Monotonicity of Non-deterministic Graph Searching

Frédéric Mazoit[1] Nicolas Nisse[2]

[1]LABRI, Université Bordeaux I, France.

[2]LRI, Université Paris-Sud, France.

WG'07, Dornburg, June 2007

Graph Searching

Goal

In an undirected simple graph, stand

- an invisible omniscient arbitrary fast fugitive ;
- a team of searchers ;

To find a strategy that catch the fugitive
using the fewest searchers as possible.

Motivations

Problem related to well known graphs'parameters :
treewidth and pathwidth.

Graph Searching

Goal

In an undirected simple graph, stand

- an invisible omniscient arbitrary fast fugitive ;
- a team of searchers ;

To find a strategy that catch the fugitive
using the fewest searchers as possible.

Motivations

Problem related to well known graphs'parameters :
treewidth and pathwidth.

Search Strategy, Parson. [GTC,1978] Variant of Kirousis and Papadimitriou. [TCS,86]

Sequence of two basic operations,...

1. Place a searcher at a vertex of the graph ;
2. Remove a searcher from a vertex of the graph.

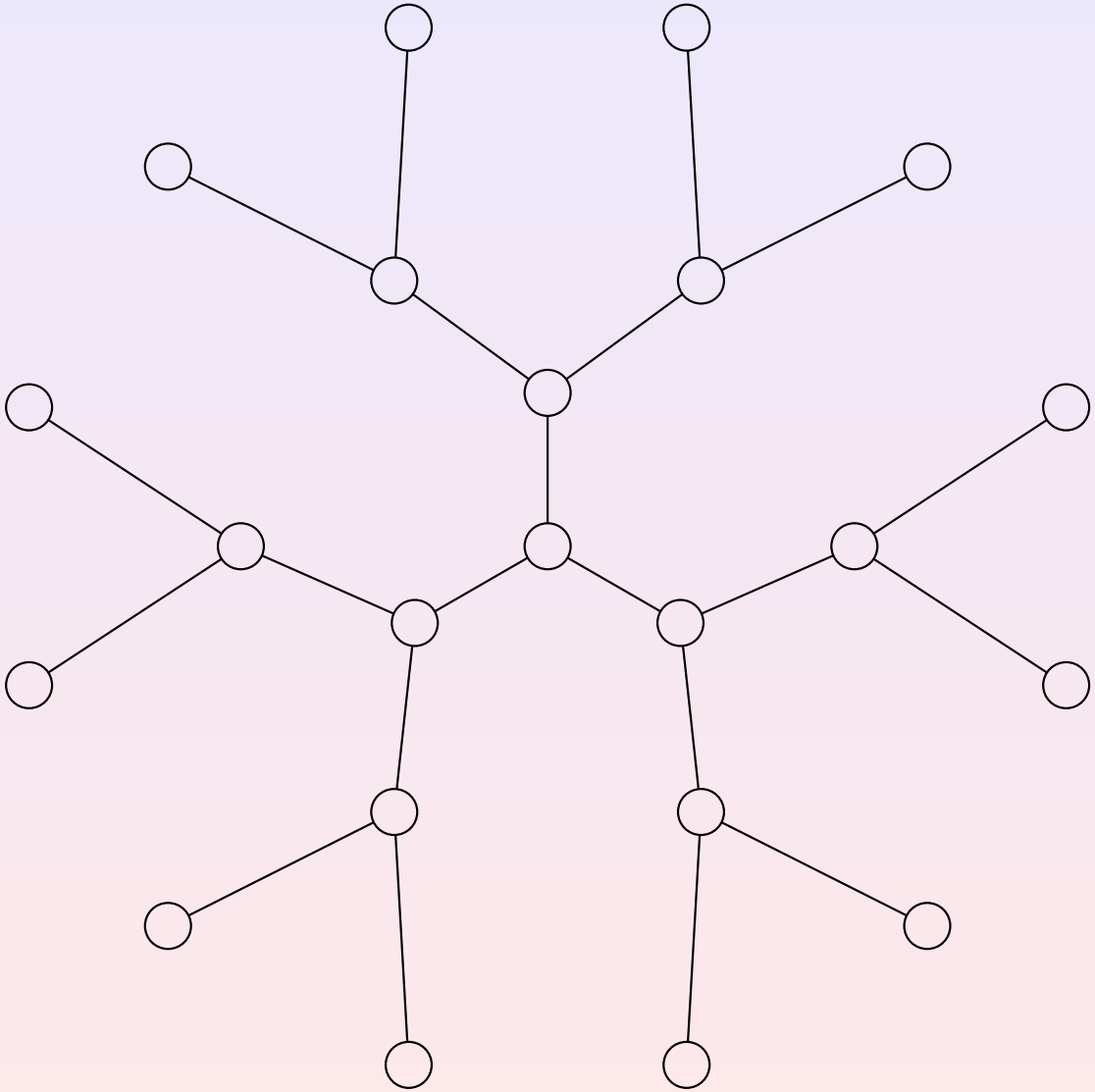
...that must result in catching the fugitive

The fugitive is caugth when it occupies (or crosses) a vertex occupied by a searcher.

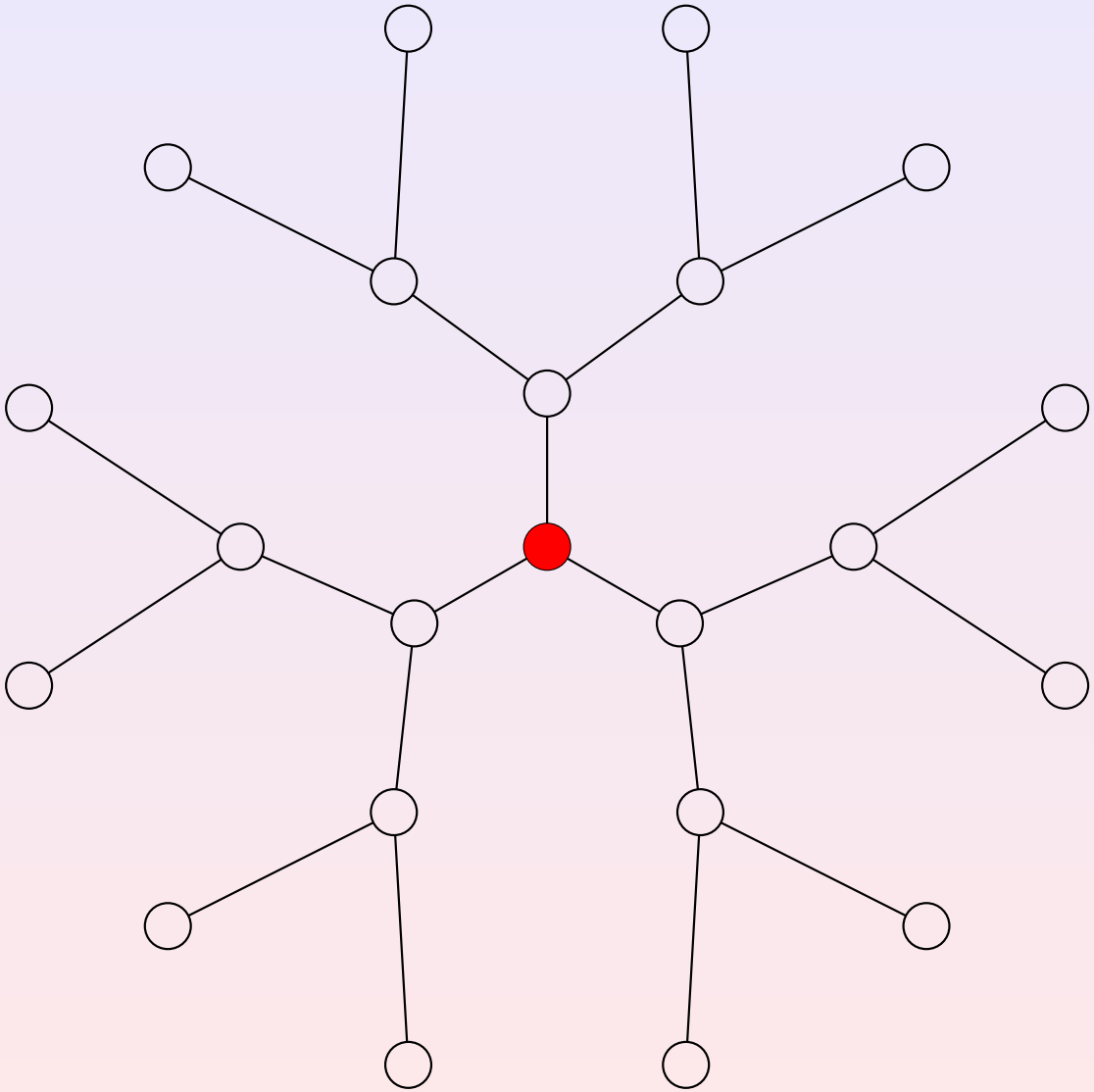
We want to minimize the number of searchers.

Let $s(G)$ be the smallest number of searchers needed to catch an invisible fugitive in a graph G.

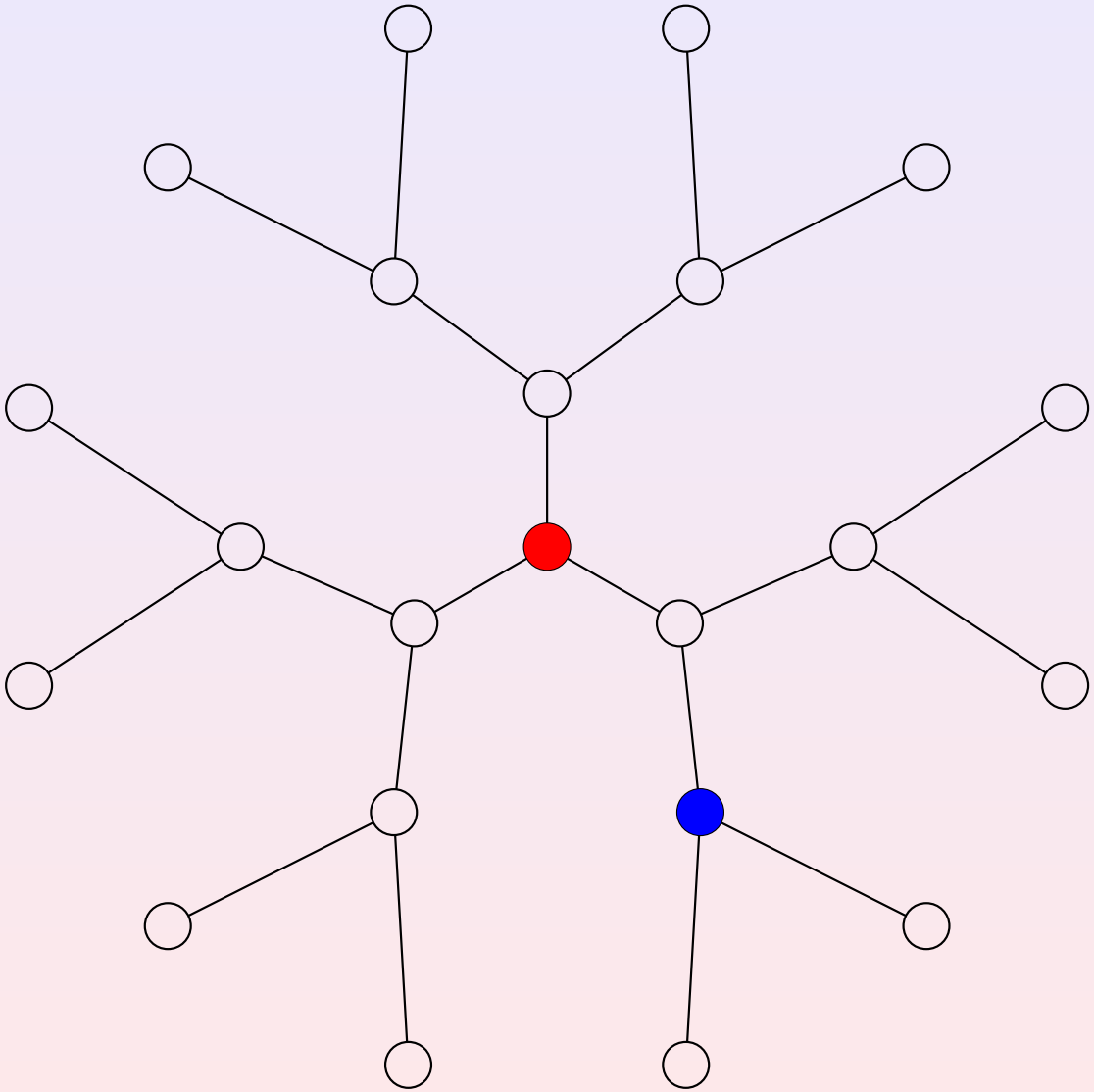
A simple example : a ternary tree



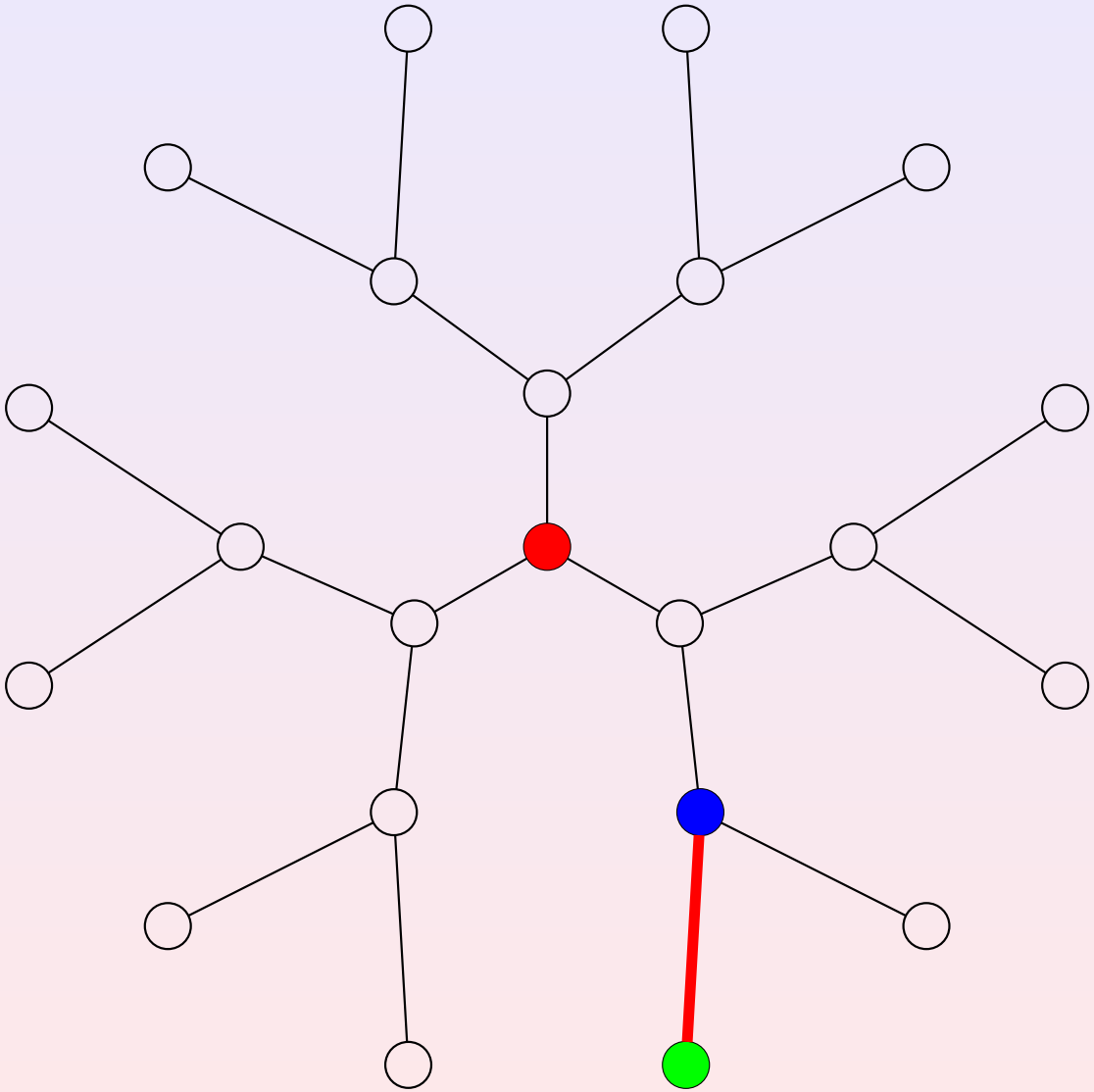
A simple example : a ternary tree



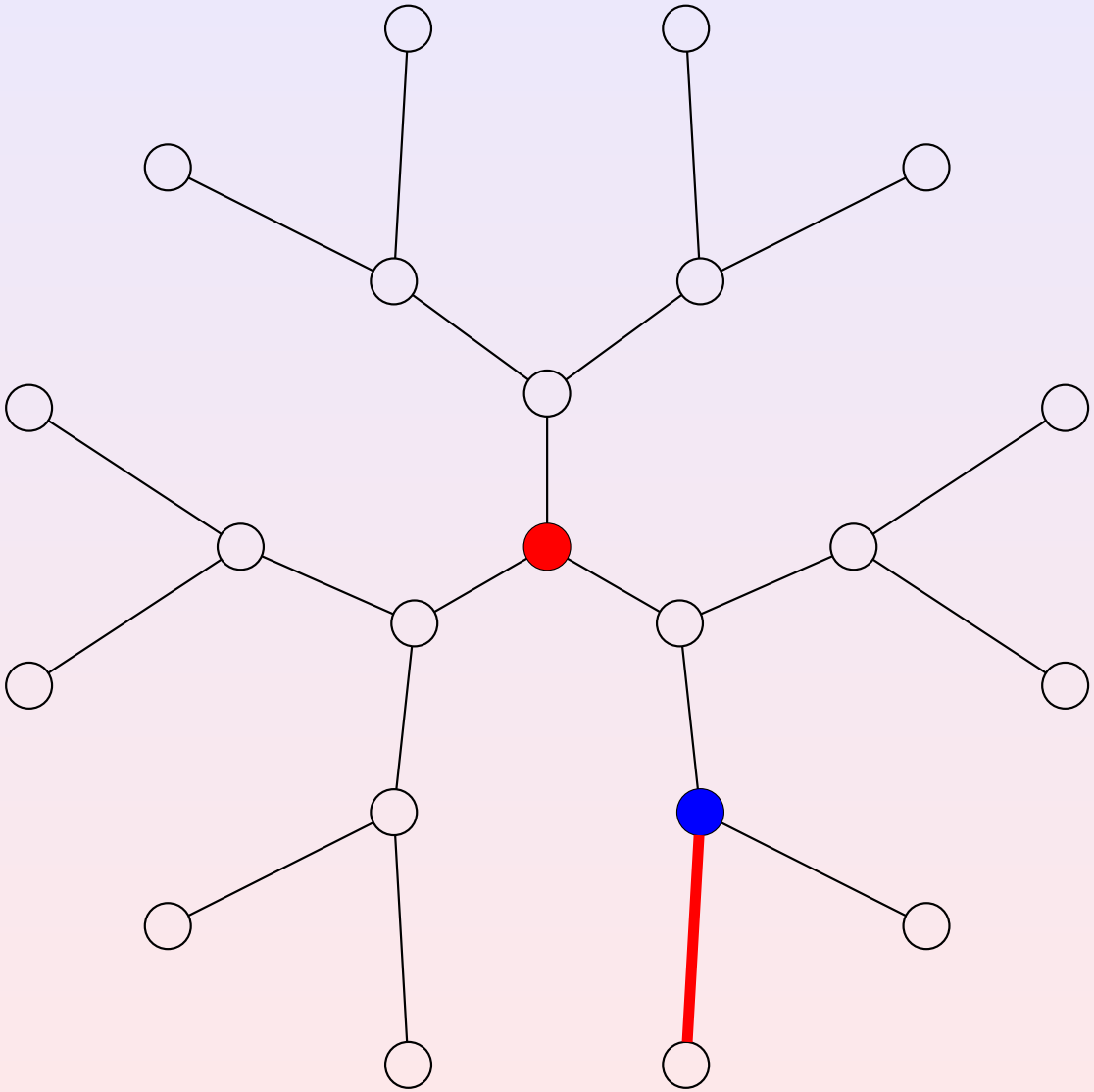
A simple example : a ternary tree



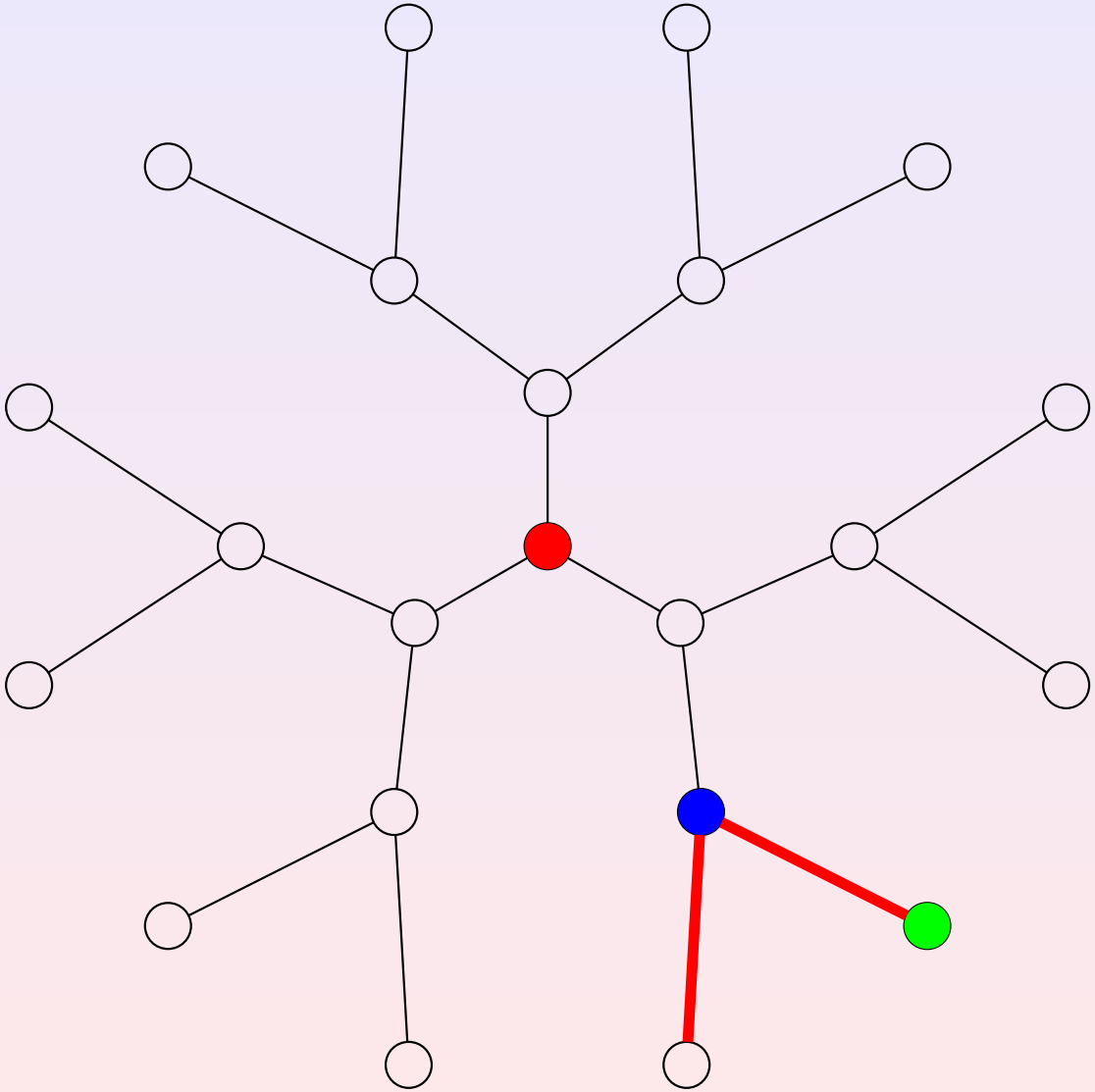
A simple example : a ternary tree



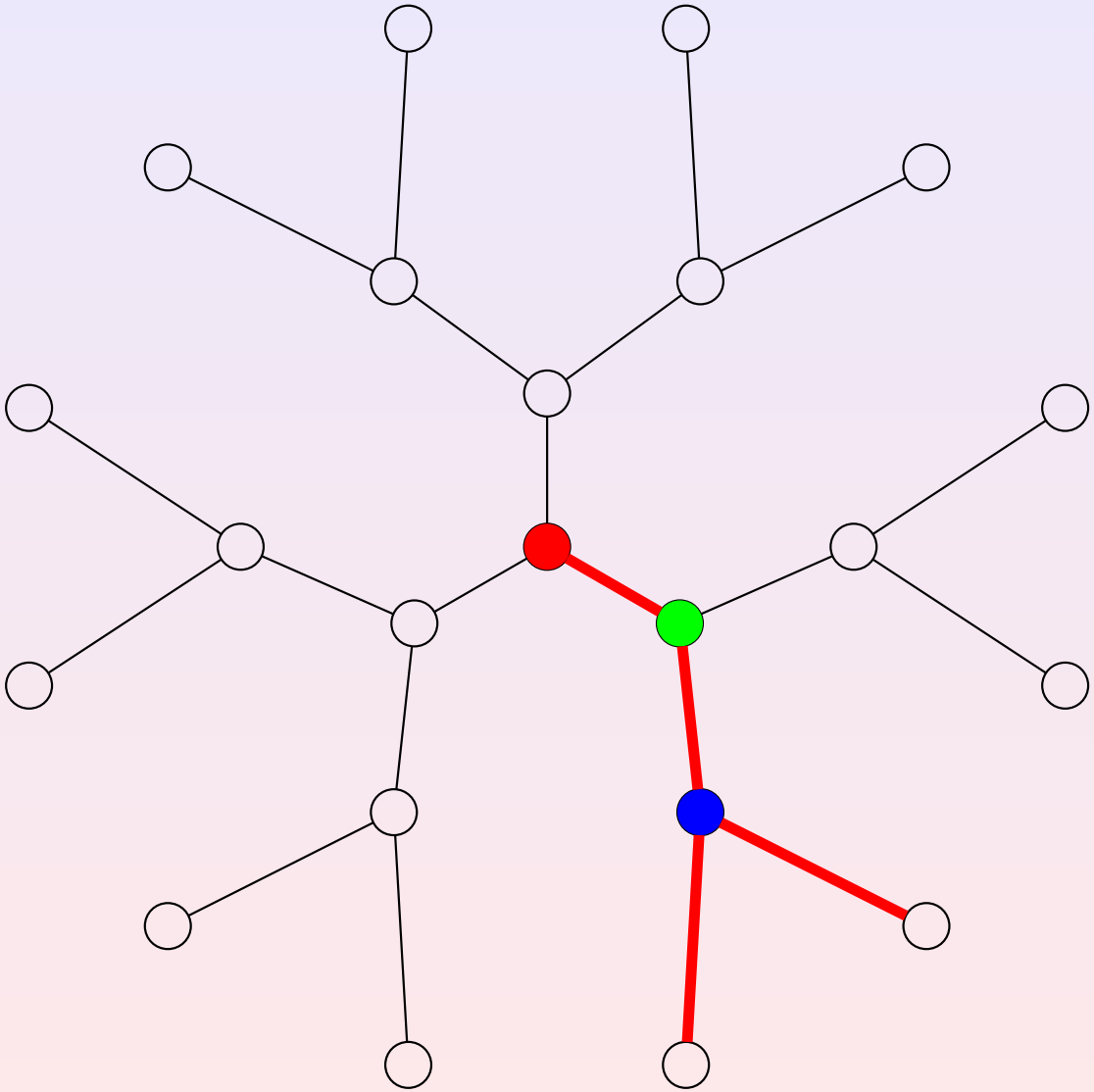
A simple example : a ternary tree



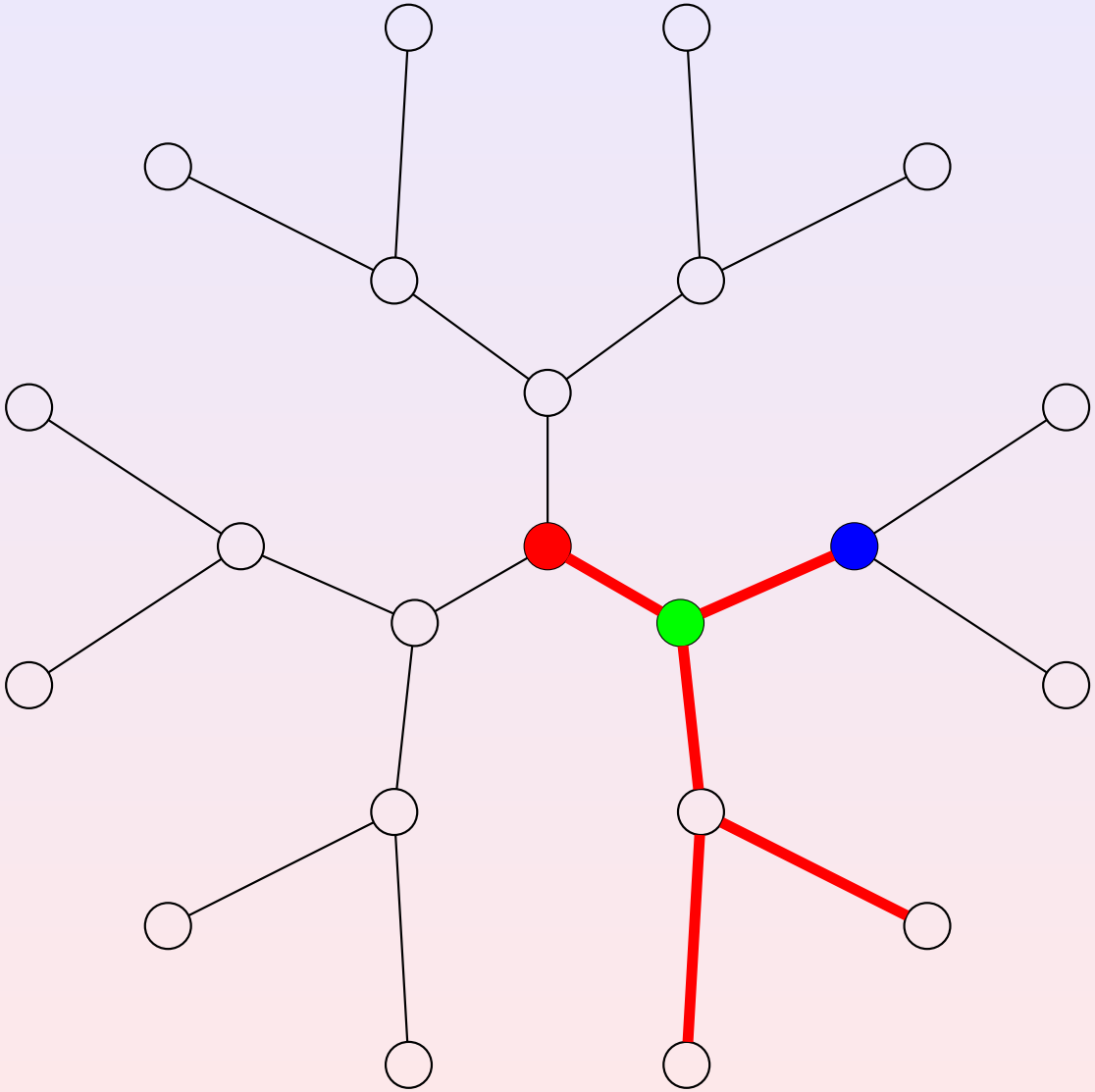
A simple example : a ternary tree



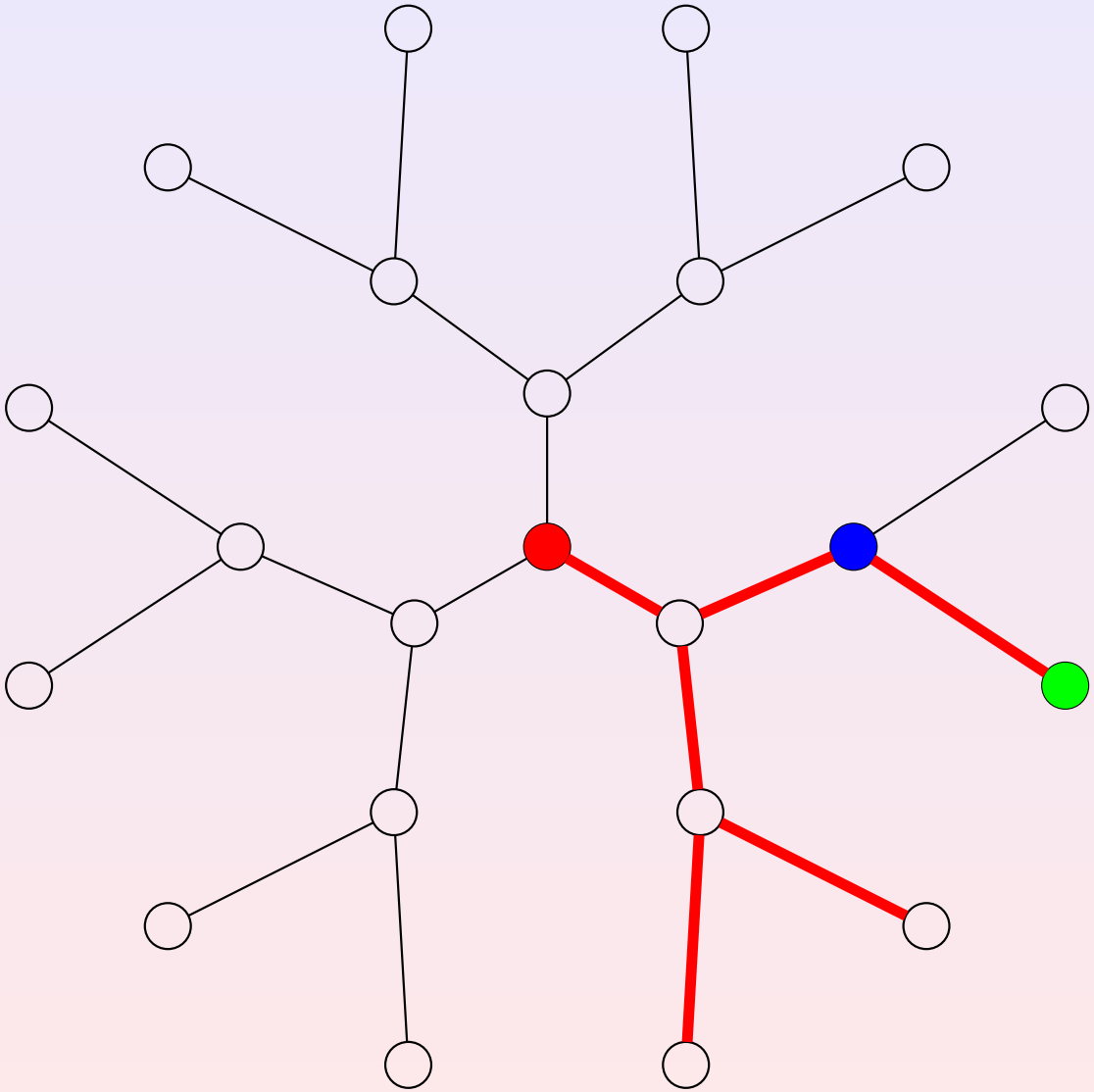
A simple example : a ternary tree



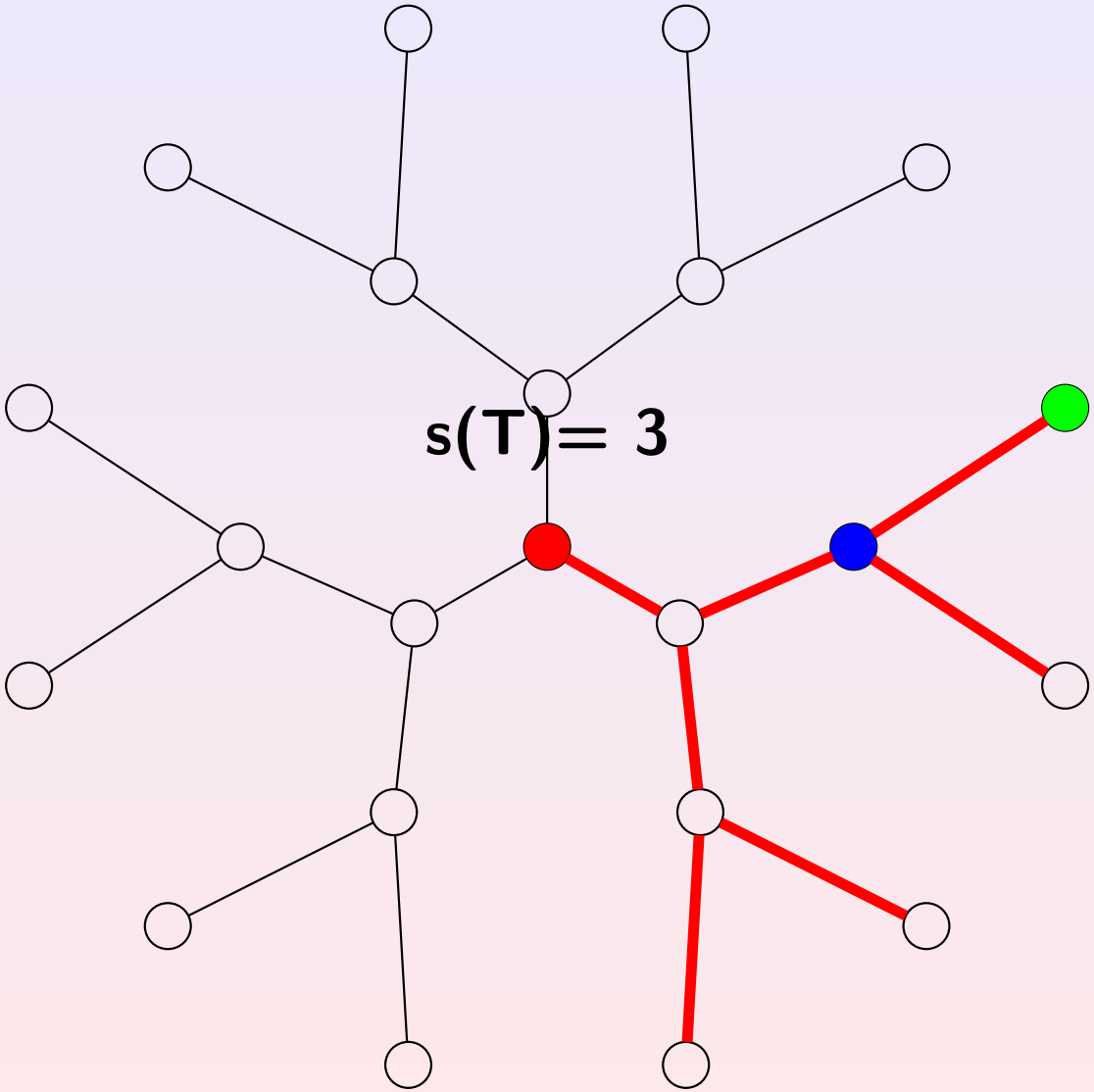
A simple example : a ternary tree



A simple example : a ternary tree



A simple example : a ternary tree



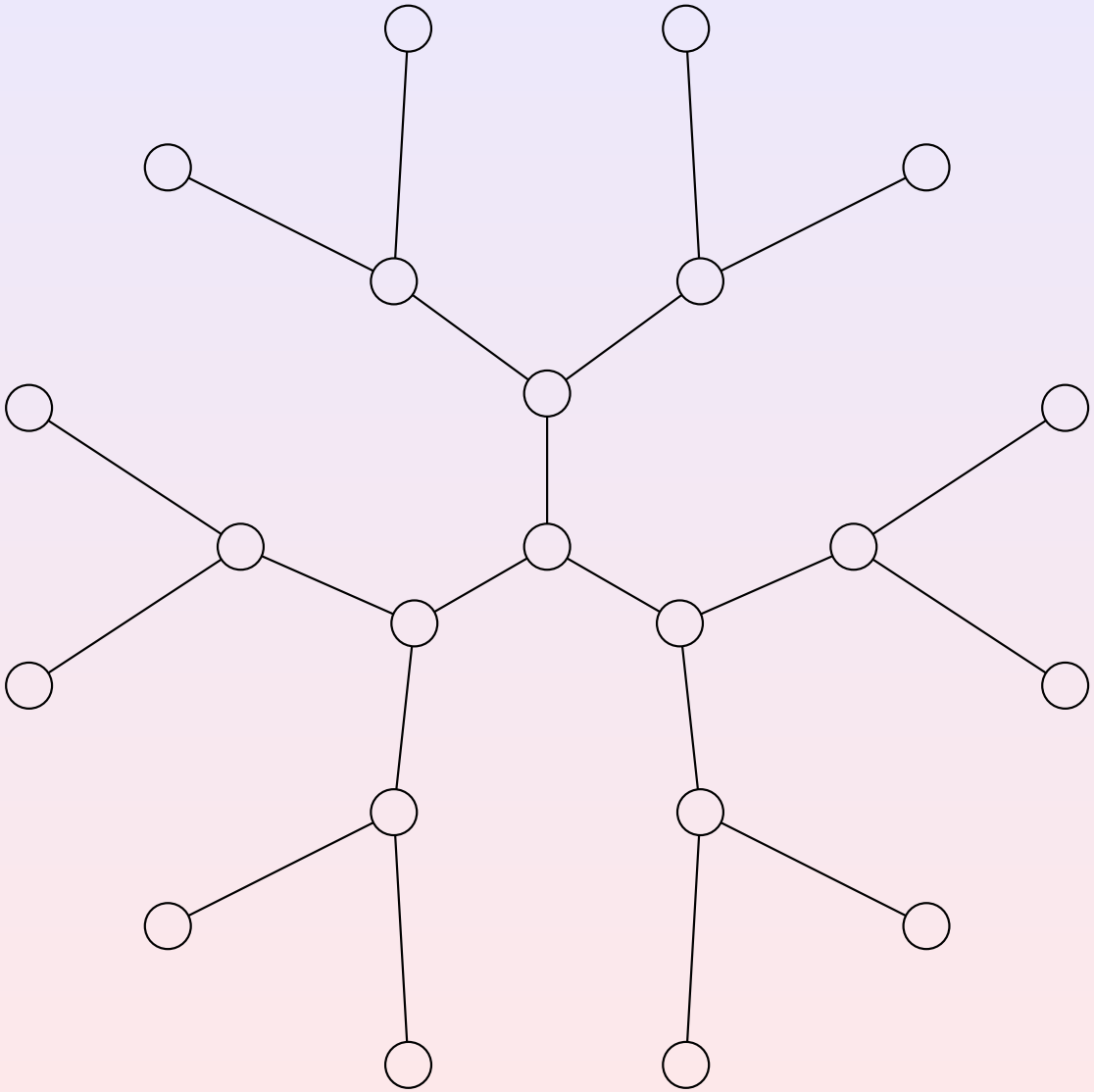
Visible Graph Searching

Visible fugitive

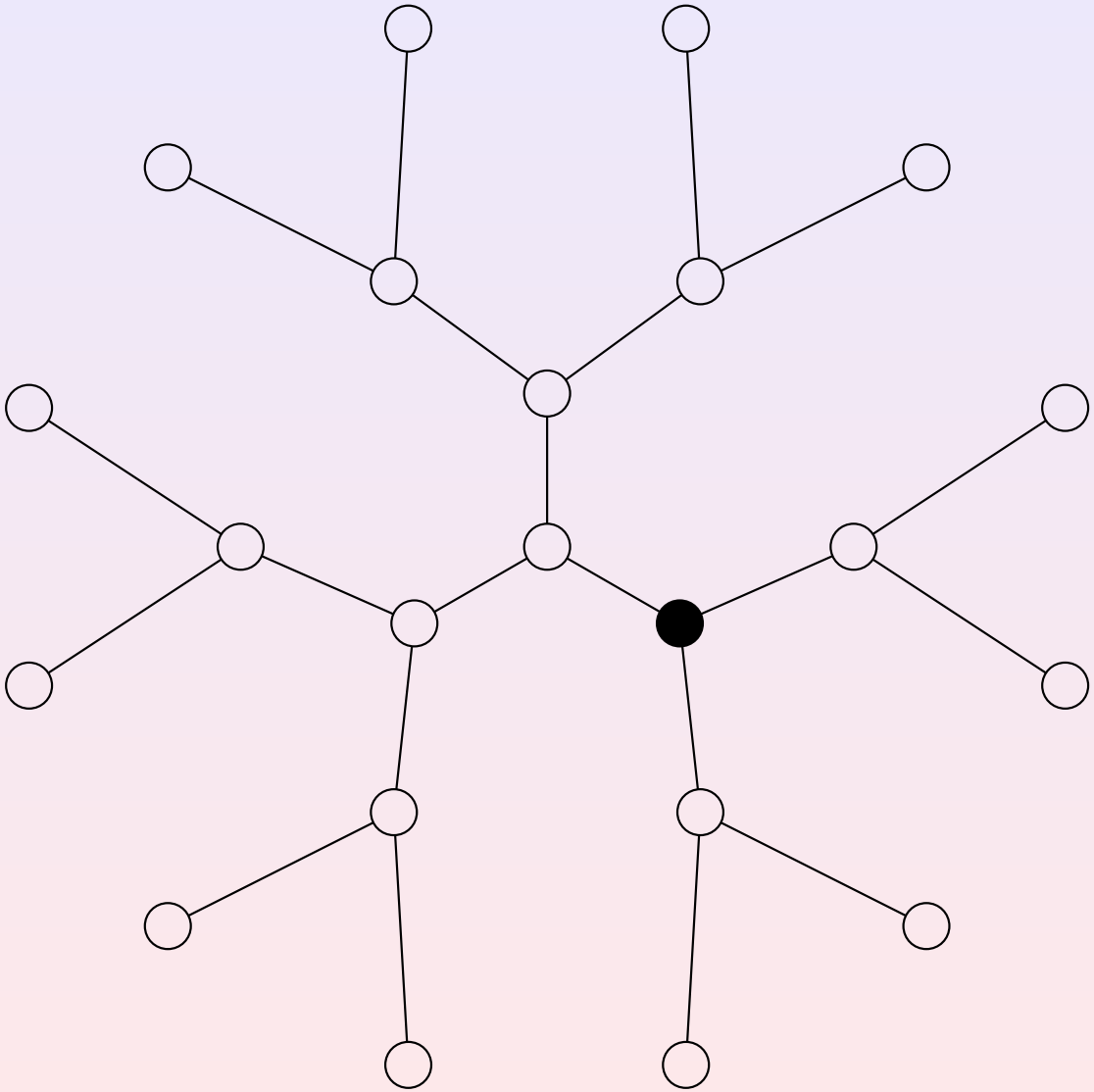
The fugitive is visible if, at every step, searchers know its position.

Let $vs(G)$ be the visible search number of the graph G.
Obviously, $vs(G) \leq s(G)$ for any graph G.

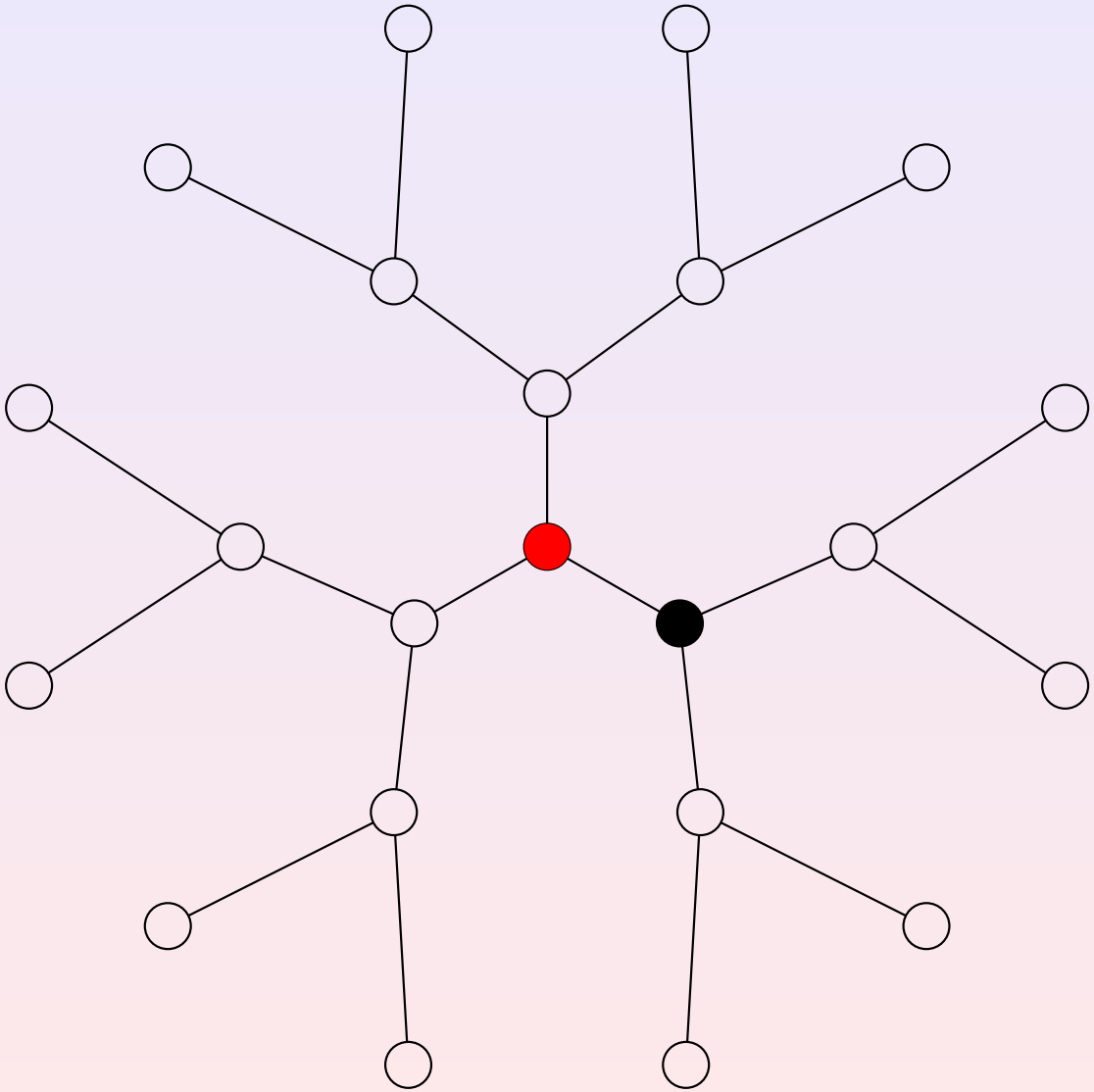
Visible graph searching in a tree



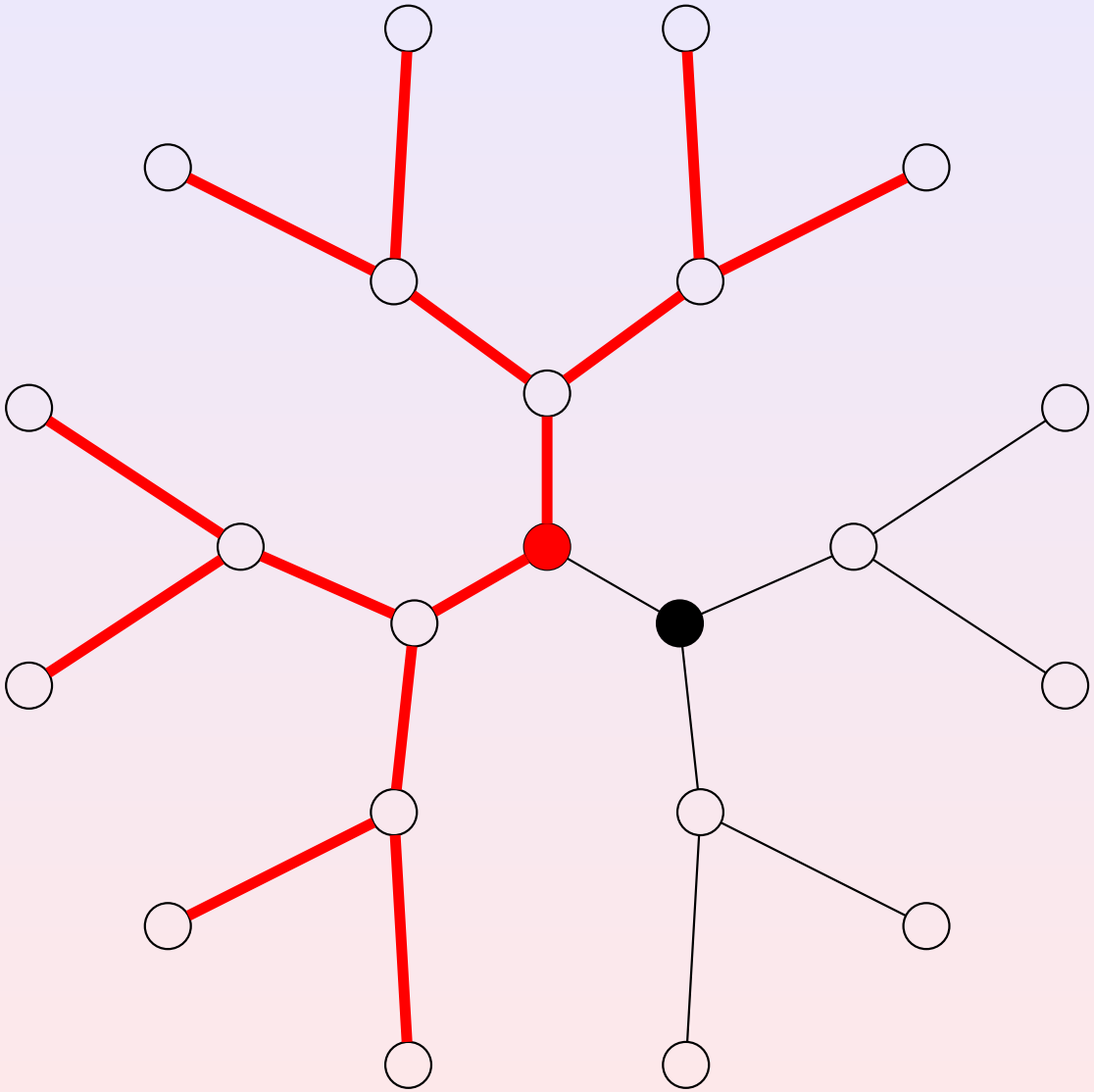
Visible graph searching in a tree



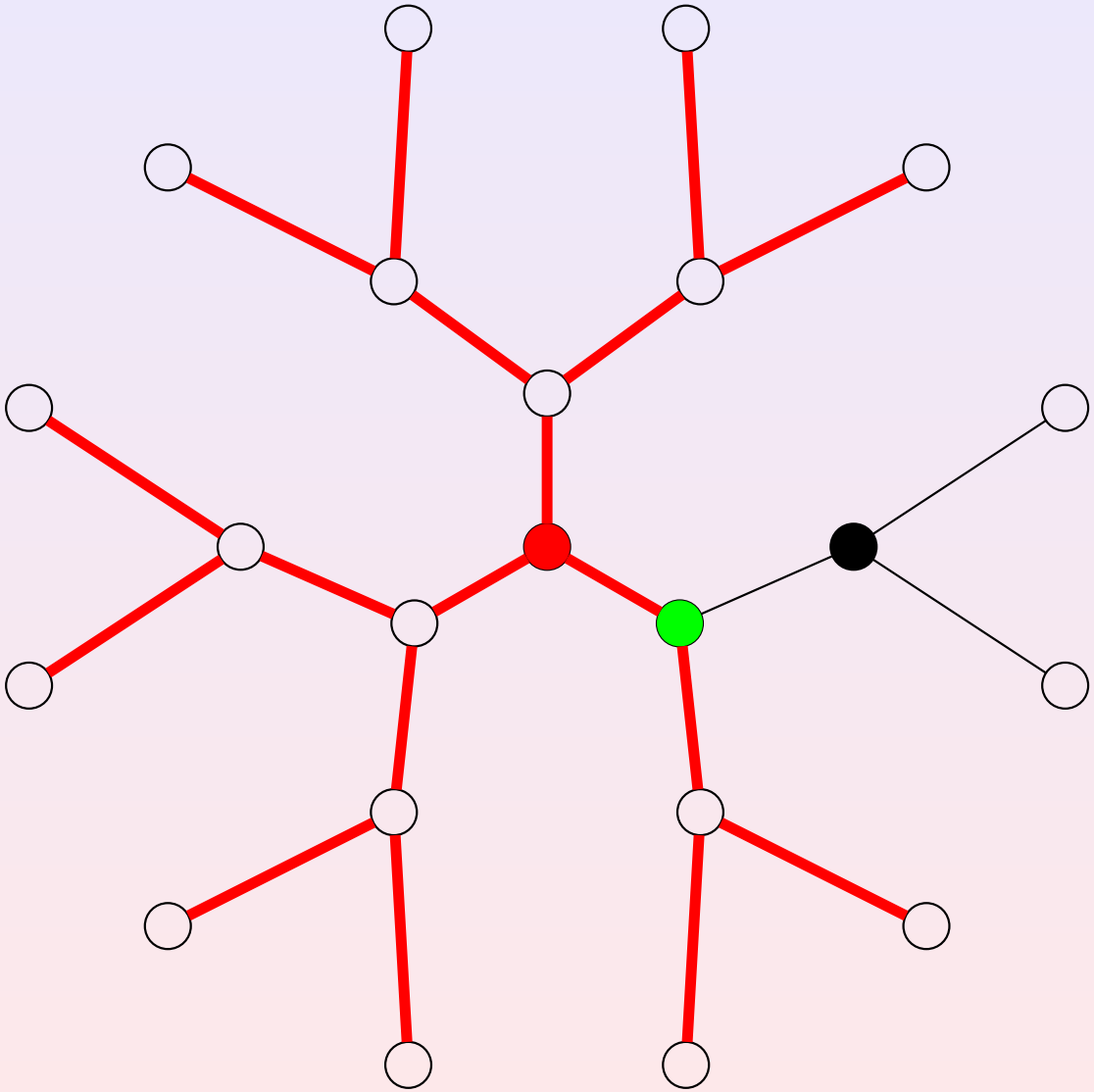
Visible graph searching in a tree



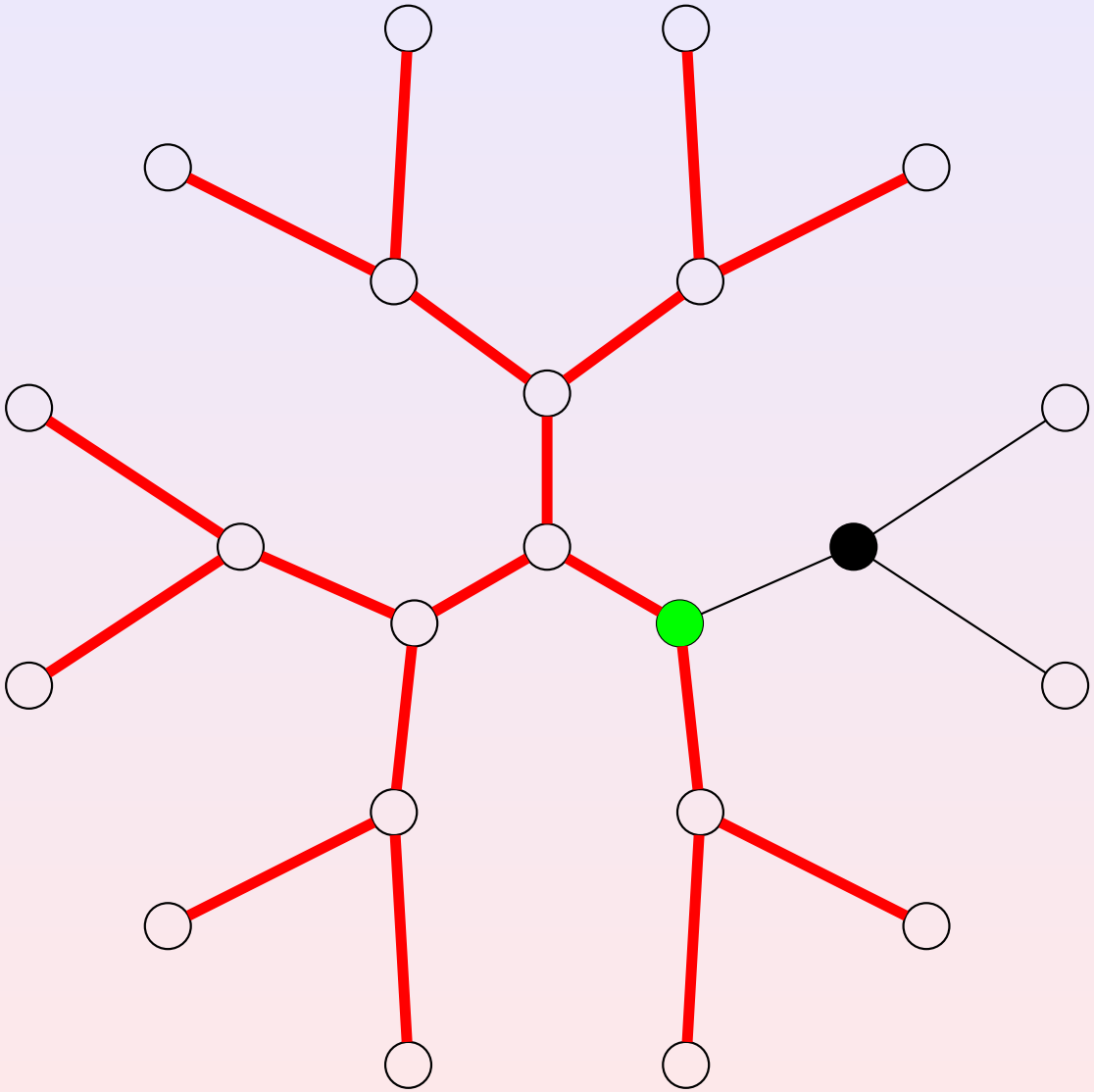
Visible graph searching in a tree



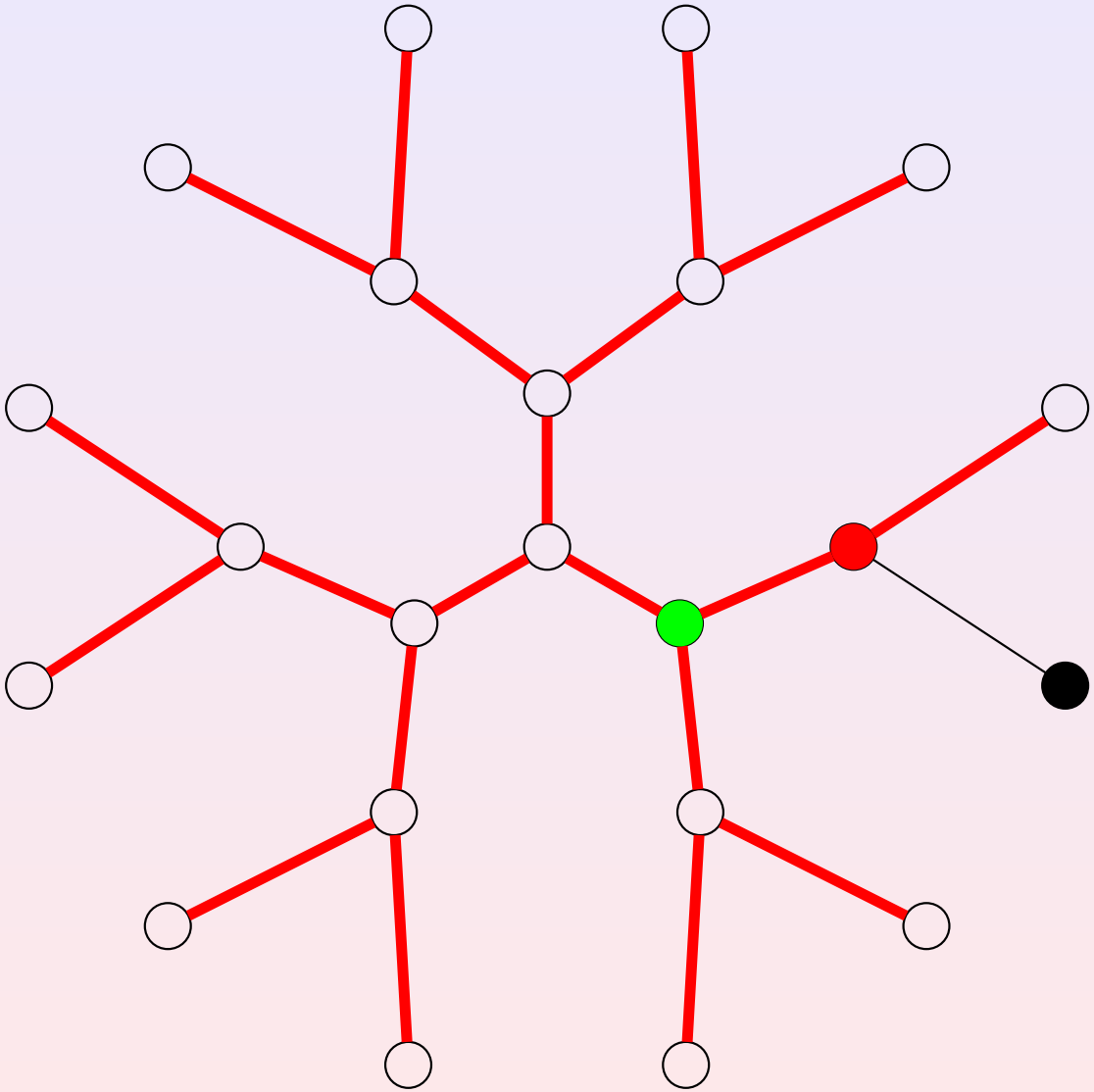
Visible graph searching in a tree



Visible graph searching in a tree

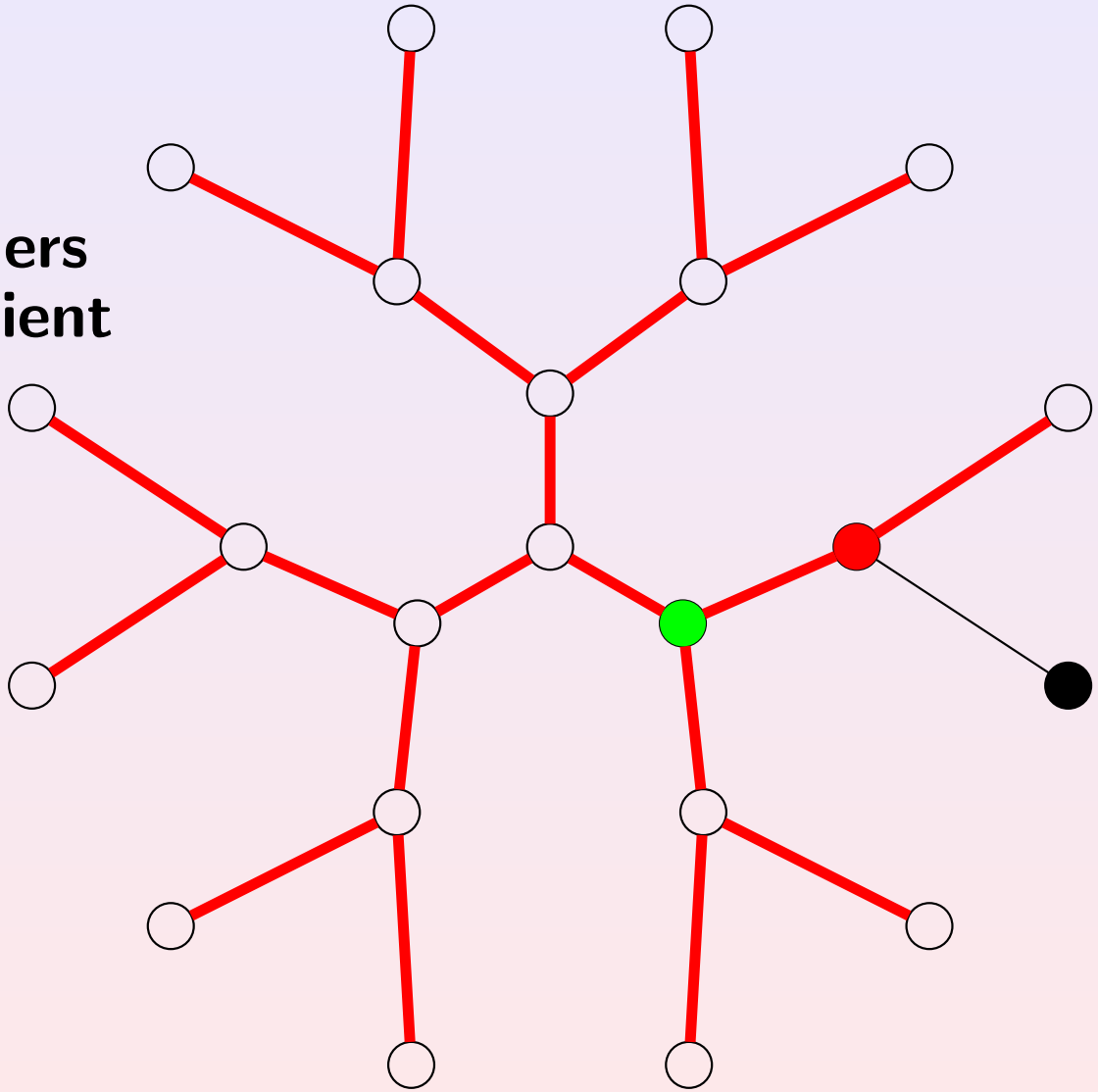


Visible graph searching in a tree



Visible graph searching in a tree

2 searchers are sufficient



Complexity and Monotonicity

s$(G) \leq k$ is NP-hard

- **Megiddo** *et al*, J.of ACM, 1988
  The complexity of searching a graph.

vs$(G) \leq k$ is NP-hard

- **Seymour and Thomas**, J. of Comb. Th., 1993.
  Graph searching and a min-max theorem for tree-width

Question : NP-membership ?
A strategy is a certificate, but what is its size ?

Complexity and Monotonicity

$\mathbf{s}(G) \leq k$ is NP-hard

- **Megiddo** *et al*, J.of ACM, 1988
  The complexity of searching a graph.

$\mathbf{vs}(G) \leq k$ is NP-hard

- **Seymour and Thomas**, J. of Comb. Th., 1993.
  Graph searching and a min-max theorem for tree-width

Question : NP-membership ?
A strategy is a certificate, but what is its size ?

Monotonicity

Monotone strategies

A search strategy is monotone if the cleared part can only increase.

$\Leftrightarrow$ any vertex is occupied only once.

$\Leftrightarrow$ *recontamination* never occurs.

A monotone strategy consists of a polynomial number of steps

Question : Does recontamination help ?

In other words, for any graph, does there always exist a monotone strategy using the smallest number of searchers ?

Monotonicity

Monotone strategies

A search strategy is monotone if the cleared part can only increase.

$\Leftrightarrow$ any vertex is occupied only once.

$\Leftrightarrow$ *recontamination* never occurs.

A monotone strategy consists of a polynomial number of steps

Question : Does recontamination help ?

In other words, for any graph, does there always exist a monotone strategy using the smallest number of searchers ?

Monotonicity

Monotone strategies

A search strategy is monotone if the cleared part can only increase.

$\Leftrightarrow$ any vertex is occupied only once.

$\Leftrightarrow$ *recontamination* never occurs.

A monotone strategy consists of a polynomial number of steps

Question : Does recontamination help ?

In other words, for any graph, does there always exist a monotone strategy using the smallest number of searchers ?

Recontamination does not help

Case of an invisible fugitive

- **Bienstock and Seymour**, J.of Alg., 1991
  Monotonicity in graph searching.
- **LaPaugh**, J.of ACM, 1993
  Recontamination does not help to search a graph.

Constructive proofs : *LaPaugh*

- Any strategy using k searchers
- can be turn into a *monotone* strategy using $\leq k$ searchers.

Recontamination does not help

Case of an invisible fugitive

- **Bienstock and Seymour**, J.of Alg., 1991
  Monotonicity in graph searching.
- **LaPaugh**, J.of ACM, 1993
  Recontamination does not help to search a graph.

Constructive proofs : *Bienstock and Seymour*

- strategy using k searchers
- $\Rightarrow$ crusade of width $\leq k$
- $\Rightarrow$ *monotone* crusade of width $\leq k$
- $\Rightarrow$ *monotone* strategy using $\leq k$ searchers.

Recontamination does not help

Case of a visible fugitive

- **Seymour and Thomas**, J. of Comb. Th., 1993.
  Graph searching and a min-max theorem for tree-width

Non constructive proof :

- no monotone strategy using $\leq k$ searchers
- $\Rightarrow$ evasion strategy for a fugitive versus $\leq k$ "monotone" searchers
- $\Rightarrow$ evasion strategy for a fugitive versus $\leq k$ searchers
- $\Rightarrow$ no strategy using $\leq k$ searchers

Recontamination does not help

Consequences

- NP-membership
- relationship invisible search number/pathwidth **pw** : $\mathbf{s}(G) = \mathbf{pw}(G) + 1$
- relationship visible search number/treewidth **tw** : $\mathbf{vs}(G) = \mathbf{tw}(G) + 1$

Non-deterministic Graph Searching

Invisible fugitive
An Oracle permanently knows the position of the fugitive

One extra operation

Searchers can perform a query to the oracle :
"What is the current position of the fugitive ?"

A non-deterministic search strategy consists of a sequence of the following basic operations :

- Place a searcher at a vertex of the graph ;
- Remove a searcher from a vertex of the graph ;
- Perform a query to the Oracle.

Non-deterministic Graph Searching

Invisible fugitive
An Oracle permanently knows the position of the fugitive

One extra operation

Searchers can perform a query to the oracle :
"What is the current position of the fugitive ?"

A non-deterministic search strategy consists of a sequence of the following basic operations :

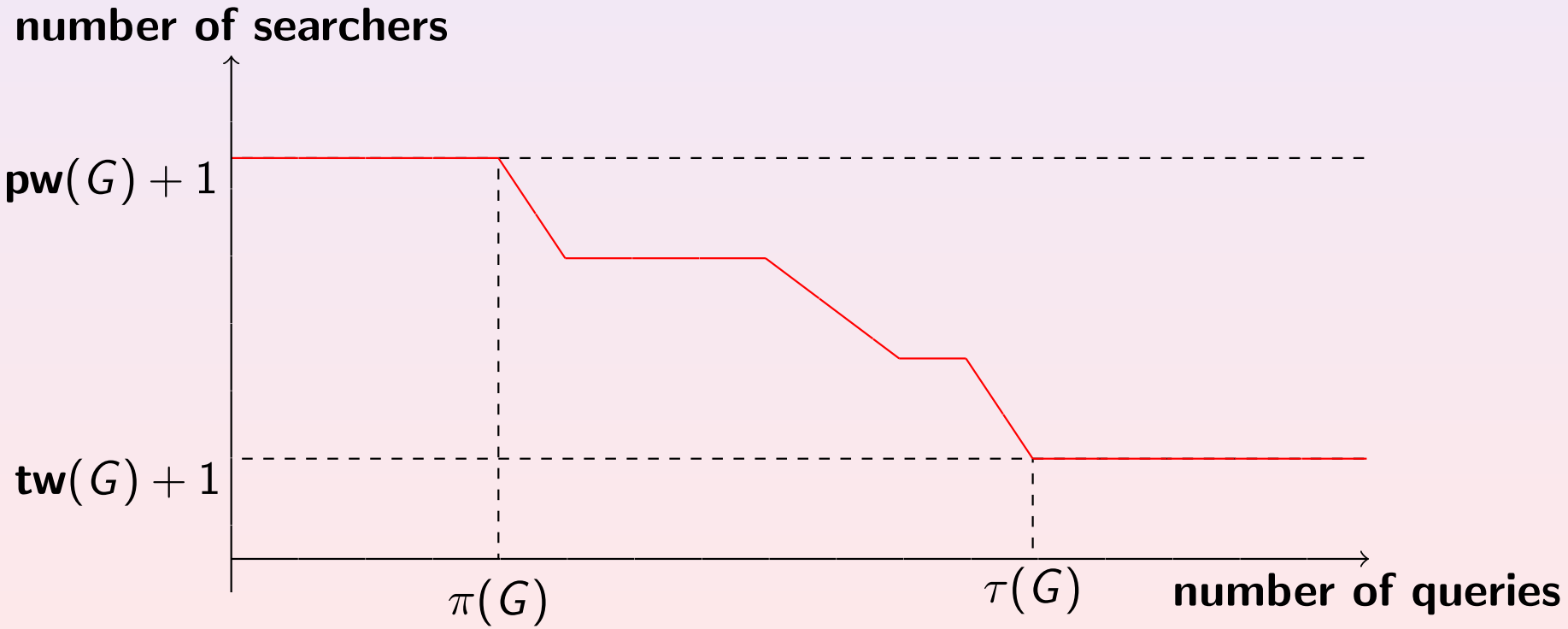
- Place a searcher at a vertex of the graph ;
- Remove a searcher from a vertex of the graph ;
- Perform a query to the Oracle.

q-limited search number

Tradeoff number of searchers / number of queries

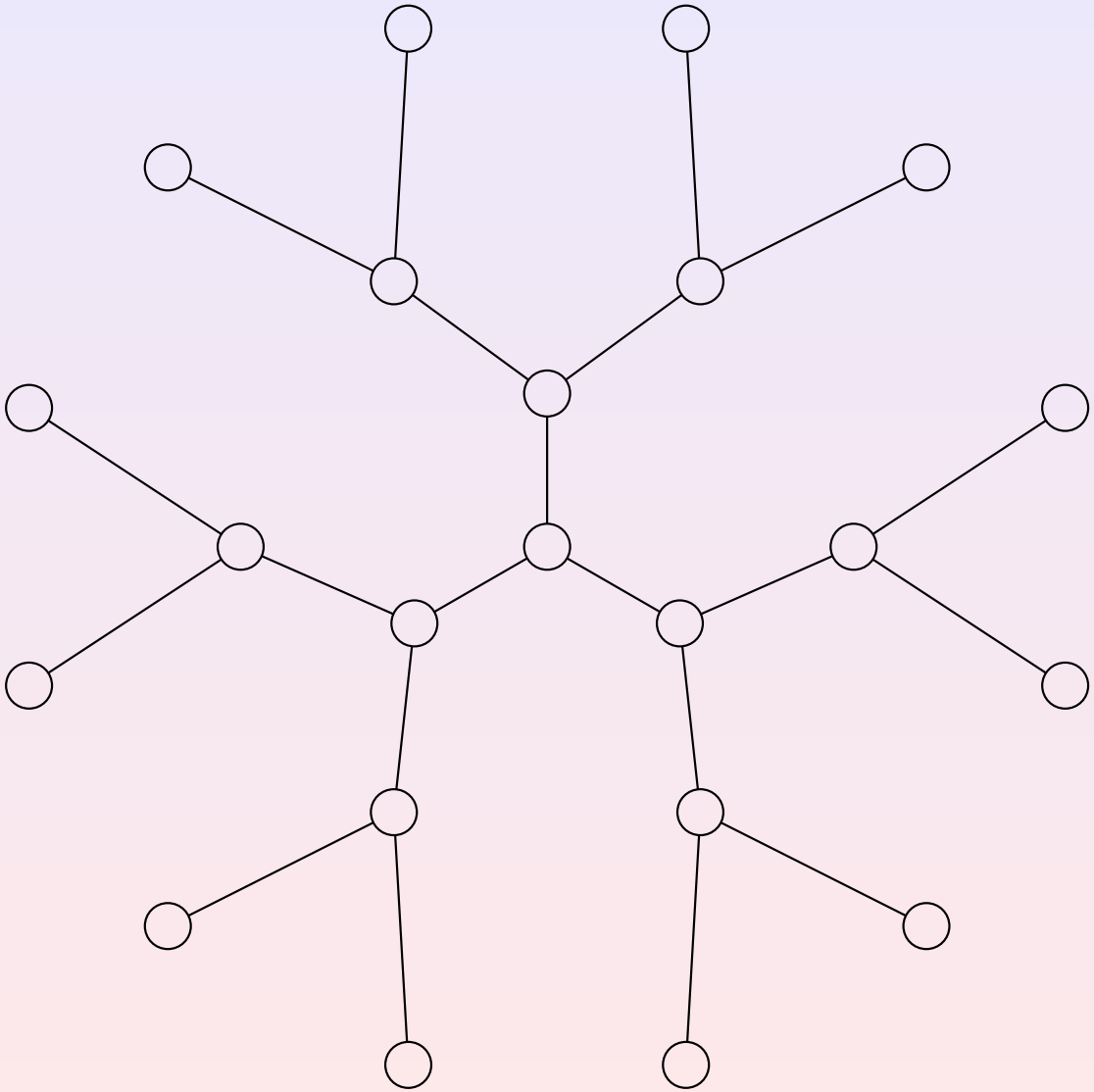
q-limited (non-deterministic) search number, $s_q(G)$

- $\mathbf{s}_0(G) = \mathbf{pw}(G) + 1$, invisible search number of G ;
- $\mathbf{s}_\infty(G) = \mathbf{tw}(G) + 1$, visible search number of G.



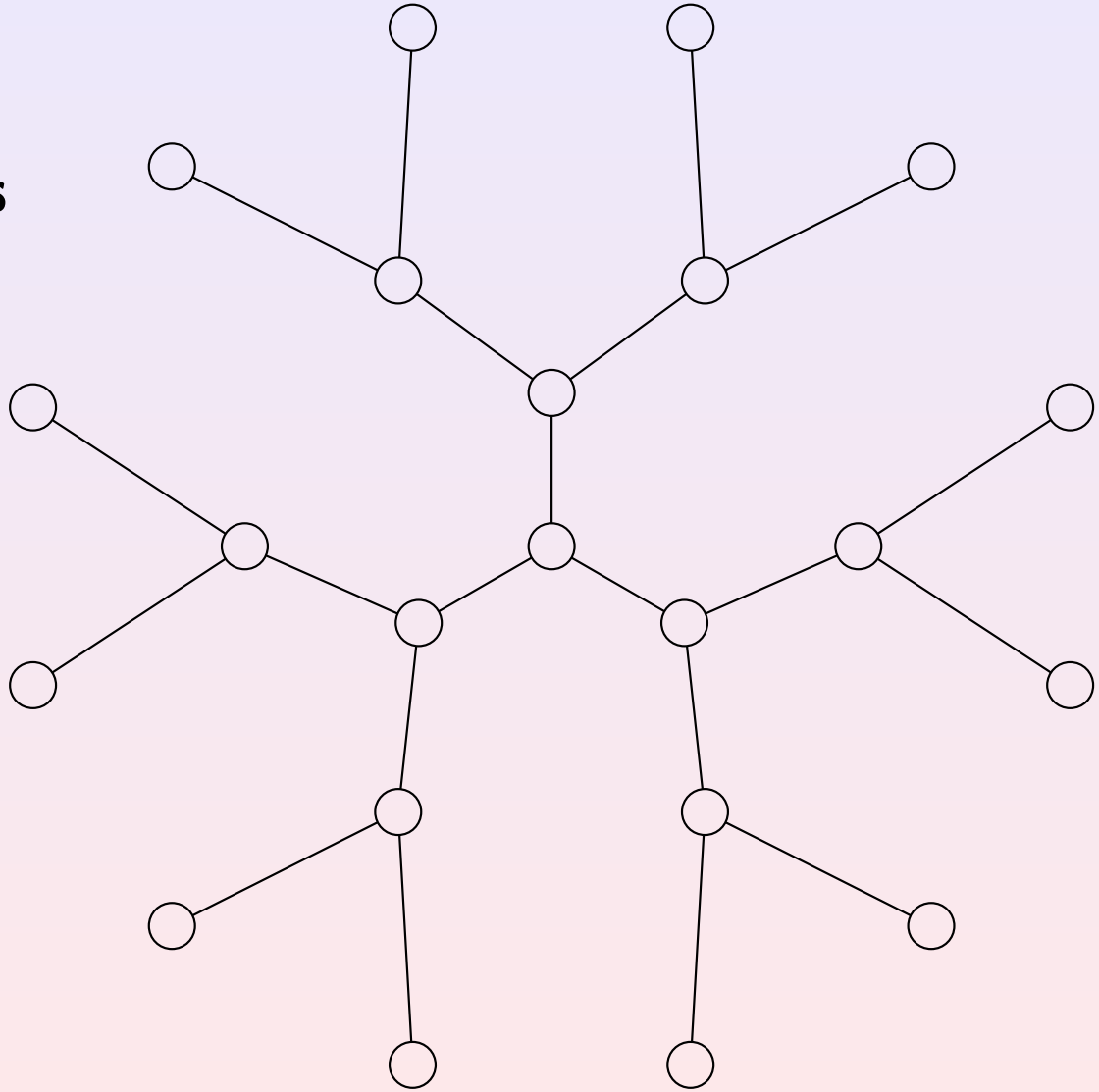
Still the same example :

$s_0(T)=3$



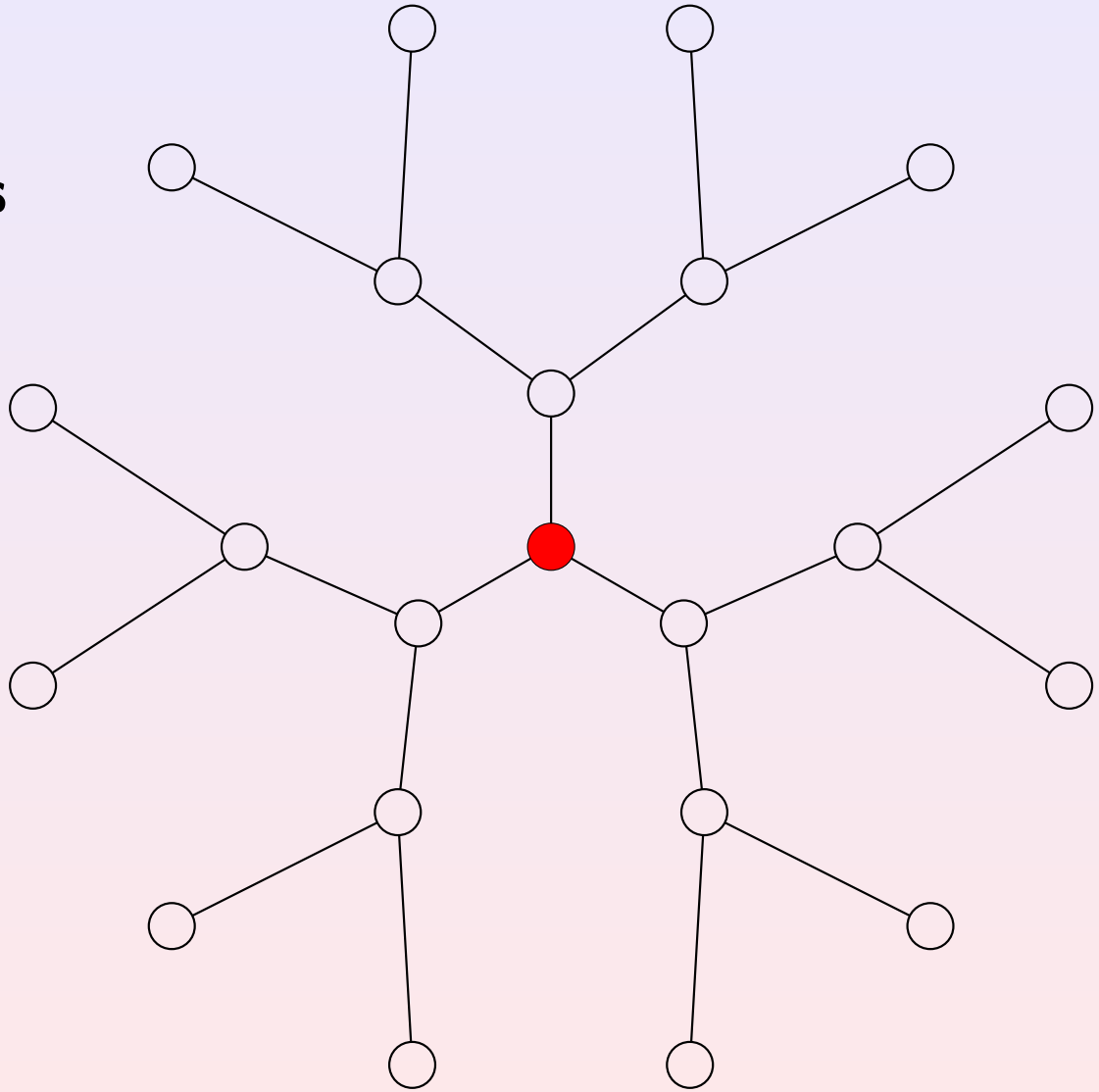
Still the same example :

2 queries



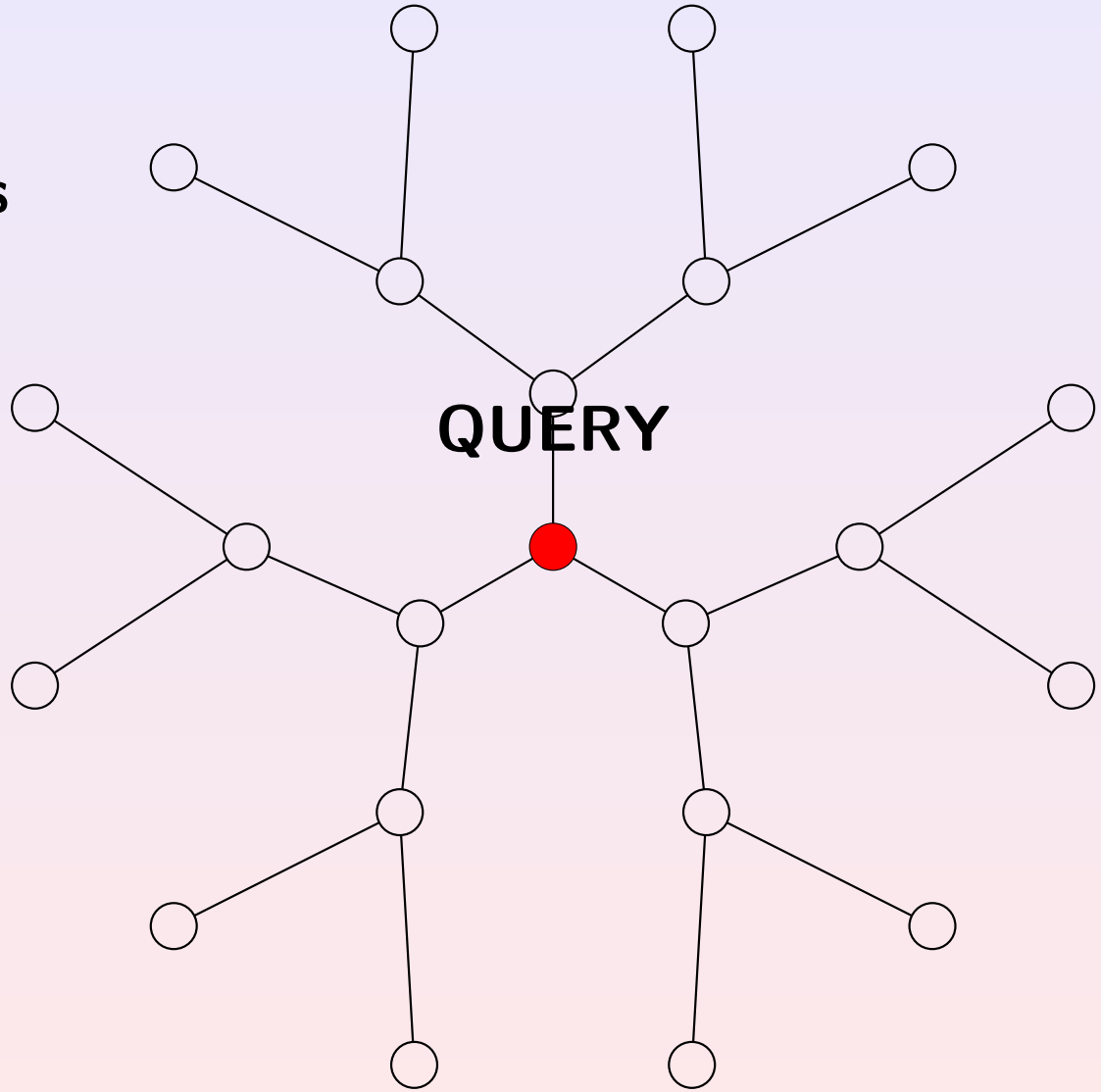
Still the same example :

2 queries



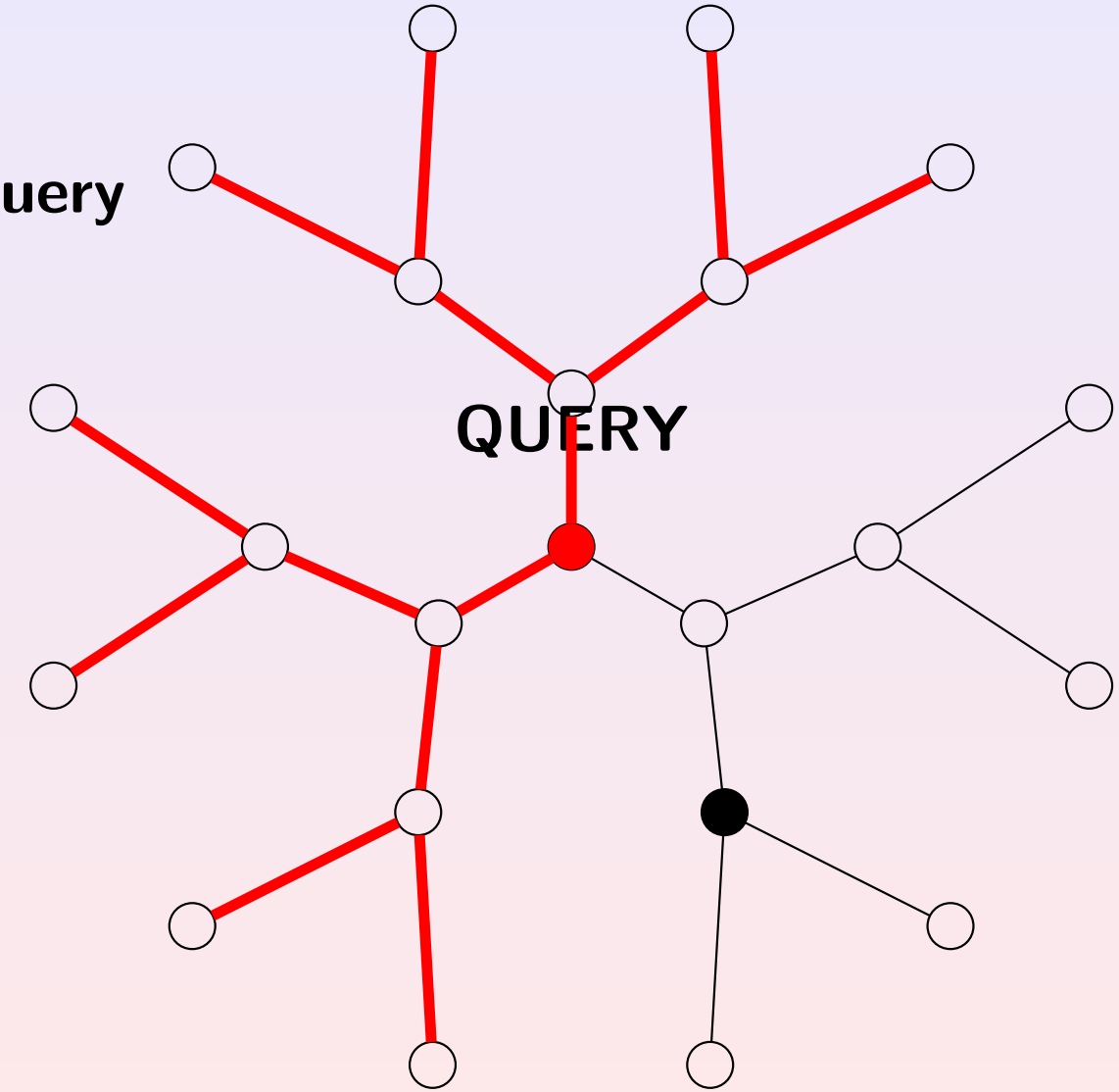
Still the same example :

2 queries



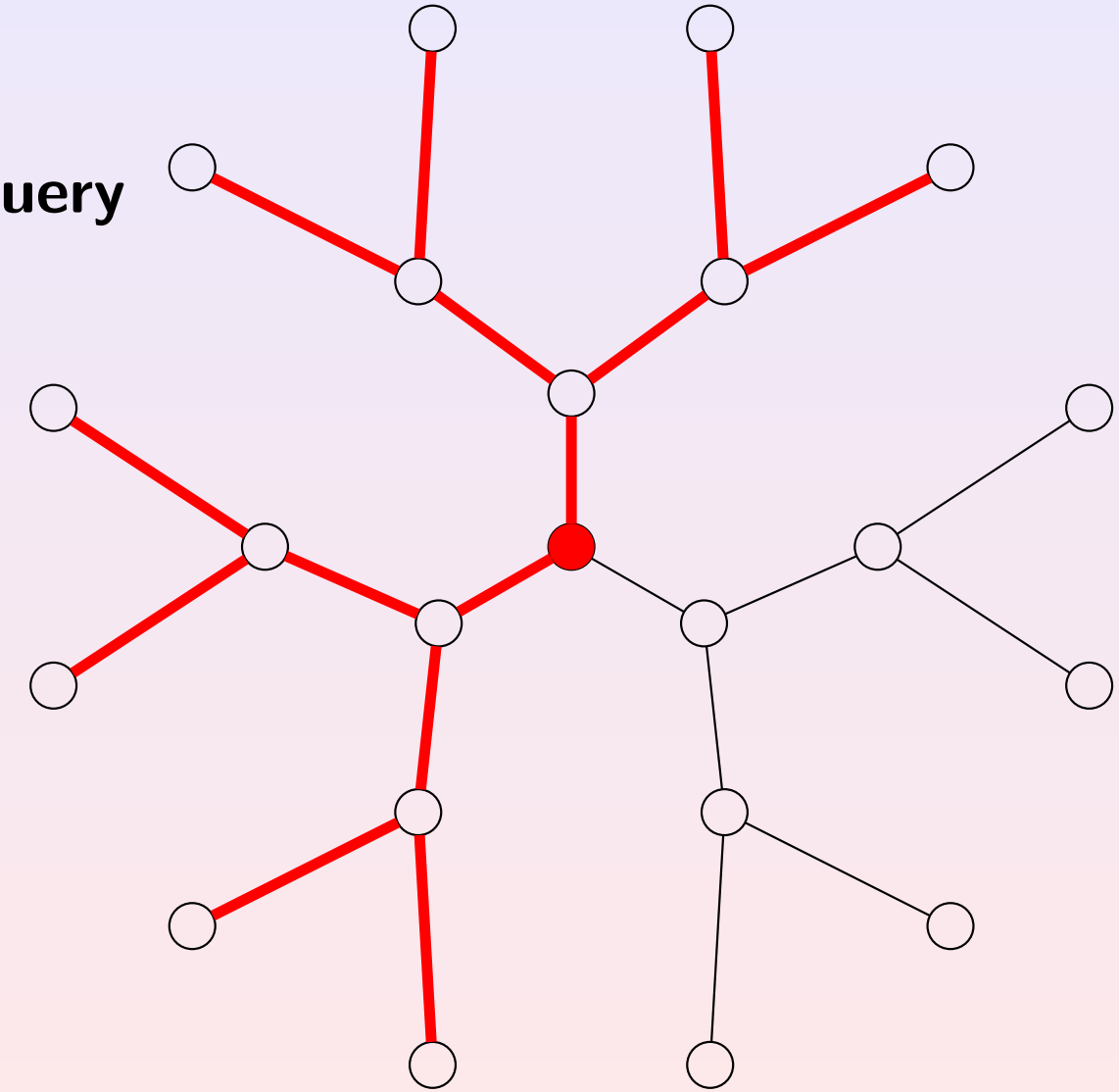
Still the same example :

1 remaining query



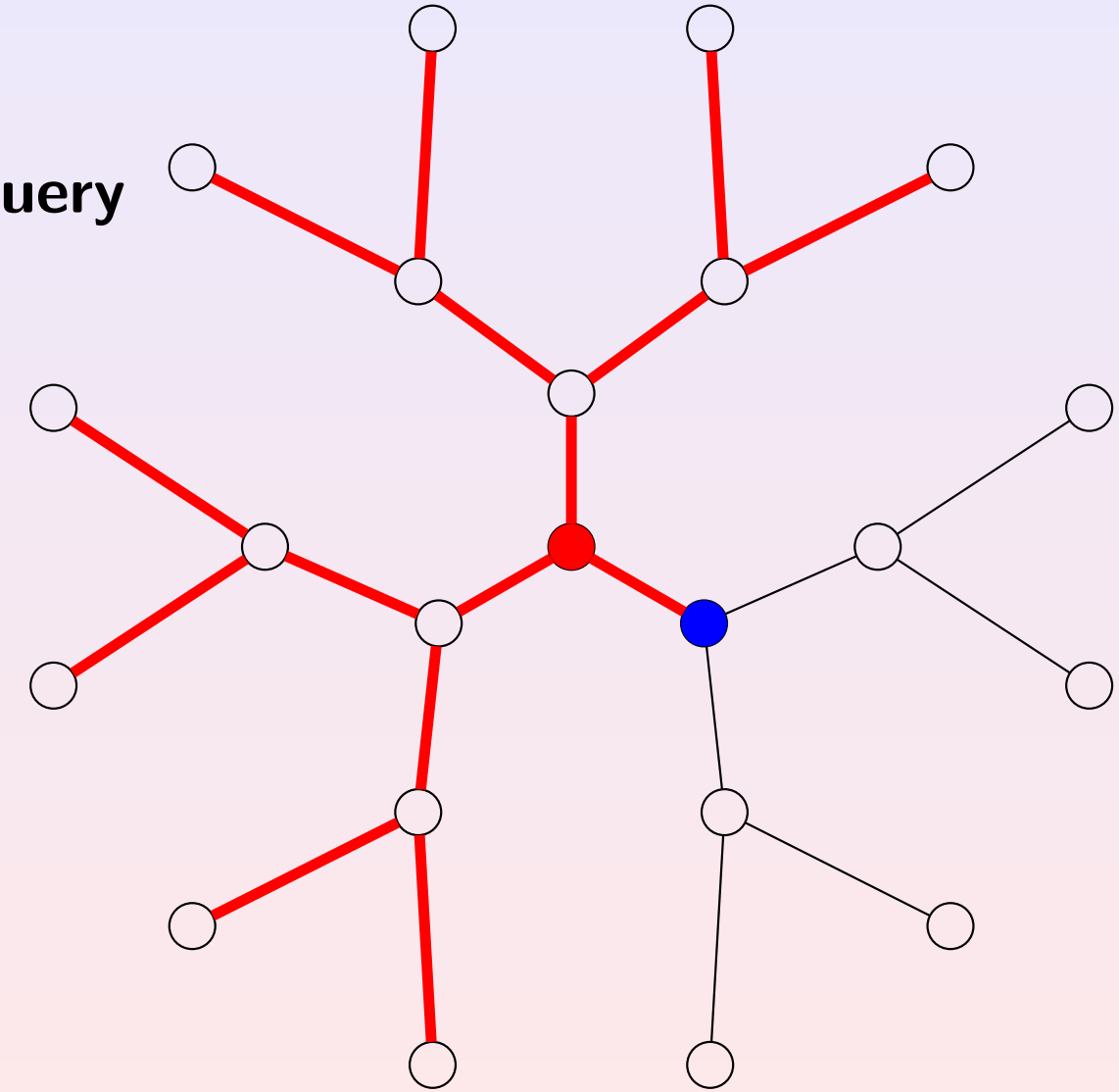
Still the same example :

1 remaining query



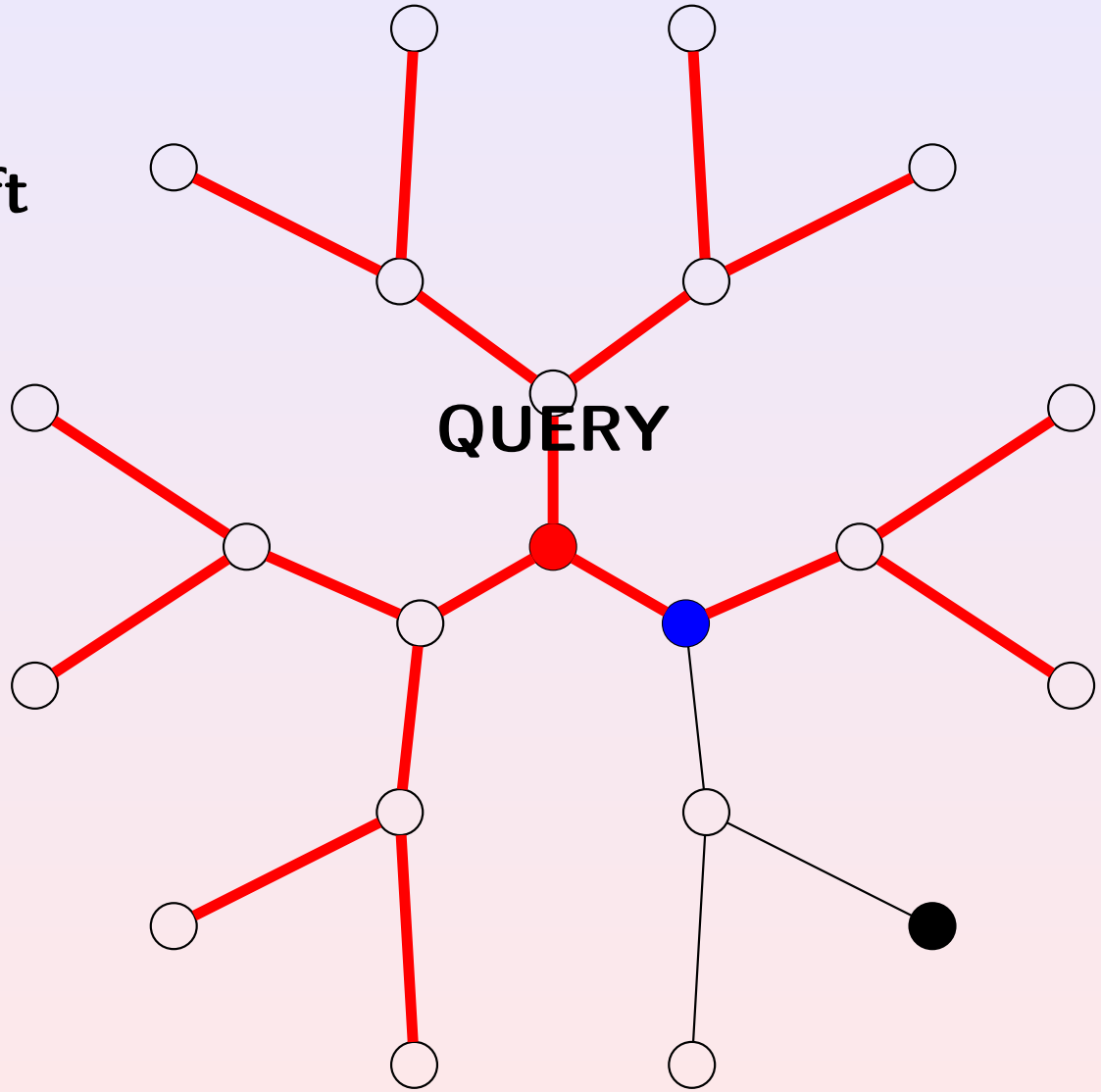
Still the same example :

1 remaining query



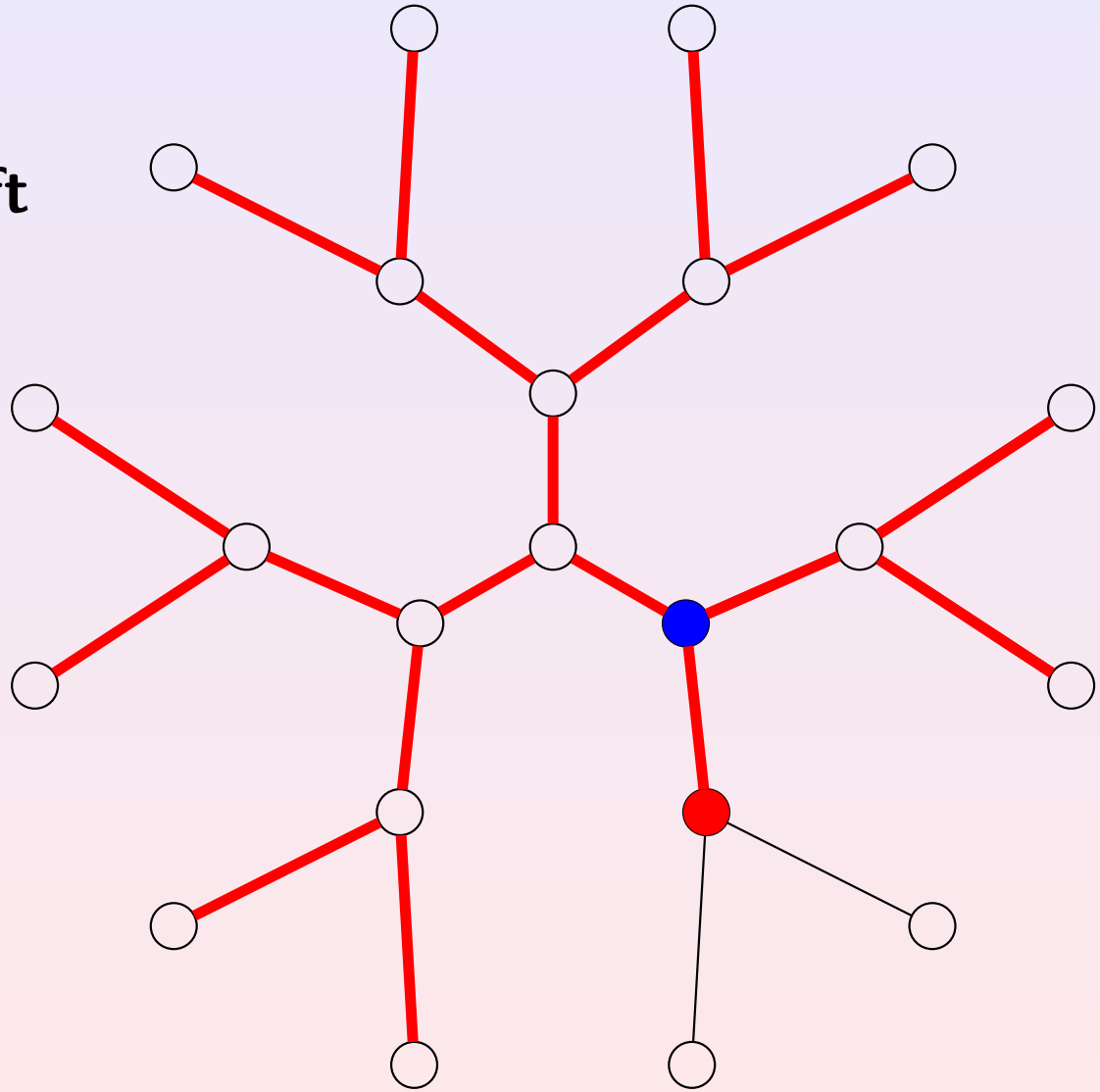
Still the same example :

no query left



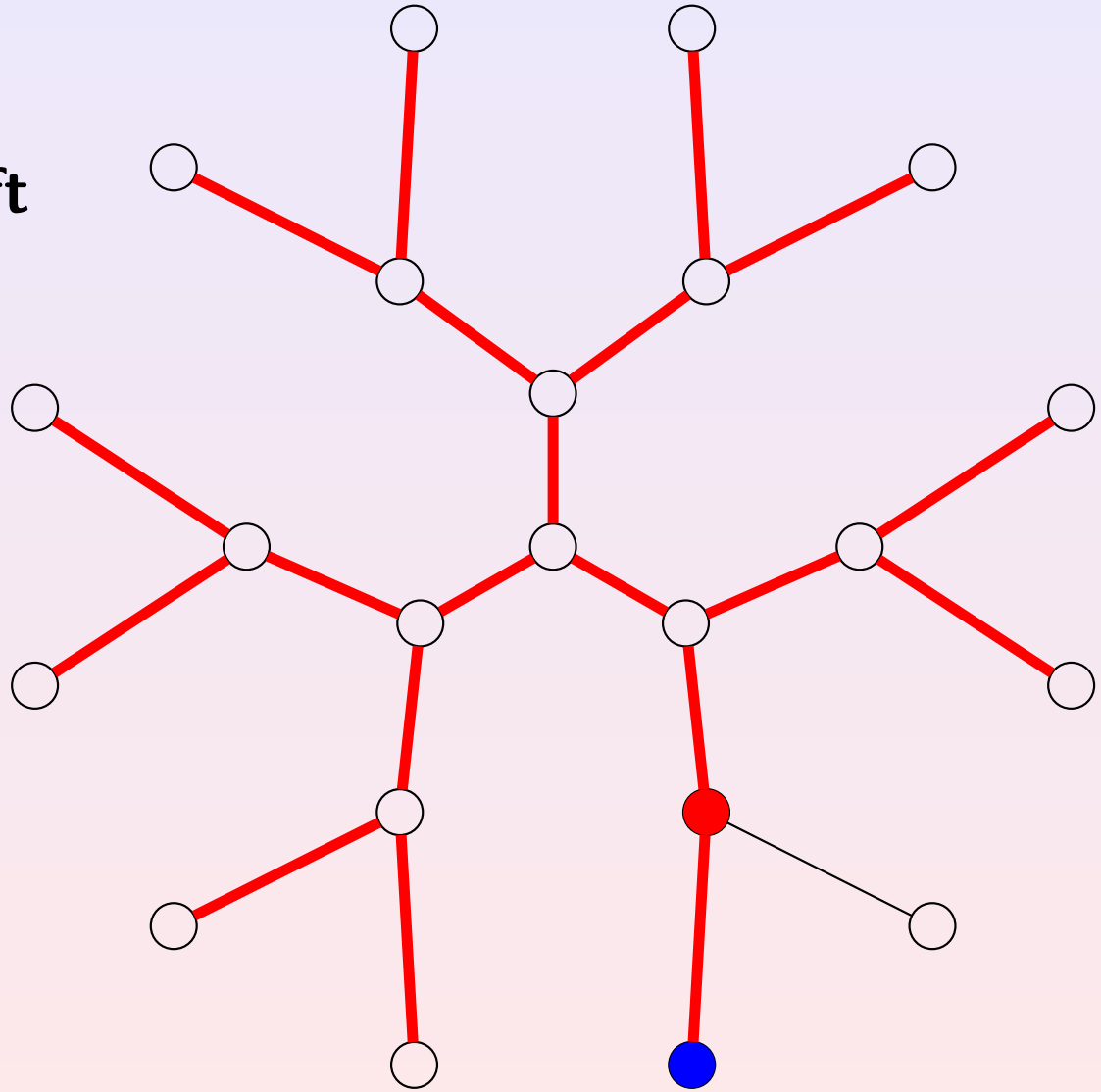
Still the same example :

no query left



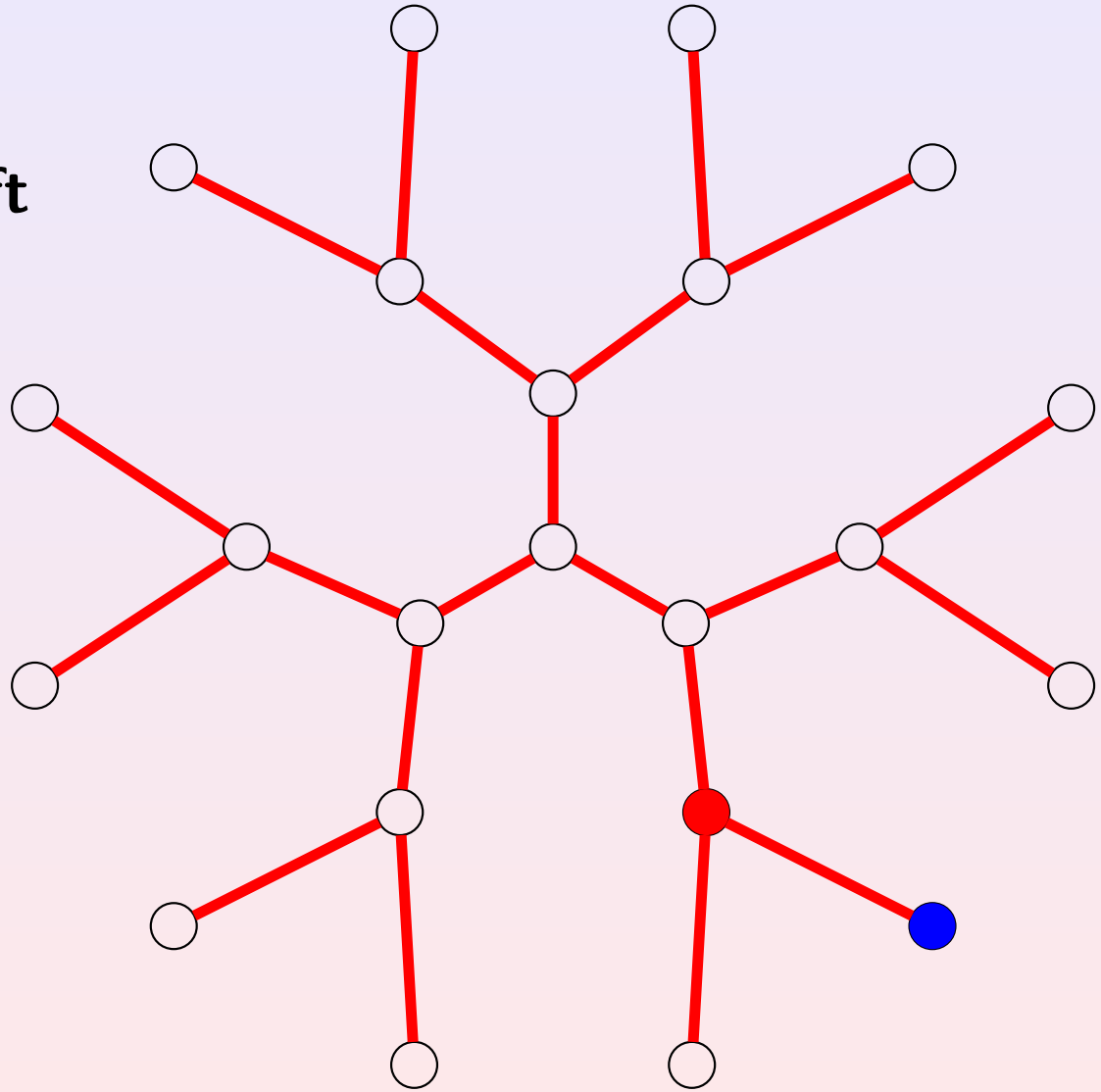
Still the same example :

no query left



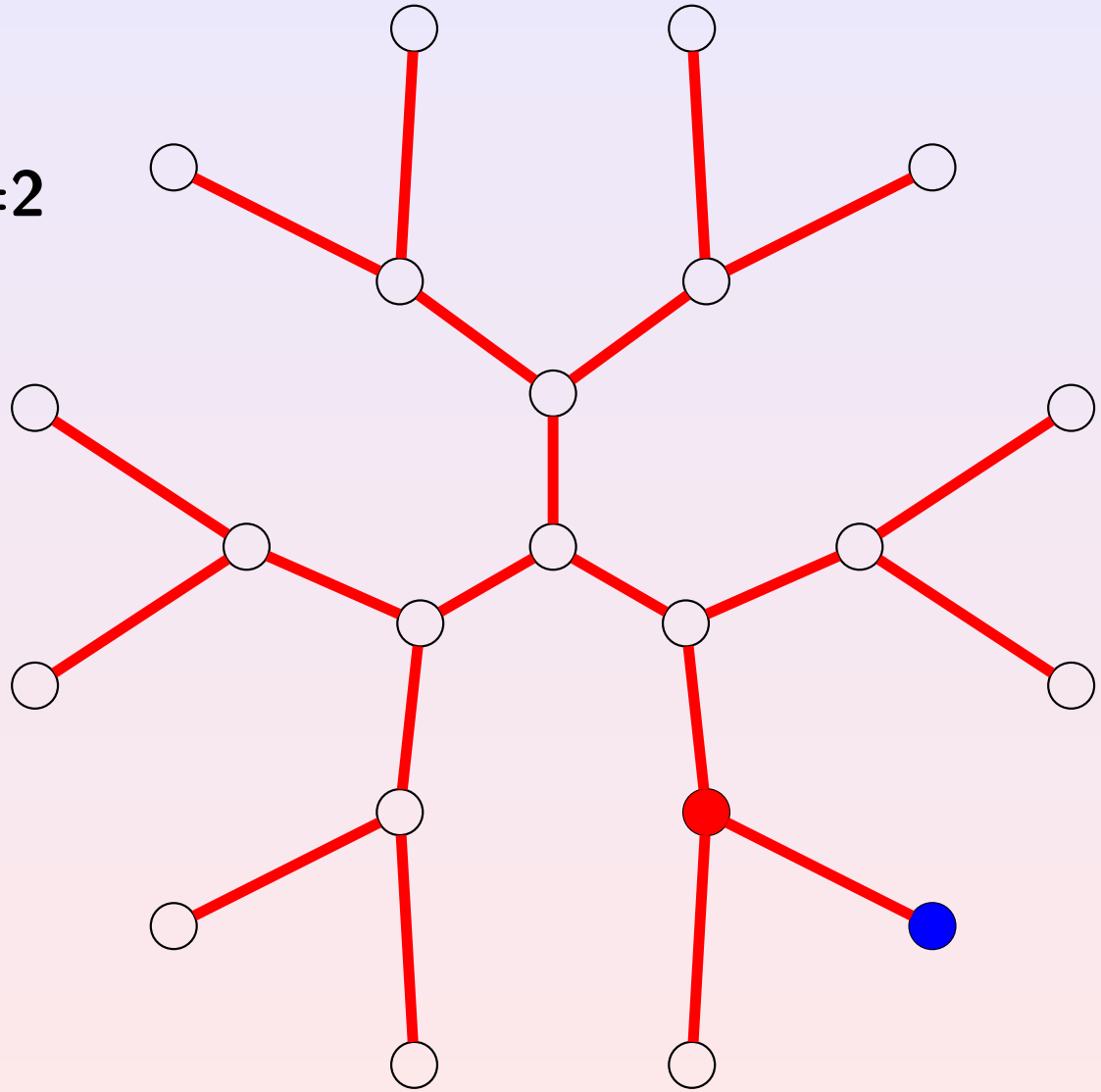
Still the same example :

no query left



Still the same example :

$s(T) > s_2(T) = 2$



Monotone non-deterministic graph searching

$ms_q(G)$: smallest number of searchers needed to catch a fugitive, in a monotone way, using at most q queries.

Theorem[Fomin, Fraigniaud, Nisse, MFCS 2005] :

1. Equivalence between monotone q-limited search number and q-branched treewidth (parametrized variant of treewidth) ;
2. $\mathbf{ms}_q(G) \leq k$ is NP-complete ;
3. Exponential exact algorithm ($O^*(2^n)$) computing $ms_q(G)$

Question : Does recontamination help ?
("No" for $q = 0$ and $q = \infty$.)

Monotone non-deterministic graph searching

$ms_q(G)$: smallest number of searchers needed to catch a fugitive, in a monotone way, using at most q queries.

Theorem[Fomin, Fraigniaud, Nisse, MFCS 2005] :

1. Equivalence between monotone q-limited search number and q-branched treewidth (parametrized variant of treewidth) ;
2. $\mathbf{ms}_q(G) \leq k$ is NP-complete ;
3. Exponential exact algorithm ($O^*(2^n)$) computing $ms_q(G)$

Question : Does recontamination help ?
("No" for $q = 0$ and $q = \infty$.)

Our result

Non-derministic graph searching is monotone

For any $q \geq 0$ and any graph G, recontamination does not help to catch a fugitive in G, performing at most q queries.

Remarks

- Constructive proof that unifies existing proofs ;
- Above results related to $ms_q(G)$ are valid for $s_q(G)$.

Sketch of the proof

auxiliary structure inspired by the tree-labelling [Robertson and Seymour, Graph Minor X] :
search-tree $\approx$ relaxed tree-decomposition

Sketch of the proof

- search-tree $\Leftrightarrow$ (possibly non monotone) strategy
- weight function over the search trees
- minimal search tree $\Rightarrow$ monotone strategy
- local optimization without increasing neither the number of searcher nor the number of queries.

Search-tree

A tree T labelled with subsets of $E(G)$

For any vertex $v \in V(T)$ incident to $e_1, \ldots, e_p$:

- label of v : $\ell(v) \subseteq E(G)$
- label of e_i : $\ell_v(e_i) \subseteq E(G)$

Any edge has two labels : one for each extremity.

2 Properties

1. $\{\ell(v), \ell_v(e_1), \ell_v(e_2), \ldots, \ell_v(e_p)\}$ partition of $E(G)$;
2. $\forall e = \{u, v\} \in E(T)$, $\ell_v(e)$ and $\ell_u(e)$ are disjoint.

A search tree T represents a non-deterministic strategy.

If, for any $e = \{u, v\} \in E(T)$, $\ell_v(e)$ and $\ell_u(e)$ are complementary, the corresponding strategy is monotone

Search-tree

A tree T labelled with subsets of $E(G)$

For any vertex $v \in V(T)$ incident to $e_1, \ldots, e_p$:

- label of v : $\ell(v) \subseteq E(G)$
- label of e_i : $\ell_v(e_i) \subseteq E(G)$

Any edge has two labels : one for each extremity.

2 Properties

1. $\{\ell(v), \ell_v(e_1), \ell_v(e_2), \ldots, \ell_v(e_p)\}$ partition of $E(G)$;
2. $\forall e = \{u, v\} \in E(T)$, $\ell_v(e)$ and $\ell_u(e)$ are disjoint.

A search tree T represents a non-deterministic strategy.

If, for any $e = \{u, v\} \in E(T)$, $\ell_v(e)$ and $\ell_u(e)$ are complementary, the corresponding strategy is monotone

Search-tree

A tree T labelled with subsets of $E(G)$

For any vertex $v \in V(T)$ incident to $e_1, \ldots, e_p$:

- label of v : $\ell(v) \subseteq E(G)$
- label of e_i : $\ell_v(e_i) \subseteq E(G)$

Any edge has two labels : one for each extremity.

2 Properties

1. $\{\ell(v), \ell_v(e_1), \ell_v(e_2), \ldots, \ell_v(e_p)\}$ partition of $E(G)$;
2. $\forall e = \{u, v\} \in E(T)$, $\ell_v(e)$ and $\ell_u(e)$ are disjoint.

A search tree T represents a non-deterministic strategy.

If, for any $e = \{u, v\} \in E(T)$, $\ell_v(e)$ and $\ell_u(e)$ are complementary, the corresponding strategy is monotone

Search-tree

A tree T labelled with subsets of $E(G)$

For any vertex $v \in V(T)$ incident to $e_1, \ldots, e_p$:

- label of v : $\ell(v) \subseteq E(G)$
- label of e_i : $\ell_v(e_i) \subseteq E(G)$

Any edge has two labels : one for each extremity.

2 Properties

1. $\{\ell(v), \ell_v(e_1), \ell_v(e_2), \ldots, \ell_v(e_p)\}$ partition of $E(G)$;
2. $\forall e = \{u, v\} \in E(T)$, $\ell_v(e)$ and $\ell_u(e)$ are disjoint.

A search tree T represents a non-deterministic strategy.

If, for any $e = \{u, v\} \in E(T)$, $\ell_v(e)$ and $\ell_u(e)$ are complementary, the corresponding strategy is monotone

Conclusion and further work

About monotonicity

Generalyse our result to other variants of graph searching problems : directed graphs.

About non-deterministic graph searching

Polynomial time algorithm for the class of trees ?
FPT algorithms ?

Recent result using search-tree

With O. Amini, F. Mazoit, and S. Thomassé
Generalization of the min-max for treewidth

Conclusion and further work

About monotonicity

Generalyse our result to other variants of graph searching problems : directed graphs.

About non-deterministic graph searching

Polynomial time algorithm for the class of trees ?
FPT algorithms ?

Recent result using search-tree

With O. Amini, F. Mazoit, and S. Thomassé
Generalization of the min-max for treewidth