Nondeterministic Graph Searching: From Pathwidth to Treewidth

Fedor V.Fomin[1] Pierre Fraigniaud[2] Nicolas Nisse[2]

Department of Informatics, University of Bergen,
PO Box 7800, 5020 Bergen, Norway.

CNRS, Lab. de Recherche en Informatique, Université Paris-Sud,
91405 Orsay, France.

Workshop on Graph Classes, Width Parameters and Optimisation, Prague, October 19, 2005

Graph Searching

Goal

In an undirected simple graph,

- omniscient and arbitrary fast fugitive ;
- a team of searchers ;

We want to find a strategy that catch the fugitive
using the fewest searchers as possible.

Motivation

- game related to treewidth and pathwidth ;
- we introduce a parametrized version of treewidth.

Graph Searching

Goal

In an undirected simple graph,

- omniscient and arbitrary fast fugitive ;
- a team of searchers ;

We want to find a strategy that catch the fugitive
using the fewest searchers as possible.

Motivation

- game related to treewidth and pathwidth ;
- we introduce a parametrized version of treewidth.

Search Strategy, Parson. [GTC,1978]

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 is at a vertex occupied by a searcher and it has no way to escape.

We want to minimize the number of searchers.

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

Search Strategy, Parson. [GTC,1978]

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 is at a vertex occupied by a searcher and it has no way to escape.

We want to minimize the number of searchers.

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

Search Strategy, Parson. [GTC,1978]

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.

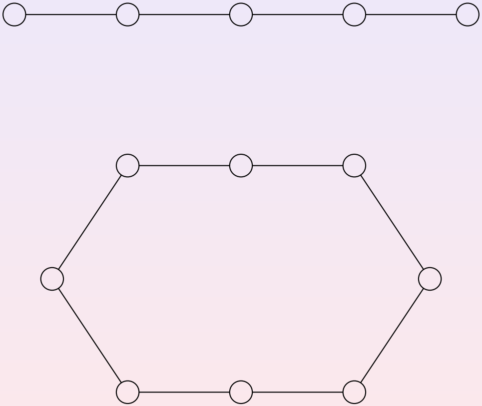
...that must result in catching the fugitive

The fugitive is caugth when it is at a vertex occupied by a searcher and it has no way to escape.

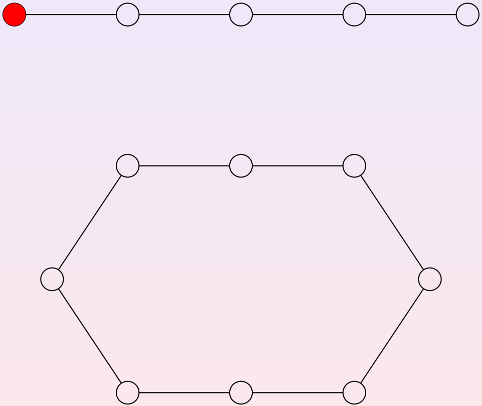
We want to minimize the number of searchers.

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

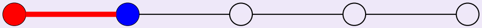
Simple Examples : Path and Ring

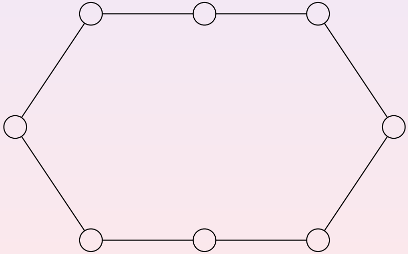


Simple Examples : Path and Ring

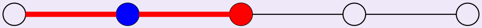


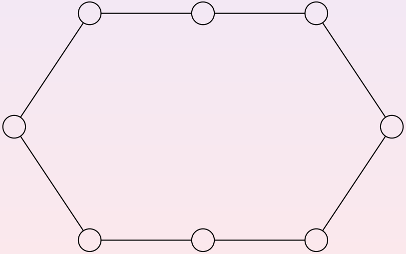
Simple Examples : Path and Ring



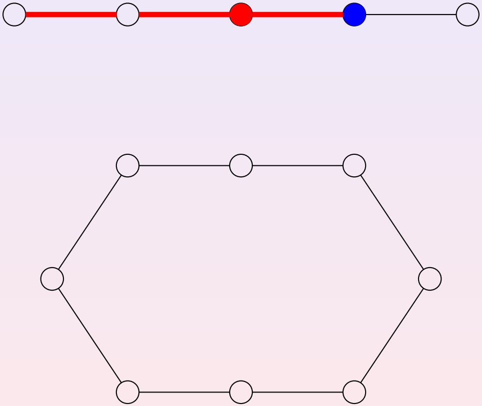


Simple Examples : Path and Ring

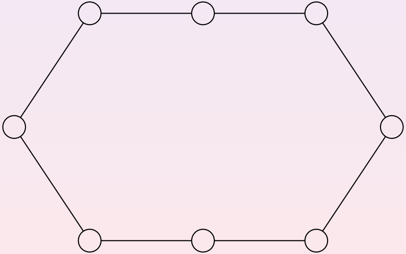




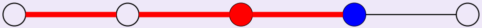
Simple Examples : Path and Ring



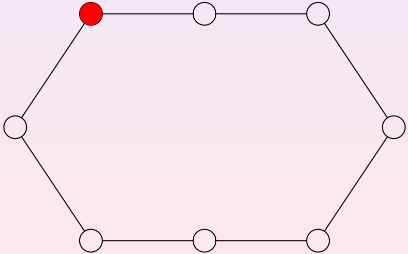
s(Path)=2



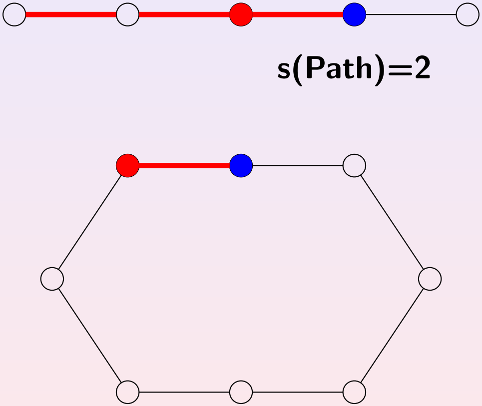
Simple Examples : Path and Ring



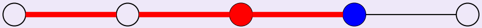
s(Path)=2



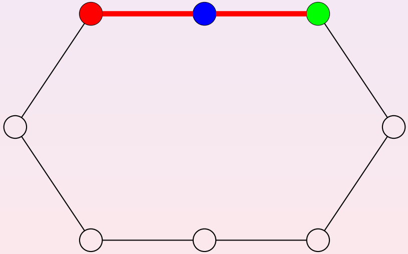
Simple Examples : Path and Ring



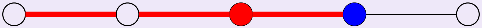
Simple Examples : Path and Ring



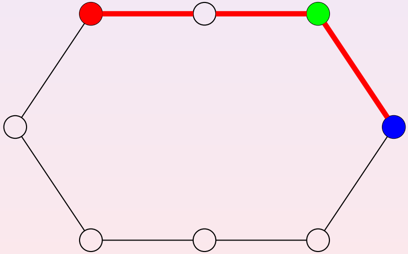
s(Path)=2



Simple Examples : Path and Ring

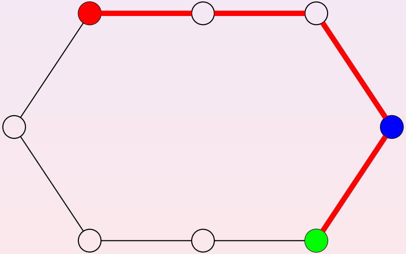


s(Path)=2

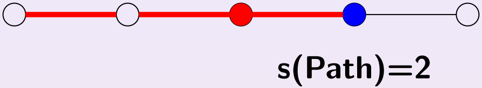


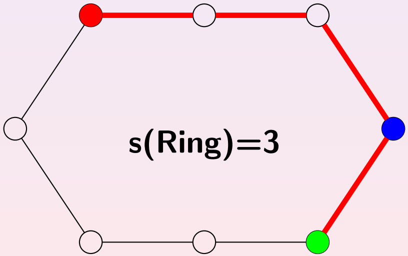
Simple Examples : Path and Ring

s(Path)=2

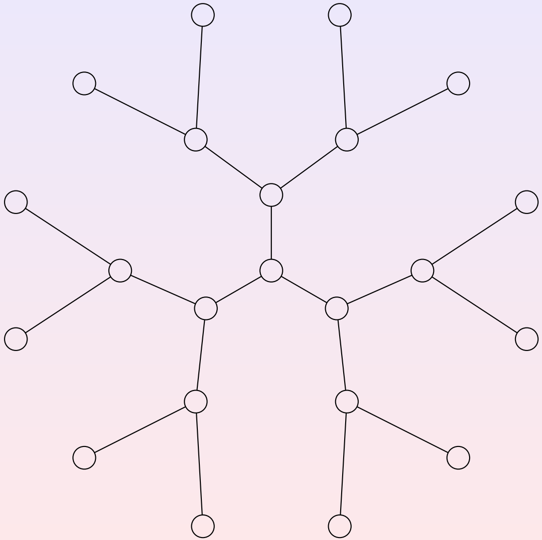


Simple Examples : Path and Ring

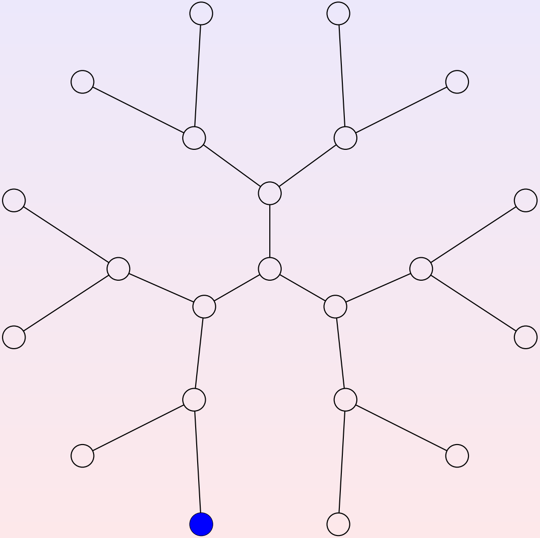




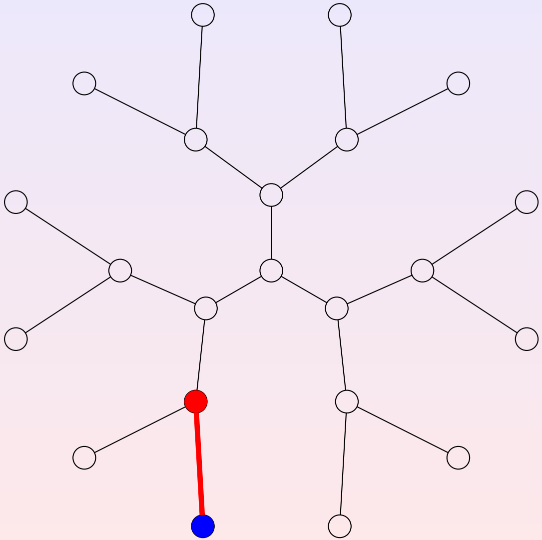
Graph searching in a tree



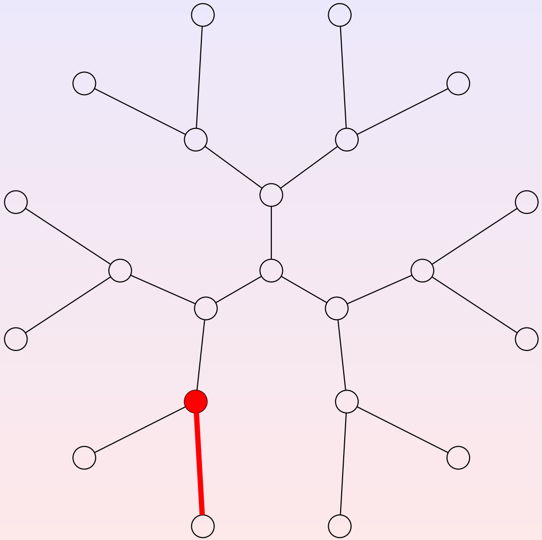
Graph searching in a tree



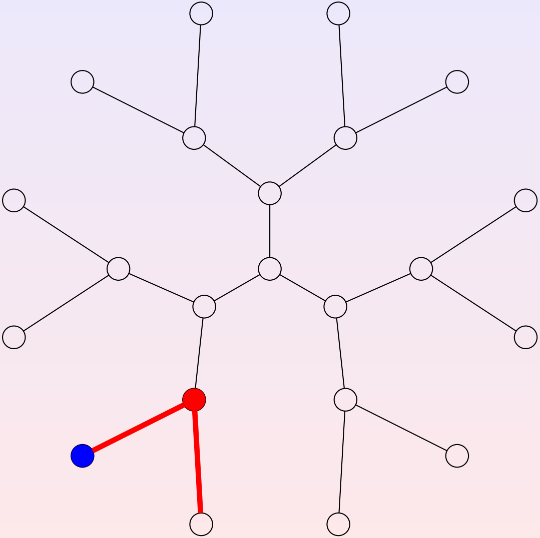
Graph searching in a tree



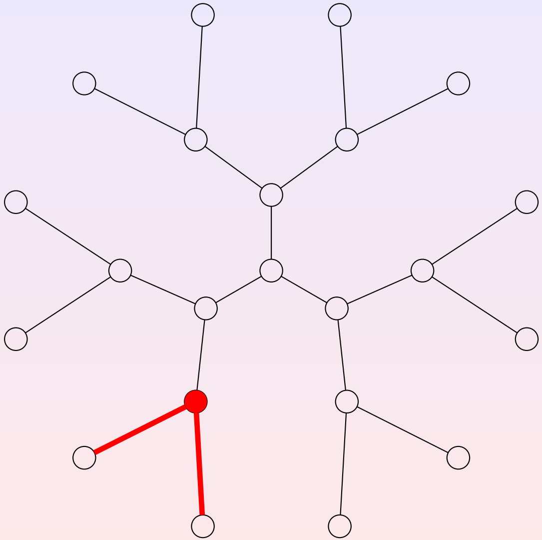
Graph searching in a tree



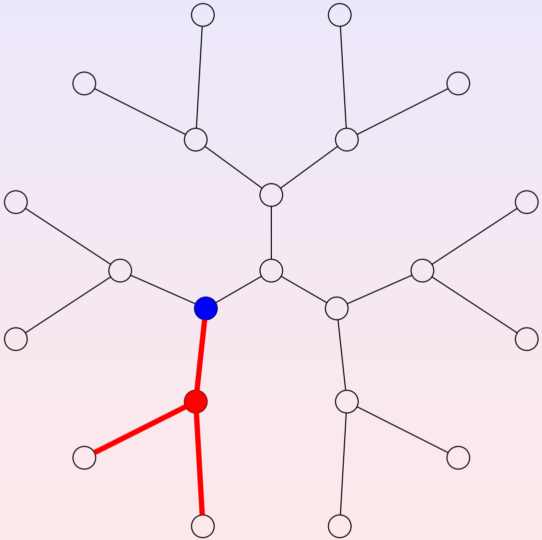
Graph searching in a tree



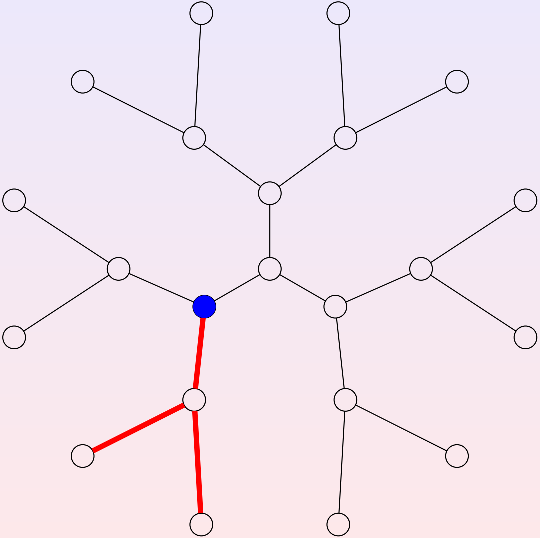
Graph searching in a tree



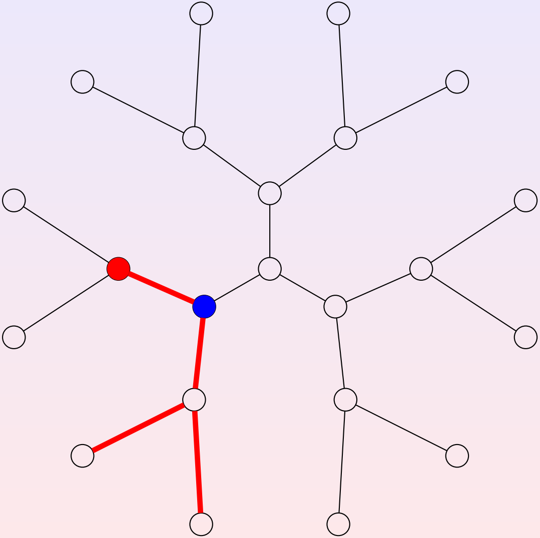
Graph searching in a tree



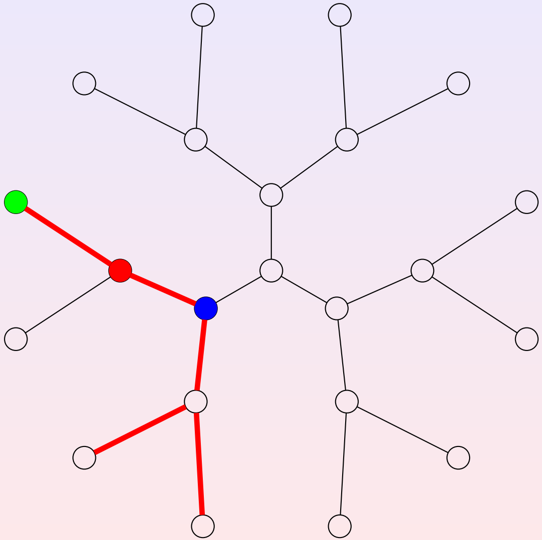
Graph searching in a tree



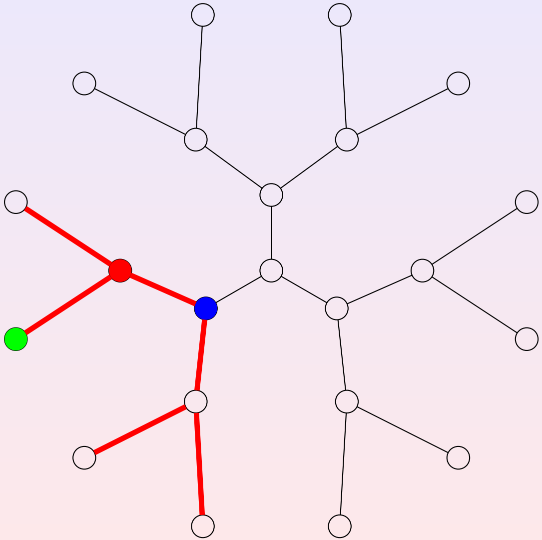
Graph searching in a tree



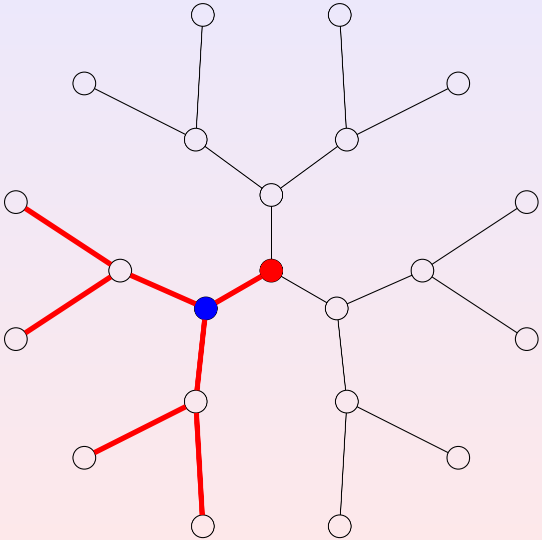
Graph searching in a tree



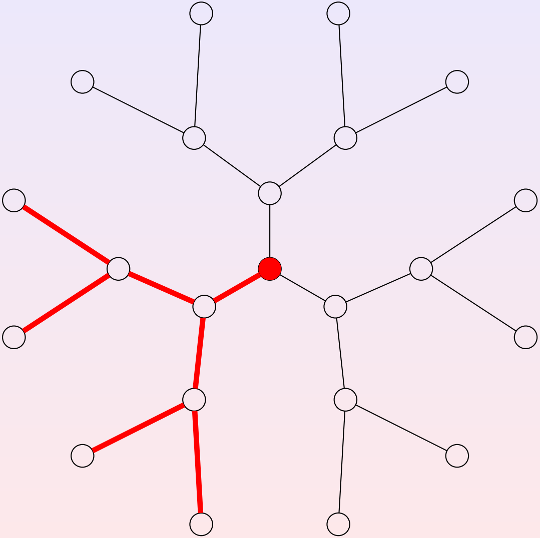
Graph searching in a tree



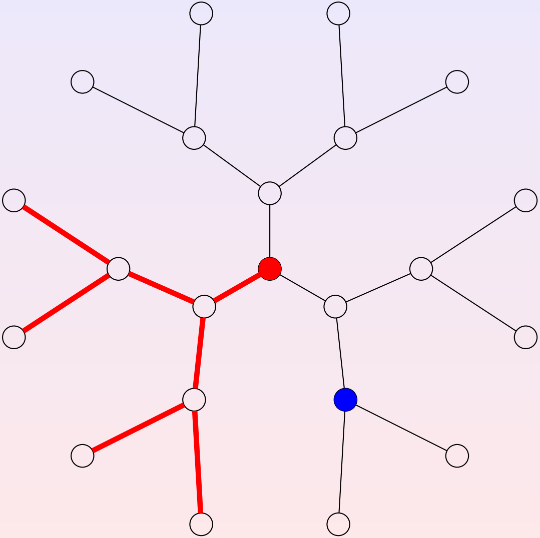
Graph searching in a tree



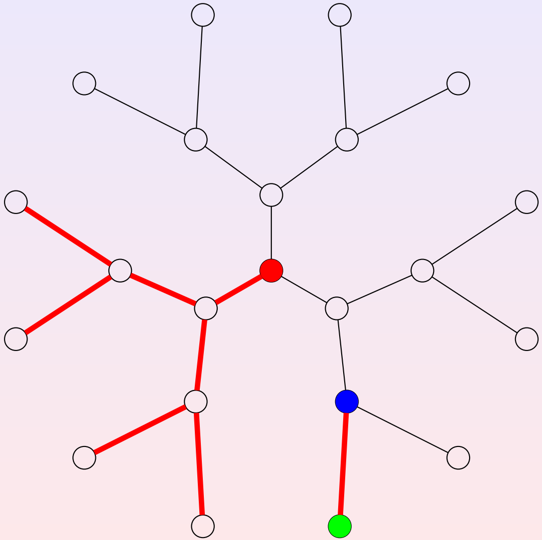
Graph searching in a tree



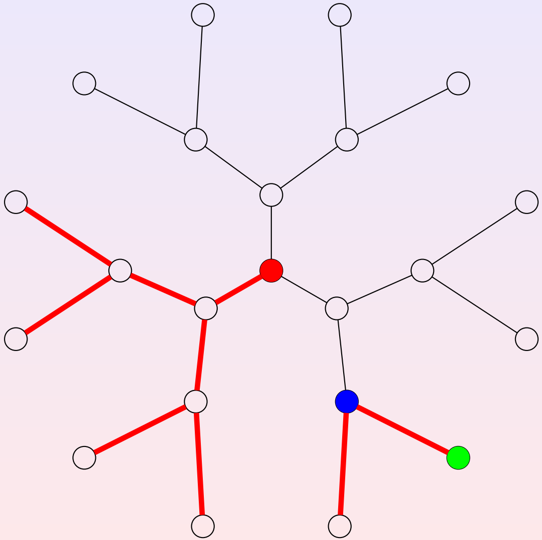
Graph searching in a tree



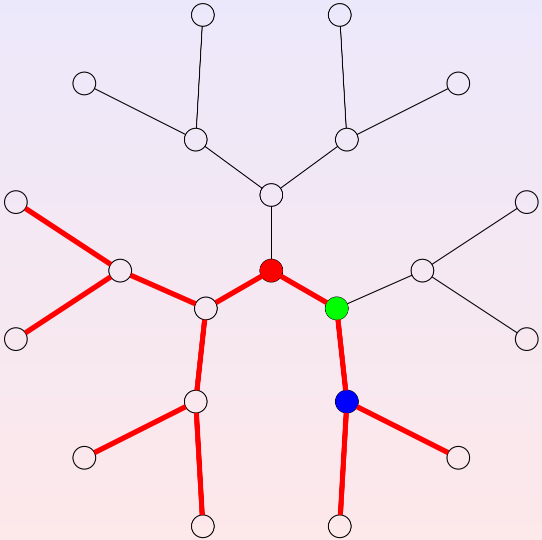
Graph searching in a tree



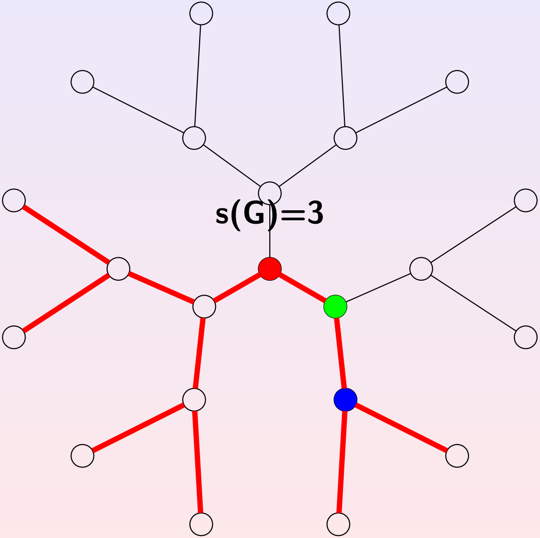
Graph searching in a tree



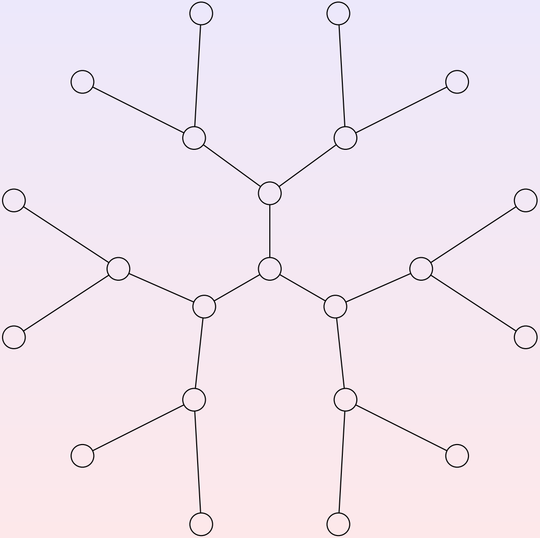
Graph searching in a tree



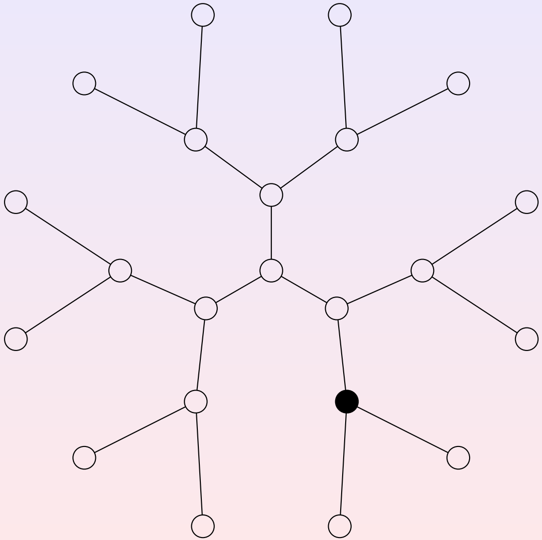
Graph searching in a tree



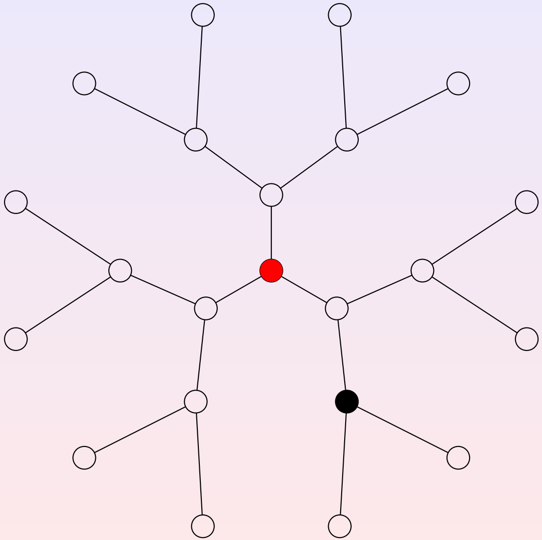
Visible graph searching in a tree



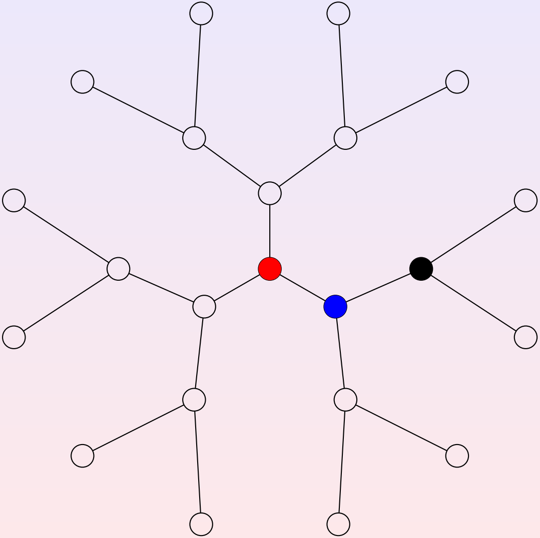
Visible graph searching in a tree



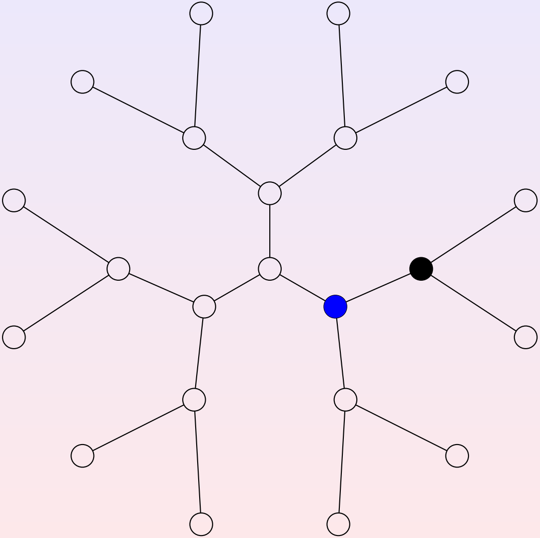
Visible graph searching in a tree



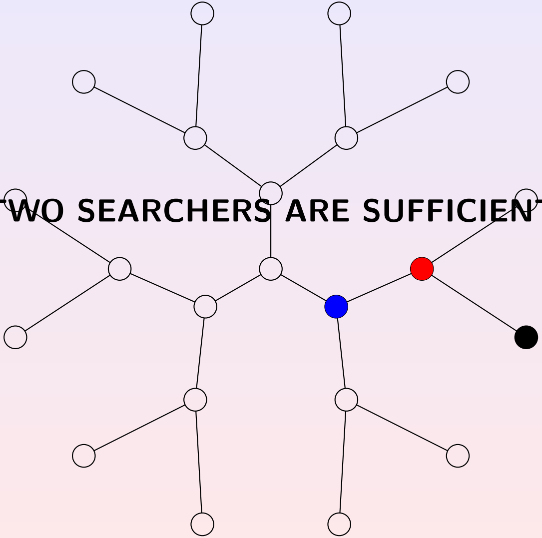
Visible graph searching in a tree



Visible graph searching in a tree

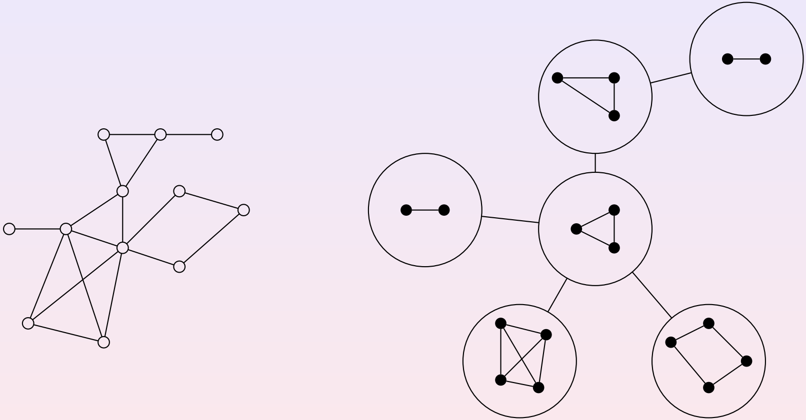


Visible graph searching in a tree

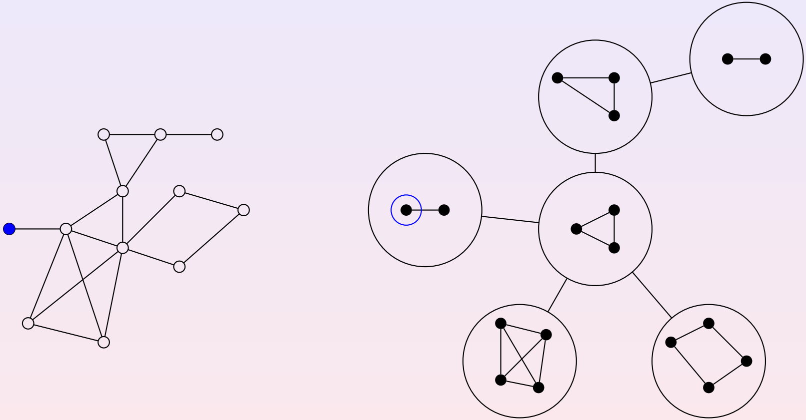


TWO SEARCHERS ARE SUFFICIENT

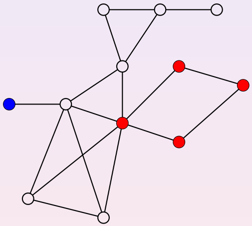
Tree Decomposition and Visible Search

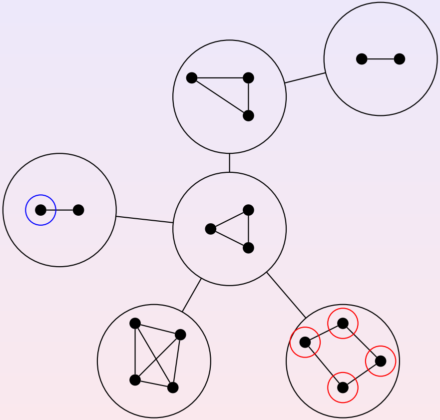


Tree Decomposition and Visible Search

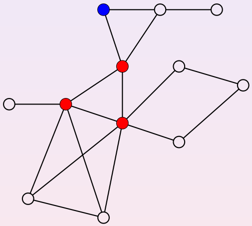


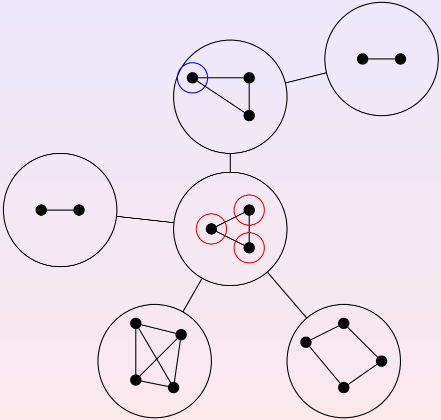
Tree Decomposition and Visible Search



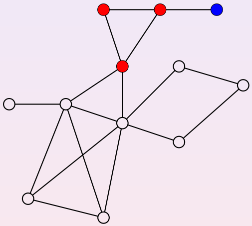


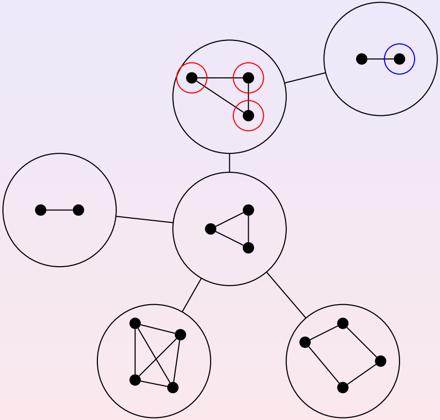
Tree Decomposition and Visible Search



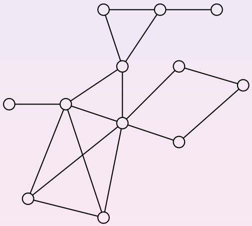


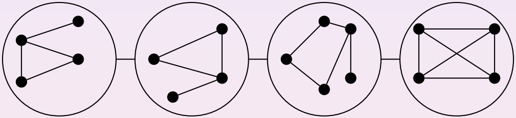
Tree Decomposition and Visible Search



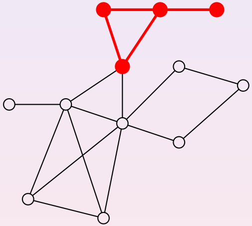


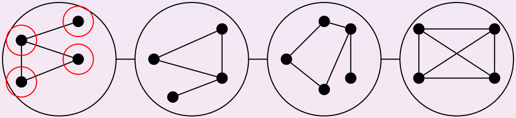
Path Decomposition and Invisible Search



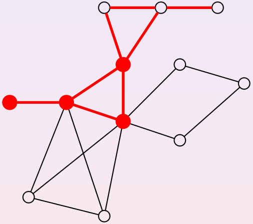


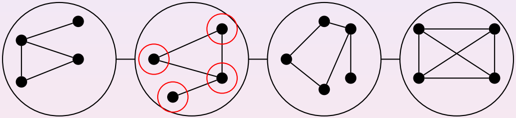
Path Decomposition and Invisible Search



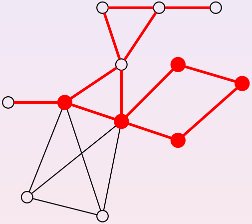


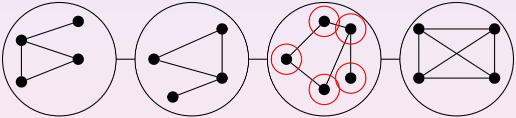
Path Decomposition and Invisible Search





Path Decomposition and Invisible Search





Related Work

$\mathbf{s}(G) = \mathbf{pw}(G) + 1$

- **Kirousis and Papadimitriou**, Theor. Comp. Sc., 1986
  Searching and Pebbing.
- **Ellis, Sudborough and Turner**. Inf. and Comp., 1994
  The vertex separation and search number of a graph

$\mathbf{s}_{visible}(G) = \mathbf{tw}(G) + 1$

- **Seymour and Thomas**, J. Combin. Theory, 1993
  Graph searching and min-max theorem for treewidth.

Nondeterministic Graph Searching

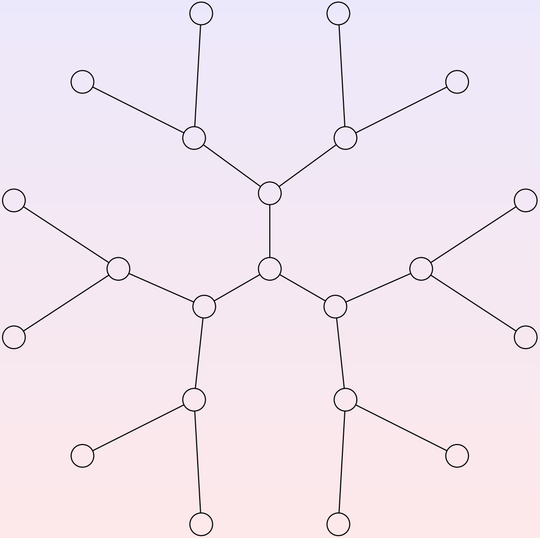
Query operation

- oracle ;
- tradeoff between number of searchers and number of query steps ;
- q-limited nondeterministic search number, $s_q(G)$.

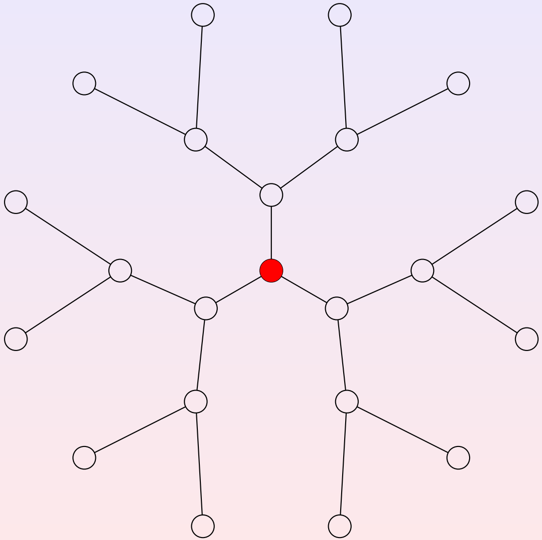
Link with graph searching

- $s(G) = s_0(G)$;
- $s_{visible}(G) = s_\infty(G)$.

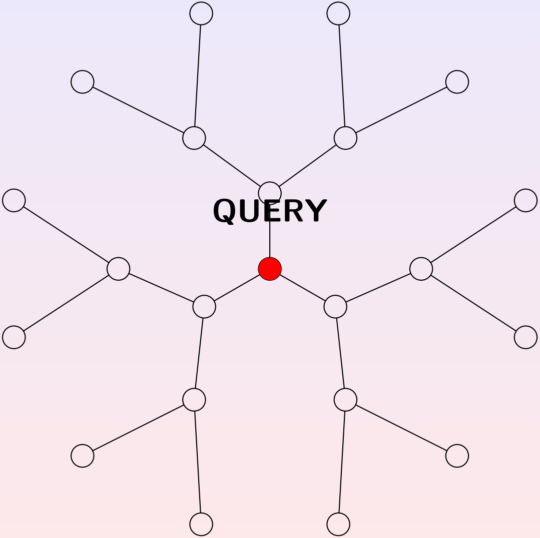
Limited Graph searching with 2 queries



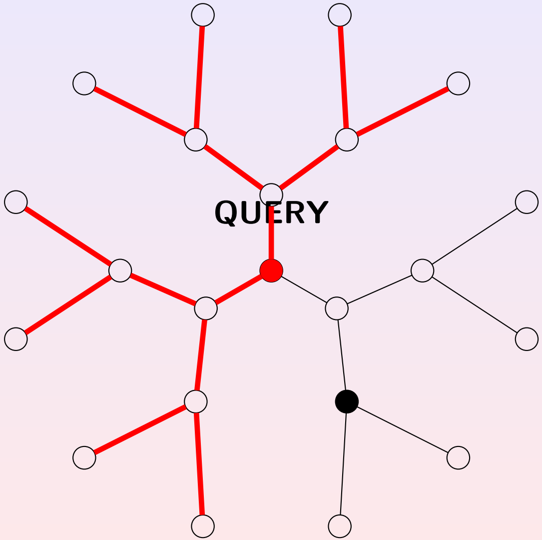
Limited Graph searching with 2 queries



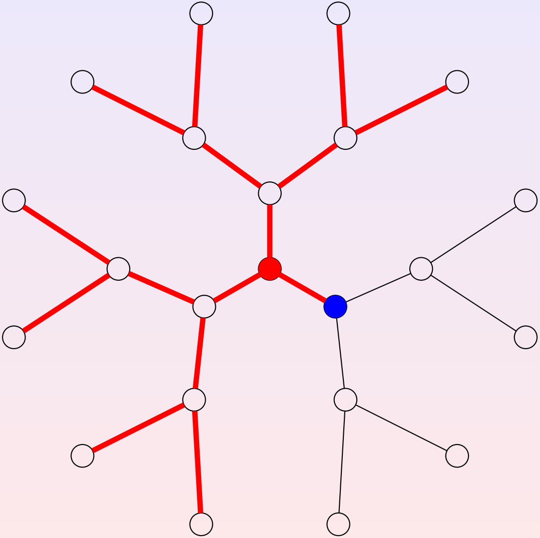
Limited Graph searching with 2 queries



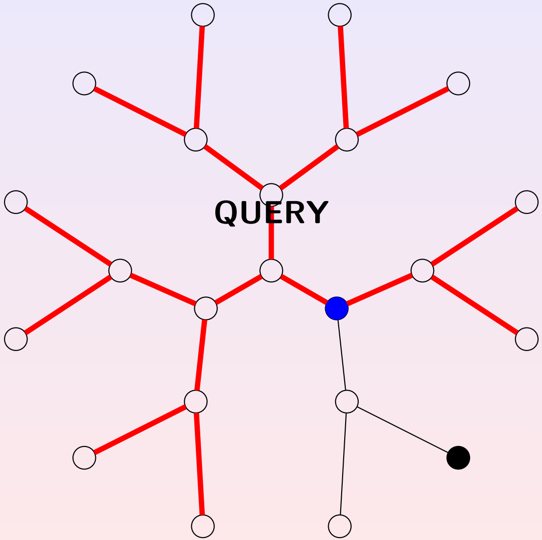
Limited Graph searching with 2 queries



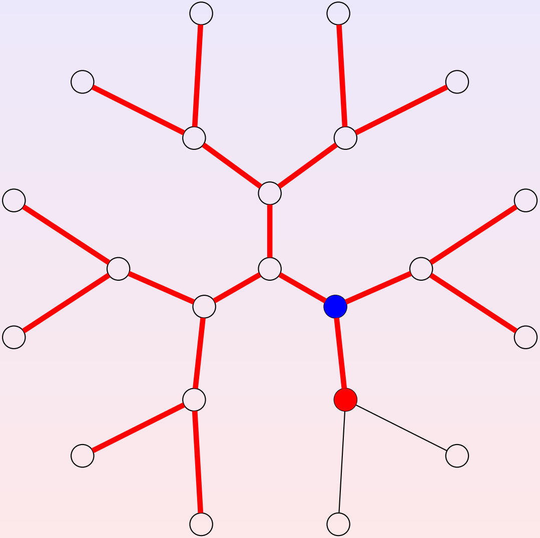
Limited Graph searching with 2 queries



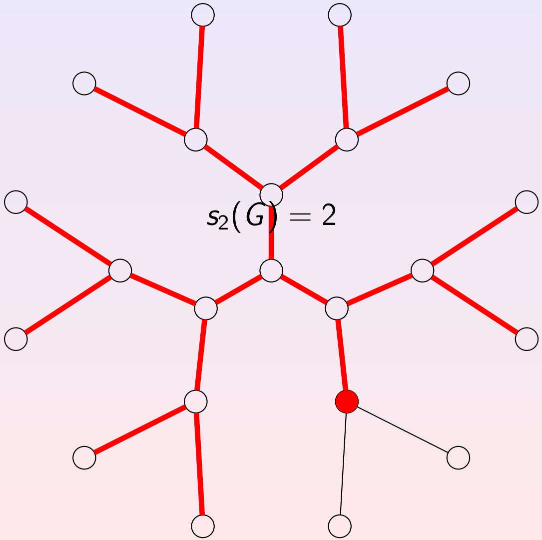
Limited Graph searching with 2 queries



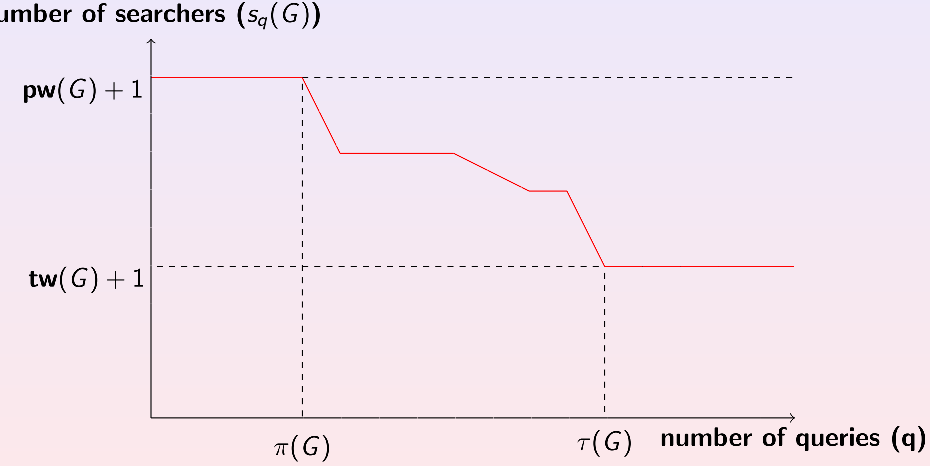
Limited Graph searching with 2 queries



Limited Graph searching with 2 queries



Controlled Amount of Nondeterminism



Branched Tree Decomposition

q-branched treewidth, $tw_q(G)$.

- rooted tree decomposition ;
- branching node (at least two children) ;
- every path from the root to a leaf contains at most q branching nodes.

- path decomposition = 0-branched tree decomposition $\mathbf{pw}(G) = \mathbf{tw}_0(G)$;
- tree decomposition = ∞-branched tree decomposition $\mathbf{tw}(G) = \mathbf{tw}_\infty(G)$;

Main Results (1)

Branched Treewidth vs. Limited Graph Searching

Theorem 1 : For any $q \geq 0$, for any graph G,
$\mathbf{s}_q(G) = \mathbf{tw}_q(G) + 1$.

NP-Completeness

Computing whether $\mathbf{tw}_q(G) \leq k$ is NP-complete for any q.

Main Results (2)

Exact Exponential Algorithm

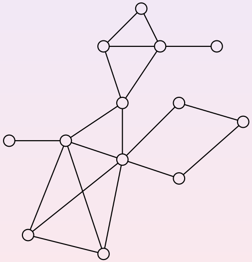
Theorem 2 : There exists an algorithm that, for any graph G and any $q \geq 0$, computes $\mathbf{tw}_q(G)$ and an optimal q-branched tree decomposition of G.

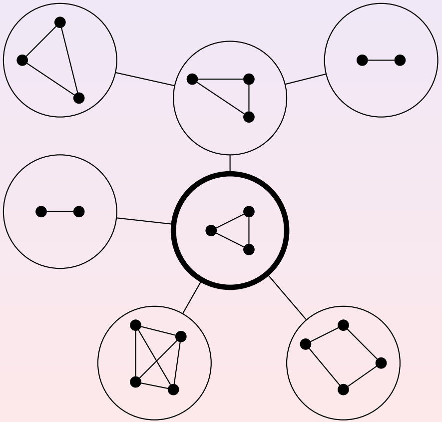
Bounding the Nondeterminism

Theorem 3 : For any $q \geq 1$, for any graph G, $\mathbf{tw}_{q-1}(G) \leq 2\ \mathbf{tw}_q(G)$.

Theorem 1 : $\mathbf{s}_q(G) = \mathbf{tw}_q(G) + 1$

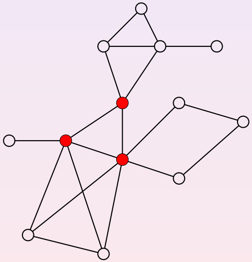
- place searchers at vertices of the root ;
- perform a query at any branching node we meet.

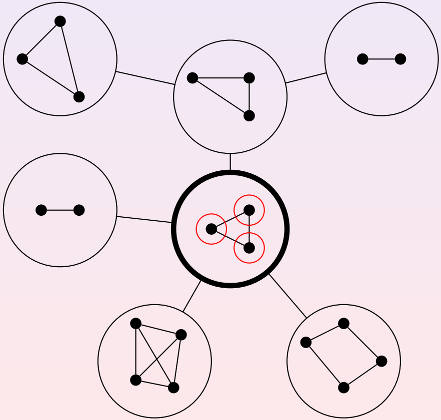




Theorem 1 : $\mathbf{s}_q(G) = \mathbf{tw}_q(G) + 1$

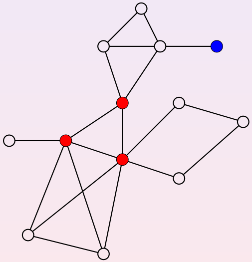
- place searchers at vertices of the root ;
- perform a query at any branching node we meet.

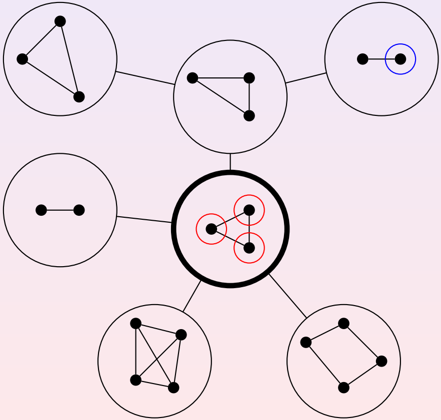




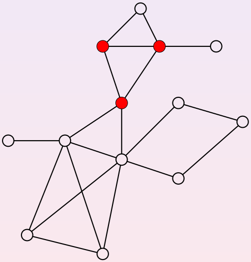
Theorem 1 : $\mathbf{s}_q(G) = \mathbf{tw}_q(G) + 1$

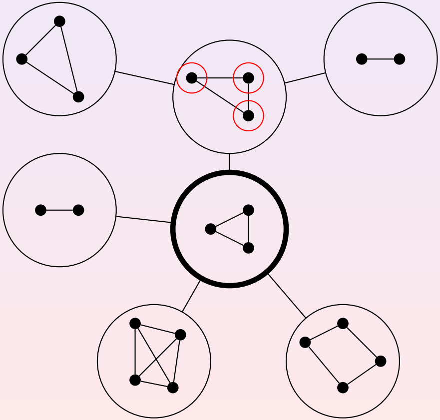
- place searchers at vertices of the root ;
- perform a query at any branching node we meet.





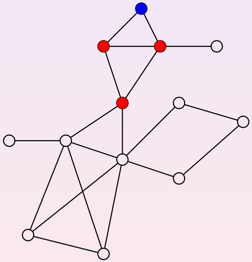
- place searchers at vertices of the root ;
- perform a query at any branching node we meet.

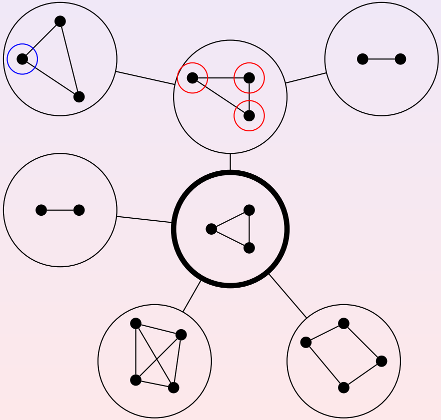




Theorem 1 : $\mathbf{s}_q(G) = \mathbf{tw}_q(G) + 1$

- place searchers at vertices of the root ;
- perform a query at any branching node we meet.





Exact Exponential Algorithm

Theorem 2

There exists an algorithm that, for any graph G and any $q \geq 0$, computes $\mathbf{tw}_q(G)$ and an optimal q-branched tree decomposition of G.

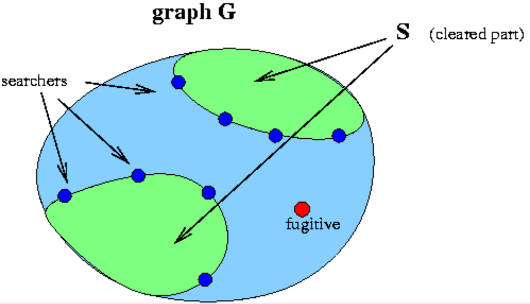
Thanks to Theorem 1, the following is sufficient :

For any k and any graph G, our algorithm computes the minimum q such that there exists a non-deterministic search strategy of G, using k searchers and q queries and the corresponding strategy.

Exact Exponential Algorithm

Configuration digraph H

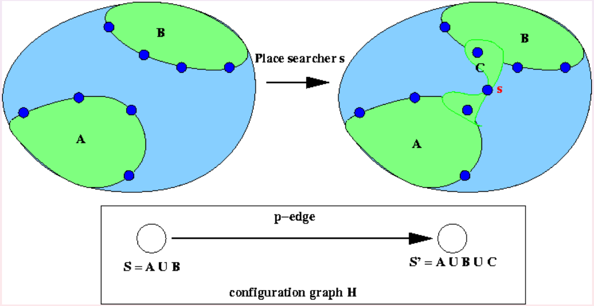
- a **vertex** S of H is a *set of clear vertices* of G *reachable* by a search program using k searchers.
  $V(H) = \{S \subseteq V(G)\ s.t.\ |\delta(S)| \leq k\}$;



Exact Exponential Algorithm

Configuration digraph H

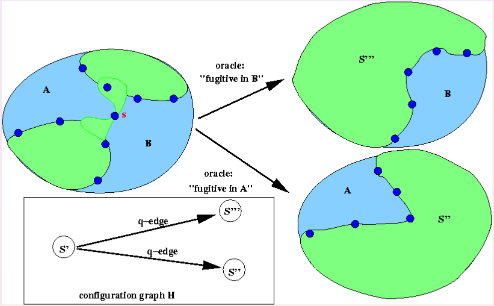
- a **directed edge** $\{S, S'\}$ of H is a *search step* that allows to reach S' from S.
  place edges and *query* edges ;



Exact Exponential Algorithm

Configuration digraph H

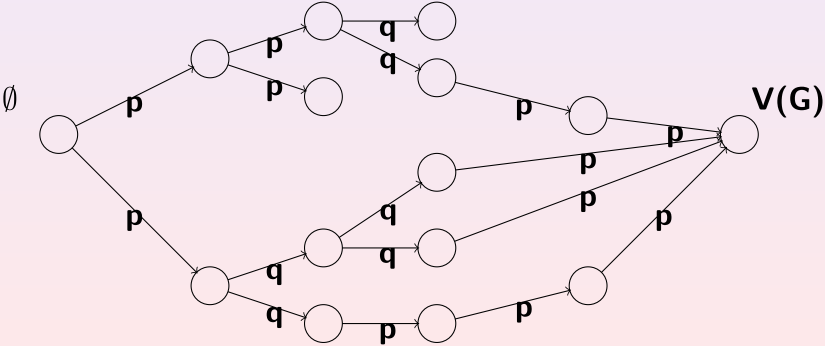
- a **directed edge** $\{S, S'\}$ of H is a *search step* that allows to reach S' from S.
  place edges and **query** edges ;



Exact Exponential Algorithm

Principle of the Algorithm

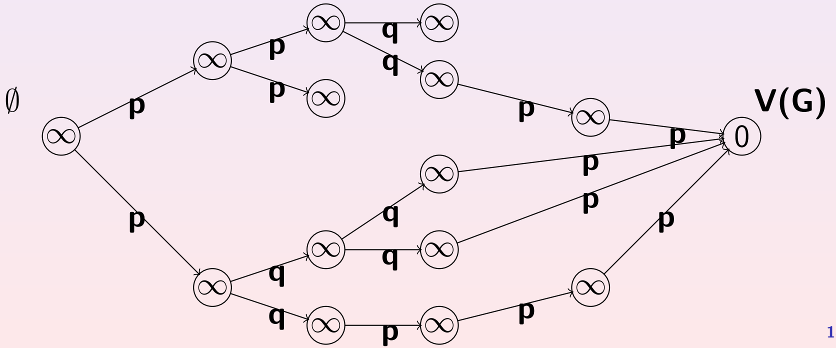
- To find a path in H from $S = \emptyset$ to $S = V(G)$. Correspondence with a search strategy.
- Labelling of vertices of the configuration graph.



Exact Exponential Algorithm

Principle of the Algorithm

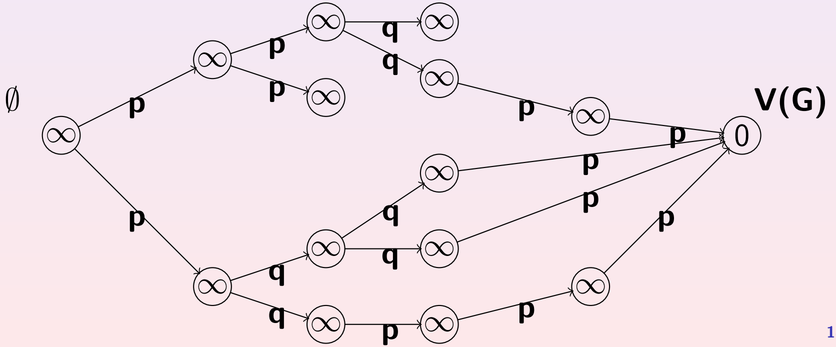
- *label*(S)= number of queries required to reach $V(G)$;
- Propagation of the labelling from $label(V(G)) = 0$;
  $label(\emptyset)$ is the minimum q such that $\mathbf{s}_q(G) = k$;



Exact Exponential Algorithm

Rules of labelling

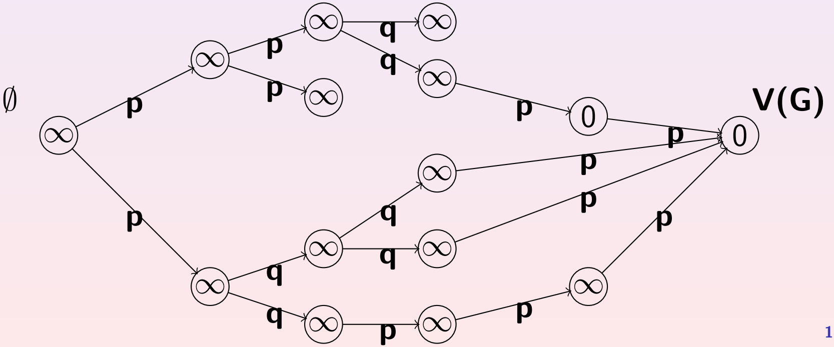
- (S, S') p-edge : $label(S) = \min(label(S), label(S'))$;
- (S, S_i) q-edges :
  $label(S) = \min(label(S), 1 + \max label(S_i))$.



Exact Exponential Algorithm

Rules of labelling

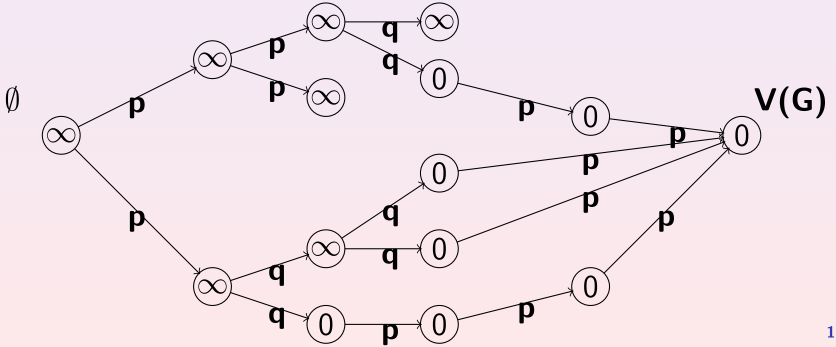
- (S, S') p-edge : $label(S) = \min(label(S), label(S')$;
- (S, S_i) q-edges :
  $label(S) = \min(label(S), 1 + \max label(S_i))$.



Exact Exponential Algorithm

Rules of labelling

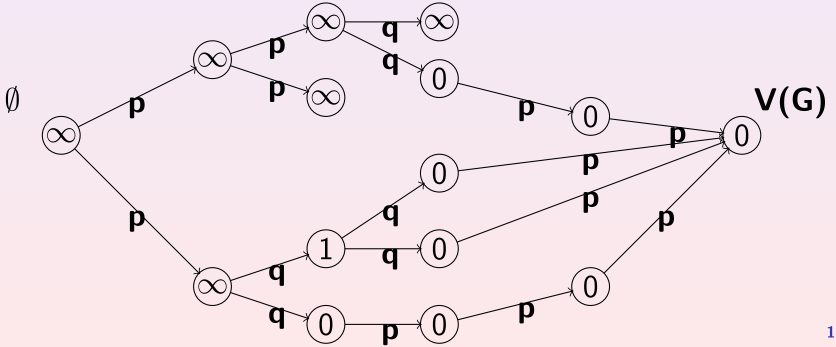
- (S, S') p-edge : $label(S) = \min(label(S), label(S'))$;
- (S, S_i) q-edges :
  $label(S) = \min(label(S), 1 + \max label(S_i))$.



Exact Exponential Algorithm

Rules of labelling

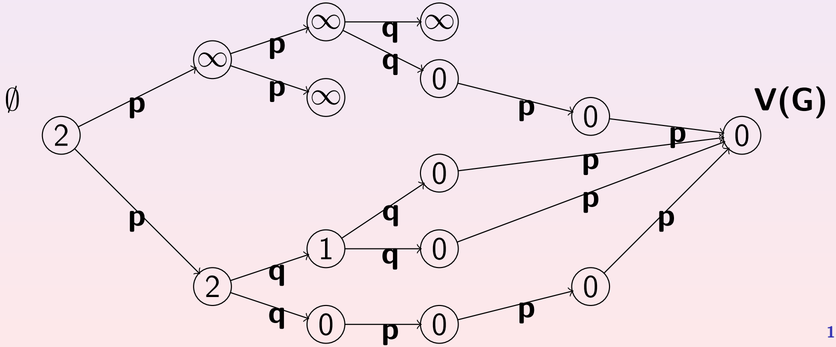
- (S, S') p-edge : $label(S) = \min(label(S), label(S'))$;
- (S, S_i) q-edges :
  $label(S) = \min(label(S), 1 + \max label(S_i))$.



Exact Exponential Algorithm

Rules of labelling

- (S, S') p-edge : $label(S) = \min(label(S), label(S')$;
- (S, S_i) q-edges :
  $label(S) = \min(label(S), 1 + \max label(S_i))$.



Conclusion and Further Work

New variant of search game

- nondeterministic search game ;
- unify pathwidth and treewidth.

Open Problems

- role of recontamination ;
- design of an $O(c^n)$-time exact algorithm ($c < 2$**) ;**
- **design of a linear algorithm for trees.**