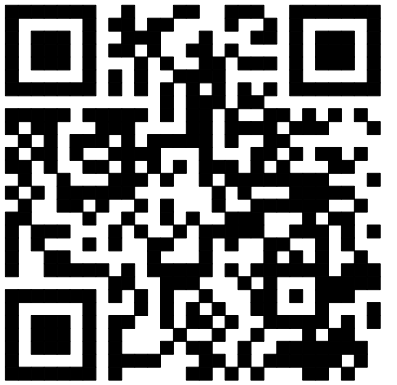
Karlsruhe Institute of Technology

Algorithmic Graph Theory

Problem Class 3 | 21 May 2025

Laura Merker, Samuel Schneider

Problems



(0) Download the following two papers:
`https://epubs.siam.org/doi/epdf/10.1137/060664690`
`https://www.sciencedirect.com/science/article/pii/0166218X81900135`

(1) Both papers define cographs. Is it the same graph class?
If yes, how do you know?
If no, is one a strict subclass of the other? Find an example.

(2) **Def:** True (false) twin of vertex v: vertex w with $N[v] = N[w]$ $(N(v) = N(w))$
Let $\mathcal{G}$ be the set of graphs obtained from a single vertex by adding true or false twins.
Can the graphs in $\mathcal{G}$ be efficiently recognized?
I.e., given a graph G, can it be efficiently decided whether or not $G \in \mathcal{G}$?

(3) Is the LexBFS algorithm of Bretscher, Corneil, Habib, Paul the same as in the lecture?
What does *same* mean here? Where do they differ, what are similarities?

(4) Is there a connection between LexBFS, LexBFS$^-$, G, and $\overline{G}$?
If yes, can this be exploited to simplify the algorithm? If no, where is the problem?

(5) Characterize the graphs with twinwidth 0. Find a suitable reference.
https://scholar.google.com

Definitions of cographs

Bretscher, Corneil, Habib, Paul:

1. Introduction. We introduce a new[1] linear time algorithm to recognize *cographs*, namely, graphs with no induced path on four vertices (P_4).

Corneil, Lerchs, Stewart Burlingham:

A *complement reducible graph* (also called a *cograph*) is defined recursively as follows:

(i) A graph on a single vertex is a complement reducible graph.

(ii) If $G_1, G_2, \ldots, G_k$ are complement reducible graphs, then so is their union $G_1 \cup G_2 \cup \cdots \cup G_k$.

(iii) If G is a complement reducible graph, then so is its complement $\bar{G}$.

?

Definitions of cographs

Bretscher, Corneil, Habib, Paul:

> **1. Introduction.** We introduce a new[1] linear time algorithm to recognize *cographs*, namely, graphs with no induced path on four vertices (P_4).

Corneil, Lerchs, Stewart Burlingham:

> **Theorem 2.** ***Given a graph G, the following statements are equivalent:***
>
> **(1)** ***G is a cograph.***
>
> **(4)** ***G does not contain P_4 as a subgraph.***

Isn't that false?

K_4 is a cograph but does contain P_4 as subgraph!

So, not the same graph class?

Definitions of cographs

Bretscher, Corneil, Habib, Paul:

> **1. Introduction.** We introduce a new[1] linear time algorithm to recognize *cographs*, namely, graphs with no induced path on four vertices (P_4).

Corneil, Lerchs, Stewart Burlingham:

> **Theorem 2.** ***Given a graph G, the following statements are equivalent:***
>
> **(1)** ***G is a cograph.***
>
> **(4)** ***G does not contain P_4 as a subgraph.***

Cograph means the same in both papers!

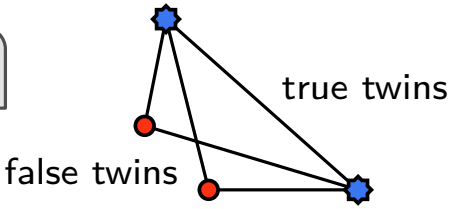
> **2. Terminology**
>
> **The terminology used in this paper is compatible with [3]. We assume that all graphs are finite and that unless stated otherwise the term subgraph always refers to the notion of induced subgraph.**

Twins and cographs

Let G be a graph and $u, v \in V(G)$ be distinct vertices.

Definition: u and v are **true twins** if $N[u] = N[v]$.

$N(v) \cup \{v\}$



Definition: u and v are **false twins** if $N(u) = N(v)$.

Let $\mathcal{G}$ be the set of graphs obtained from a single vertex by adding true or false twins.
Can the graphs in $\mathcal{G}$ be efficiently recognized?
I.e., given a graph G, can it be efficiently decided whether or not $G \in \mathcal{G}$?

Goal: $\mathcal{G}$ is exactly the class of *cographs*

Theorem 2. *Given a graph G, the following statements are equivalent:*

(1) *G is a cograph.*
(2) *Any nontrivial subgraph of G has at least one pair of siblings.*
(3) *Any subgraph of G has the CK-property.*
(4) *G does not contain P_4 as a subgraph.*
(5) *The complement of any nontrivial connected subgraph of G is disconnected.*
(6) *G is an HD-graph.*
(7) *Every connected subgraph of G has diameter ≤2.*
(8) *G is the comparability graph of a multitree.*

Twins and cographs

Let $\mathcal{G}$ be the set of graphs obtained from a single vertex by adding true or false twins.
Can the graphs in $\mathcal{G}$ be efficiently recognized?
I.e., given a graph G, can it be efficiently decided whether or not $G \in \mathcal{G}$?

Goal: $\mathcal{G}$ is exactly the class of *cographs* (with at least two vertices)

Theorem 2: G is a cograph $\Leftrightarrow$ every **induced** subgraph of G has a pair of twins.

Claim: $\mathcal{G}$ is closed under taking induced subgraphs

- Let $v \in V(G)$ be the vertex that was added last and call induction on $G - v$
- By induction all induced subgraphs G_A of $G - v$ in $\mathcal{G}$
- W. l. o. g twin of v in $A \Rightarrow G_{A+v} \in \mathcal{G}$

Twins and cographs

Let $\mathcal{G}$ be the set of graphs obtained from a single vertex by adding true or false twins.
Can the graphs in $\mathcal{G}$ be efficiently recognized?
I.e., given a graph G, can it be efficiently decided whether or not $G \in \mathcal{G}$?

Goal: $\mathcal{G}$ is exactly the class of *cographs* (with at least two vertices)

Theorem 2: G is a cograph $\Leftrightarrow$ every **induced** subgraph of G has a pair of twins.

Claim: $\mathcal{G}$ is closed under taking induced subgraphs

- Every $G \in \mathcal{G}$ has pair of twins by construction $\Rightarrow$ cograph by **Theorem 2** and **Claim**
- We show the other direction by induction on the number of vertices:
  - Let G be a cograph and $u, v \in V(G)$ be two twins.
  - $G - u$ is a cograph $\Rightarrow$ by induction $G - u \in \mathcal{G} \Rightarrow G \in \mathcal{G}$

> MAIN RESULT. *There exists a simple, linear time, LexBFS based recognition algorithm for the family of cographs that returns a certificate in the form of a cotree or an induced P_4.*

LexBFS

Is the LexBFS algorithm of Bretscher, Corneil, Habib, Paul the same as in the lecture?
What does *same* mean here? Where do they differ, what are similarities?

Algorithm LexBFS(G, τ)
Input: *A graph* $G = (V, E)$ *and an initial ordering* τ *of the vertices.*
Output: *An ordering* σ *of the vertices of* G.

1. $L \leftarrow (V)$
2. $i \leftarrow 1$
3. **while** $\exists P_i \neq \emptyset$ in $L = (P_1, \ldots, P_k)$ **do**
4. Let P_l be the leftmost nonempty cell
5. Remove the first vertex x (smallest with respect to τ) from P_l
6. $\sigma(x) \leftarrow i$
7. $i \leftarrow i + 1$
8. **for** each cell $P_j, j \geq l$ **do**
9. Let $P' = \{v | v \in N(x) \cap P_j\}$;
10. **if** P' is nonempty and $P' \neq P_j$, **then**
11. Remove P' from P_j
12. Insert P' to the left of P_j in L
13. **end for**
14. **end while**
15. **return** (σ)

end LexBFS

(Good) papers have a text description of the algorithm that explains the key ideas.

?!

LexBFS

Is the LexBFS algorithm of Bretscher, Corneil, Habib, Paul the same as in the lecture?
What does *same* mean here? Where do they differ, what are similarities?

3.1. LexBFS. For a graph $G = (V, E)$, a *LexBFS* of G, denoted *LexBFS(G)*, is a breadth first search in which preference is given to vertices whose neighbors have been visited *earliest* in the search [25].

sounds familiar → continue with assumption that it is the same algorithm

indeeed, same algorithm as in the lecture

The term *lexicographic* stems from the original *labeling* paradigm in which vertices pass a label to each unnumbered neighbor. The next vertex is chosen such that it has the lexicographically largest label. Another conceptualization of LexBFS involves pivots and partitioning [18]. The partitioning paradigm will prove more useful for illustrating the variant LexBFS$^-$.

But we should continue reading and learn more!

LexBFS$^-$

Is there a connection between LexBFS, LexBFS$^-$, G, and $\overline{G}$?
If yes, can this be exploited to simplify the algorithm? If no, where is the problem?

A *LexBFS minus*, or LexBFS$^-$, of a graph G is a modified version of a LexBFS of $\overline{G}$.

There seems to be some connection.

LexBFS$^-$

Is there a connection between LexBFS, LexBFS$^-$, G, and $\overline{G}$?
If yes, can this be exploited to simplify the algorithm? If no, where is the problem?

A *LexBFS minus*, or LexBFS$^-$, of a graph What is a pivot? ersion of a LexBFS of $\overline{G}$. The modification arises in how the pivots are selected. The idea is to have an initial LexBFS ordering τ of G. LexBFS$^-(G, \tau)$ does a LexBFS of $\overline{G}$ *and* selects the next pivot to be the vertex in the leftmost cell that was numbered earliest in τ.

τ is the result of a LexBFS, something we know!

maybe not so clear

Never stop at a sentence you don't get!
The explanation is often in the next sentence.

LexBFS$^-$

Is there a connection between LexBFS, LexBFS$^-$, G, and $\overline{G}$?
If yes, can this be exploited to simplify the algorithm? If no, where is the problem?

A *LexBFS minus*, or LexBFS$^-$, of a graph G is a modified version of a LexBFS of $\overline{G}$. The modification arises in how the pivots are selected. The idea is to have an initial LexBFS ordering τ of G. LexBFS$^-(G, \tau)$ does a LexBFS of $\overline{G}$ *and* selects the next pivot to be the vertex in the leftmost cell that was numbered earliest in τ. To implement this "*tie-breaking mechanism*," simply pass τ as the initial ordering of the vertex set. Notice that the leftmost vertex in the leftmost cell *is* the vertex that was numbered earliest in τ.

LexBFS$^-$ is a LexBFS on the complement with τ as tie-breaker.

Never stop at a sentence you don't get!
The explanation is often in the next sentence.

Now, you can verify what you understood by checking how the pseudocode is changed.

LexBFS$^-$

Is there a connection between LexBFS, LexBFS$^-$, G, and $\overline{G}$?
If yes, can this be exploited to simplify the algorithm? If no, where is the problem?

LexBFS$^-$ is a LexBFS on the complement with τ as tie-breaker.

So, can we simplify the algorithm?
$\rightarrow$ Omit LexBFS$^-$ by a LexBFS on the complement (respecting the tie-breaker)?

✓ correctness
X runtime $O(n^2)$ for computing the complement instead of $O(m)$

Twinwidth

Characterize the graphs with twinwidth 0. Find a suitable reference.

Bonnet, Kim, Thomassé, Watrigant: *Twin-width I: Tractable FO Model Checking*
https://dl.acm.org/doi/pdf/10.1145/3486655

Graphs of twinwidth 0 are exactly the cographs.

Learning: Papers can be helpful, even if you do not understand the abstract.

Try to skip everything you do not understand and find the part you are familiar with, e.g., cographs or twinwidth 0 (after reading the definition).