Un autre point de vue sur les parcours de graphes,

Michel Habib
habib@irif.fr
http://www.irif.fr/~habib

22 décembre 2017

Schedule

Graph searches are very well known and used in many situations :

1. "Fil d'ariane" in the Greek mythology.
2. Euler (1735) for solving the famous walk problem in Kœnisberg city
3. Euler's theorem proved by Hierholzer in 1873.
4. Tremaux (1882) and Tarry (1895) using DFS to solve maze problems
5. Fleury, proposed a nice algorithm to compute an Euler Tour, cited in E. Lucas, Récréations mathématiques, Paris, 1891.
6. Computer scientists from 1950, in particular in the 70's, Tarjan for new applications of DFS....
7. 4 points characterizations Corneil, Krueger (2008), and the definition of LDFS a new interesting basic search.

First course of Graph Properties, I know in Paris, at CNAM

Graphs or networks :
A. Sainte-Laguë, *Les réseaux ou graphes*, **Gauthier-Villars**, Paris,1926.

Some definitions

Graph Search

The graph is supposed to be connected so as the set of visited vertices. After choosing an initial vertex, a search of a connected graph visits each of the vertices and edges of the graph such that a new vertex is visited only if it is adjacent to some previously visited vertex.
At any point there may be several vertices that may possibly be visited next. To choose the next vertex we need a tie-break rule. The breadth-first search (BFS) and depth-first search (DFS) algorithms are the traditional strategies for determining the next vertex to visit.

Our main question

Main Problem

What kind of knowledge or properties can we learn about the structure of a given graph via graph searching (i.e. with one or a series of successive graph searches) ?
A good example : Exact diameter computations on huge graphs via BFS.

Goals

- Building bottom up graph algorithms from well-known graph searches
- Develop basic theoretic tools for the structural analysis of graphs
- Applications on huge graphs : No need to store sophisticated data structures, just some labels on each vertex,

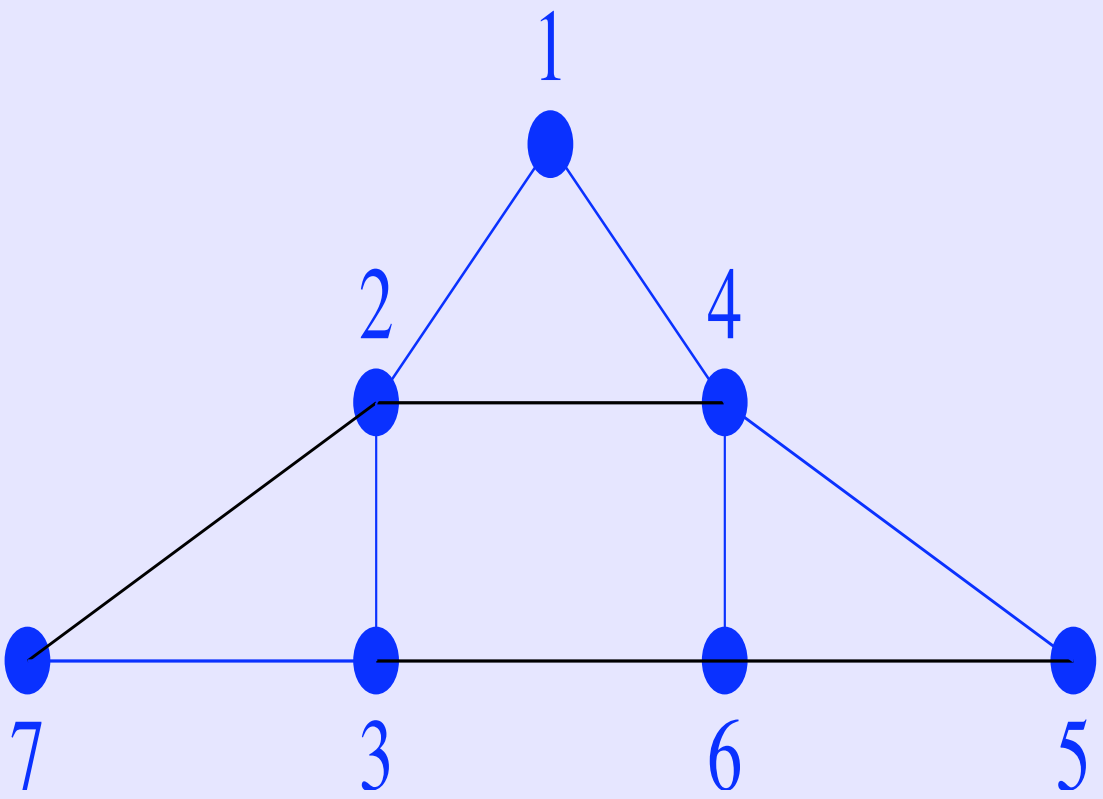
We can play with :

1. Find new uses of already known searches or describe new interesting searches designed for special purpose
2. Seminal paper :
   D.G. Corneil et R. M. Krueger, A unified view of graph searching, SIAM J. Discrete Math, 22, Num 4 (2008) 1259-1276

Basic graph searches

- Generic search, BFS, DFS
- LBFS, LDFS
- But also MNS, MCS

Generic Search



Invariant

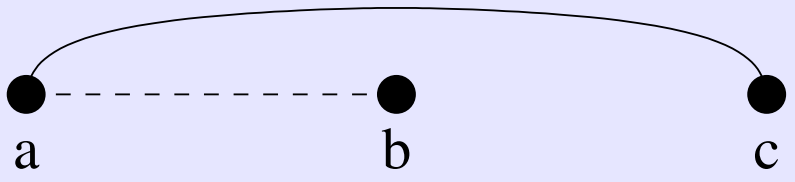
At each step, an edge between a visited vertex and a unvisited one is selected

Generic search

$S \leftarrow \{s\}$
for $i \leftarrow 1$ **to** 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 ?

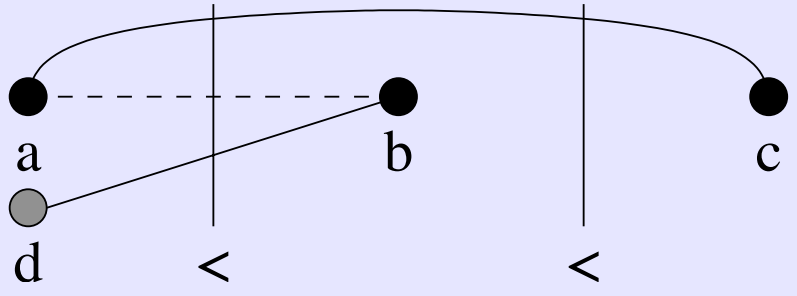
Let a, b et c be 3 vertices such that $ab \notin E$ et $ac \in E$.



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

Property (Generic)

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



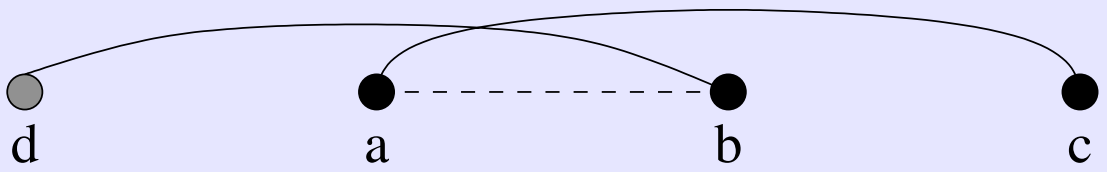
Theorem

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

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 vertex to be visited
Graph searches mainly differ by the management of the tie-break set

Property (BFS)

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



Theorem

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

Applications of BFS

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.

Lexicographic Breadth First Search (LBFS)

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

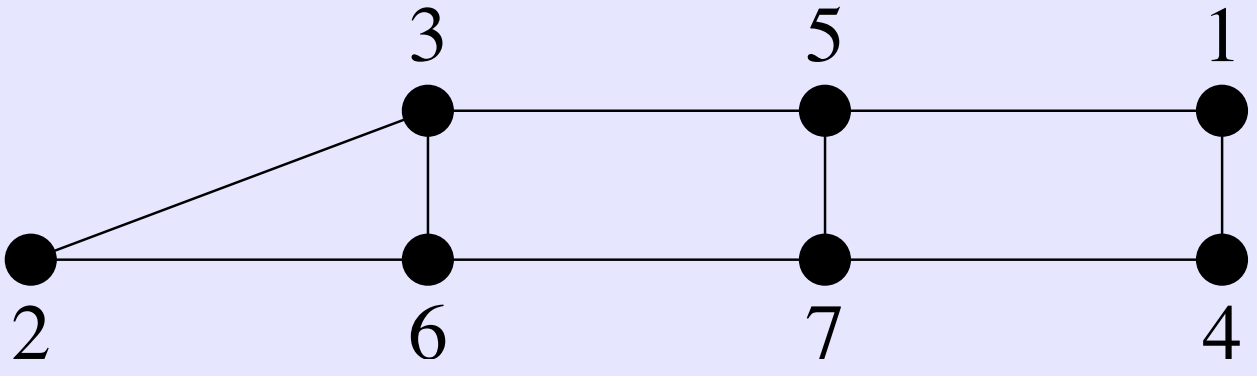
    $\sigma(i) \leftarrow v$

    foreach *unnumbered vertex* w *adjacent to* v **do**

        $label(w) \leftarrow label(w).\{i\}$

    end

end

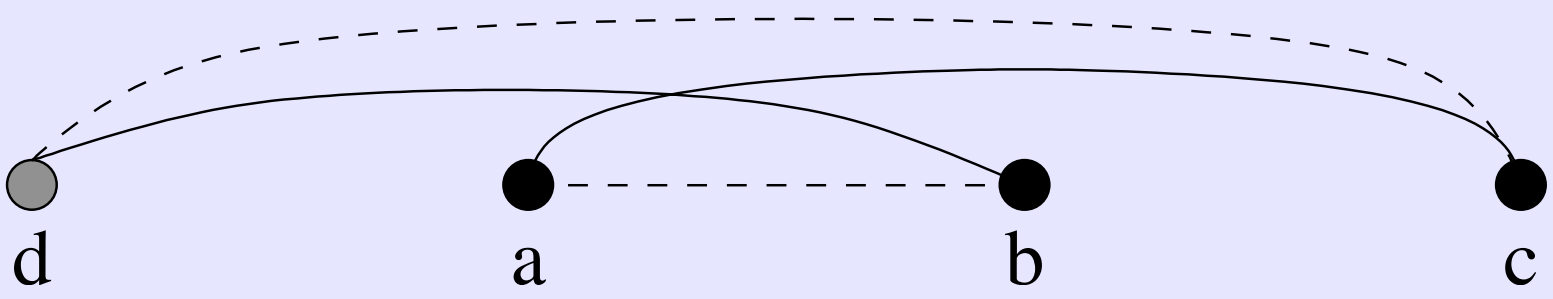

3
5
1
2
6
7
4

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

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

Property (LexB)

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



Theorem

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

Why LBFS behaves so nicely on well-structured graphs

A nice recursive property

On every tie-break set S, LBFS operates on $G(S)$ as a LBFS.

proof

Consider $a, b, c \in S$ such that $a <_\sigma b <_\sigma c$ and $ac \in E$ and $ab \notin E$, then it must exist a vertex d such that $d <_\sigma a$ et $db \in E$ et $dc \notin E$. But then necessarily $d \in S$.

Remark

Analogous properties are false for other classical searches.

LexBFS versus LBFS !

Google Images query : LBFS (thanks to Fabien)

yields :

First Answer

Applications of LBFS

1. Most famous one : chordal graph recognition via simplicial elimination schemes (easy application of the 4-points condition)
2. For many classes of graphs using LBFS ordering "backward" provides structural information on the graph.
3. Last visited vertex (or clique) has some property (example simplicial for chordal graph)
4. Of course property LexB was known by authors such as Tarjan or Golumbic to study chordal graphs but they did not noticed that it was a characterization of LBFS.

Parcours en profondeur (DFS)

Données: Un graphe $G = (V, E)$ et un sommet source s

Résultat: Un ordre total σ de V

Initialiser la pile S à s

pour $i \leftarrow 1$ **à** n **faire**

    Extraire le sommet v du haut de la pile S

    $\sigma(i) \leftarrow v$

    pour chaque *sommet non-numéroté* $w \in N(v)$ **faire**

        si w *n'est pas dans* S **alors**

            Ajouter w en haut de la pile S

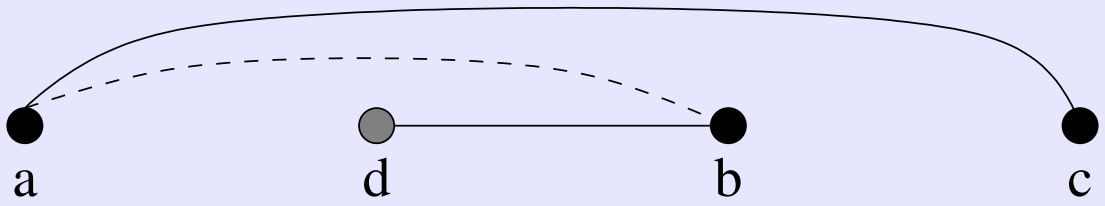
        fin

    fin

fin

Propriété (D)

Étant donné un ordre σ sur V, si $a < b < c$ et $ac \in E$ et $ab \notin E$, alors il existe un sommet d tel que $a < d < b$ et $db \in E$.



Théorème

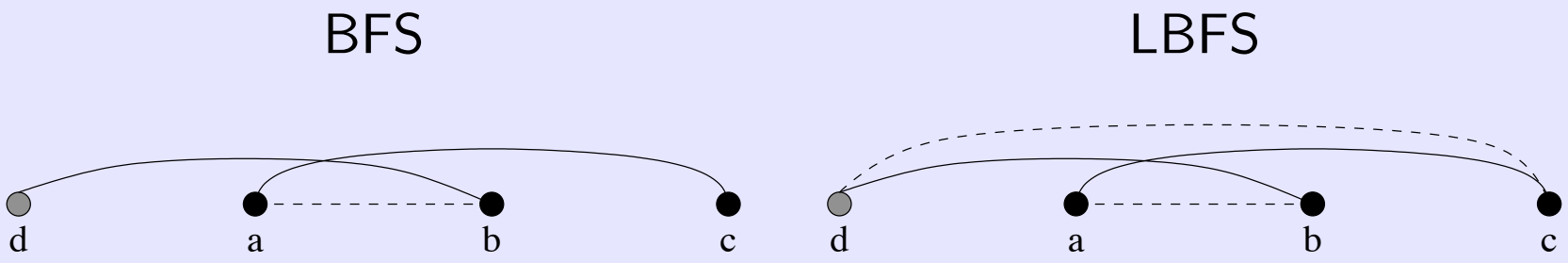
Pour un graphe $G = (V, E)$, un ordre σ sur V est un parcours DFS de G ssi σ vérifie la propriété (D).

Quelques exemples d'application

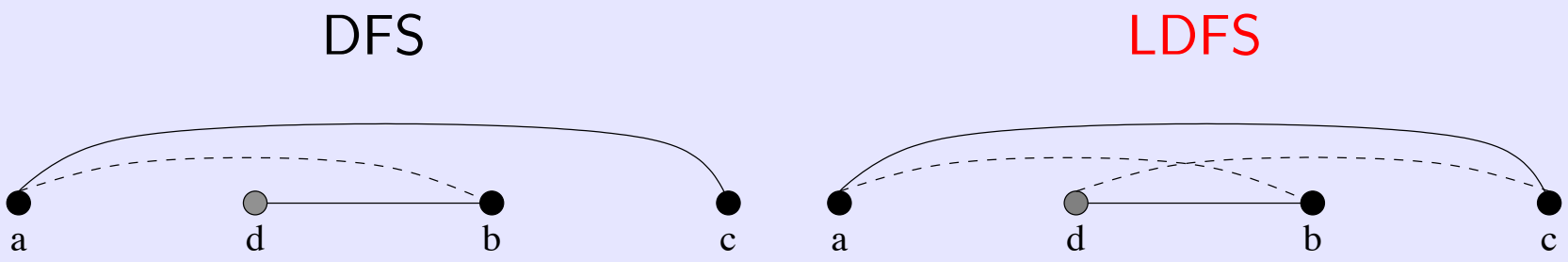
- Test de planarité (utilisation d'un DFS pour placer les arêtes autour de l'arbre associé au parcours).
- Composantes 2-connexes (resp. composantes fortement connexes dans le cas des graphes orientés).
- Tri topologique (extension linéaire) des graphes sans circuits, applications aux méchanismes d'héritage. . . .

LDFS

BFS vs LBFS

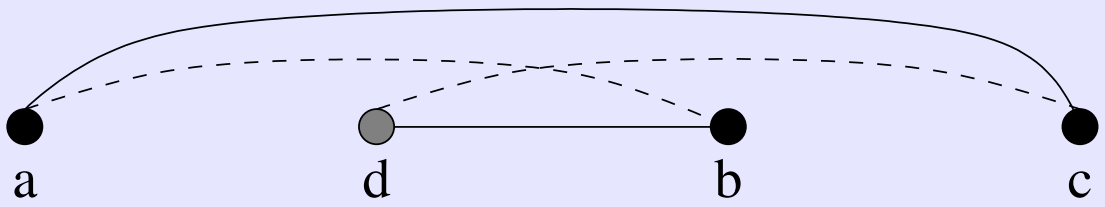


DFS vs LDFS



Property (LD)

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



Theorem

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

Lexicographic Depth First Search (LDFS)

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

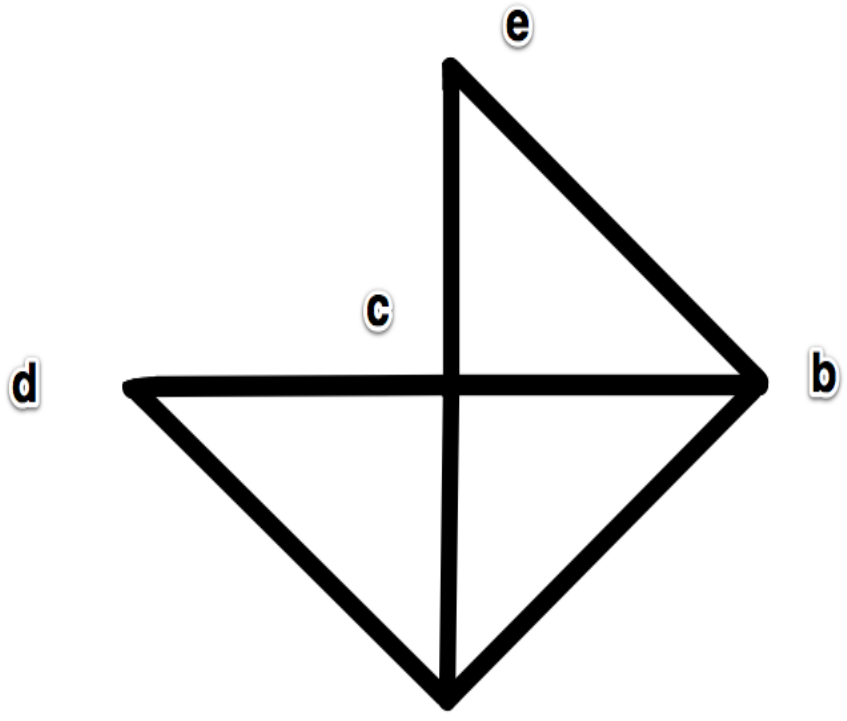
    $\sigma(i) \leftarrow v$

    foreach *unnumbered vertex* w *adjacent to* v **do**

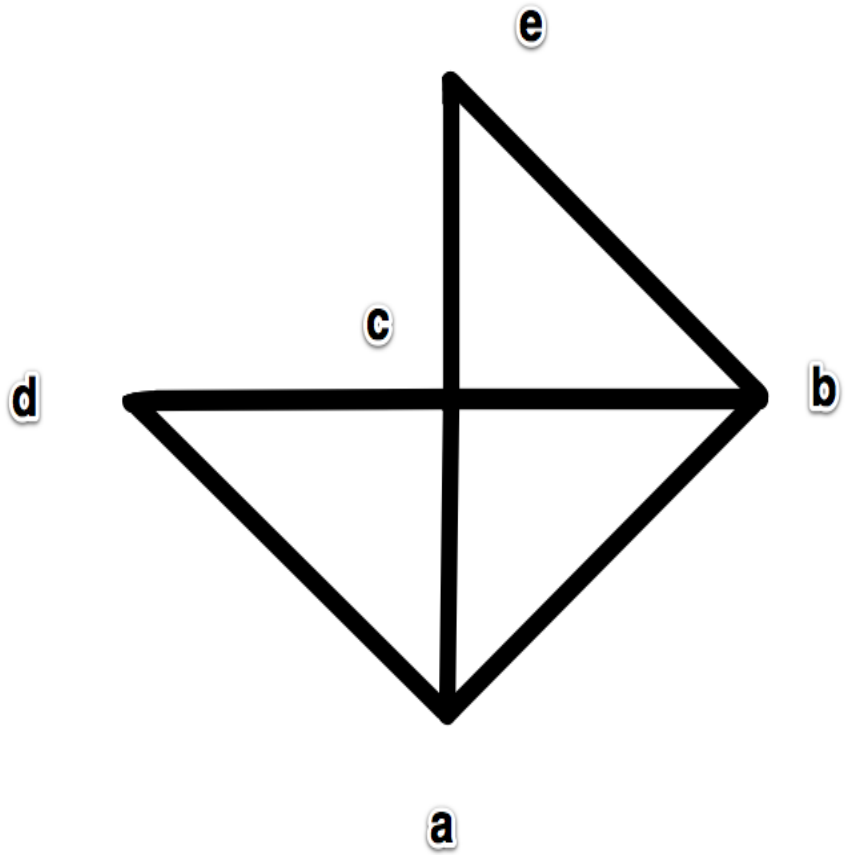
        $label(w) \leftarrow \{i\}.label(w)$

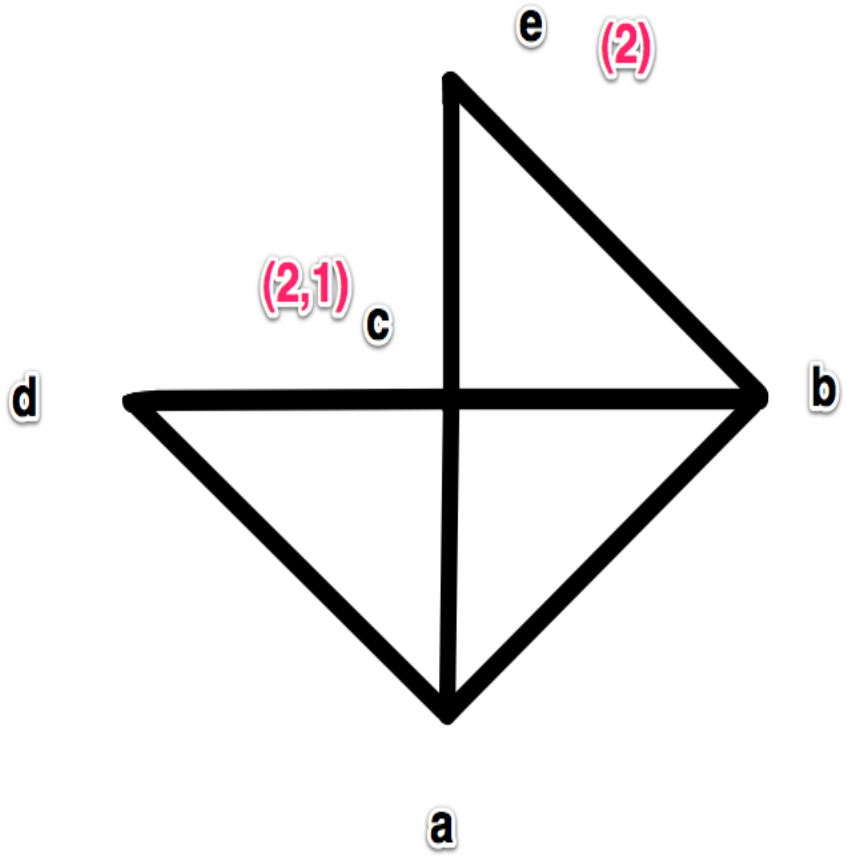
    end

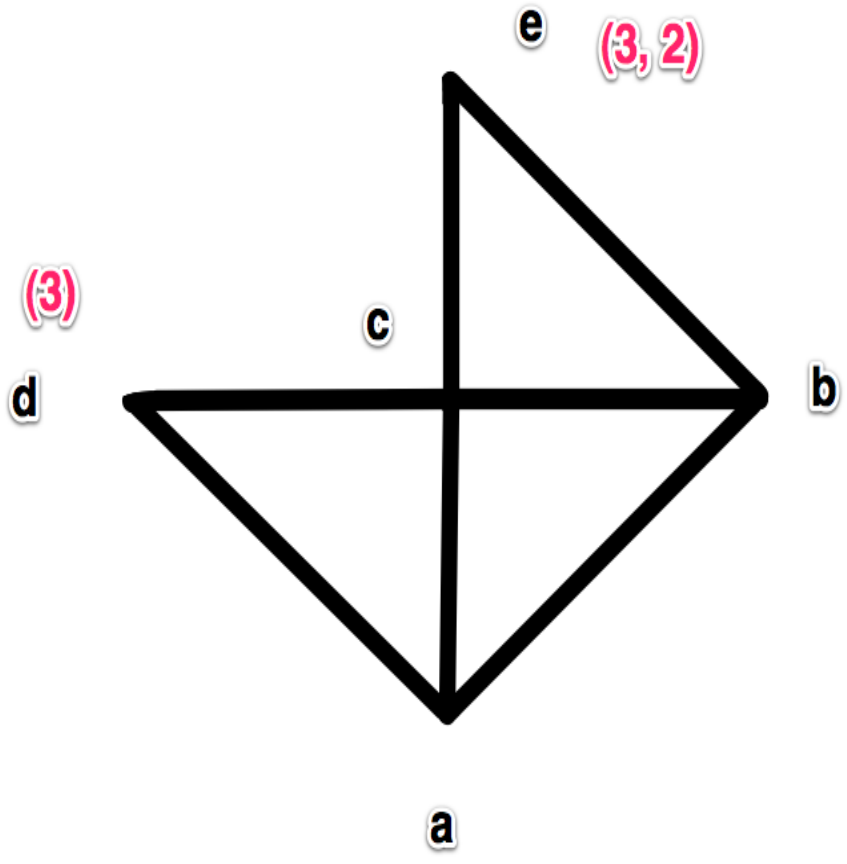
end

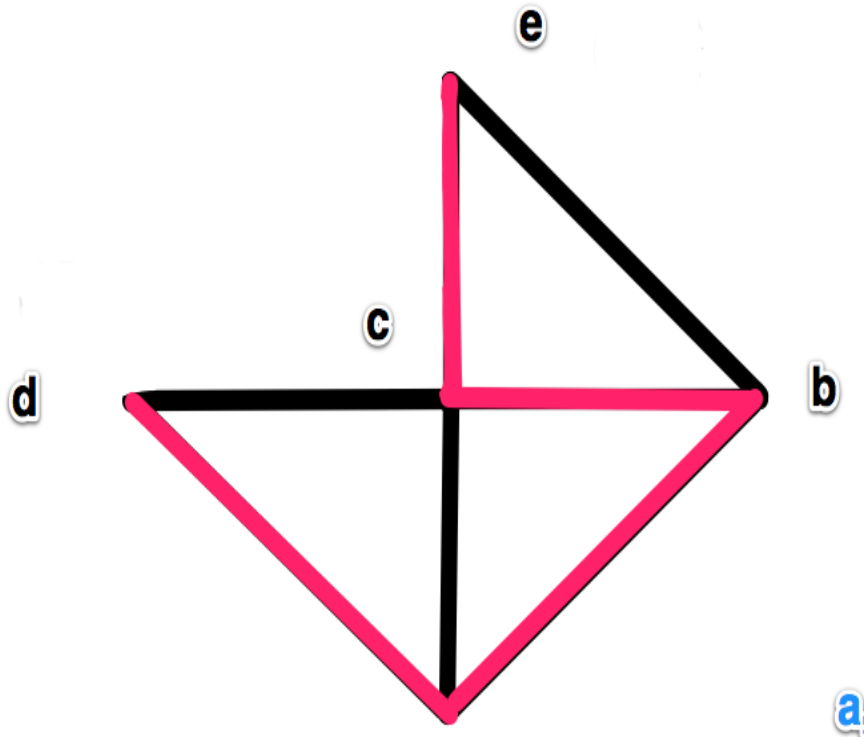


an example for LexDFS


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


e
(2)
(2,1)
c
d
b
a
start with a and first visit b
we must choose c next


e
(3, 2)
(3)
c
d
b
a
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

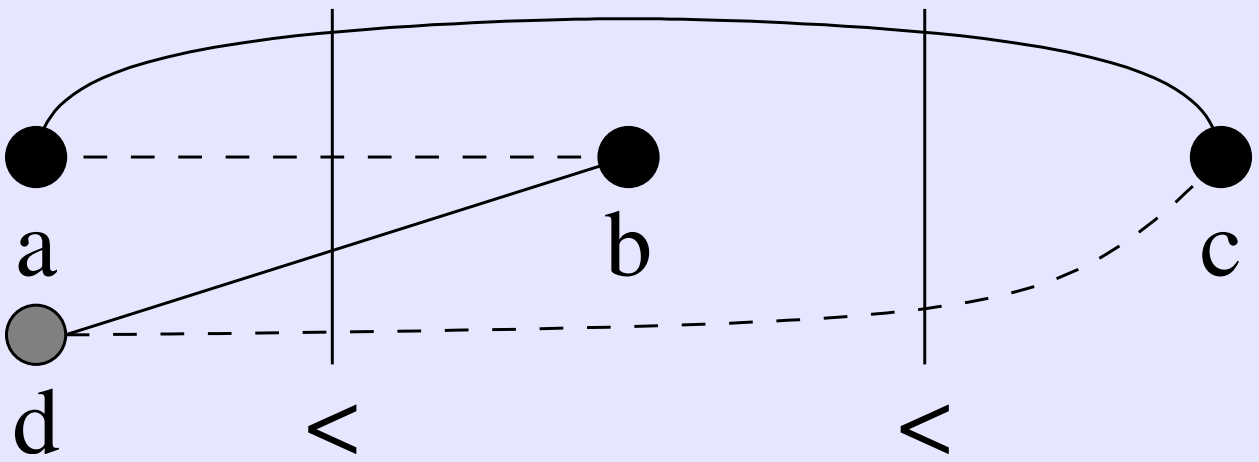
Parcours MNS

Parcours par voisinage maximal (MNS)

Données: Un graphe $G = (V, E)$ et un sommet source s
Résultat: Un ordre total σ de V
Affecter l'étiquette $\emptyset$ à chaque sommet
$label(s) \leftarrow \{n\}$
pour $i \leftarrow n$ **à** 1 **faire**
    Choisir un sommet v d'étiquette maximal pour l'inclusion.
    $\sigma(i) \leftarrow v$
    pour chaque *sommet non-numéroté* $w \in N(v)$ **faire**
        $label(w) \leftarrow \{i\} \cup label(w)$
    fin
fin

Propriété (MNS)

Étant donné un ordre σ sur V, si $a < b < c$ et $ac \in E$ et $ab \notin E$, alors il existe un sommet d tel que $d < b$, $db \in E$ et $dc \notin E$.

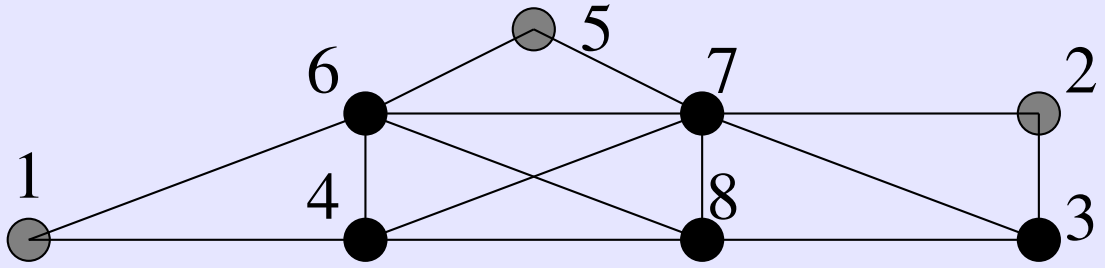


After a graph search $\mathcal{S}$ we may use two things :

1. The visiting ordering σ of the vertices
2. The 4-points conditions of the graph search $\mathcal{S}$
3. I shall try to convince you with some examples that this is enough to prove theorems on algorithms.

Recall the defintion of chordal graphs :
No induced cycle of length ≥ 4 (or equivalently : every cycle of length ≥ 4 has a chord).

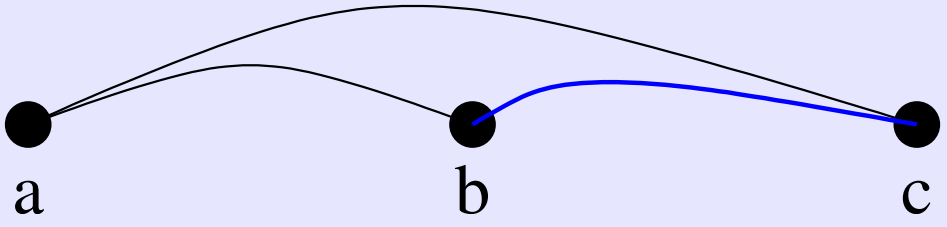
Chordal graph

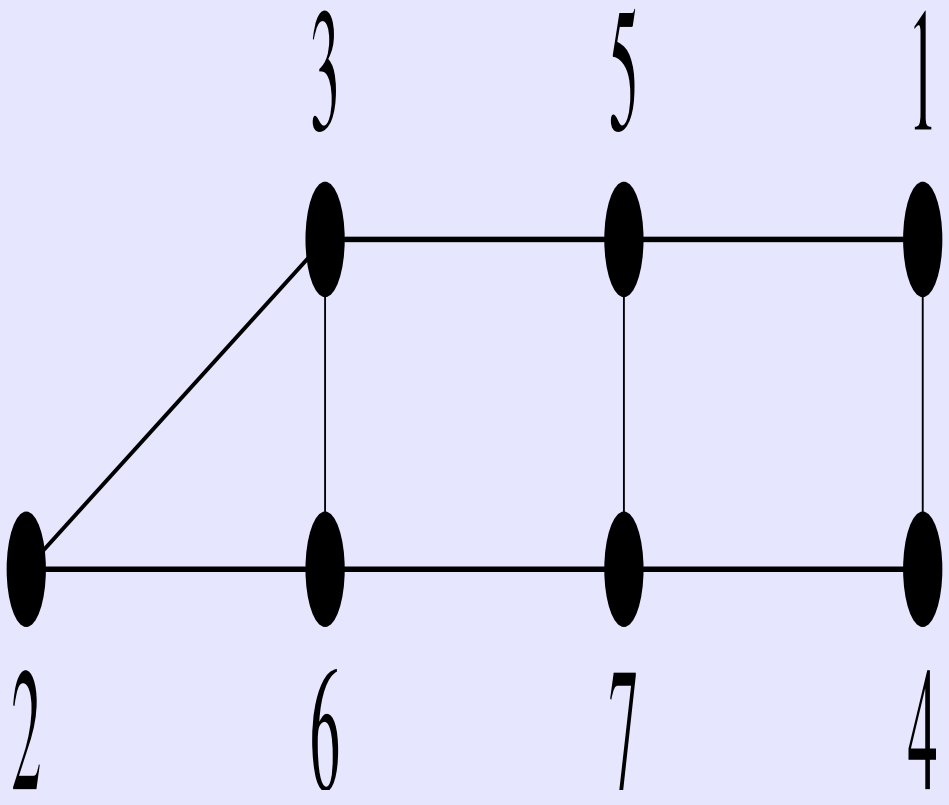


A vertex is simplicial if its neighbourhood is a clique.

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 $G_i = G[\{x_i \dots x_n\}]$



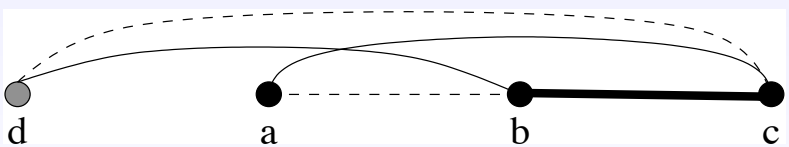

3
5
1
2
6
7
4

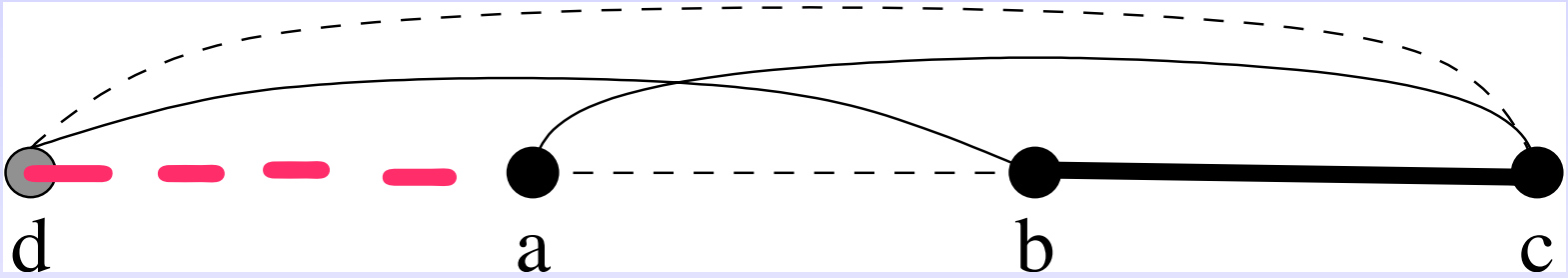
Theorem [Tarjan et Yannakakis, 1984]

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

Proof :

Let c be a non simplicial vertex.
There exist $a < b \in N(c)$ avec $ab \notin E$.
Using characterization of LexBFS orderings, it exists $d < a$ with $db \in E$ and $dc \notin E$. Since G is chordal, necessarily $ad \notin E$.





But then from the triple d, a, b, it exists $d' < d$ with $d'a \in E$ and $d'b \notin E$. Furthermore $d'd \notin E$ ...
And using the triple d', d, a , we start an infinite chain .....

Remark

Most of the proofs based on some characteristic ordering of the vertices are like that, with no extra reference to the algorithm itself.

Chordal graphs recognition so far

Chordal graph recognition

1. Apply a LexBFS on G $O(n+m)$
2. Check if the reverse ordering is a simplicial elimination scheme $O(n+m)$
3. In case of failure, exhibit a certificate : i.e. a cycle of length ≥ 4, without a chord. $O(n)$

Exercises

1. Is Tarjan Yannakakis's theorem also true for BFS, DFS or LDFS ?
2. Question sur G^2

Research problem

Design a linear time algorithm to recognize if a given ordering σ of $V(G)$ is LBFS-ordering
(same question for LDFS).

First let us consider a characterization of DFS and BFS.

proposition

For every beginning ordering of the vertices of a DFS (resp. BFS) $\sigma_i = x_1, \ldots, x_i$, the next vertex to be visited is adjacent to the latest (resp. first) vertex σ_i having a neighbour in $V(G)$-σ_i.

Theorem

Let G be a directed graph, σ a total ordering of the vertices is a DFS ordering iff for every triple $a <_\sigma b <_\sigma c$ with $ac \in A(G)$ and $ab \notin A(G)$, it exists necessarily some vertex d between a and b such that $db \in A(G)$.

proof

$\Rightarrow$ If σ is a DFS ordering, with $a <_\sigma b <_\sigma c$, then necessarily b must be reachable with a path from a. Just take for d the last vertex of this path and the condition is trivially true.
$\Leftarrow$ Let us prove it by induction that $\sigma_i = x_1, \ldots, x_i$ is a legitimate DFS ordering on $G(\{x_1, \ldots, x_i\})$. Using the previous proposition, suppose that the last vertex in σ_i connected to x_i is x_l and that there exists a vertex x_k with $l < k \leq i$-1 and $x_k y \in A(G)$ with $y \notin \sigma_i$. But then we have a contradiction with the condition that σ is supposed to respect, on the triple (x_k, x_i, y).

We have a similar result for BFS.

Theorem

Let G be a directed graph, σ a total ordering of the vertices is BFS ordering iff for every triple $a <_\sigma b <_\sigma c$ with $ac \in A(G)$ and $ab \notin A(G)$, it exists necessarily some vertex d with $d <_\sigma a$ such that $db \in A(G)$.

Consequences

Similar proof as for the above theorem.
Therefore we can also define directed LDFS and LBFS by analogy with the undirected case.

Application to Tarjan's strongly connected components algorithm

The following lemma captures the recursivity of DFS.

The Factor lemma J. Dusart 2014

Let σ be a DFS-ordering of a directed graph G. Let μ be a factor of σ, then μ is a legitimate DFS-ordering of the induced subgraph $G(\mu)$.

Proof

Let us consider a triple of vertices (a, b, c) in $G(\mu)$ such that : $ac \in A(G)$ and $ab \notin A(G)$. Using the DFS 4-points conditions it exists necessarily some vertex d between a and b such that $db \in A(G)$. Since d is between a and b in σ, and μ a factor, necessarily $d \in G(\mu)$.

Algorithm 1: The Tarjan's Strongly Connected Components

DFS(G**)**;

Data: A directed graph G

Result: a DFS-ordering of the vertices σ and the lists of strongly connected components of G

$i \leftarrow 1$;

$Result \leftarrow \emptyset$;

foreach $x \in V(G)$ **do**

    $Closed(x) \leftarrow False$; $Stack(x) = False$

end

foreach $x \in V(G)$ **do**

    if $Closed(x) = False$ **then**

        $Explore(G, x)$

    end

end

Explore(G, x);
$Push(x, Result)$; $Stack(x) = True$; $Closed(x) \leftarrow True$;
$\sigma(i) \leftarrow x$; $root(x) \leftarrow i$; $i \leftarrow i + 1$;
foreach $xy \in A(G)$ **do**
    if $Closed(y) = False$ **then**
        $Explore(G, y)$; $root(x) \leftarrow min\{root(x), root(y)\}$;
    end
    else
        if $Stack(y) = True$ **then**
            $root(x) \leftarrow min\{root(x), root(y)\}$
        end
    end
end
if $root(x) = \sigma^{-1}(x)$ **then**
    Pop Result until x included, print these vertices as a list and update their value in the array Stack to false ;
end

In the above algorithm : Result is a stack, Stack is a boolean array describing if a vertex belongs to Result. σ is the DFS-ordering yielded by this DFS search.

Definition 1

For an ordering σ of the vertices of G, a **flyer** is $xy \in A(G)$ such that there exists $z \in V(G)$ with $x <_\sigma z <_\sigma y$.

Definition 2

A vertex x is called a **root** if during the execution of the algorithm, when the work is finished at x (i.e. at the end of $Explore(G, x)$), $root(x) = \sigma^{-1}(x)$.

Theorem

Tarjan's algorithm applied on a directed graph G computes its strongly connected components.

The proof

It goes by induction on the size of G. If G is reduced to a vertex z, the stack *Result* contains z which is by default a strongly connected component.
The proof will rely on σ the DFS-ordering generated by the execution of Tarjan's algorithm on a graph G.
Let z be the last root vertex in σ. It always exists at least one such vertex, since the first vertex of the DFS is necessarily a root. Let us denote by $D(z)$ the set of descendants of z after z in σ.

- **Claim 1** $G(z \cup D(z))$ is strongly connected.
  It suffices to prove that for every vertex $y \in D(z)$ there exists a path form y to z.
  Since y is not a root it admits a successor t previously considered in σ. If $z <_\sigma t$ we apply the same reasoning on t. Else $t <_\sigma z$ implies that z cannot be root. So we construct a path from y that must necessarily end in z.

Claim 2 $G(z \cup D(z))$ is maximal because it cannot be extended in σ in both directions.

- **Claim 3** $z \cup D(z)$ is a factor of σ.
  Else let us consider the closest to z, vertex b such that : $z <_\sigma b <_\sigma t$, with $t \in D(z)$, $b \notin D(z)$. Let us consider the smallest flyer across b. Such an arch is an arc ac with $a, c \in D(z)$ and $a <_\sigma b <_\sigma c$. using the DFS 4 points condition it exists d in between a and b in σ, such that : $db \in A(G)$. d cannot belong to $D(z)$ else b would also belong to $D(z)$. But then b is not the closest to z, a contradiction.
  If z is the first element of σ then $z \cup D(z) = V(G)$ and G is strongly connected, and all vertices belong to the stack *Result*, and therefore Tarjan's algorithm finds the right solution in this case.
  Else we can apply the factor Lemma , since $z \cup D(z)$ is a factor of σ, by induction the algorithm works on $G(z \cup D(z))$.

Let σ' the restriction of σ to $V(G)$-$z \cup D(z)$. It suffices to prove that σ' is a legitimate DFS on this graph denoted by G'.
Consider a triple $(a, b, c) \in G'$ with $a <_\sigma b <_\sigma c$, $ac \in A(G)$ and $ab \notin A(G)$. Using the DFS 4 points condition on G, it exists d in between a and b in σ, such that : $db \in A(G)$. Let us consider the position of z in σ with respect to this triple. The only interesting case is :
$a <_\sigma z <_\sigma b$
if $d \in z \cup D(z)$ then $b \in z \cup D(z)$ which is not possible, therefore the 4 points condition is satisfied in G'.
So the proof terminates using induction on G'.

Maximal Cardinality Search : MCS

Data: a graph $G = (V, E)$ and a start vertex s

Result: an ordering σ of V

Assign the label 0 to all vertices

$label(s) \leftarrow 1$

for $i \leftarrow n$ **à** 1 **do**

    Pick an unumbered vertex v with largest label

    $\sigma(i) \leftarrow v$

    foreach *unnumbered vertex* w *adjacent to* v **do**

        $label(w) \leftarrow label(w) + 1$

    end

end

Maximal Neighbourhood Search (MNS)

MNS

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 $i \leftarrow 1$ **to** n **do**

    Pick an unumbered vertex v with a maximal under inclusion label

    $\sigma(i) \leftarrow v$

    foreach *unnumbered vertex* w *adjacent to* v **do**

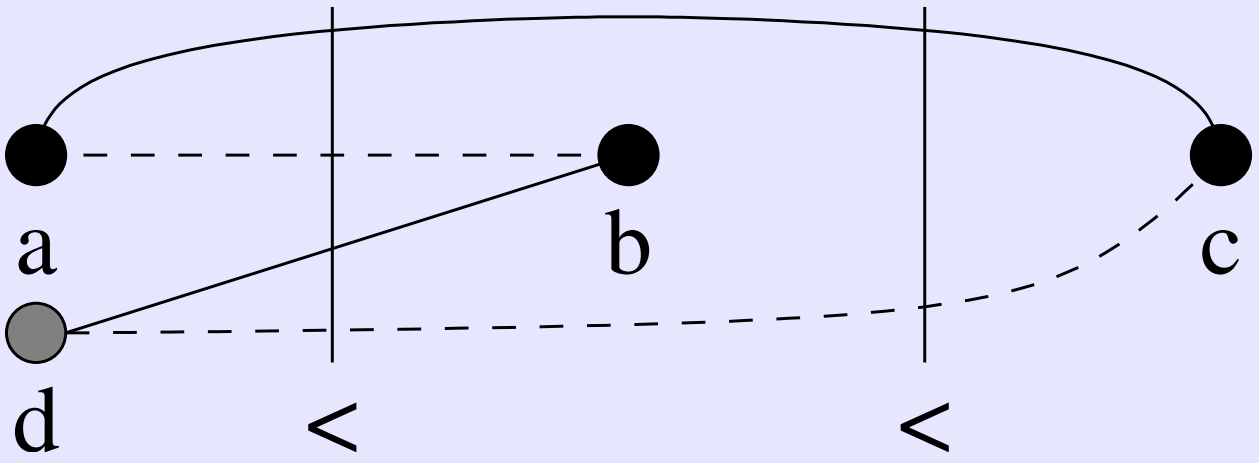
        $label(w) \leftarrow \{i\} \cup label(w)$

    end

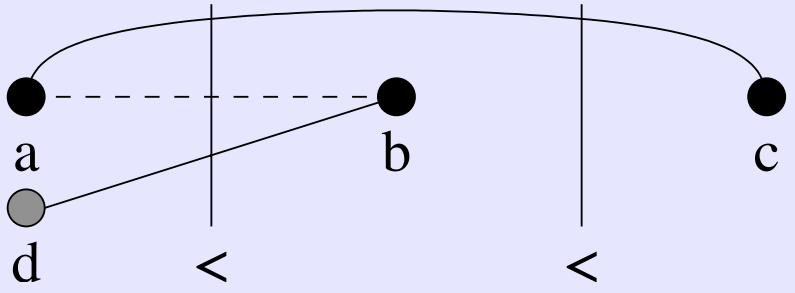
end

MNS property

Let σ be a total ordering $V(G)$, if $a < b < c$ and $ac \in E$ and $ab \notin E$, then it exsits d such that $d < b$, $db \in E$ and $dc \notin E$.



Generic search



MNS

MNS is a kind of completion of Generic search similar to BFS versus LBFS (resp. DFS versus LDFS). This explains why MNS was first named LexGen.

Theorem [Tarjan et Yannakakis, 1984]

G is a chordal graph iff every MNS computes a simplicial ordering.

Proof :

Let c be a non simplicial vertex (to the left). Thus it exists $a < b < c \in N(c)$ with $ab \notin E$. Using MNS property, it exsits $d < b$ with $db \in E$ and $dc \notin E$. Since G is chordal, necessarily $ad \notin E$.
Either $d < a$, considering the triple d, a, b, it exists $d' < a$ such that $d'a \in E$ and $d'b \notin E$. Furthermore $d'd \notin E$.
Or $a < d$, considering the triple a, d, c, it exists $d' < d$ such that $d'd \in E$ and $d'c \notin E$. Furthermore $ad' \notin E$.
In both cases a pattern is propagating to the left, a contradiction.

Corollary

G is a chordal graph iff every MCS, LBFS, LDFS computes a simplicial ordering.

Proof

Maximal for the cardinality, or maximal lexicographically are particular cases of maximality under inclusion.

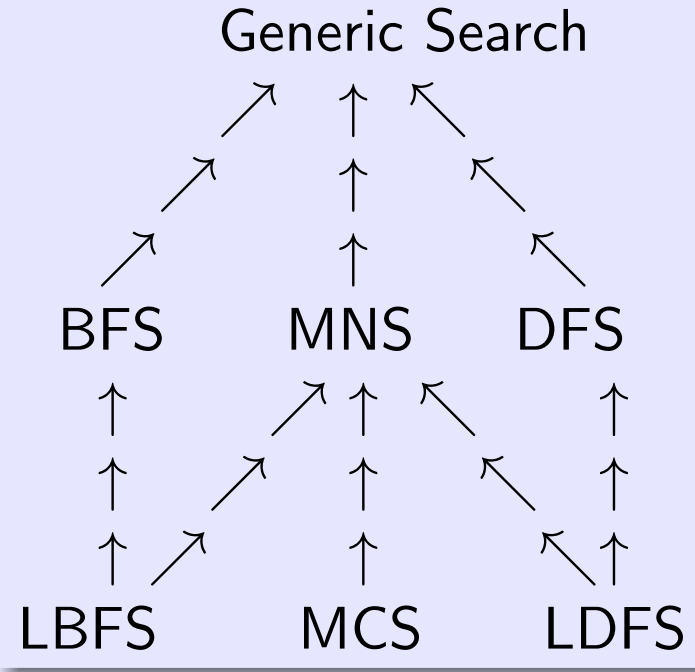
Implementation

MCS, LBFS provide linear time particular implementation sof MNS. But they are many others, less famous.
But in its full generality no linear time implementation is known.

Conclusions

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

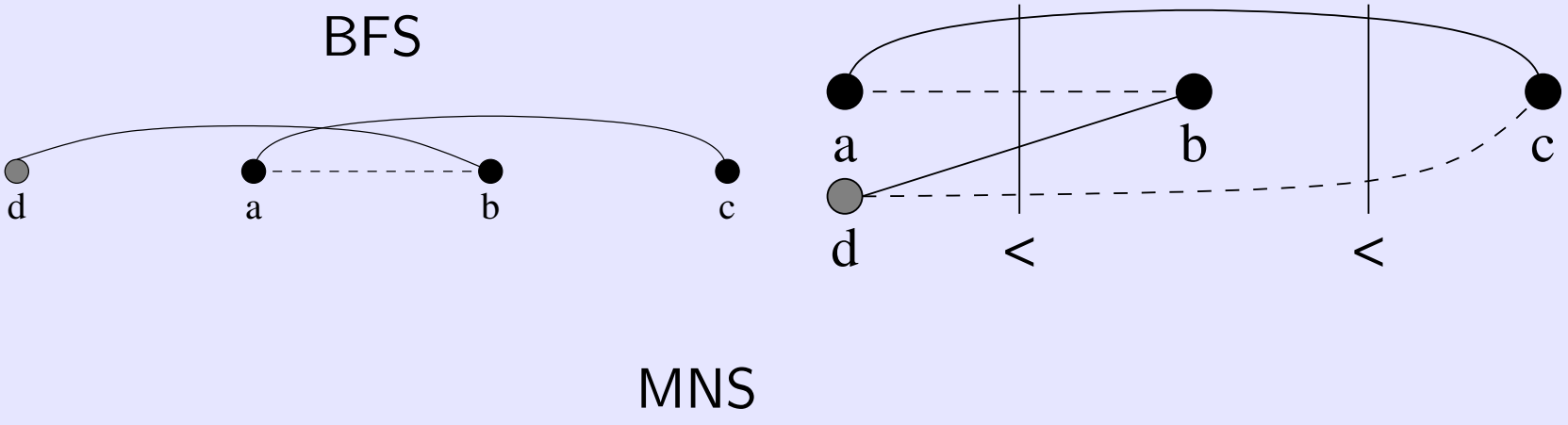
Strict inclusions



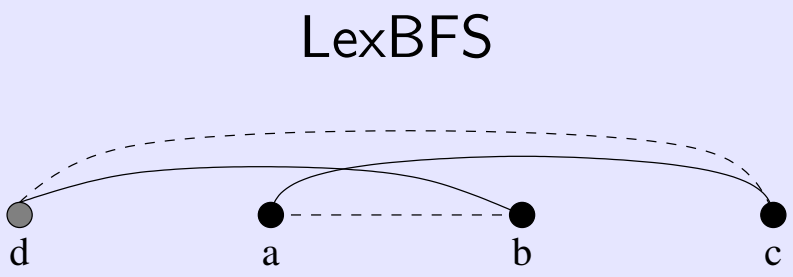
Exercices

1. Vérifier que les inclusions sont bien strictes.
2. Que peut-on dire d'un parcours à la fois BFS et MNS ?
   A-t-on BFS+MNS =LexBFS ?
3. Idem pour DFS et MNS
   A-t-on DFS + MNS = LexDFS ?

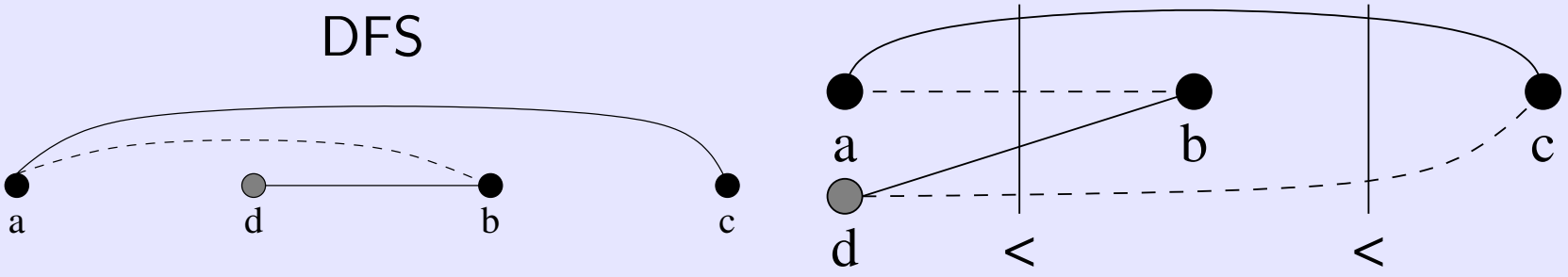
BFS + MNS



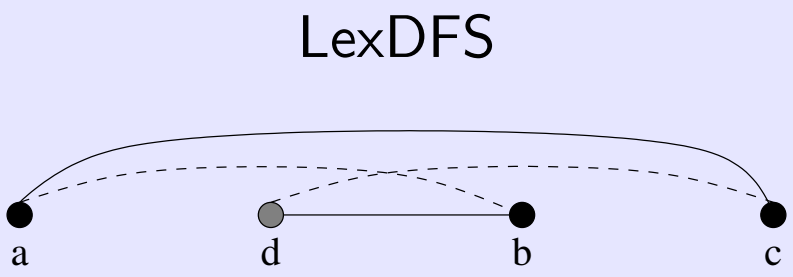
= LexBFS ?



DFS + MNS



= LexDFS ?



Layer ordering or distance level ordering

un ordre préservant les distances au sommet initial
$\neq$ BFS

Search classification

Search	Tie-break management
Generic search	none (random)
BFS	queue
DFS	stack
LBFS	Lexicographic maximal
LDFS	Lexicographic maximal
MNS	Maximal under inclusion
MCS	Maximal for the cardinality

Applications

- BFS to compute distances, diameter, centers
  Heuristics for diameter
- DFS planarity, strongly connected components, 2-SAT, ...
- LBFS, recognition of chordal graphs, interval graphs ...
  Recursive behavior on tie-break sets.
  Heuristics for one consecutiveness property
- LDFS, long paths, minimum path cover
  For cocomparability graphs LDFS computes layered ordering of the complement partial order.
  Heuristics for graph clustering, still many applications to be discovered.

Multisweep algorithms

In this talk a graph search is identified with the visiting ordering of the vertices it produces and therefore 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 the vertex x_0.

Definition of the + Rule

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

Why this Rule ?

The + Rule forces to keep the ordering of the previous sweep in case of tie-break

This + rule was introduced for LBFS by Ma and Simon in the 1980's when dealing with interval graph recognition.

We already have used this idea for diameter computations : a series of dependant BFS.
Many conjectures on this subject (about the result of the iteration). Hard to handle.