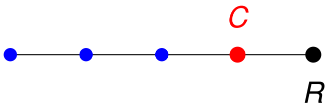
Examples



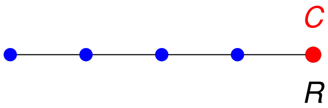
Examples



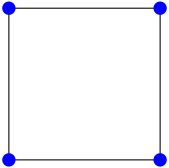
Examples



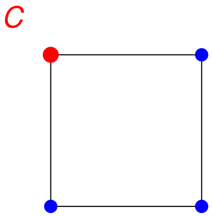
Examples



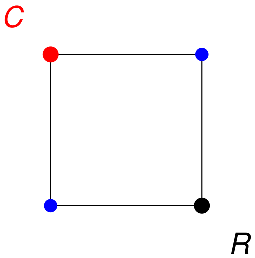
Examples



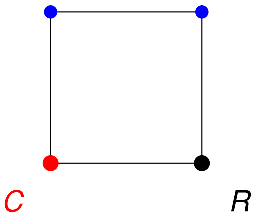
Examples



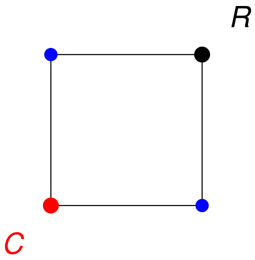
Examples



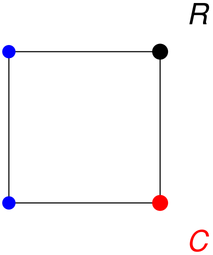
Examples



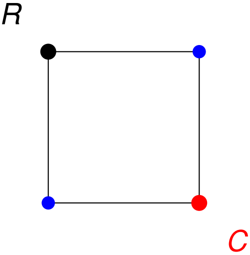
Examples



Examples



Examples



Motivating Questions

Definition

*A graph is **cop-win** if there is a strategy under which the Cop arrests the Robber in a finite number of moves. Otherwise, a graph is **robber-win**.*

Motivating Questions

Definition

*A graph is **cop-win** if there is a strategy under which the Cop arrests the Robber in a finite number of moves. Otherwise, a graph is **robber-win**.*

Example

A tree T is cop-win.

Motivating Questions

Definition

A graph is ***cop-win*** *if there is a strategy under which the Cop arrests the Robber in a finite number of moves. Otherwise, a graph is* ***robber-win****.*

Example

A tree T is cop-win.

1. Which graphs are cop-win?
2. For a graph G, what is the minimum number of Cops required to capture the Robber?
3. What is the minimum guaranteed capture time?

Early Results

Theorem (Nowakowski, Winkler 1983)

A finite graph G is cop-win if and only if G is dismantlable.

Early Results

Theorem (Nowakowski, Winkler 1983)

A finite graph G is cop-win if and only if G is dismantlable.

Theorem (Aigner, Fromme 1984)

If G is planar with 3 Cops, then G is cop-win.

Theorem (Quilliot 1985)

If G is genus k with $3 + 2k$ Cops, then G is cop-win.

Early Results

Theorem (Nowakowski, Winkler 1983)

A finite graph G is cop-win if and only if G is dismantlable.

Theorem (Aigner, Fromme 1984)

If G is planar with 3 Cops, then G is cop-win.

Theorem (Quilliot 1985)

If G is genus k with $3 + 2k$ Cops, then G is cop-win.

Theorem (Seymour, Thomas 1993)

A graph G has treewidth $k - 1$ iff k is the minimum number of cops for which G is cop-win.

On Large Classes

- Directed graphs (Hahn, MacGillivray 2006)
- Edge Critical graphs (Clarke, Fitzpatrick, Nowakowski 2010)
- Forbidden (induced) subgraphs (Joret, Kamiński, Theis 2010)
- Interval graphs (Gavenčiak 2011)
- Geometric graphs (Beveridge, Dudek, Frieze, Müller 2012)
- Random graphs (Scott, Sudakov 2011; Bollobas, Kun, Leader 2013)

Mobility Variations

- Robber can Hide and Ride (Chalopin, Chepoi 2011)
- Lazy Robber (Richerby, Thilikos 2011)
- Fast Robber (Alon, Mehrabian 2011; Frieze, Krivelevich, Loh 2012)
- Drunk Robber (Kehagias, Prałat 2012)

Search Variations

- Helicopter Search (Fomin 1998)
- Tandem Search (Clarke, Nowakowski 2005)
- Reduced Visibility (Isler, Karnad 2008)
- Witness (Clarke 2009)
- Fuel, Cost, Time (Fomin, Golovach, Prałat 2012)
- Distance Query (Seager 2012)

Search Variations

- Helicopter Search (Fomin 1998)
- Tandem Search (Clarke, Nowakowski 2005)
- Reduced Visibility (Isler, Karnad 2008)
- Witness (Clarke 2009)
- Fuel, Cost, Time (Fomin, Golovach, Prałat 2012)
- **Distance Query** (Seager 2012)

Distance Query

Game

Distance Query

Game

1. *Robber chooses location*

Distance Query

Game

1. *Robber chooses location*
2. *Cop 'probes' a vertex and discovers the distance from that vertex to Robber*

Distance Query

Game

1. *Robber chooses location*
2. *Cop 'probes' a vertex and discovers the distance from that vertex to Robber*
3. *Robber may move to any adjacent vertex not the probe vertex (no-backtrack condition)*

Distance Query

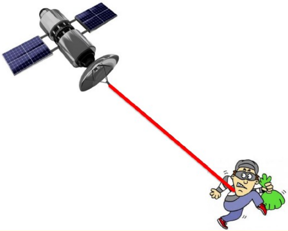
Game

1. *Robber chooses location*
2. *Cop 'probes' a vertex and discovers the distance from that vertex to Robber*
3. *Robber may move to any adjacent vertex not the probe vertex (no-backtrack condition)*
4. *Cop wins if he determines unique location of Robber*

Distance Query

Game

1. *Robber chooses location*
2. *Cop 'probes' a vertex and discovers the distance from that vertex to Robber*
3. *Robber may move to any adjacent vertex not the probe vertex (no-backtrack condition)*
4. *Cop wins if he determines unique location of Robber*



Distance Query

Game

1. *Robber chooses location*
2. *Cop 'probes' a vertex and discovers the distance from that vertex to Robber*
3. *Robber may move to any adjacent vertex not the probe vertex (no-backtrack condition)*
4. *Cop wins if he determines unique location of Robber*

Definition

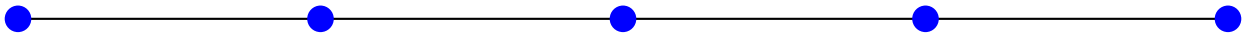
A graph G is ***locatable*** *if the Cop has a winning strategy.*

For locatable G, the ***location number*** *of G, $loc(G)$, is the minimum number of probes guaranteed to locate the Robber.*

Location Number of Paths

Proposition (Seager 2012)

$loc(G) = 1$ if and only if G is a path.

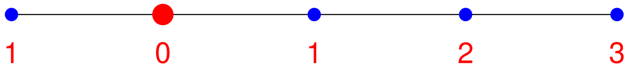


P_5

Location Number of Paths

Proposition (Seager 2012)

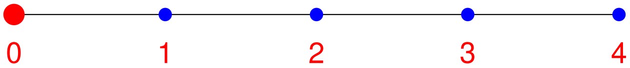
$loc(G) = 1$ if and only if G is a path.



Location Number of Paths

Proposition (Seager 2012)

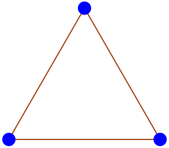
$loc(G) = 1$ if and only if G is a path.



Location Number of K_3

Proposition (Seager 2012)

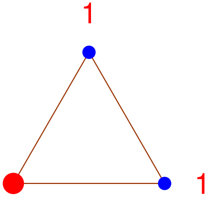
$loc(K_3) = 2.$



K_3

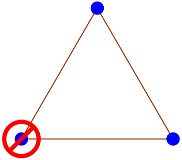
Proposition (Seager 2012)

$loc(K_3) = 2.$



Proposition (Seager 2012)

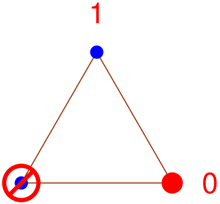
$loc(K_3) = 2.$



Location Number of K_3

Proposition (Seager 2012)

$loc(K_3) = 2.$



Seager's Results

Proposition (Seager 2012)

$loc(G) = 1$ if and only if G is a path.

Seager's Results

Proposition (Seager 2012)

$loc(G) = 1$ if and only if G is a path.

Proposition (Seager 2012)

$loc(K_3) = 2$ and $loc(K_{2,3}) = 3$.

Seager's Results

Proposition (Seager 2012)

$loc(G) = 1$ if and only if G is a path.

Proposition (Seager 2012)

$loc(K_3) = 2$ and $loc(K_{2,3}) = 3$.

Proposition

If G contains K_4 as a subgraph, then G is not locatable.

Seager's Results

Proposition (Seager 2012)

$loc(G) = 1$ if and only if G is a path.

Proposition (Seager 2012)

$loc(K_3) = 2$ and $loc(K_{2,3}) = 3$.

Proposition

If G contains K_4 as a subgraph, then G is not locatable.

Proposition

If G contains $K_{3,3}$ as an induced subgraph, then G is not locatable.

Seager's Results

Proposition

$loc(C_4) = 2$

Seager's Results

Proposition

$loc(C_4) = 2$

Proposition

C_5 is not locatable.

Seager's Results

Proposition

$loc(C_4) = 2$

Proposition

C_5 is not locatable.

Proposition

C_n is locatable for $n > 5$. With, $loc(C_n) = 3$ for $6 \leq n \leq 11$ and $loc(C_n) = 2$ for $n > 11$.

Seager's Results For Trees

Theorem

For any tree T, $loc(T) \geq \Delta(T) - 1$.

Seager's Results For Trees

Theorem

For any tree T, $loc(T) \geq \Delta(T) - 1$.

Theorem

If T is an n-vertex spider with $n \geq 3$, then $loc(T) = \Delta(T) - 1$.

Seager's Results For Trees

Theorem

For any tree T, $loc(T) \geq \Delta(T) - 1$.

Theorem

If T is an n-vertex spider with $n \geq 3$, then $loc(T) = \Delta(T) - 1$.

Corollary

$loc(K_{1,n-1}) = n - 2$ for $n \geq 3$.

Seager's Results For Trees

Theorem

For any tree T, $loc(T) \geq \Delta(T) - 1$.

Theorem

If T is an n-vertex spider with $n \geq 3$, then $loc(T) = \Delta(T) - 1$.

Corollary

$loc(K_{1,n-1}) = n - 2$ for $n \geq 3$.

Theorem

If T is a tree with $n \geq 3$ vertices, then $loc(T) \leq n - 2$, with equality if and only if $T = K_{1,n-1}$.

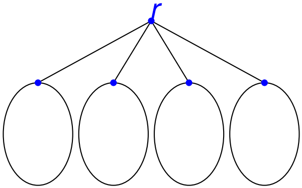
r-Locating Strategy

Definition

*An **r-locating strategy** locates the robber if he is ever at $r \in V(T)$.*
*We say an r-locating strategy S has the **Root Location Property** for r.*

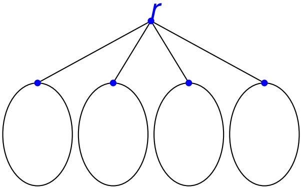
Definition

*An **r-locating strategy** locates the robber if he is ever at $r \in V(T)$.*
*We say an r-locating strategy S has the **Root Location Property** for r.*



Definition

An ***r-locating strategy*** *locates the robber if he is ever at $r \in V(T)$.*
We say an r-locating strategy S has the ***Root Location Property*** *for r.*



Remark

Under an r-locating strategy, the Robber will never be able to move from one component of $T - r$ to another.

Theorem (Seager 2012)

If T is a tree with $n \geq 3$ vertices, then $loc(T) \leq n - 2$, with equality if and only if $T = K_{n-1,1}$.

Location Number of Trees

Theorem (Seager 2012)

If T is a tree with $n \geq 3$ vertices, then $loc(T) \leq n - 2$, with equality if and only if $T = K_{n-1,1}$.

Open Problems

- Find a strategy for trees that uses $loc(T)$ probes

Theorem (Seager 2012)

If T is a tree with $n \geq 3$ vertices, then $loc(T) \leq n-2$, with equality if and only if $T = K_{n-1,1}$.

Open Problems

- Find a strategy for trees that uses $loc(T)$ probes
- Improve bound for $loc(T)$ using parameters other than n

Small Diameter

d=1 Edge $\Rightarrow$ $loc(T) = \ell - 1$

Small Diameter

d=1 Edge $\Rightarrow$ $loc(T) = \ell - 1$

d=2 Star $\Rightarrow$ $loc(T) = \ell - 1$

Small Diameter

d=1 Edge $\Rightarrow$ $loc(T) = \ell - 1$

d=2 Star $\Rightarrow$ $loc(T) = \ell - 1$

d=3 Double-Star $\Rightarrow$ $loc(T) < \ell - 1$.

Small Diameter

d=1 Edge $\Rightarrow$ $loc(T) = \ell - 1$

d=2 Star $\Rightarrow$ $loc(T) = \ell - 1$

d=3 Double-Star $\Rightarrow$ $loc(T) < \ell - 1$.

d=4 Star of stars $\Rightarrow$ $loc(T) < \ell - 1$.

Small Diameter

d=1 Edge $\Rightarrow$ $loc(T) = \ell - 1$

d=2 Star $\Rightarrow$ $loc(T) = \ell - 1$

d=3 Double-Star $\Rightarrow$ $loc(T) < \ell - 1$.

d=4 Star of stars $\Rightarrow$ $loc(T) < \ell - 1$.

d=5 One arm is star of stars, each other arms is a star $\Rightarrow < \ell - 1$.

Main Result

Theorem (Brandt, Diemunsch, Erbes, LeGrand, M. 2013+)

If T is a tree with $diam(T) = d \geq 6$ and $\Delta(T) = \Delta$, then

$$loc(T) \leq (2\Delta^2 - 6\Delta + 4)\lceil \frac{d}{2} \rceil - 4\Delta^2 + 13\Delta - 9.$$

Main Result

Theorem (Brandt, Diemunsch, Erbes, LeGrand, M. 2013+)

If T is a tree with $diam(T) = d \geq 6$ and $\Delta(T) = \Delta$, then

$$loc(T) \leq (2\Delta^2 - 6\Delta + 4)\lceil \frac{d}{2} \rceil - 4\Delta^2 + 13\Delta - 9.$$

Answering Seager's Questions

- Algorithmic construction of a strategy.

Main Result

Theorem (Brandt, Diemunsch, Erbes, LeGrand, M. 2013+)

If T is a tree with $diam(T) = d \geq 6$ and $\Delta(T) = \Delta$, then

$$loc(T) \leq (2\Delta^2 - 6\Delta + 4)\lceil\frac{d}{2}\rceil - 4\Delta^2 + 13\Delta - 9.$$

Answering Seager's Questions

- Algorithmic construction of a strategy.
- For $diam(T) \geq 6$, $f(\Delta, d) < n - 2$.

Main Result

Theorem (Brandt, Diemunsch, Erbes, LeGrand, M. 2013+)

If T is a tree with $diam(T) = d \geq 6$ and $\Delta(T) = \Delta$, then

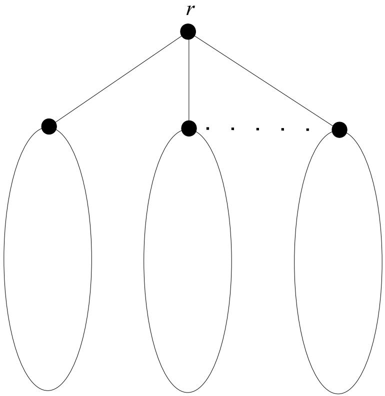
$$loc(T) \leq (2\Delta^2 - 6\Delta + 4)\lceil \frac{d}{2} \rceil - 4\Delta^2 + 13\Delta - 9.$$

Answering Seager's Questions

- Algorithmic construction of a strategy.
- For $diam(T) \geq 6$, $f(\Delta, d) < n - 2$.
- Conjecture $loc(T) \leq \ell - 1$...?

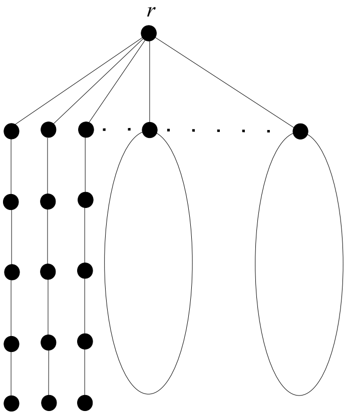
Algorithm Phase 1:

- Pick a Root.



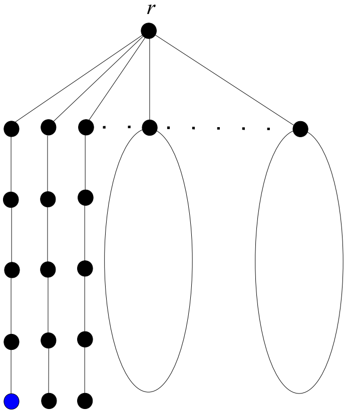
Algorithm Phase 1:

- Pick a Root.
- Remove paths.



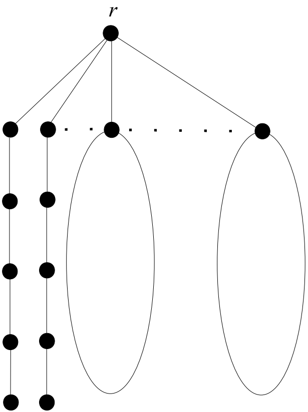
Algorithm Phase 1:

- Pick a Root.
- Remove paths.



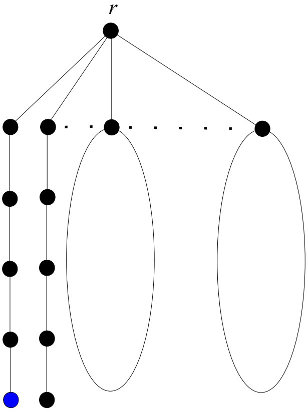
Algorithm Phase 1:

- Pick a Root.
- Remove paths.



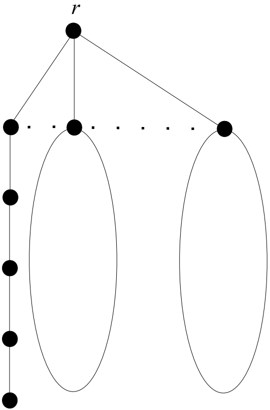
Algorithm Phase 1:

- Pick a Root.
- Remove paths.



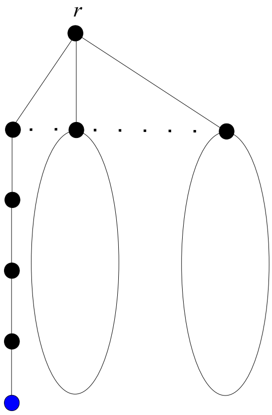
Algorithm Phase 1:

- Pick a Root.
- Remove paths.



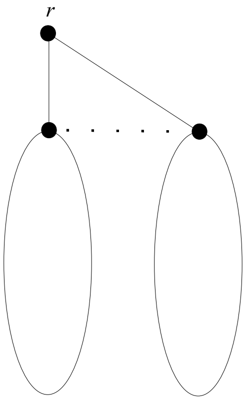
Algorithm Phase 1:

- Pick a Root.
- Remove paths.



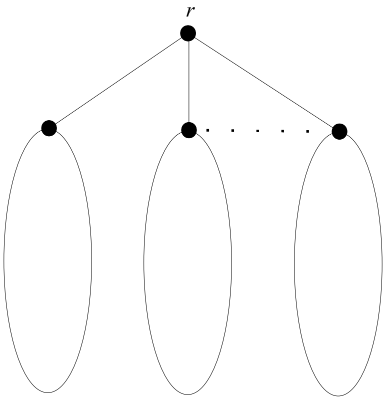
Algorithm Phase 1:

- Pick a Root.
- Remove paths.



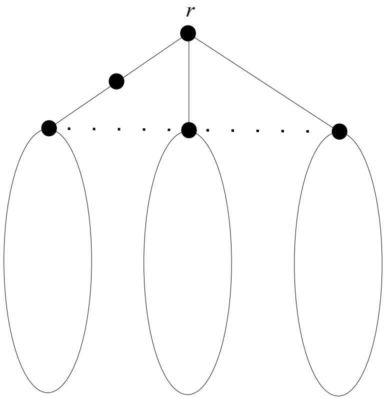
Algorithm Phase 1:

- Pick a Root.
- Remove paths.

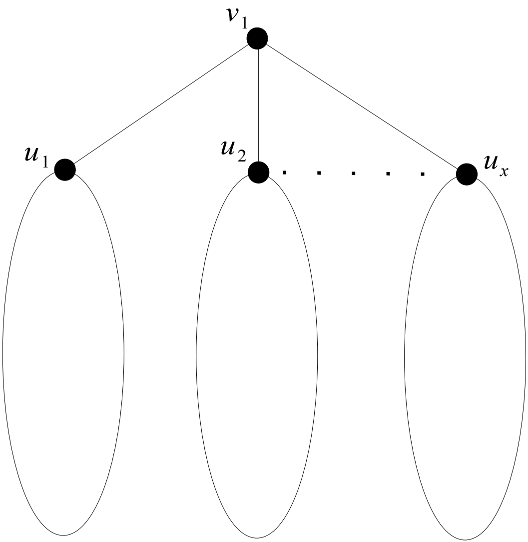


Algorithm Phase 1:

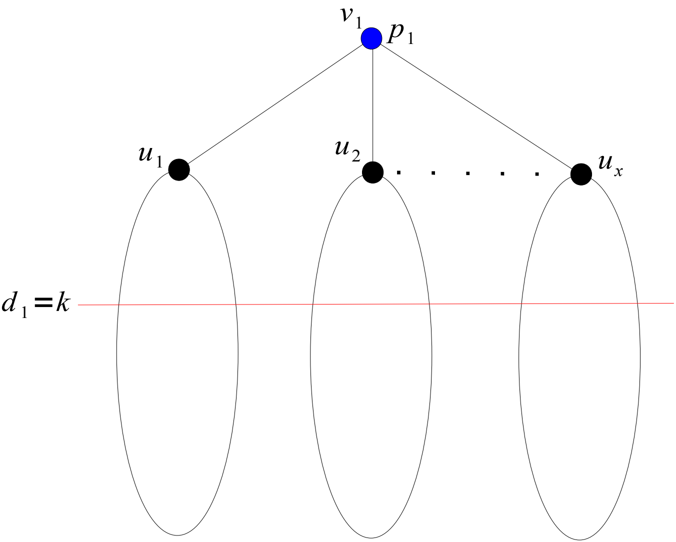
- Pick a Root.
- Remove paths.
- Remove Long Arms.



Algorithm Phase 2:

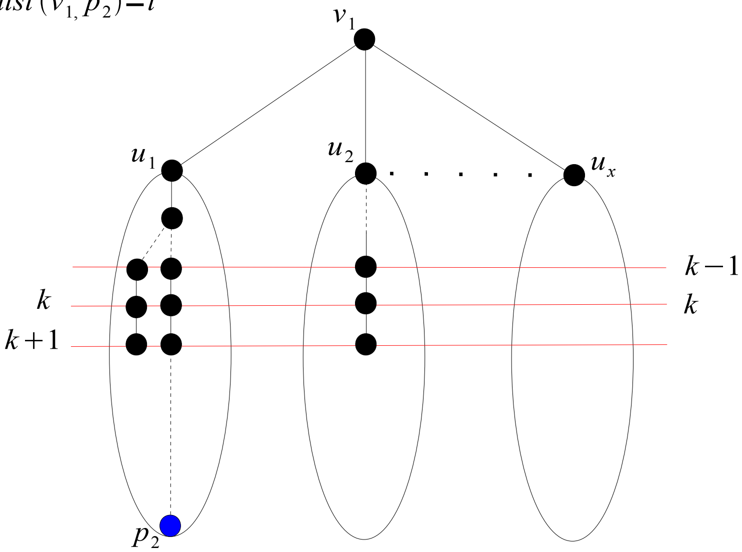


Algorithm Phase 2:



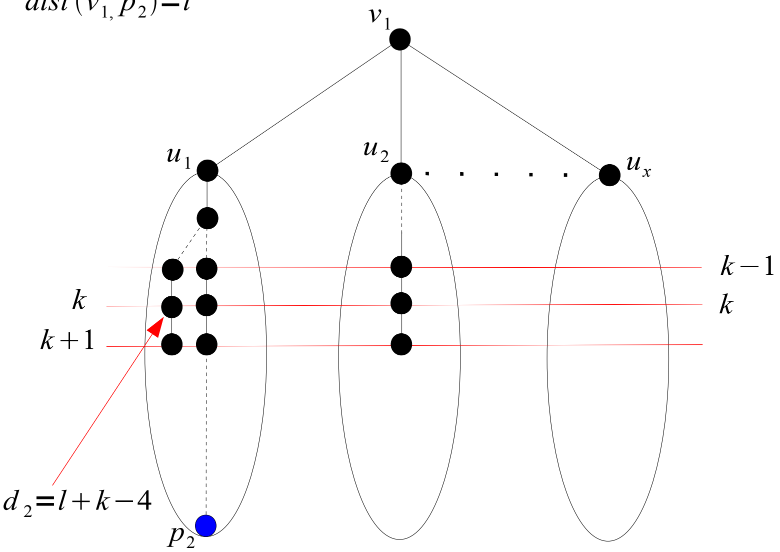
Algorithm Phase 2:

$dist(v_{1,} p_2) = l$



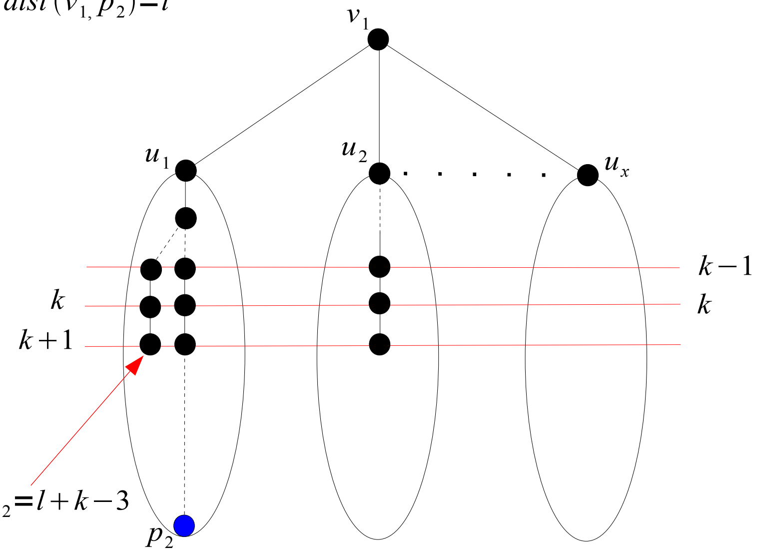
Algorithm Phase 2:

$dist(v_{1,} p_2) = l$



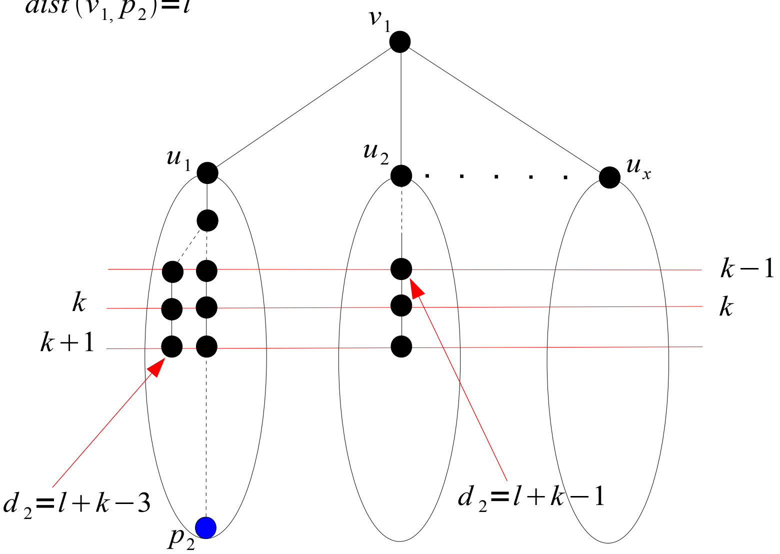
Algorithm Phase 2:

$dist(v_{1,} p_2) = l$



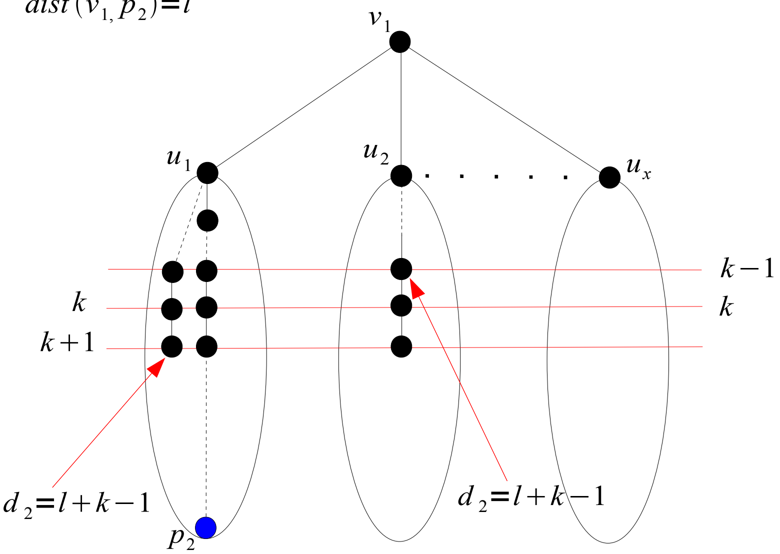
Algorithm Phase 2:

$dist(v_{1,} p_2) = l$

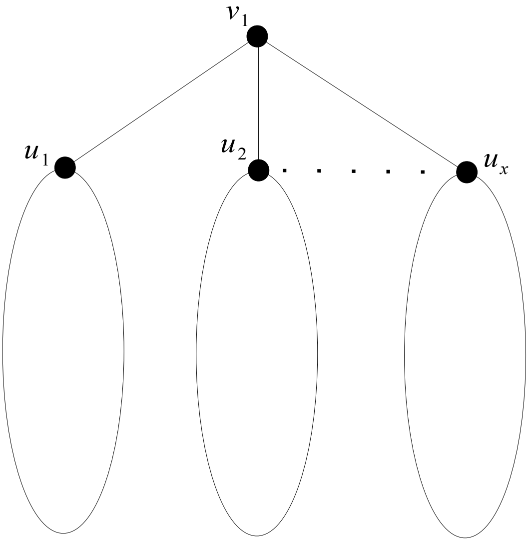


Algorithm Phase 3:

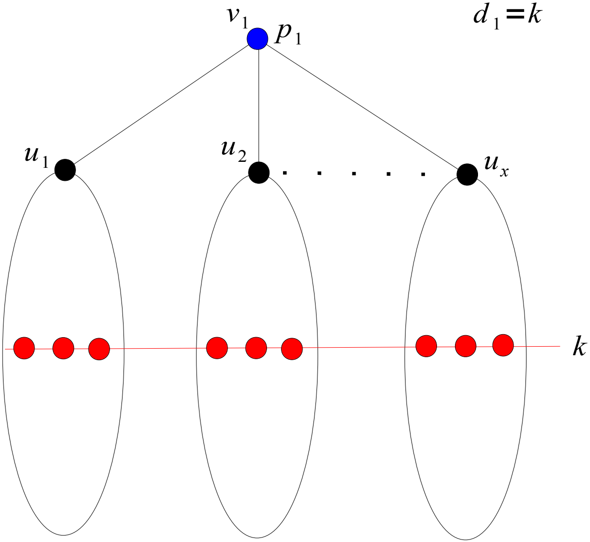
$dist(v_1, p_2) = l$



Algorithm Phase 3:

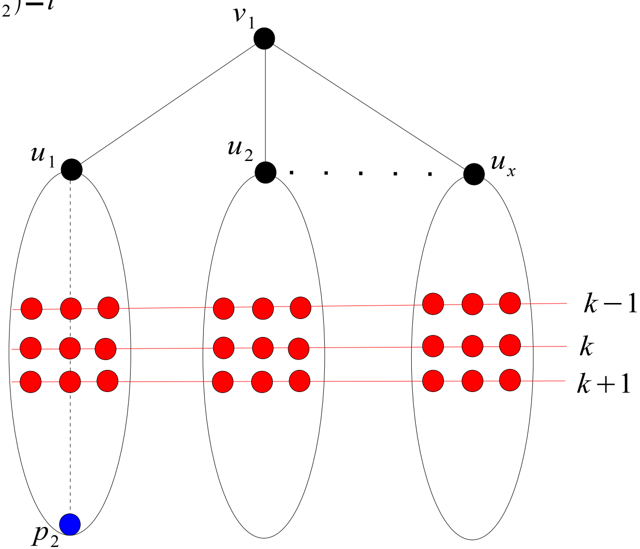


Algorithm Phase 3:



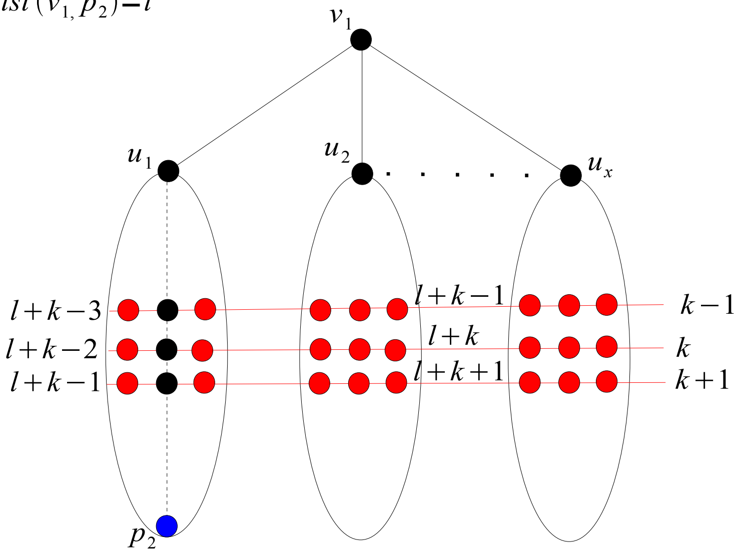
Algorithm Phase 3:

$dist(v_1, p_2) = l$



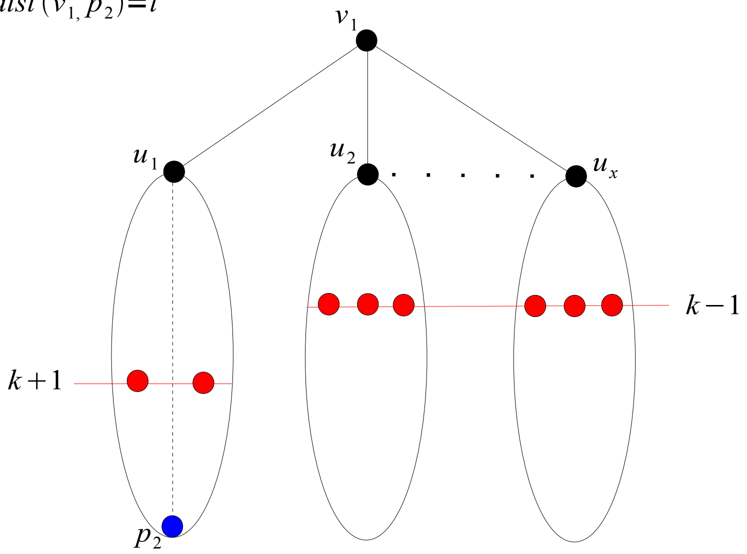
Algorithm Phase 3:

$dist(v_{1,} p_2) = l$



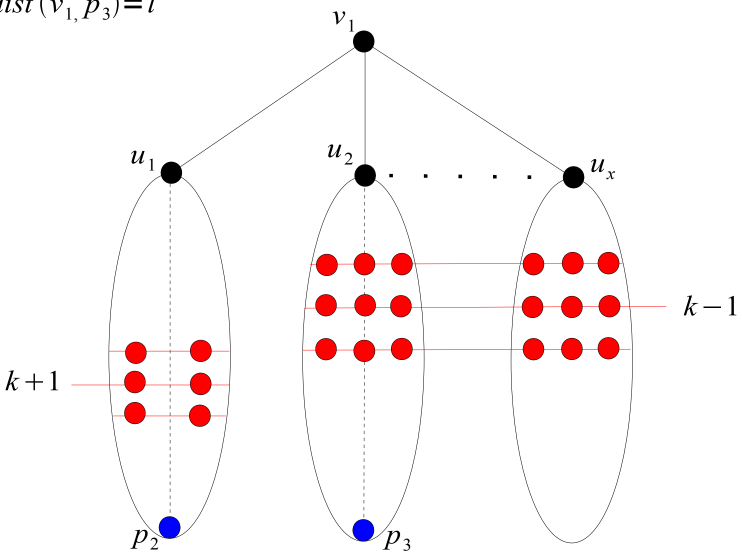
Algorithm Phase 3:

$dist(v_{1,} p_2) = l$



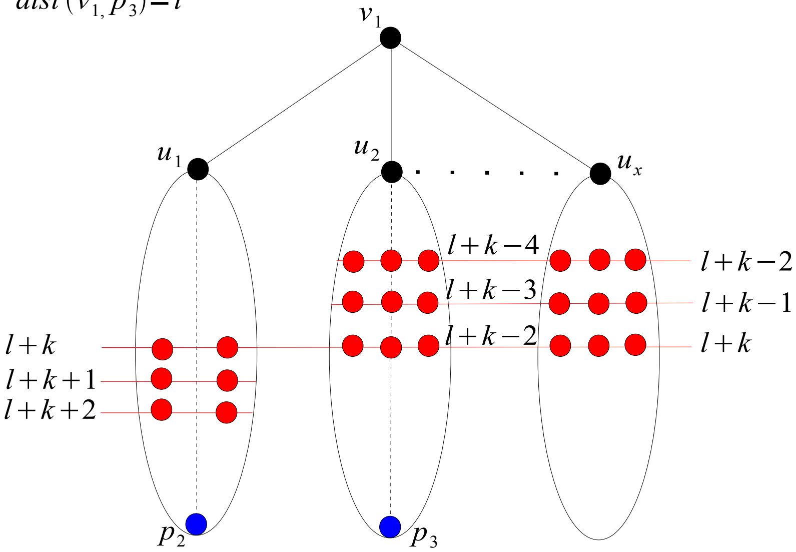
Algorithm Phase 3:

$dist(v_{1}, p_3) = l$



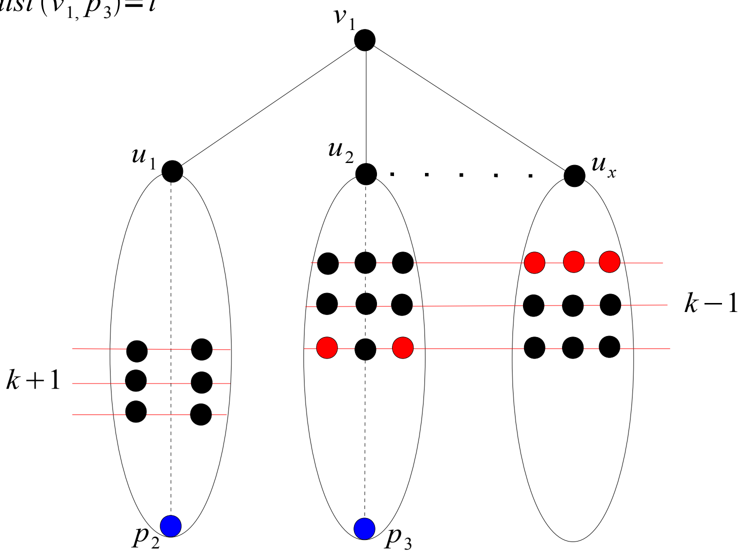
Algorithm Phase 3:

$dist(v_{1,} p_3) = l$



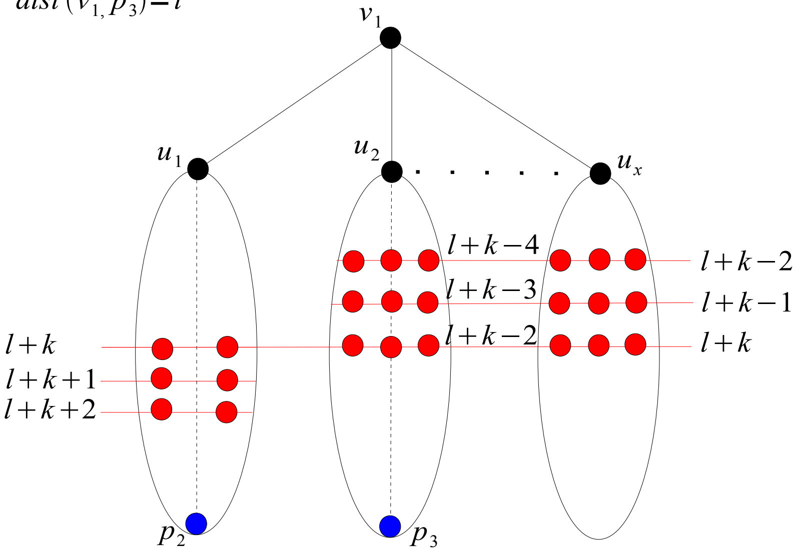
Algorithm Phase 3:

$dist(v_{1,} p_3) = l$



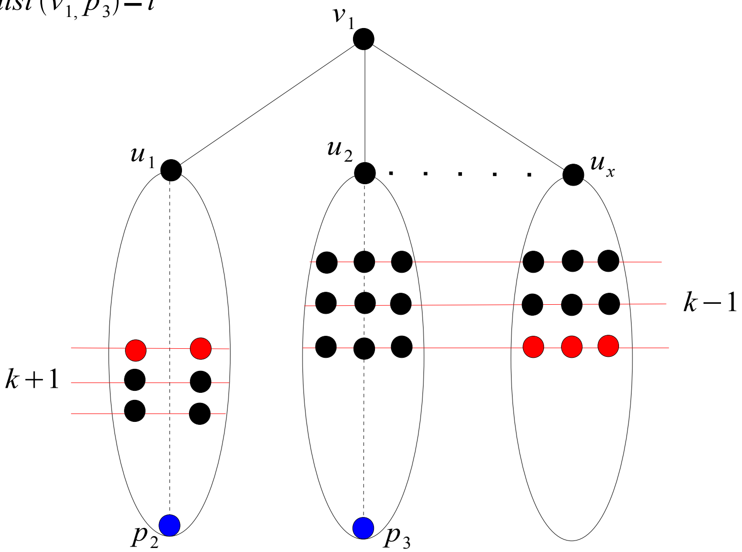
Algorithm Phase 3:

$dist(v_{1,} p_3) = l$

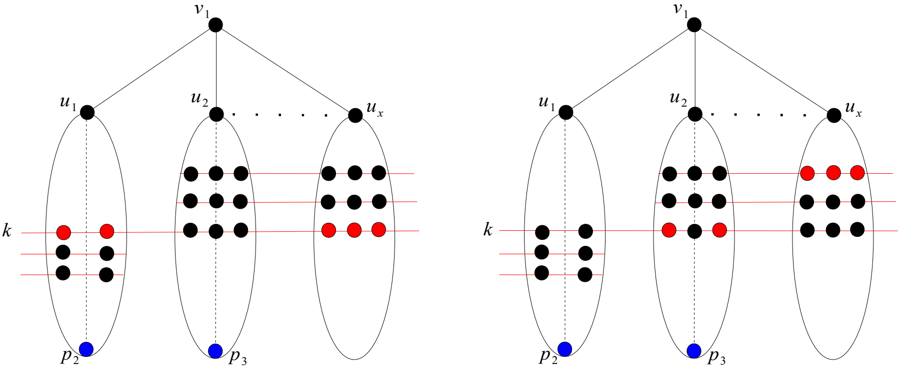


Algorithm Phase 3:

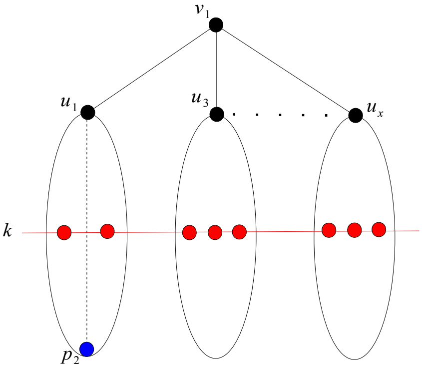
$dist(v_{1,} p_3) = l$

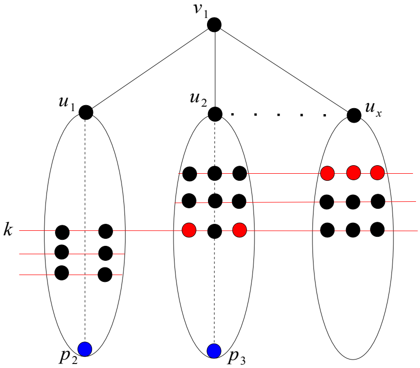


Algorithm Phase 3:

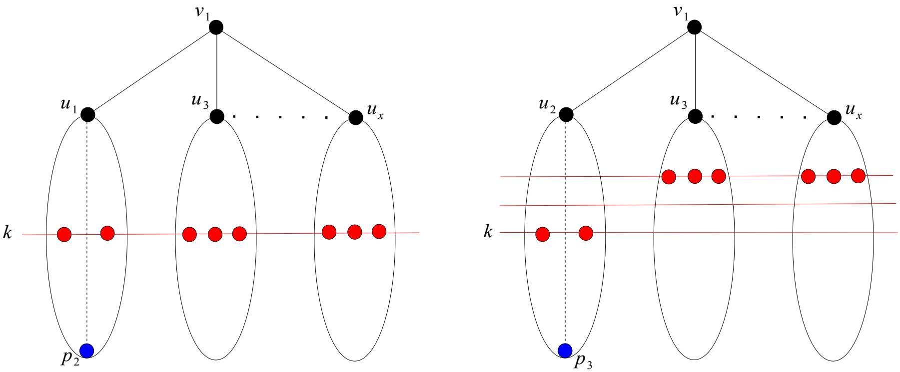


Algorithm Phase 3:

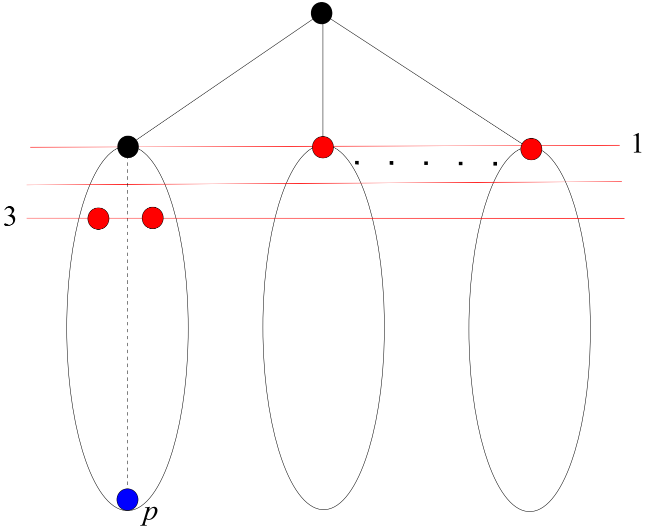




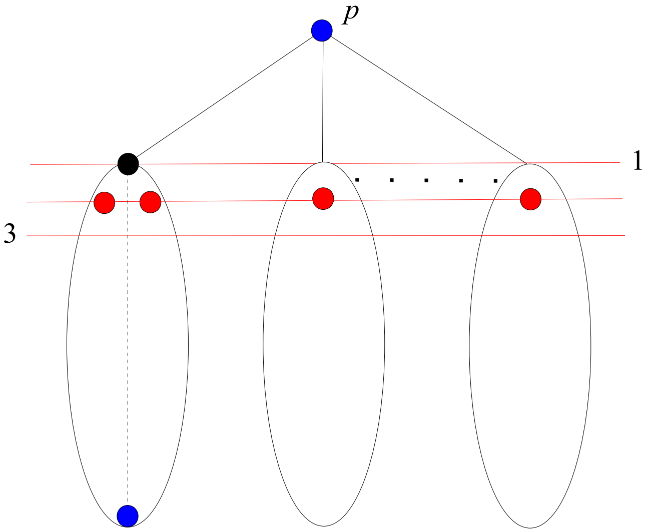
Algorithm Phase 3:



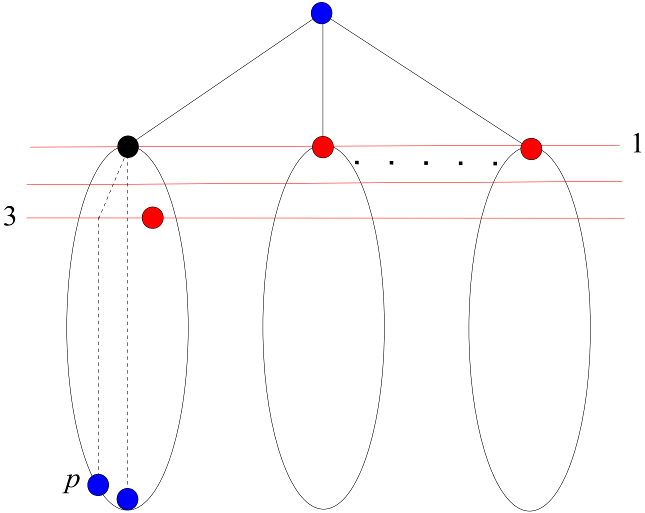
Algorithm Phase 3:



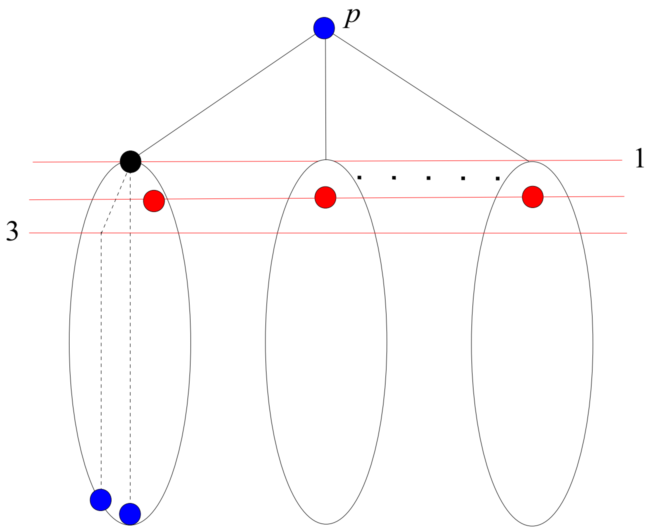
Algorithm Phase 3:



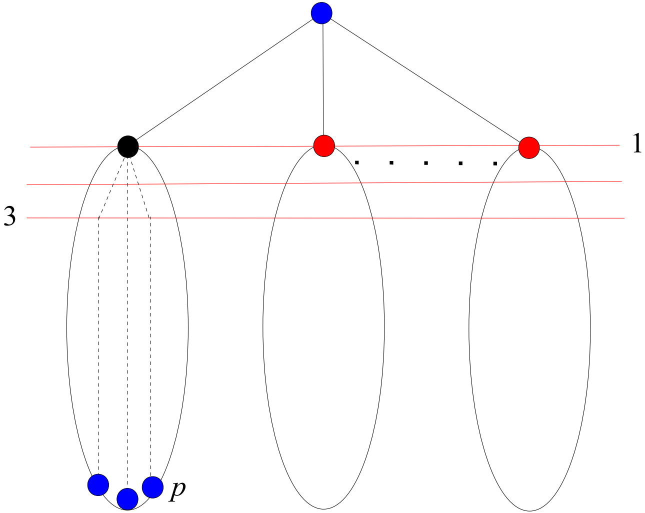
Algorithm Phase 3:



Algorithm Phase 3:

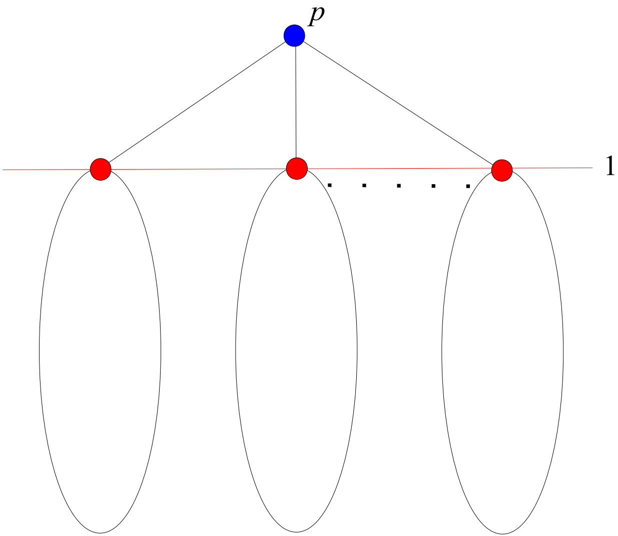


Algorithm Phase 3:

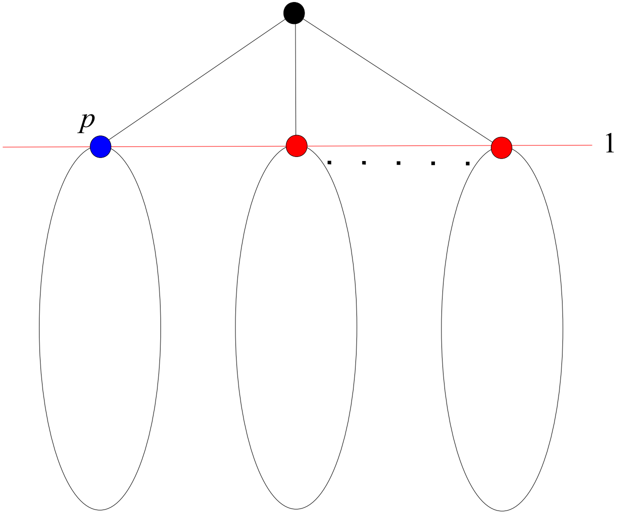


Algorithm Phase 4:

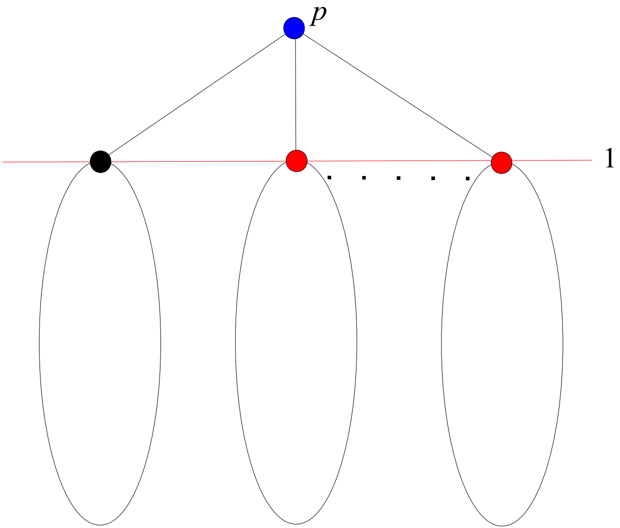
Algorithm Phase 4:



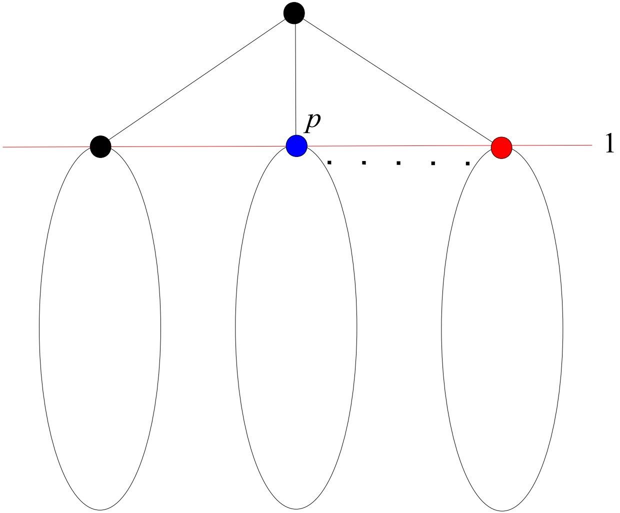
Algorithm Phase 4:



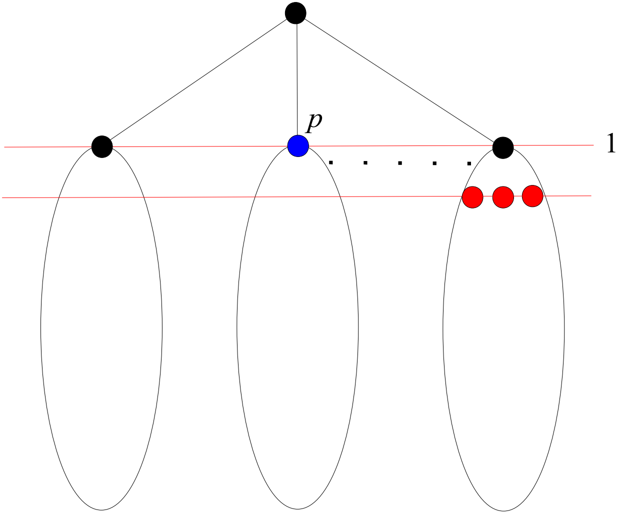
Algorithm Phase 4:



Algorithm Phase 4:



Algorithm Phase 4:



Algorithm Upper Bound

Lemma

If $T \subseteq T'$ *then* $loc(T) \leq loc(T')$

Algorithm Upper Bound

Lemma

If $T \subseteq T'$ then $loc(T) \leq loc(T')$

- Let T' be the smallest Δ-inary tree such that $T \subseteq T'$.

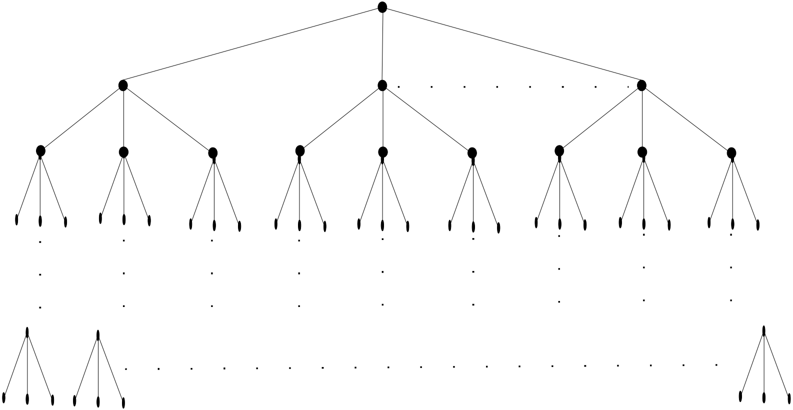
Algorithm Upper Bound

Lemma

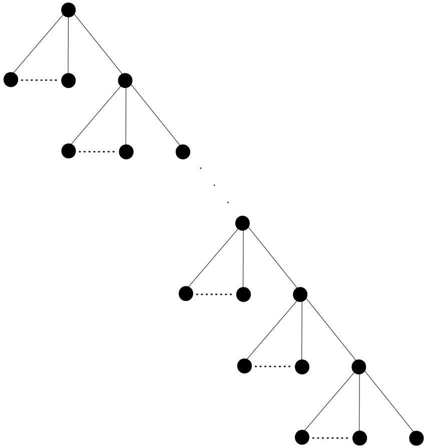
If $T \subseteq T'$ then $loc(T) \leq loc(T')$

- Let T' be the smallest Δ-inary tree such that $T \subseteq T'$.
- Determining $loc(T')$ gives an upper bound on $loc(T)$.

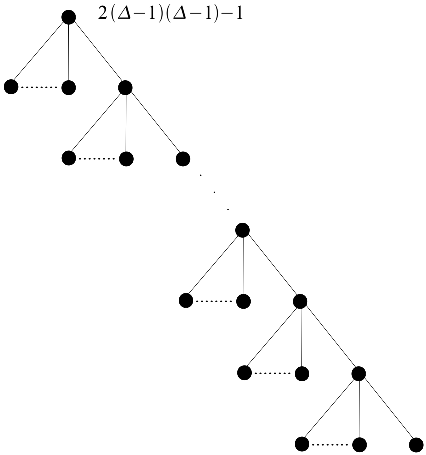
Algorithm Upper Bound (Diameter at least 6).



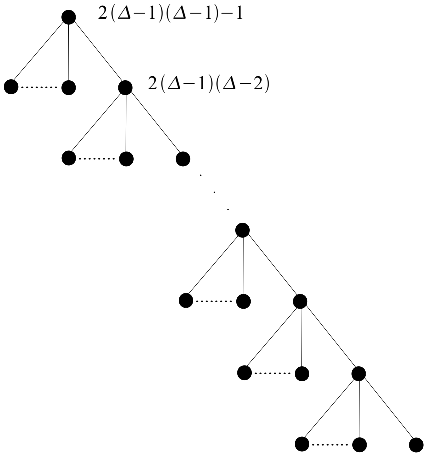
Algorithm Upper Bound (Diameter at least 6).



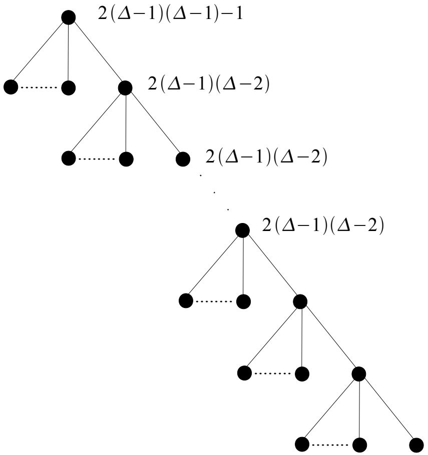
Algorithm Upper Bound (Diameter at least 6).



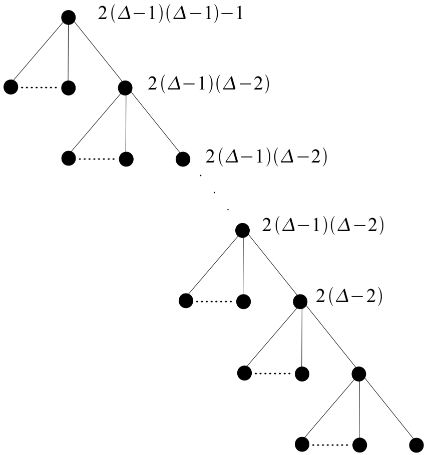
Algorithm Upper Bound (Diameter at least 6).



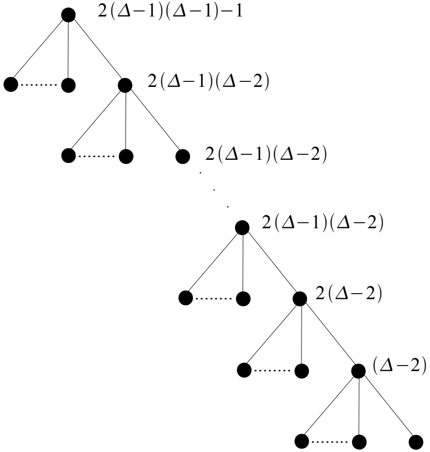
Algorithm Upper Bound (Diameter at least 6).



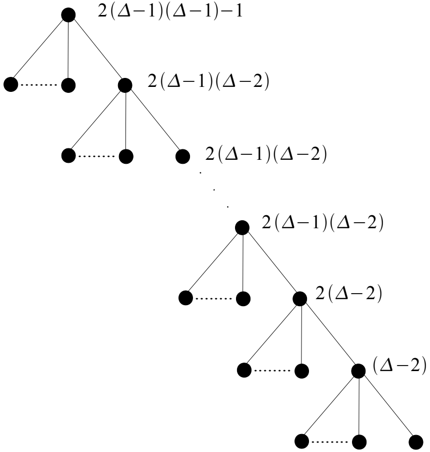
Algorithm Upper Bound (Diameter at least 6).



Algorithm Upper Bound (Diameter at least 6).

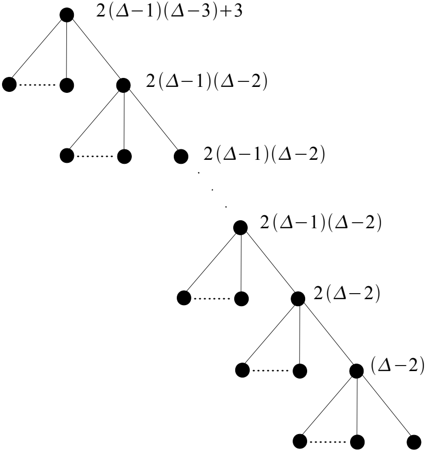


Algorithm Upper Bound (Diameter at least 6).



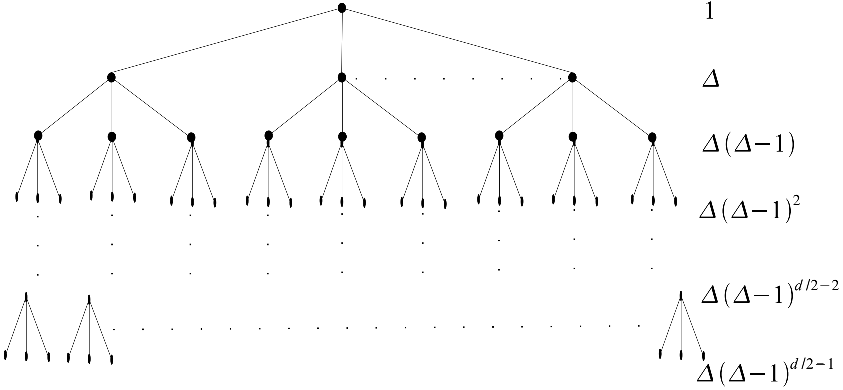
$$loc(T) \leq (2\Delta^2 - 6\Delta + 4)\lceil \tfrac{d}{2} \rceil - 4\Delta^2 + 17\Delta - 17$$

Algorithm Upper Bound (Diameter at least 6).



$$loc(T) \leq (2\Delta^2 - 6\Delta + 4)\lceil \tfrac{d}{2} \rceil - 4\Delta^2 + 13\Delta - 9$$

Compare with $n - 2$ For k-inary Tree.



$$\sum_{i=0}^{d/2-1} \Delta(\Delta-1)^i + 1$$

Compare with $n-2$ For k-inary Tree.

$$f(\Delta, d) = (2\Delta^2 - 6\Delta + 4)\lceil \frac{d}{2} \rceil - 4\Delta^2 + 13\Delta - 9$$

- $f(\Delta, d) \leq d\Delta^2$

Compare with $n-2$ For k-inary Tree.

$$f(\Delta, d) = (2\Delta^2 - 6\Delta + 4)\lceil \frac{d}{2} \rceil - 4\Delta^2 + 13\Delta - 9$$

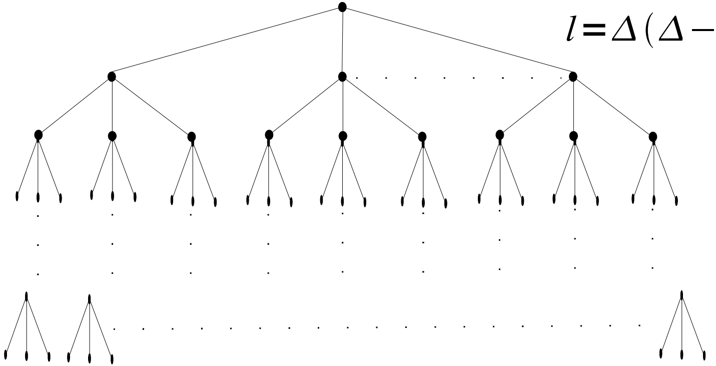
- $f(\Delta, d) \leq d\Delta^2$
- $d\Delta^2 < \sum_{i=0}^{\frac{d}{2}-1} (\Delta)(\Delta - 1)^i - 1$

Compare with $n-2$ For k-inary Tree.

$$f(\Delta, d) = (2\Delta^2 - 6\Delta + 4)\lceil \frac{d}{2} \rceil - 4\Delta^2 + 13\Delta - 9$$

- $f(\Delta, d) \le d\Delta^2$
- $d\Delta^2 < \sum_{i=0}^{\frac{d}{2}-1} (\Delta)(\Delta - 1)^i - 1$
- So $f(\Delta, d) \ll n - 2$ for Δ-inary tree.

Compare with $\ell - 1$...?



$$l = \Delta(\Delta - 1)^{d/2-1}$$

$$loc(T) \leq (2\Delta^2 - 6\Delta + 4)\lceil \frac{d}{2} \rceil - 4\Delta^2 + 13\Delta - 9$$

Compare with $\ell - 1$...?

Phase 1: Each ping we delete one leaf.

Compare with $\ell - 1$...?

Phase 1: Each ping we delete one leaf.

Phase 2: Every two pings we delete at least two leaves.

Compare with $\ell - 1$...?

Phase 1: Each ping we delete one leaf.

Phase 2: Every two pings we delete at least two leaves.

Phase 3: Each leaf takes at most two pings (except first/last).

Compare with $\ell - 1$...?

Phase 1: Each ping we delete one leaf.

Phase 2: Every two pings we delete at least two leaves.

Phase 3: Each leaf takes at most two pings (except first/last).

Phase 4: Half of the leaves take at most two pings.

Compare with $\ell - 1$...?

Phase 1: Each ping we delete one leaf.

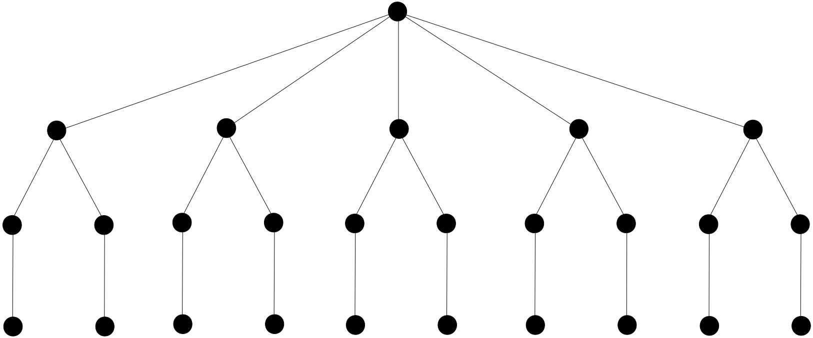
Phase 2: Every two pings we delete at least two leaves.

Phase 3: Each leaf takes at most two pings (except first/last).

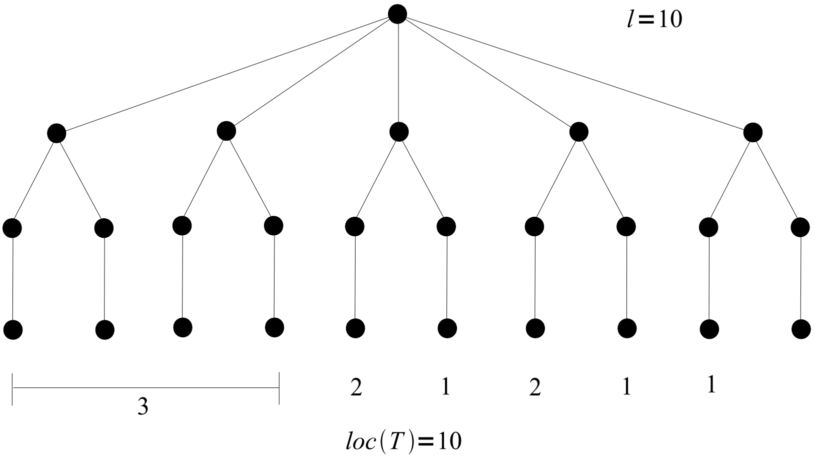
Phase 4: Half of the leaves take at most two pings.

$$\Rightarrow loc(T) < 2\ell.$$

Counter Example

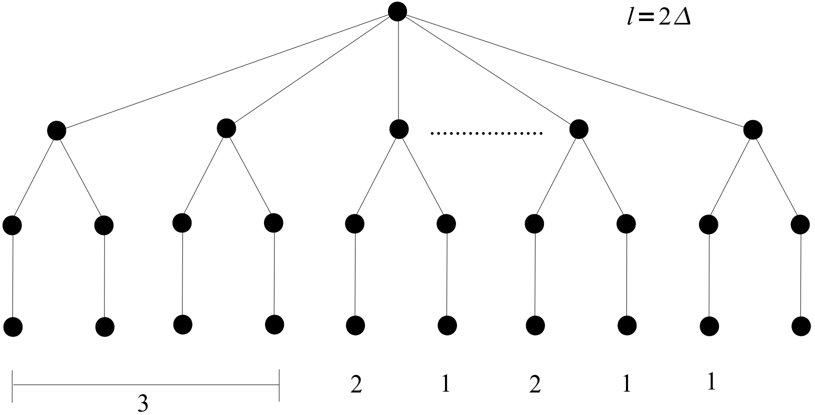


Counter Example



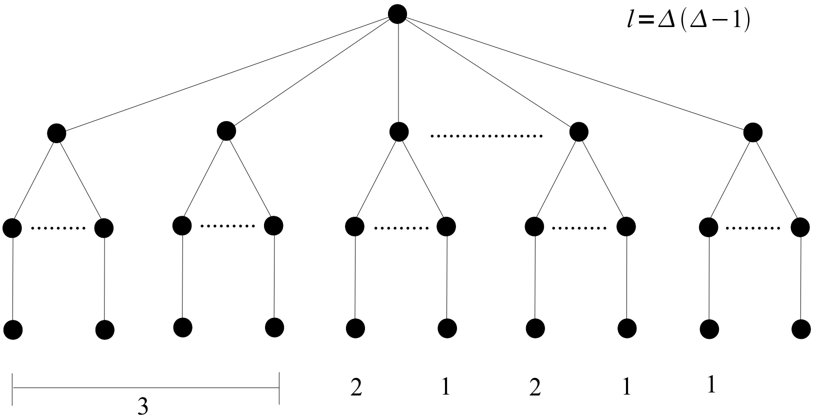
Counter Example

$l = 2\Delta$



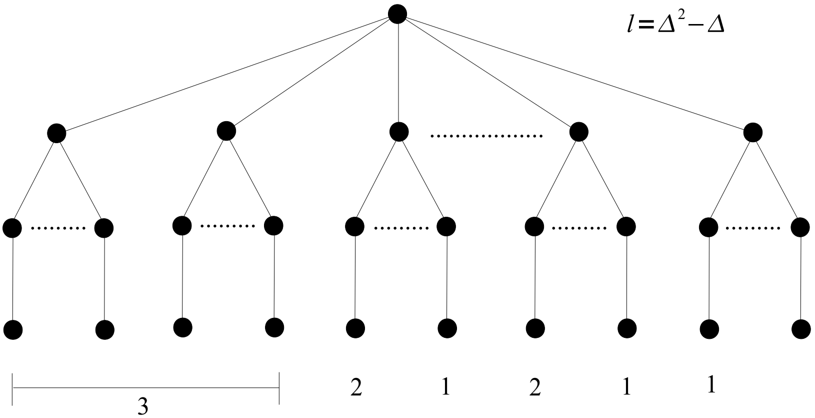
$loc(T) = 3\Delta - 5$

Counter Example



$$loc(T) = 3 + (2(\Delta-2)-1)(\Delta-3)+1$$

Counter Example



$$loc(T) = 2\Delta^2 - 11\Delta + 19$$

Future Work

Question

Lower Bound as a Function of ℓ?

Future Work

Question

Lower Bound as a Function of ℓ?

Question

Can we generalize this to larger classes?

Thank You

Thank You

Seager. **Locating a robber on a graph.** *Discrete Math.* 312 (2012), no. 22, 3265-3269.