Graph Searches Revisited : A lego of graph searches

Michel Habib
habib@liafa.jussieu.fr
http://www.liafa.jussieu.fr/~habib

Pretty Structures 2011, IHP, may 2011

Schedule

- Generic Search
- Breadth First Search
- Depth First Search
- Lexicographic Breadth First Search LexBFS
- Lexicographic Depth First Search LexDFS
- Maximal Neighbourhood Search (MNS)
- Lego with basic graph searches
- Importance of 4 points conditions for graph recognition
- The set of all simplicial schemes
- Principle of a Composition of Searches
- Hamiltonicity on co-comparability graphs

Graph searches are very well known and used

1. Euler (1735) for solving the famous walk problem in Kœnisberg
2. Tremeaux (1882) and Tarry (1895) using DFS to solve maze problems
3. Computer scientists from 1950 ....

Two main aspects for a graph search :

1. its principle or its algorithm
   (i.e. the description of the tie-break rules for the choice of the next vertex (edge) to be explored)
2. its implementation or its program

We will focus here on a third one :

- its characterization using forbidden structures
  and the relationships with the algorithm
- I will try to convince you that these characterizations can be helpful

Problem

For an undirected graph $G = (V, E)$,
explore the vertices of G "traversing or following" the edges.

Result

- a tree structure rooted at the first visited vertex
- an ordering σ of the vertices

Questions

- Under which conditions an ordering σ of the vertices corresponds to a search ?
- What are the properties of these orderings ?

Main reference for this today lecture :

These easy questions have been only recently systematically considered :
D.G. Corneil et R. M. Krueger, A unified view of graph searching, SIAM J. Discrete Math, 22, N°4 (2008) 1259-1276

Generic Search

Breadth First Search

Depth First Search

Lexicographic Breadth First Search LexBFS

Lexicographic Depth First Search LexDFS

Maximal Neighbourhood Search (MNS)

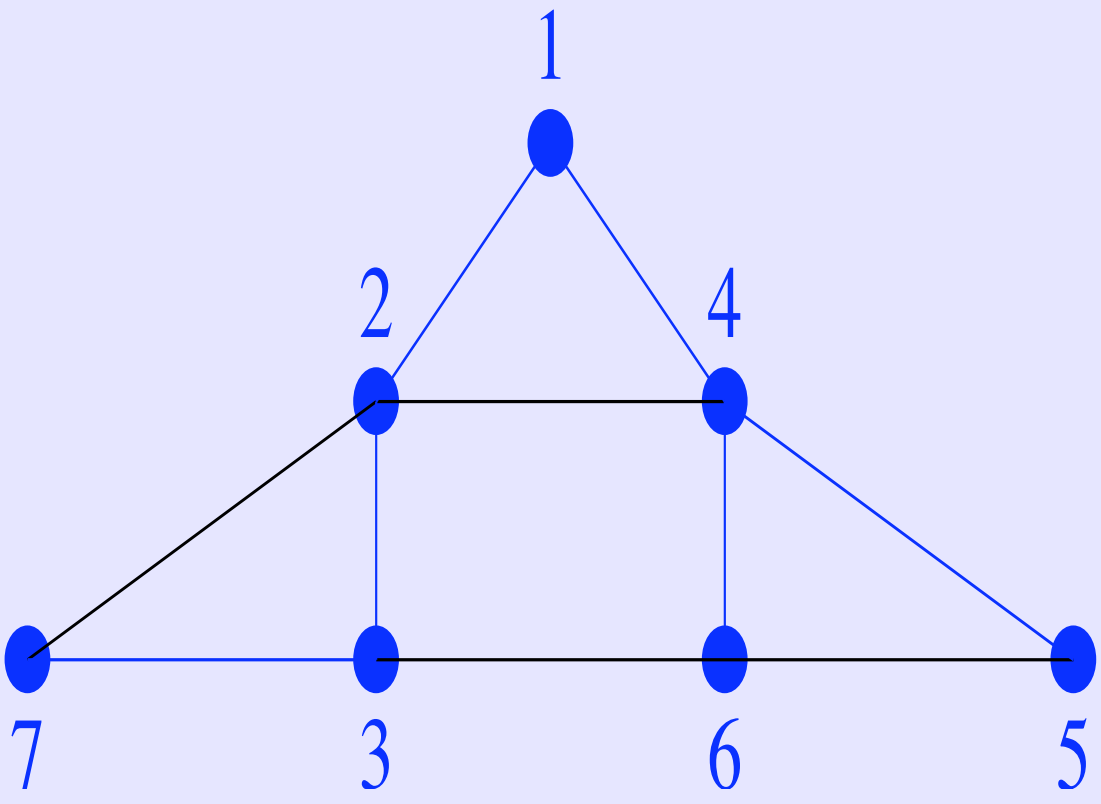
Lego with basic graph searches

Importance of 4 points conditions for graph recognition

The set of all simplicial schemes

Principle of a Composition of Searches

Hamiltonicity on co-comparability graphs



Invariant

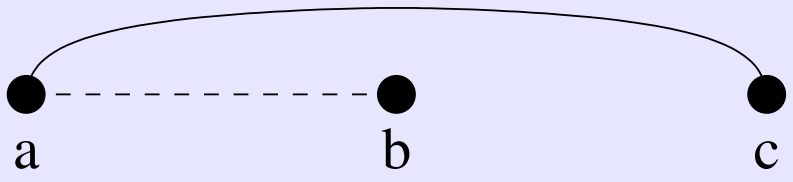
At each step, an edge between a visited vertex and a unvisited one is selected

Generic search

$S \leftarrow \{s\}$
for $i \leftarrow 1$ **à** n **do**
    Pick an unumbered vertex v of S
    $\sigma(i) \leftarrow v$
    foreach *unumbered vertex* $w \in N(v)$ **do**
        if $w \notin S$ **then**
            Add w to S
        end
    end
end

Generic question ?

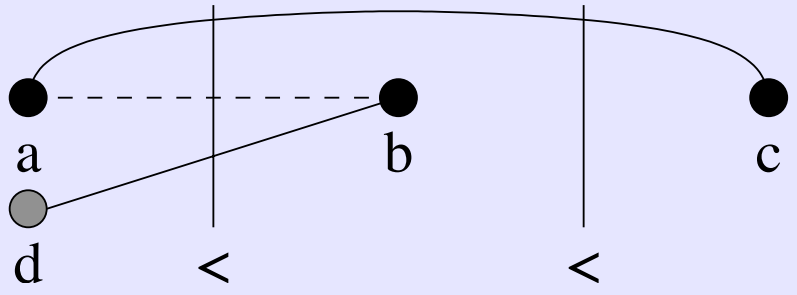
Let a, b et c be 3 vertices such that $ab \notin E$ et $ac \in E$.



Under which condition could we visit first a then b and last c ?

Property (Gen)

For an ordering σ on V, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then it must exist a vertex d such that $d < b$ et $db \in E$



Theorem

For a graph $G = (V, E)$, an ordering σ sur V is a generic search of G iff σ satisfies property (Gen).

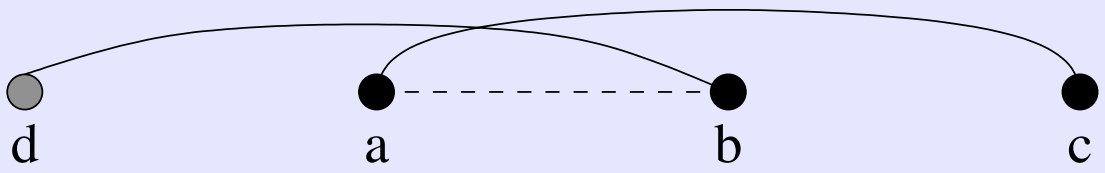
Most of the searches that we will study are refinement of this generic search
i.e. we just add new rules to follow for the choice of the next to be visited

BFS

Data: a graph $G = (V, E)$ and a start vertex s
Result: an ordering σ of V
Create a queue S ; $S \leftarrow \{s\}$
for $i \leftarrow 1$ **à** n **do**
    Extract v from head of S
    $\sigma(i) \leftarrow v$
    foreach *unumbered vertex* $w \in N(v)$ **do**
        if $w \notin S$ **then**
            Add w at tail of S
        end
    end
end

Property (B)

For an ordering σ on V, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then it must exist a vertex d such that $d < a$ et $db \in E$



Theorem

For a graph $G = (V, E)$, an ordering σ sur V is a BFS of G iff σ satisfies property (B).

Applications

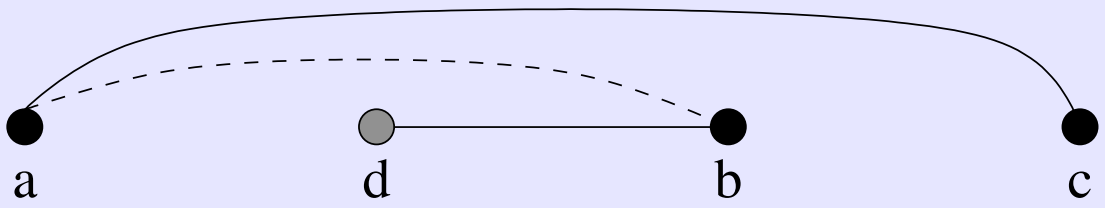
1. Distance computations (unit length), diameter and centers
2. BFS provides a useful layered structure of the graph
3. Using BFS to search an augmenting path provides a polynomial implementation of Ford-Fulkerson maximum flow algorithm.

Classical DFS

Data: a graph $G = (V, E)$ and a start vertex s
Result: an ordering σ of V
Create an empty stack S
Add s on top of S
for $i \leftarrow n$ **à** 1 **do**
    Extract v from top of the stack S
    $\sigma(i) \leftarrow v$
    foreach *unnumbered vertex* w *adjacent to* v **do**
        if $w \notin S$ **then**
            Add w on top of the stack S
        end
    end
end

Proprerty (D)

For an ordering σ on V, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then it must exist a vertex d such that $a < d < b$ and $db \in E$.



Theorem

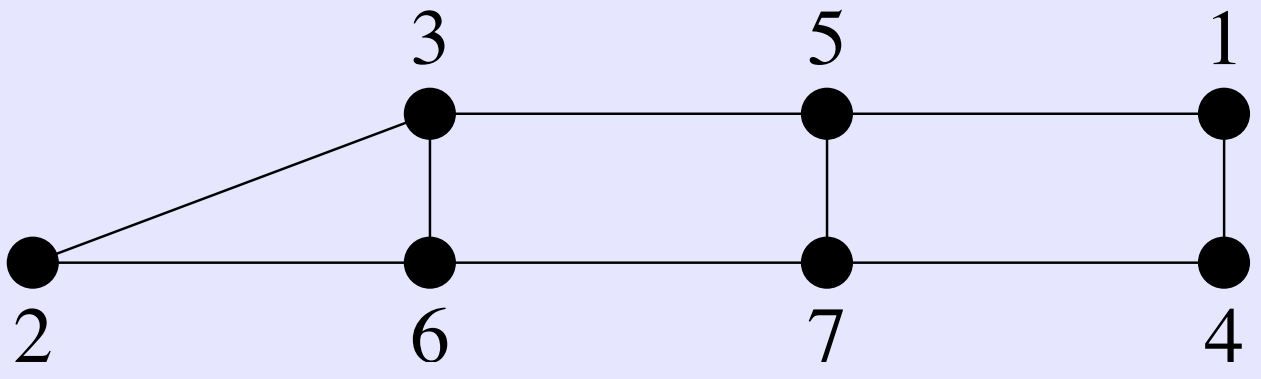
For a graph $G = (V, E)$, an ordering σ sur V is a DFS of G iff σ satisfies property (D).

Some applications

- Planarity testing.
- Computation of 2-connected (resp. strongly connected) components.
- Computation of a linear extension (topological sorting) for an acyclic digraph, applications to inheritance mechanisms. . . .

Lexicographic Breadth First Search (LexBFS)

Data: a graph $G = (V, E)$ and a start vertex s
Result: an ordering σ of V
Assign the label $\emptyset$ to all vertices
$label(s) \leftarrow \{n\}$
for $i \leftarrow n$ **à** 1 **do**
    Pick an unumbered vertex v with lexicographically largest label
    $\sigma(i) \leftarrow v$
    foreach *unnumbered vertex* w *adjacent to* v **do**
        $label(w) \leftarrow label(w).\{i\}$
    end
end

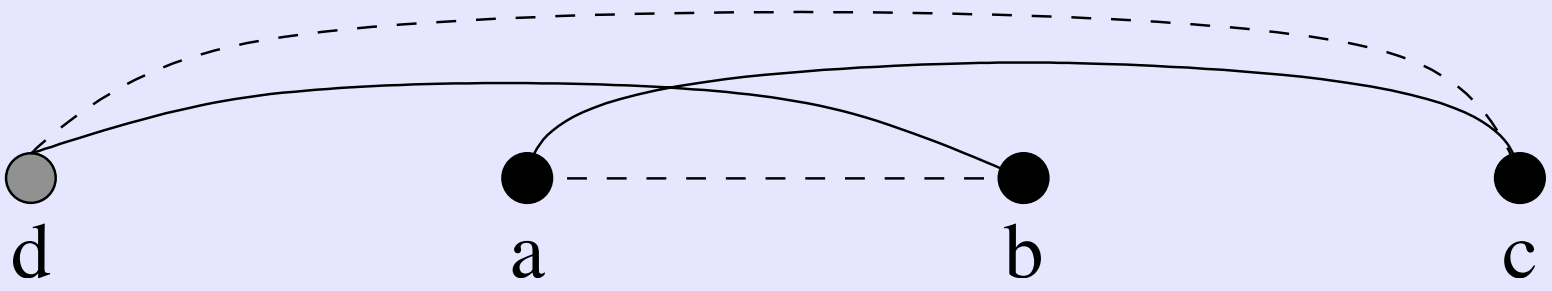

3
5
1
2
6
7
4

It is just a breadth first search with a tie break rule.

We are now considering a characterization of the order in which a LexBFS explores the vertices.

Property (LexB)

For an ordering σ on V, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then it must exist a vertex d such that $d < a$ et $db \in E$ et $dc \notin E$.



Theorem

For a graph $G = (V, E)$, an ordering σ sur V is a LexBFS of G iff σ satisfies property (LexB).

Some Applications

1. Most famous one : Chordal graph recognition
2. For many classes of graphs using LexBFS ordering "backward" provides structural information on the graph.
3. Last visited vertex (or clique) has some property (example simplicial for chordal graph)

A characterization theorem for chordal graphs

Theorem

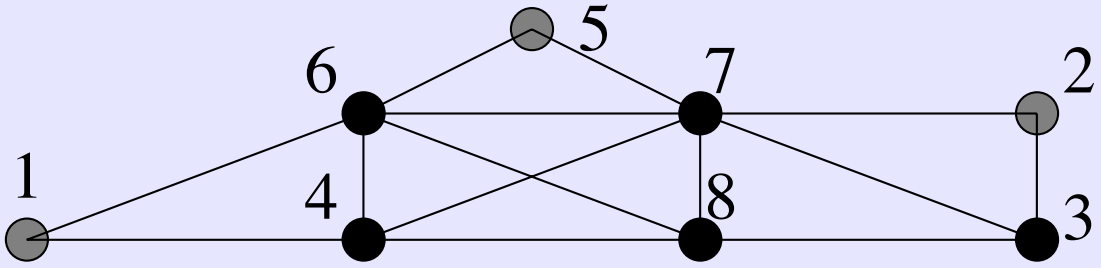
Dirac 1961, Fulkerson, Gross 1965, Gavril 1974, Rose, Tarjan, Lueker 1976.

(0) *G is chordal (every cycle of length* ≥ 4 *has a chord) .*

(i) *G has a simplicial elimination scheme*

(ii) *Every minimal separator is a clique*

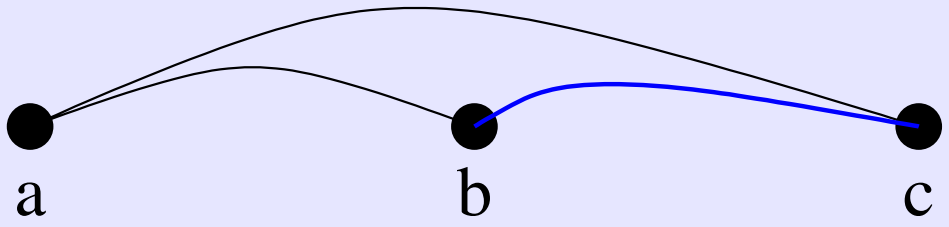
Simplicial



A vertex is simplicial if its neighbourhood is a clique.

Simplicial elimination scheme

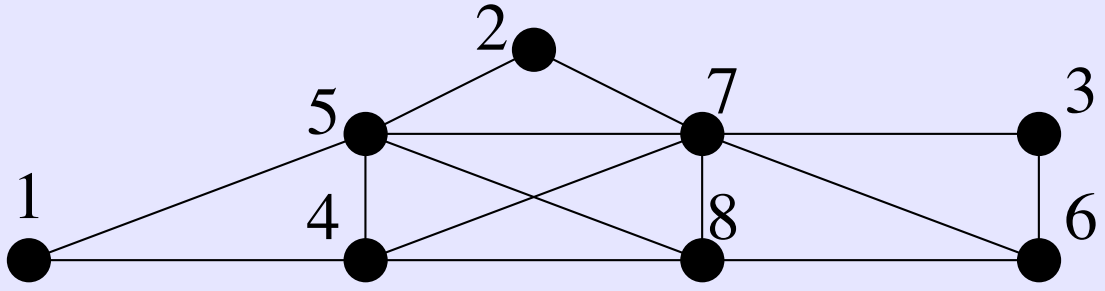
$\sigma = [x_1 \dots x_i \dots x_n]$ is a simplicial elimination scheme if x_i is simplicial in the subgraph $G_i = G[\{x_i \dots x_n\}]$



Chordal graphs are hereditary

Theorem [Tarjan et Yannakakis, 1984]

G is a chordal graph iff every LexBFS ordering provides a simplicial elimination scheme.



How can we prove such an algorithmic theorem ?

1. A direct proof, finding the invariants ?
2. Find some structure of chordal graphs
3. Understand how LexBFS explores a chordal graph

A direct proof

Theorem [Tarjan et Yannakakis, 1984]

G is a chordal graph iff every LexBFS ordering provides a simplicial elimination scheme.

proof

Let c be the leftmost non simplicial vertex.
Therefore it exists $a < b \in N(c)$ with $ab \notin E$. Using LexB property, it necessarily exists $d < a$ with $db \in E$ and $dc \notin E$.
Since G is chordal, we have $ad \notin E$ (else we would have the cycle $[a, c, b, d]$ without a chord).
But then considering the triple d, a, b, it exists $d' < d$ such that $d'a \in E$ and $d'b \notin E$.
If $dd' \in E$, using the cycle $[d, d', a, c, b]$ we must have the chord $d'c \in E$ which provides the cycle $[d, d'c, b]$ which has no chord.
Therefore $dd' \notin E$.
And we construct an infinite sequence of such d and d'.

Of course property LexB was known by authors such as M. Golumbic to study
chordal graphs but they did not noticed that it was a characterization
of LexBFS

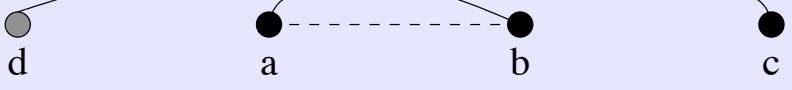
LexDFS

Can we introduce a Lexicographic depth first search ?

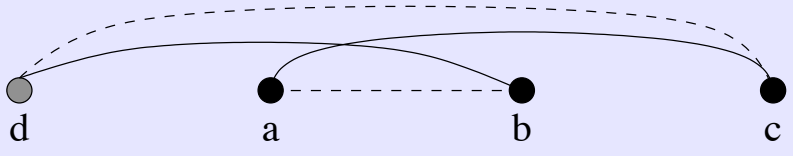
LexDFS

BFS vs LexBFS

BFS

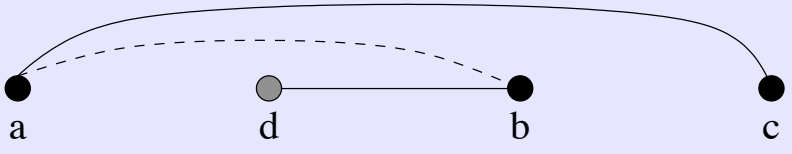


LexBFS

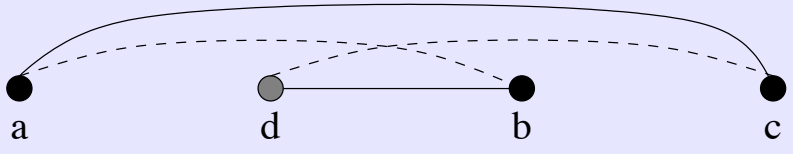


DFS vs LexDFS

DFS

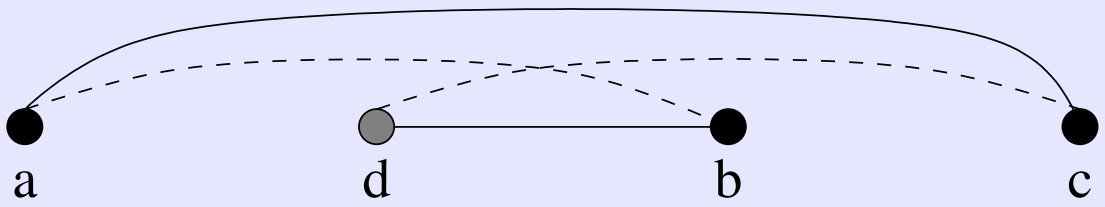


LexDFS



Property (LD)

For an ordering σ on V, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then it must exist a vertex d such that $a < d < b$ and $db \in E$ and $dc \notin E$.

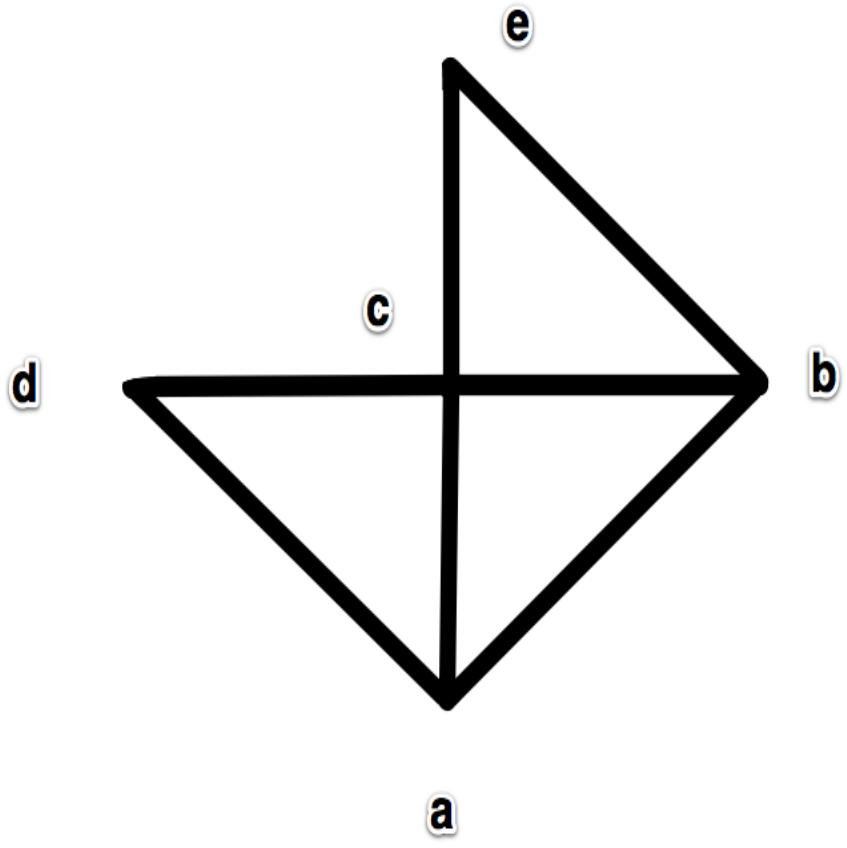


Theorem

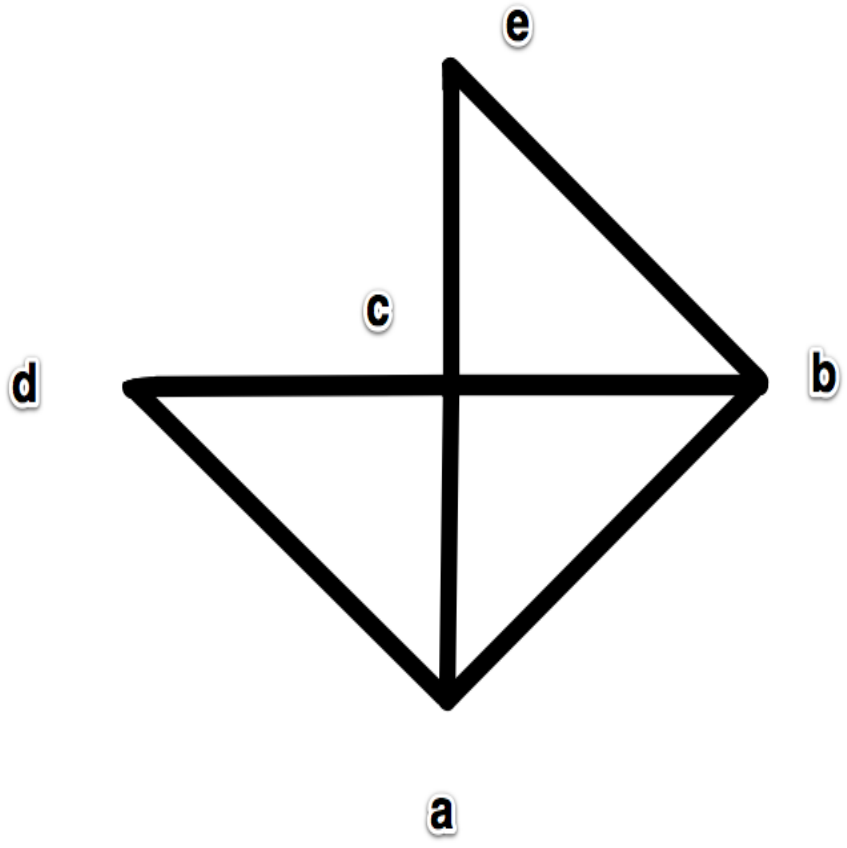
For a graph $G = (V, E)$, an ordering σ sur V is a LexDFS of G iff σ satisfies property (LD).

Lexicographic Depth First Search (LexDFS)

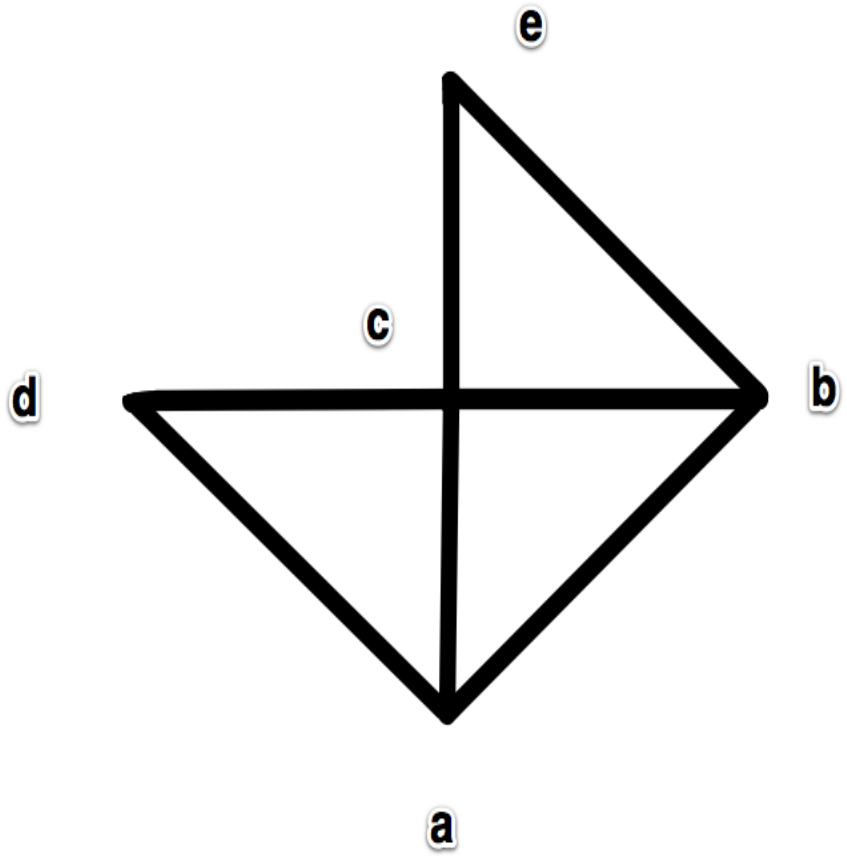
Data: a graph $G = (V, E)$ and a start vertex s
Result: an ordering σ of V
Assign the label $\emptyset$ to all vertices
$label(s) \leftarrow \{0\}$
for $i \leftarrow 1$ **à** n **do**
    Pick an unumbered vertex v with lexicographically largest label
    $\sigma(i) \leftarrow v$
    foreach *unnumbered vertex* w *adjacent to* v **do**
        $label(w) \leftarrow \{i\}.label(w)$
    end
end

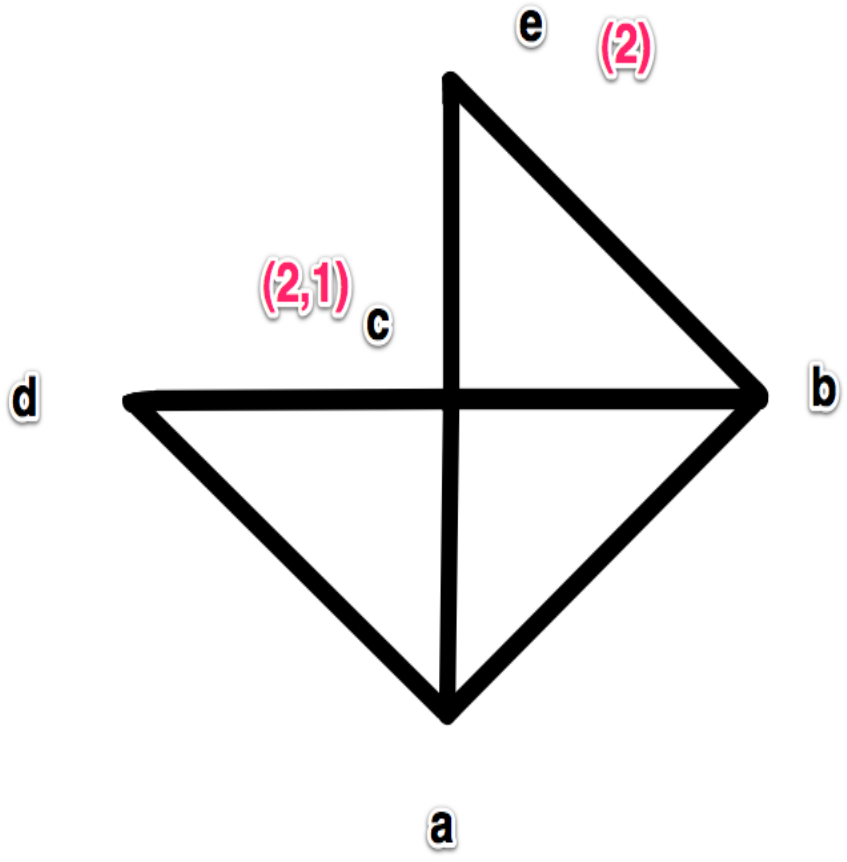


an example for LexDFS



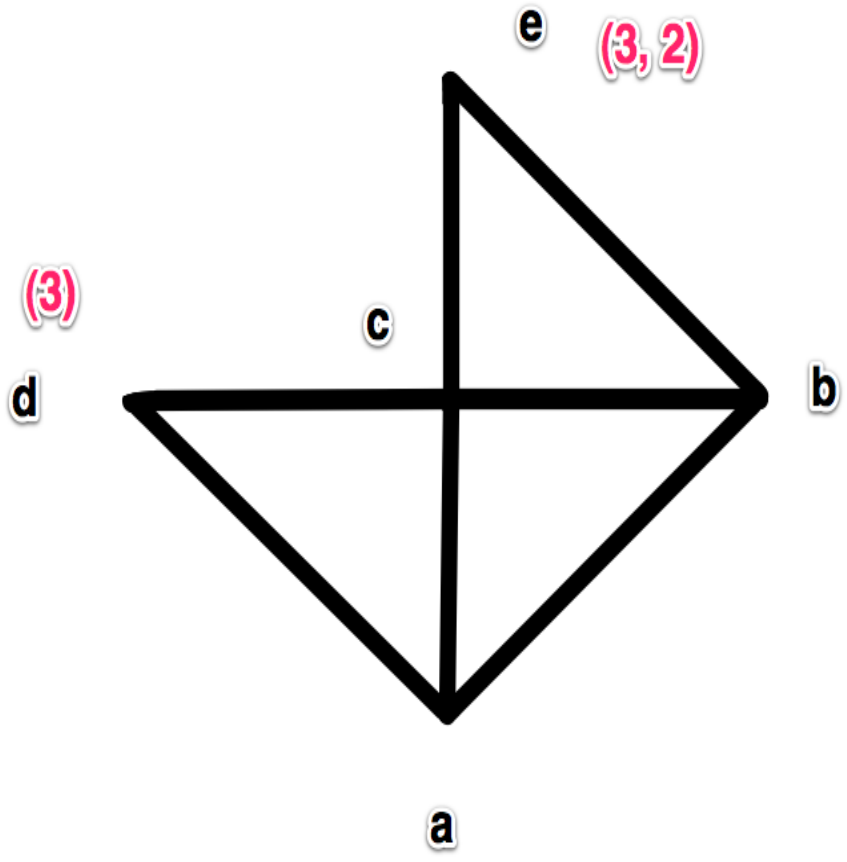
an example for LexDFS


e
c
d
b
a
start with a and first visit b



start with a and first visit b

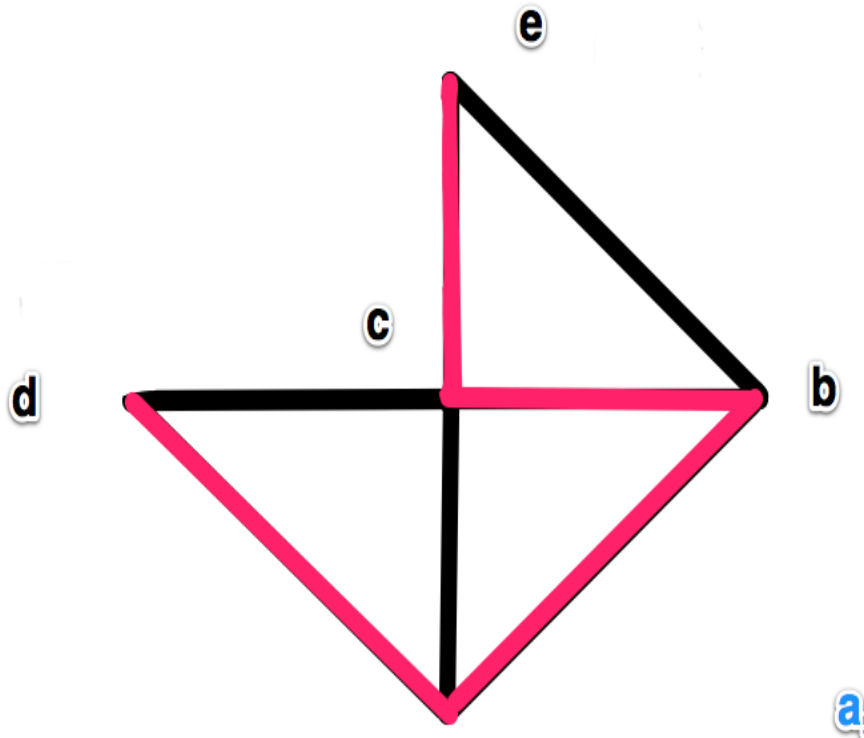
we must choose c next



start with a and first visit b

we must choose c next

then e is next and we finish in d



a, b, c, e, d

start with a and first visit b

we must choose c next

then e is next and we finish in d

LexDFS

Complexity

Is it possible to compute a LexDFS in $O(n+m)$?

Krueger and Spinrad independantly 2008

Best implementation so far needs $O(n+mloglogn)$ using Van der Boas trees.

Applications D. Corneil, B. Dalton, M. H. 2009

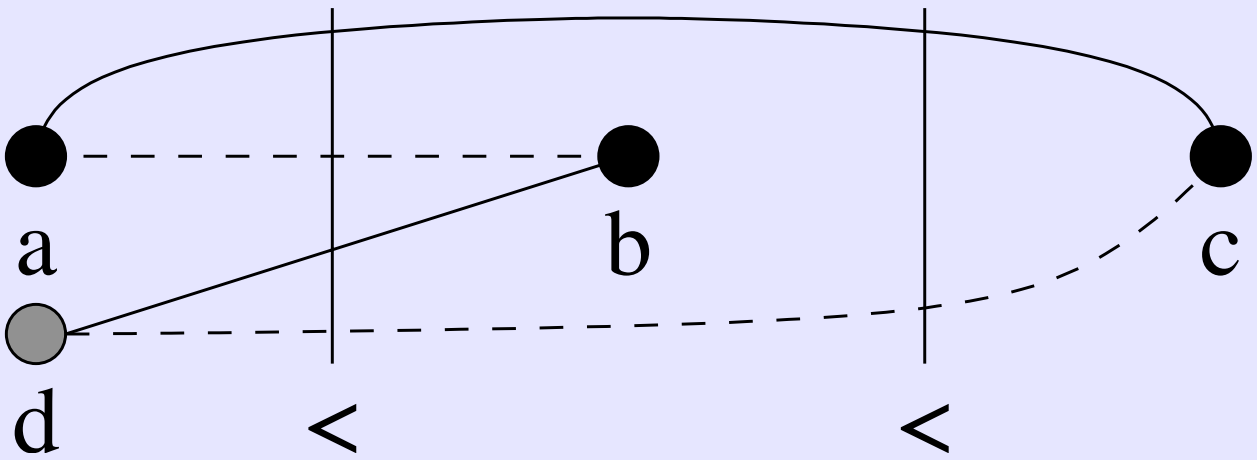
Hamiltonicity on co-comparability graphs via LexDFS.

MNS

Data: a graph $G = (V, E)$ and a start vertex s
Result: an ordering σ of V
$S \leftarrow \{s\}$
for $i \leftarrow 1$ **à** n **do**
    Pick an unumbered vertex $v \in S$ with a label maximal under inclusion
    $\sigma(i) \leftarrow v$
    foreach *unumbered vertex* $w \in N(v)$ **do**
        $label(w) \leftarrow \{i\} \cup label(w)$
    end
end

Property (MNS)

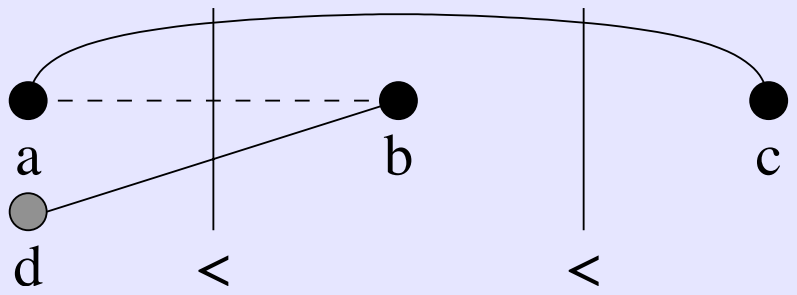
For an ordering σ on V, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then it must exist a vertex d such that $d < b$, $db \in E$ et $dc \notin E$.



Theorem

For a graph $G = (V, E)$, an ordering σ sur V is a MNS of G iff σ satisfies property MNS.

Back to the generic search



This search is a kind of Lex Generic Search (using analogy between BFS (resp. DFS) and LexBFS (resp. LexDFS). This is why MNS is sometimes named LexGen.

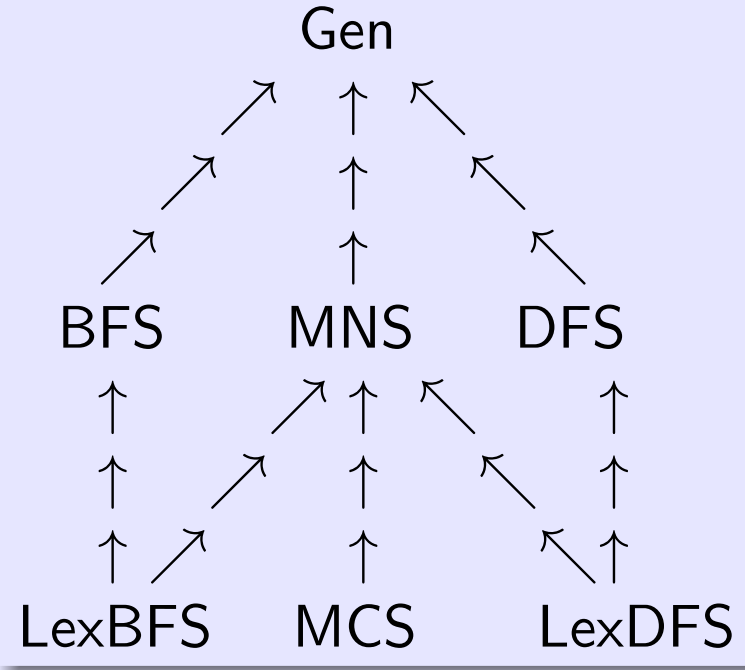
Maximal Cardinality Search

Data: a graph $G = (V, E)$ and a start vertex s
Result: an ordering σ of V
$S \leftarrow \{s\}$
for $i \leftarrow 1$ **à** n **do**
    Pick an unumbered vertex $v \in S$ with maximum label
    $\sigma(i) \leftarrow v$
    foreach *unumbered vertex* $w \in N(v)$ **do**
        $label(w) \leftarrow label(w) + 1$
    end
end

Conclusions

Using the 4-points configurations we have the following inclusion ordering between searches

Inclusions

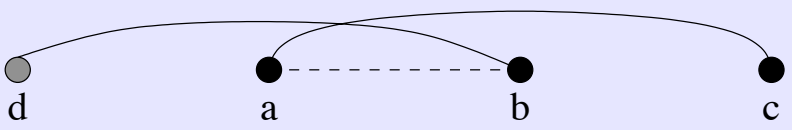


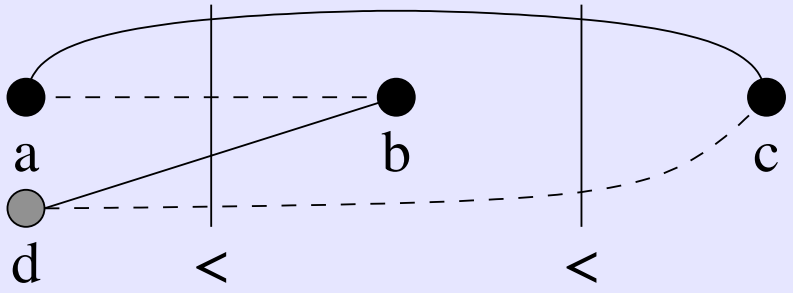
Exercises

1. What can we say of a search which is both BFS and MNS ?
   BFS+MNS =LexBFS ?
2. Same DFS and MNS
   DFS + MNS = LexDFS ?

BFS + MNS

BFS

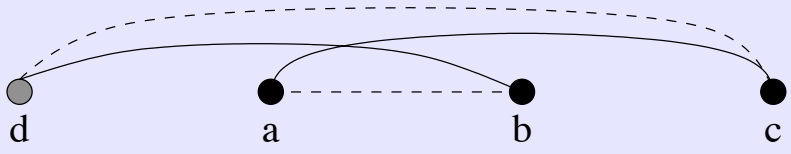




MNS

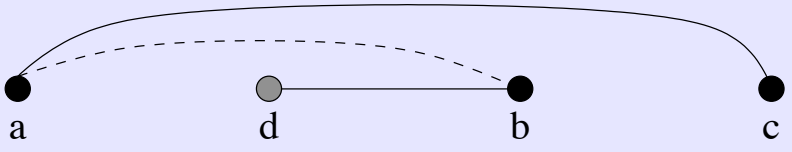
= LexBFS ?

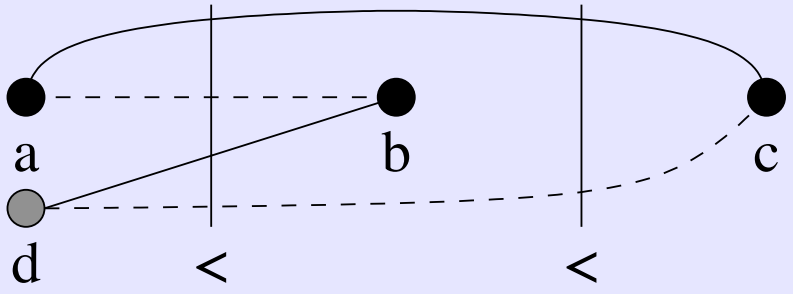
LexBFS



DFS + MNS

DFS

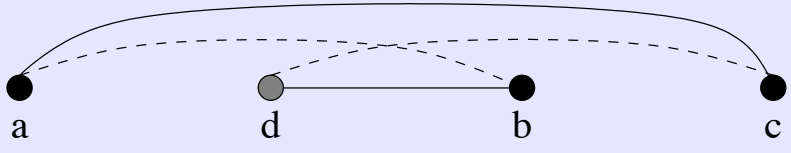




MNS

= LexDFS ?

LexDFS



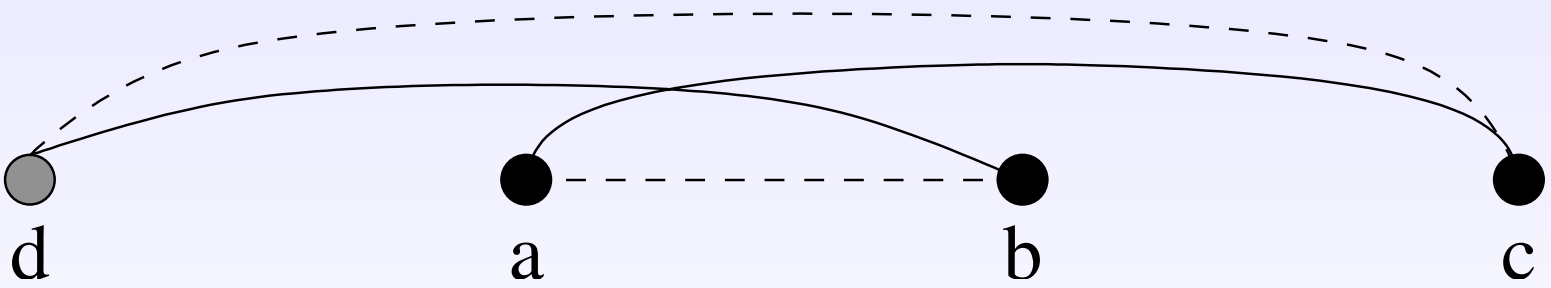
A kind of lego with simple searches

LexBFS

Data: a graph $G = (V, E)$ and a start vertex s
Result: an ordering σ of V
Assign the label $\emptyset$ to all vertices
$label(s) \leftarrow \{n\}$
for $i \leftarrow n$ **à** 1 **do**
    Pick an unumbered vertex v with lexicographically largest label
    $\sigma(i) \leftarrow v$
    foreach *unnumbered vertex w adjacent to v* **do**
        $label(w) \leftarrow label(w).\{i\}$
    end
end

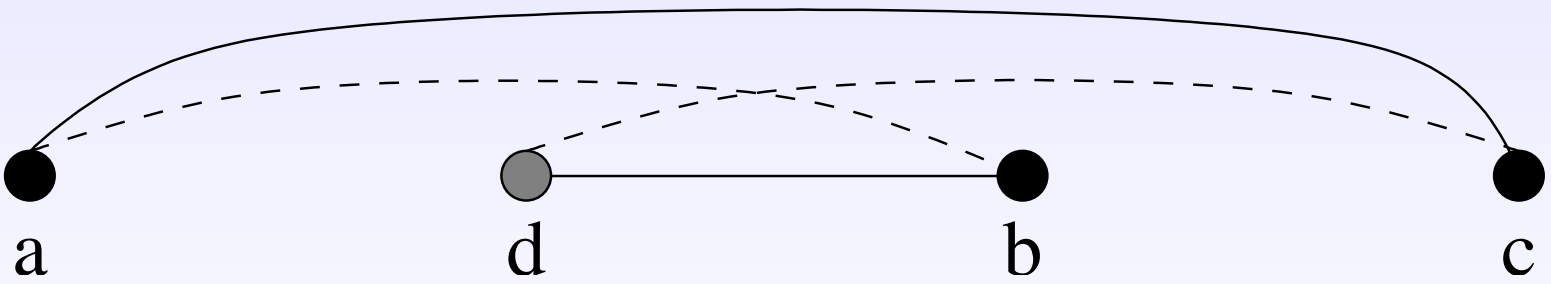
LexBFS

for $i \leftarrow n$ **à** 1 **do**
    Pick a lexicographic max
end
$label(w) \leftarrow label(w).\{i\}$



LexDFS

for $i \leftarrow 1$ **à** n **do**
    Pick a lexicographic max
end
$label(w) \leftarrow \{i\}.label(w)$



Using a remark by Berry, Krueger and Simonet 2008

co-LexBFS

for $i \leftarrow 1$ **à** n **do**
    Pick a lexicographic min
end
$label(w) \leftarrow label(w).\{i\}$

LexBFS on $\overline{G}$

co-LexDFS

for $i \leftarrow 1$ **à** n **do**
    Pick a lexicographic min
end
$label(w) \leftarrow \{i\}.label(w)$

LexDFS on $\overline{G}$

LexUP

for $i \leftarrow 1$ **à** n **do**
    Pick a lexicographic max
end
$label(w) \leftarrow label(w).\{i\}$

LexDown

for $i \leftarrow n$ **à** 1 **do**
    Pick a lexicographic max
end
$label(w) \leftarrow \{i\}.label(w)$

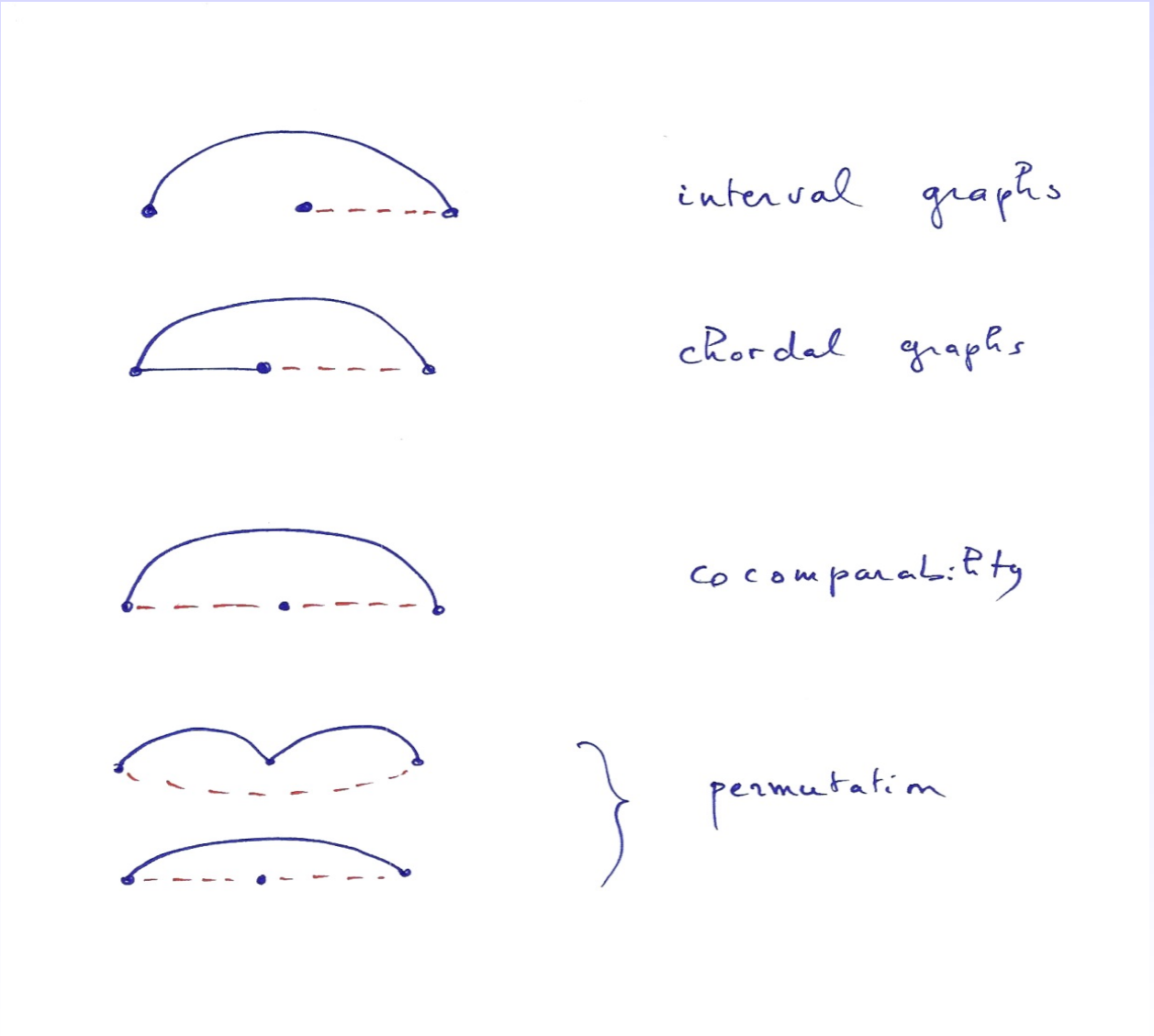
These two new searches LexUP and LexDown are studied by Jérémie Dusart (Master)

Importance of 4 points conditions for graph recognition

Many classes of graphs or partial orders can be characterized by the existence of a particular ordering of the vertices with some forbidden configuration on three points.
Examples with forbidden configuration on three points :

1. Interval graphs : ordering of the left ends of the intervals.
2. Chordal : simplicial elimination ordering.
3. Co-comparability : transitivity violation of the complement graph
4. Permutation : transitivity violation of the graph and its complement.

Forbidden 3 points suborderings



Consequences

LexBFS is involved in many recognition algorithms for these classes of graphs.

- Apply a LexBFS on $\overline{G}$ giving an ordering σ
- If G is a comparability graph the last vertex of σ, can be taken as a source in a transitive orientation of G.
- The starting point for comparability and permutation graph recognition algorithms.

There exist simplicial elimination schemes which are not MNS, LexBFS, LexDFS or MCS

M* search

Data: a graph $G = (V, E)$ and a start vertex s
Result: an ordering σ of V
$S \leftarrow \{s\}$
for $i \leftarrow 1$ **à** n **do**
    Pick an unumbered vertex v of S {applying the M selection rule on some connected component of G-{numbered vertices}}
    $\sigma(i) \leftarrow v$
    foreach *unumbered vertex* $w \in N(v)$ **do**
        Update its label applying the M rule
    end
    if $w \notin S$ **then**
        Add w to S
    end
end

Définition

So we can define : LexBFS*, LexDFS*, MCS*, MNS*.

First Properties

- GEN = GEN *
- LexBFS* is not BFS et LexDFS* is not DFS

Characterization of MNS*

Property (MNS*)

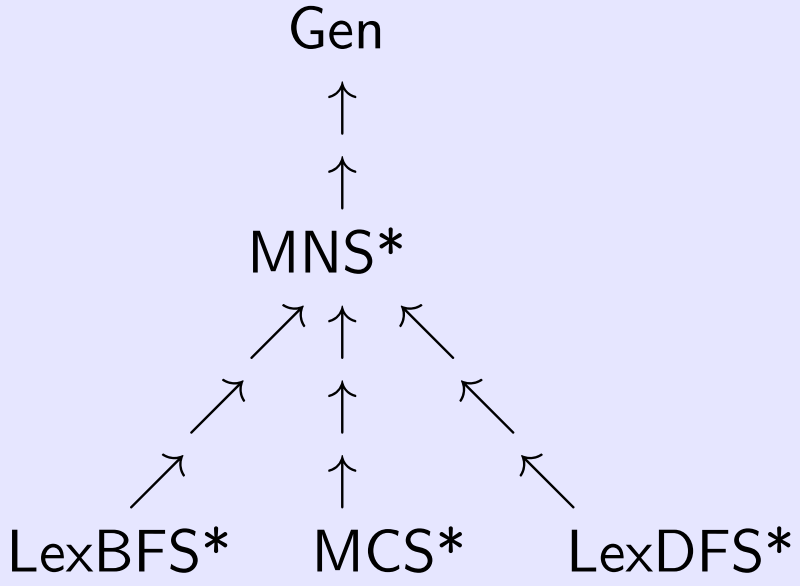
For an ordering σ on V, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then it must exist a vertex d such that $d < b$ et $db \in E$. Furthermore if b, c belong to the same connected component of $G - \{a]\}$ then $dc \notin E$.

Theorem

For a graph $G = (V, E)$, an ordering σ sur V is a MNS* of G iff σ satisfies property MNS*.
Analogous results yield for LexBFS*, LexDFS*, MCS*.

From the forbidden configurations we obtain :

Inclusions



Theorem Shier 1984

For a chordal graph MNS* =LexBFS* = LexDFS* = MCS* = { the set of all simplicial élimination schemes }.

Proof

First part of the proof.
Let c be the first non simplicial vertex to the left. It exist $a < b \in N(c)$ avec $ab \notin E$. b et c b and c belong to the same connected component of $G - a$] and therefore we can apply the proof for MNS.
Therefore every MNS* provides a simplicial elimination scheme for a chordal graph.
Therefore if G is chordal iff every search MNS* (resp. LexBFS*, LexDFS*, MCS*) provides a simplicial elimination scheme.

Since we focus on the ordering of the vertices as the result of a graph search, now we can compose graph searches in a natural way. Therefore we can denote by $M(G, x_0)$ the order of the vertices obtained by applying M on G starting from x_0.

Definition

Let M be a graph search and σ an ordering of the vertices of G, $M^+(G, \sigma)$ (resp. $M^-(G, \sigma)$) be the ordering of the vertices obtained by applying M on G starting from the vertex $\sigma(1)$ (resp. $\sigma(n)$) and tie-breaking using σ in decreasing (resp. increasing) order.

- Graph searches operate on total orderings : $M(G, M(G, \sigma)) = M^2(G, \sigma) \ldots$
- Does there exist fixed points ?
- Unfortunately a formal study of this composition remains to be done !
- Also called multisweep algorithms.

1. Such an idea was already used for planarity testing in some algorithm (de Fraysseix and Rosentiehl) with 2 consecutive DFS.
2. To compute efficiently the diameter of a graph using successive BFS

Linear time recognition algorithms for interval graphs

- Booth and Lueker 1976, using PQ-trees.
- Korte and Mohring 1981 using LexBFS and Modified PQ-trees.
- Hsu and Ma 1995, using modular decomposition and a variation on Maximal Cardinality Search.
- Corneil, Olariu and Stewart SODA 1998, using a series of 6 consecutive LexBFS, published in 2010.
- M.H, McConnell, Paul and Viennot 2000, using LexBFS and partition refinement on maximal cliques (a kind of LexBFS on cliques and minimal separators).
- New ! Peng Li, Yaokun Wu, using 3 special LexBFS, 2011

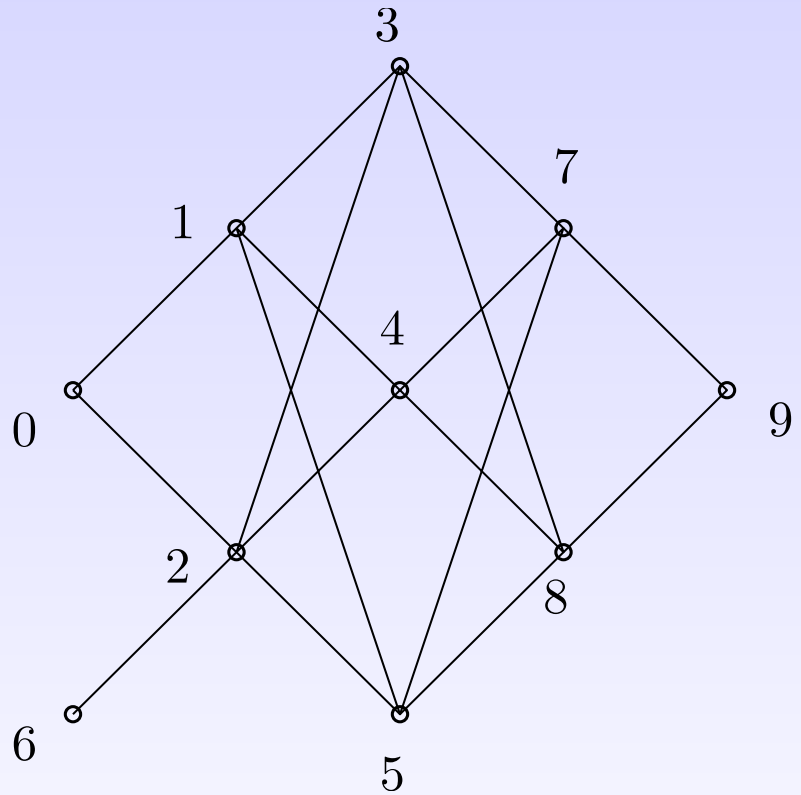
Our problem D. Corneil, B. Dalton and MH

- For a co-comparability graph G find a Minimum Path Cover
- Let P be a transitive orientation of $\overline{G}$
  our problem reduce to computing the bump number of P
  (Polynomial algorithm MH, Mohring, Steiner 1988)
- Another equivalent formulation as the 2-machine scheduling problem
  (Another polynomial algorithm Gabow, Tarjan 1985)

Our Algorithm

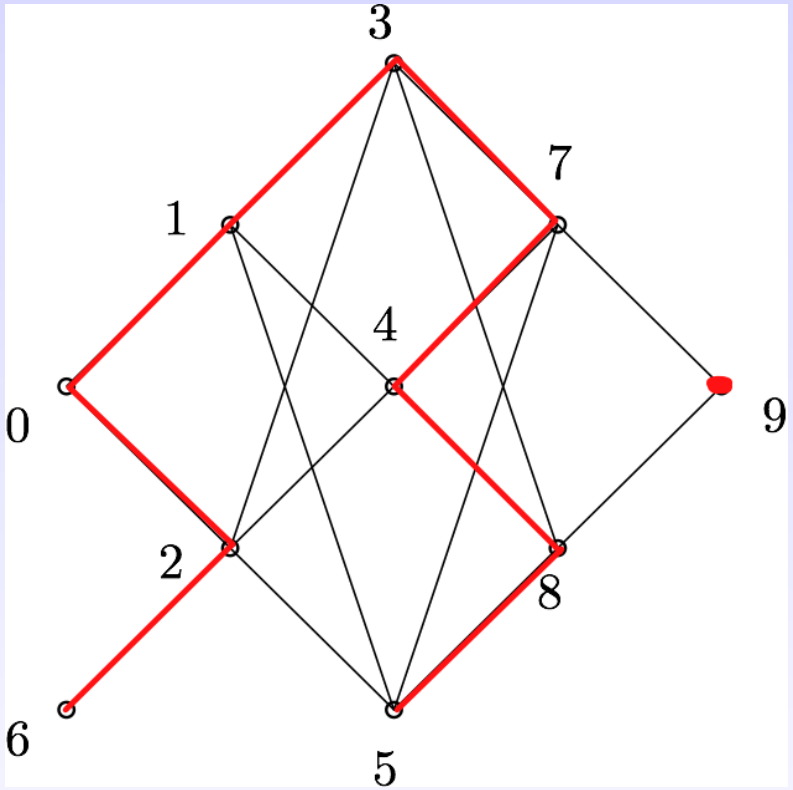
1. Start with σ any co-comparability ordering of G (a linear extension of P)
2. Apply $LDFS^+(\sigma)$ to produce an ordering τ.
3. Apply $RightMostNeighbour(\tau)$ which gives the path cover
4. Exhibit a certificate of minimality with a cut-set.

Let us take an example



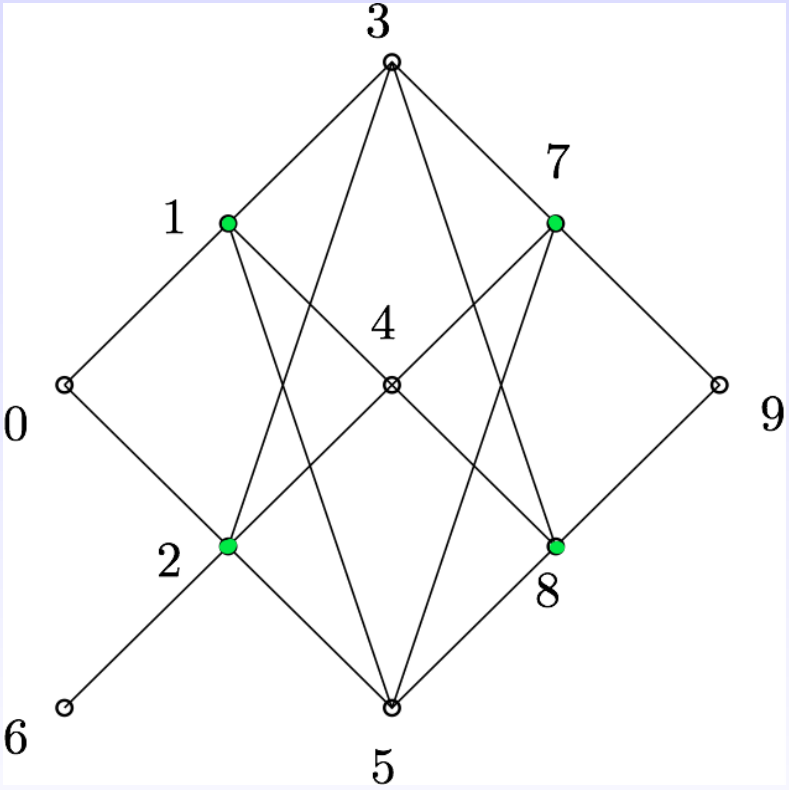
1. $\sigma = 2, 6, 0, 3, 4, 5, 1, 7, 8, 9$ a co-comparability ordering
2. $\tau = \mathit{LexDFS}^{+}(\sigma) = 9, 8, 5, 7, 4, 1, 3, 2, 0, 6$
3. $\mathit{RightMostNeighbour}(\tau) = 6, 2, 0, 1, 3, 7, 4, 8, 5, ||9$

Magic



1. $\mathit{RightMostNeighbour}(\tau) = 6, 2, 0, 1, 3, 7, 4, 8, 5, ||9$
2. The cutset $S = \{1, 7, 2, 8\}$ disconnects G into 6 connected components.

Lower bound



As a byproduct we obtain new methods to produce linear extensions

Important Lemma

If σ is a co-comp ordering of G, then $LDFS^+(\sigma)$ is a co-comp ordering of G.

Problem

Can this algorithmic method be generalized for AT-free graphs ?

Thank you for your attention !