Local certification of forbidden subgraphs

Nicolas Bousquet, Linda Cook, Laurent Feuilloley, Théo Pierron, Sébastien Zeitoun

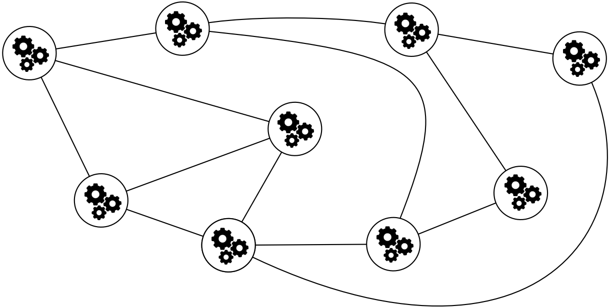
February 26, 2025

Local certification

Local certification

Context: distributed computing

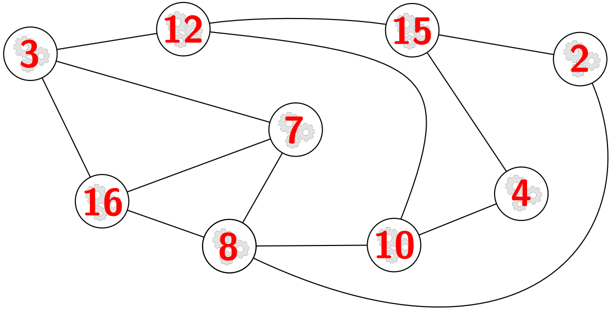
Model: graph, $\begin{cases} \text{vertices} = \text{computation units} \\ \text{edges} = \text{communication channels} \end{cases}$



Local certification

Context: distributed computing

Model: graph, $\begin{cases} \text{vertices} = \text{computation units} & \longrightarrow \text{have unique identifiers in } \{1, \ldots, n^c\} \\ \text{edges} = \text{communication channels} \end{cases}$

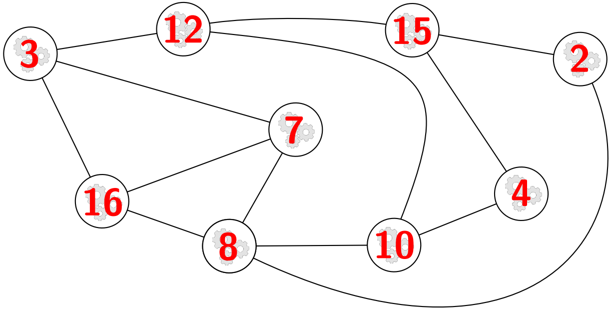


Local certification

Context: distributed computing

Model: graph, $\begin{cases} \text{vertices} = \text{computation units} \longrightarrow \text{have unique identifiers in } \{1, \ldots, n^c\} \\ \text{edges} = \text{communication channels} \end{cases}$

Goal: verify locally a graph property $\mathcal{P}$, thanks to certificates

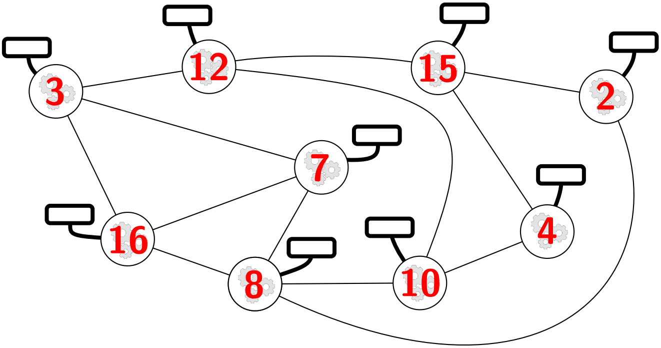


Local certification

Context: distributed computing

Model: graph, $\begin{cases} \text{vertices} = \text{computation units} \longrightarrow \text{have unique identifiers in } \{1, \ldots, n^c\} \\ \text{edges} = \text{communication channels} \end{cases}$

Goal: verify locally a graph property $\mathcal{P}$, thanks to certificates

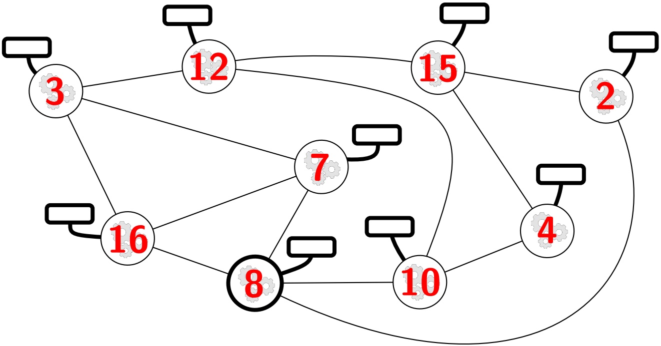


Local certification

Context: distributed computing

Model: graph, { vertices = computation units $\longrightarrow$ have unique identifiers in $\{1, \ldots, n^c\}$
edges = communication channels }

Goal: verify locally a graph property $\mathcal{P}$, thanks to certificates

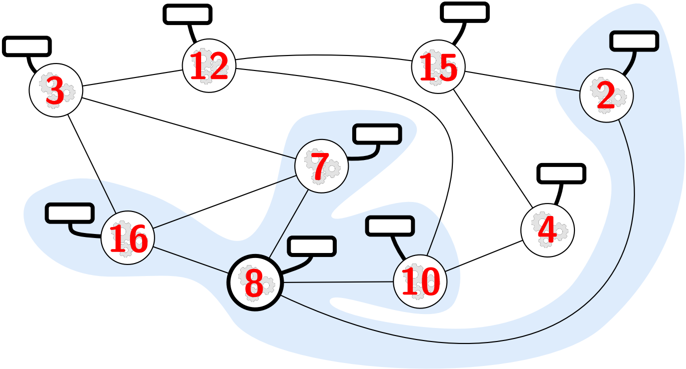


Local certification

Context: distributed computing

Model: graph, $\begin{cases} \text{vertices} = \text{computation units} \longrightarrow \text{have unique identifiers in } \{1, \dots, n^c\} \\ \text{edges} = \text{communication channels} \end{cases}$

Goal: verify locally a graph property $\mathcal{P}$, thanks to certificates

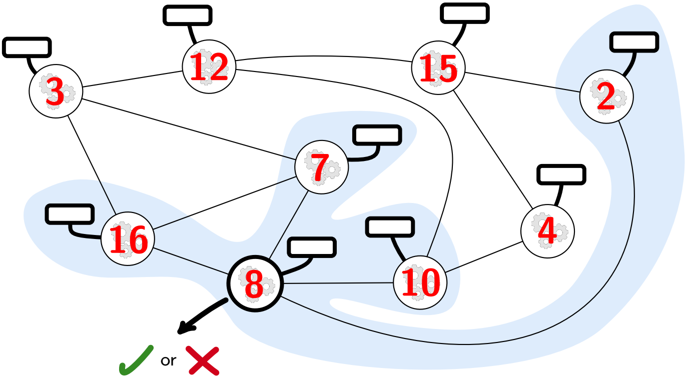


Local certification

Context: distributed computing

Model: graph, $\begin{cases} \text{vertices} = \text{computation units} \longrightarrow \text{have unique identifiers in } \{1, \ldots, n^c\} \\ \text{edges} = \text{communication channels} \end{cases}$

Goal: verify locally a graph property $\mathcal{P}$, thanks to certificates

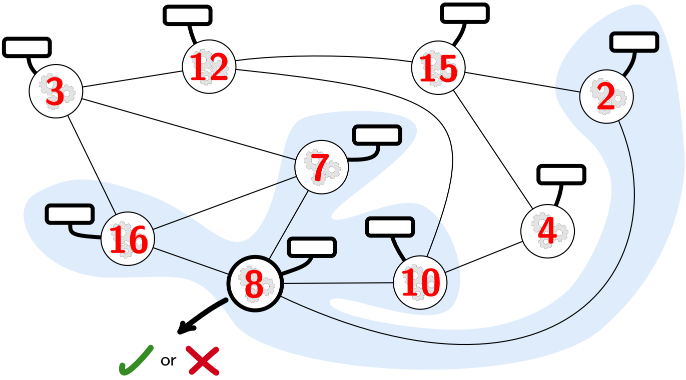


Local certification

Context: distributed computing

Model: graph, $\begin{cases} \text{vertices} = \text{computation units} \longrightarrow \text{have unique identifiers in } \{1, \ldots, n^c\} \\ \text{edges} = \text{communication channels} \end{cases}$

Goal: verify locally a graph property $\mathcal{P}$, thanks to certificates



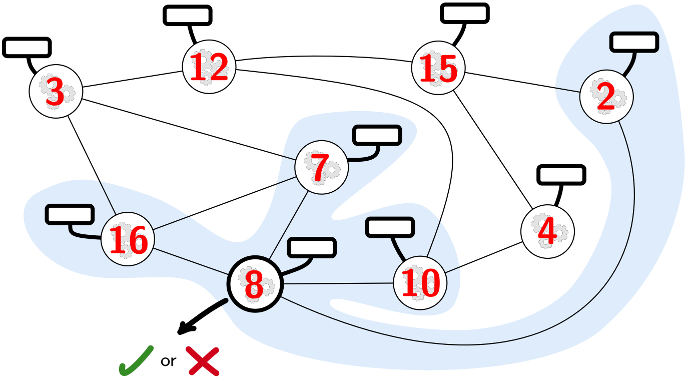
Graph (globally) accepted $\iff$ all the vertices accept (consensus)

Local certification

Context: distributed computing

Model: graph, $\begin{cases} \text{vertices} = \text{computation units} & \longrightarrow \text{have unique identifiers in } \{1, \dots, n^c\} \\ \text{edges} = \text{communication channels} \end{cases}$

Goal: verify locally a graph property $\mathcal{P}$, thanks to certificates

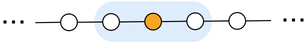


Graph (globally) accepted $\iff$ all the vertices accept (consensus)

G satisfies $\mathcal{P} \iff$ there exists an assignment of the certificates such that G is accepted

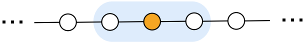
Example 1: how to certify that a graph is a path ?

Example 1: how to certify that a graph is a path ?



⟶ Path ? Cycle ?

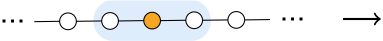
Example 1: how to certify that a graph is a path ?



$\longrightarrow$ Path ? Cycle ?

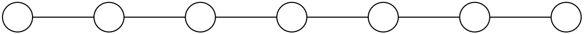
Certificate = distance to a fixed endpoint.

Example 1: how to certify that a graph is a path ?

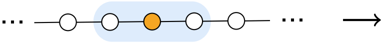


Path ? Cycle ?

Certificate = distance to a fixed endpoint.

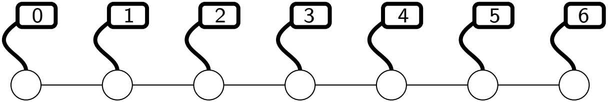


Example 1: how to certify that a graph is a path ?

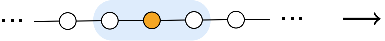


Path ? Cycle ?

Certificate = distance to a fixed endpoint.

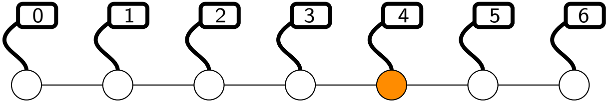


Example 1: how to certify that a graph is a path ?

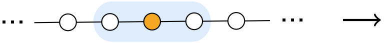


Path ? Cycle ?

Certificate = distance to a fixed endpoint.

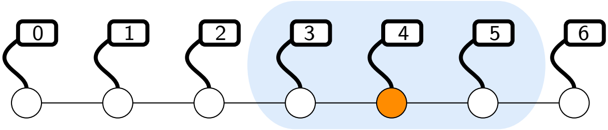


Example 1: how to certify that a graph is a path ?

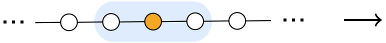


Path ? Cycle ?

Certificate = distance to a fixed endpoint.

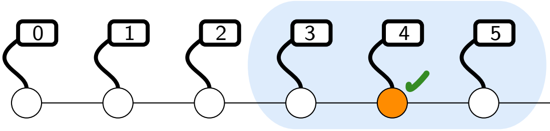


Example 1: how to certify that a graph is a path ?

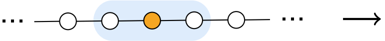


Path ? Cycle ?

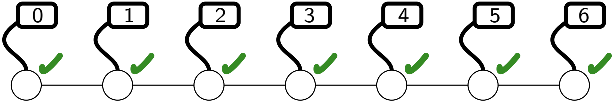
Certificate = distance to a fixed endpoint.



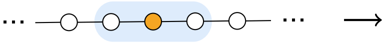
Example 1: how to certify that a graph is a path ?



Certificate = distance to a fixed endpoint.

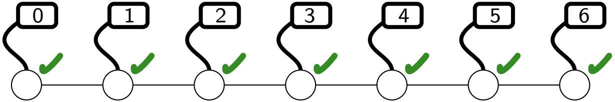


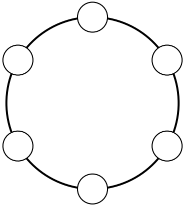
Example 1: how to certify that a graph is a path ?



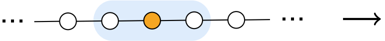
Path ? Cycle ?

Certificate = distance to a fixed endpoint.

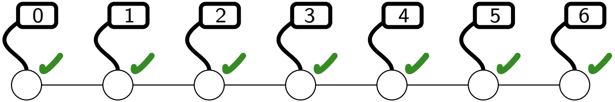


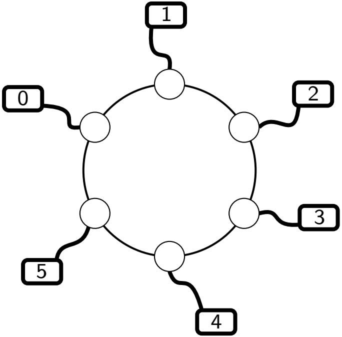


Example 1: how to certify that a graph is a path ?

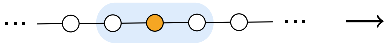


Certificate = distance to a fixed endpoint.

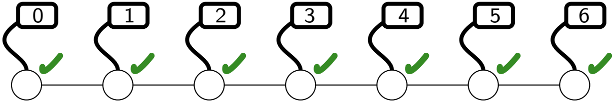


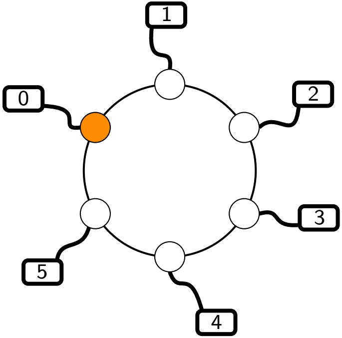


Example 1: how to certify that a graph is a path ?

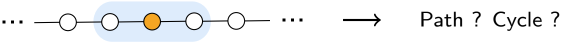


Certificate = distance to a fixed endpoint.

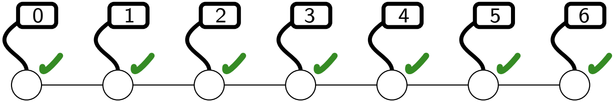


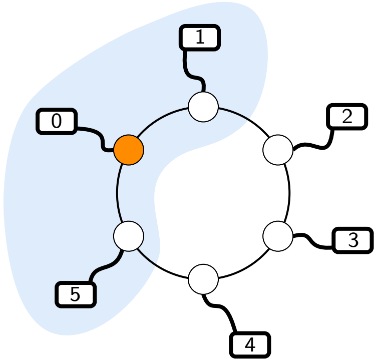


Example 1: how to certify that a graph is a path ?

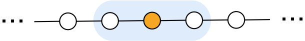


Certificate = distance to a fixed endpoint.



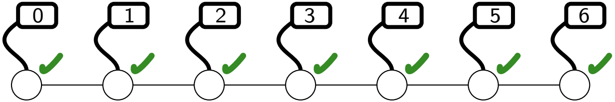


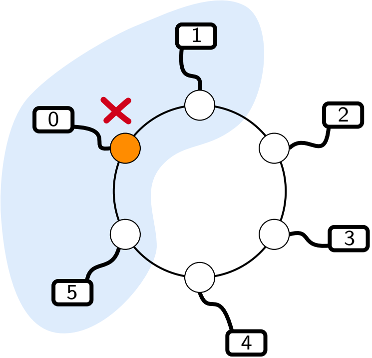
Example 1: how to certify that a graph is a path ?



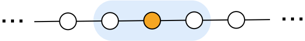
$\longrightarrow$ Path ? Cycle ?

Certificate = distance to a fixed endpoint.



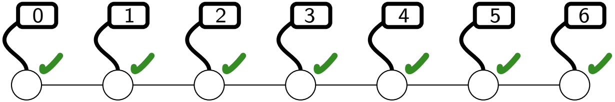


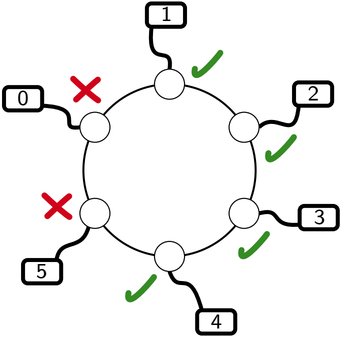
Example 1: how to certify that a graph is a path ?



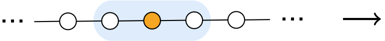
⟶ Path ? Cycle ?

Certificate = distance to a fixed endpoint.

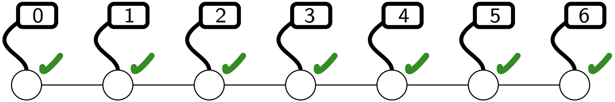


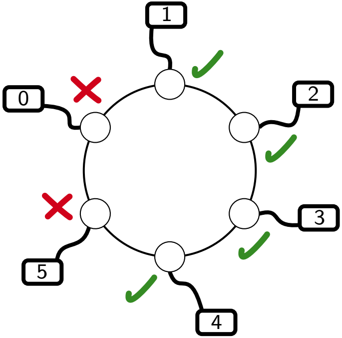


Example 1: how to certify that a graph is a path ?



Certificate = distance to a fixed endpoint.





Size of the certificates: $\lceil \log n \rceil$

Example 2: how to certify that a graph has a vertex of degree 3?

Example 2: how to certify that a graph has a vertex of degree 3?

Idea: code a rooted spanning-tree in the certificate.

Example 2: how to certify that a graph has a vertex of degree 3?

Idea: code a rooted spanning-tree in the certificate.

Root = some vertex of degree 3.

Example 2: how to certify that a graph has a vertex of degree 3?

Idea: code a rooted spanning-tree in the certificate.

Root = some vertex of degree 3.

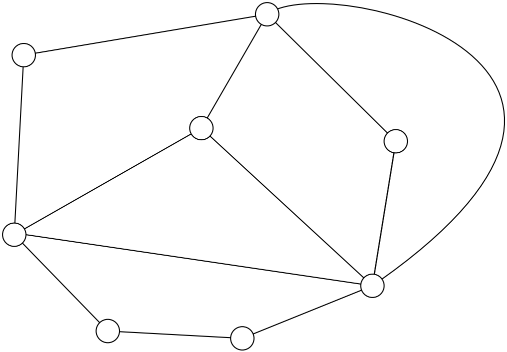
Verification:
- all vertices check the correctness of the spanning-tree
- the root checks it has degree 3

Example 2: how to certify that a graph has a vertex of degree 3?

Idea: code a rooted spanning-tree in the certificate.

Root = some vertex of degree 3.

Verification:
- all vertices check the correctness of the spanning-tree
- the root checks it has degree 3

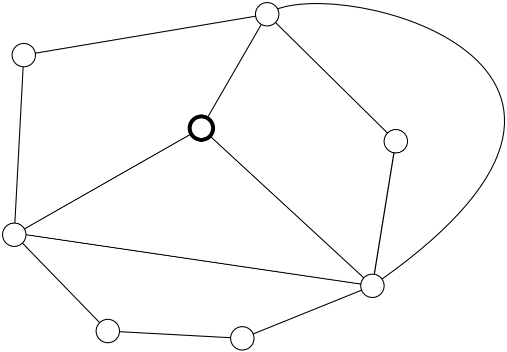


Example 2: how to certify that a graph has a vertex of degree 3?

Idea: code a rooted spanning-tree in the certificate.

Root = some vertex of degree 3.

Verification:
- all vertices check the correctness of the spanning-tree
- the root checks it has degree 3

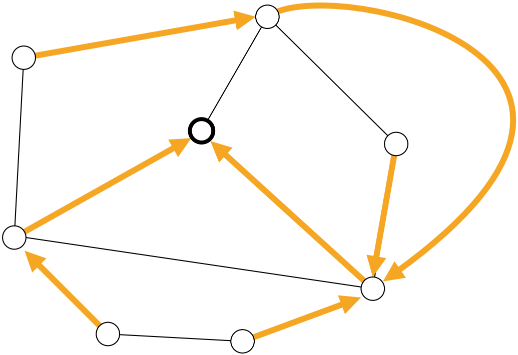


Example 2: how to certify that a graph has a vertex of degree 3?

Idea: code a rooted spanning-tree in the certificate.

Root = some vertex of degree 3.

Verification:
- all vertices check the correctness of the spanning-tree
- the root checks it has degree 3

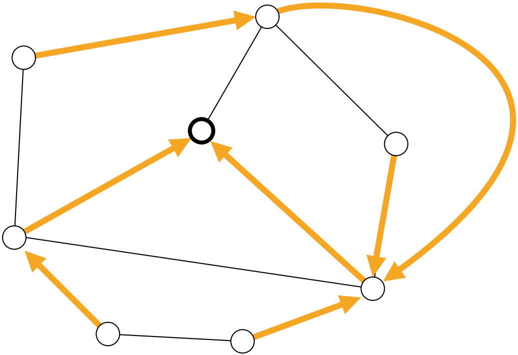


Example 2: how to certify that a graph has a vertex of degree 3?

Idea: code a rooted spanning-tree in the certificate.

Root = some vertex of degree 3.

Verification:
- all vertices check the correctness of the spanning-tree
- the root checks it has degree 3



In the certificate of every vertex, write:

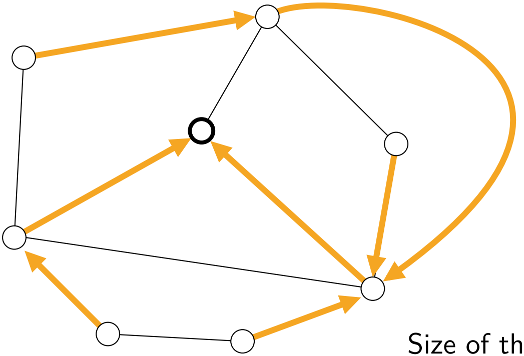
- the identifier of the root
- the identifier of its parent
- its distance to the root

Example 2: how to certify that a graph has a vertex of degree 3?

Idea: code a rooted spanning-tree in the certificate.

Root = some vertex of degree 3.

Verification:
- all vertices check the correctness of the spanning-tree
- the root checks it has degree 3



In the certificate of every vertex, write:

- the identifier of the root
- the identifier of its parent
- its distance to the root

Size of the certificates : $O(\log n)$

What is the minimum size of the certificates ?

What is the minimum size of the certificates ?

Usual parameter: n (number of vertices in the graph)

What is the minimum size of the certificates ?

Usual parameter: n (number of vertices in the graph)

Theorem

Any property can be certified with certificates of size $O(n^2)$.

↪ idea: write the full graph in the certificate of each vertex

What is the minimum size of the certificates ?

Usual parameter: n (number of vertices in the graph)

Theorem

Any property can be certified with certificates of size $O(n^2)$.

↪ idea: write the full graph in the certificate of each vertex

Typical size of certificates :

$\tilde{\Theta}(n^2)$	$\Theta(\log n)$

What is the minimum size of the certificates ?

Usual parameter: n (number of vertices in the graph)

Theorem

Any property can be certified with certificates of size $O(n^2)$.

↪ idea: write the full graph in the certificate of each vertex

Typical size of certificates :

$\tilde{\Theta}(n^2)$	$\Theta(\log n)$
▪ Non-3-colorability ▪ Non-trivial automorphism	

What is the minimum size of the certificates ?

Usual parameter: n (number of vertices in the graph)

Theorem

Any property can be certified with certificates of size $O(n^2)$.

↪ idea: write the full graph in the certificate of each vertex

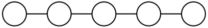
Typical size of certificates :

$\tilde{\Theta}(n^2)$	$\Theta(\log n)$
▪ Non-3-colorability	▪ Paths
▪ Non-trivial automorphism	▪ Trees
	▪ Planar graphs

Example 3: how to certify that a graph contains H as induced subgraph ?

Example 3: how to certify that a graph contains H as induced subgraph ?

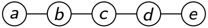
$H = P_5$



Example 3: how to certify that a graph contains H as induced subgraph ?

- Fix a name for each vertex of H.

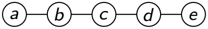
$H = P_5$



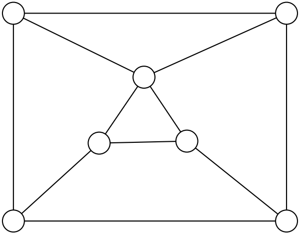
Example 3: how to certify that a graph contains H as induced subgraph ?

- Fix a name for each vertex of H.

$H = P_5$



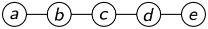
G



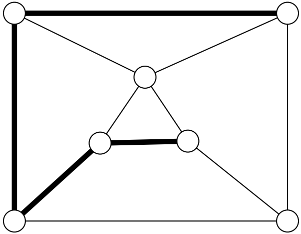
Example 3: how to certify that a graph contains H as induced subgraph ?

- Fix a name for each vertex of H.

$H = P_5$



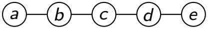
G

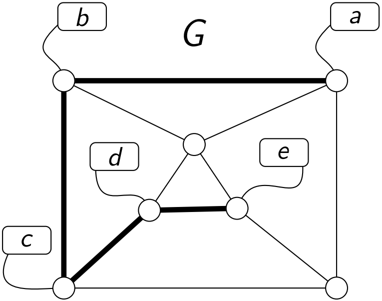


Example 3: how to certify that a graph contains H as induced subgraph ?

- Fix a name for each vertex of H.
- Tell every vertex of G to which vertex of H it corresponds (if any).

$H = P_5$

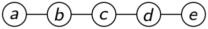


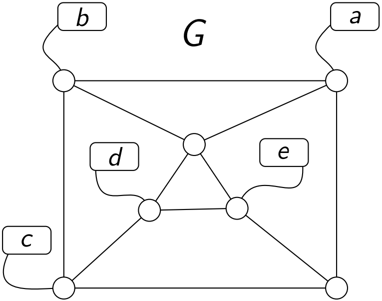


Example 3: how to certify that a graph contains H as induced subgraph ?

- Fix a name for each vertex of H.
- Tell every vertex of G to which vertex of H it corresponds (if any).
- For every vertex of H, certify that there is exactly one vertex of G which corresponds to it.

$H = P_5$

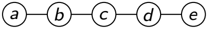


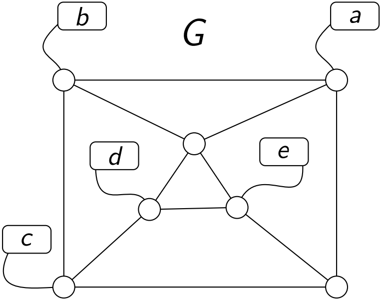


Example 3: how to certify that a graph contains H as induced subgraph ?

- Fix a name for each vertex of H.
- Tell every vertex of G to which vertex of H it corresponds (if any).
- For every vertex of H, certify that there is exactly one vertex of G which corresponds to it. $\longrightarrow$ use a spanning-tree !

$H = P_5$

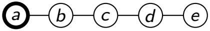


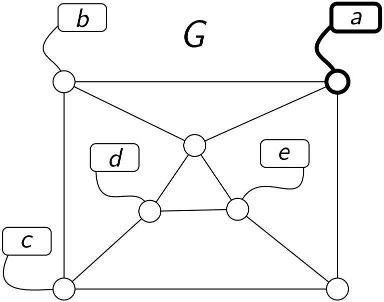


Example 3: how to certify that a graph contains H as induced subgraph ?

- Fix a name for each vertex of H.
- Tell every vertex of G to which vertex of H it corresponds (if any).
- For every vertex of H, certify that there is exactly one vertex of G which corresponds to it. $\longrightarrow$ use a spanning-tree !

$H = P_5$

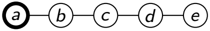


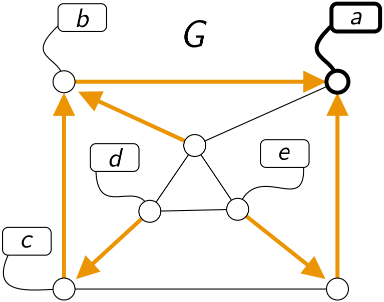


Example 3: how to certify that a graph contains H as induced subgraph ?

- Fix a name for each vertex of H.
- Tell every vertex of G to which vertex of H it corresponds (if any).
- For every vertex of H, certify that there is exactly one vertex of G which corresponds to it. $\longrightarrow$ use a spanning-tree !

$H = P_5$

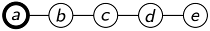


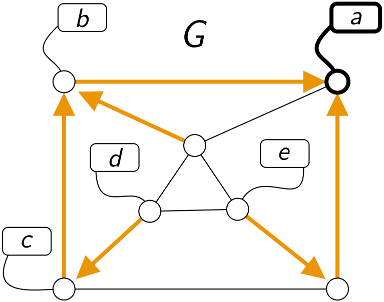


Example 3: how to certify that a graph contains H as induced subgraph ?

- Fix a name for each vertex of H.
- Tell every vertex of G to which vertex of H it corresponds (if any).
- For every vertex of H, certify that there is exactly one vertex of G which corresponds to it. $\longrightarrow$ use a spanning-tree !

$H = P_5$

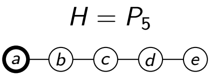




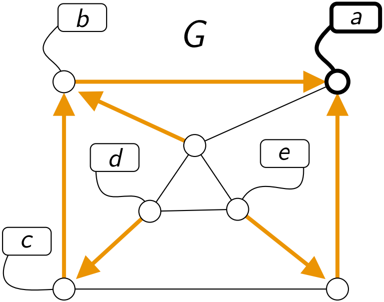
Size of the certificates :
$O(\log n)$

Example 3: how to certify that a graph contains H as induced subgraph ?

- Fix a name for each vertex of H.
- Tell every vertex of G to which vertex of H it corresponds (if any).
- For every vertex of H, certify that there is exactly one vertex of G which corresponds to it. $\longrightarrow$ use a spanning-tree !



Size of the certificates :
$O(\log n)$

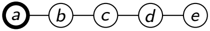


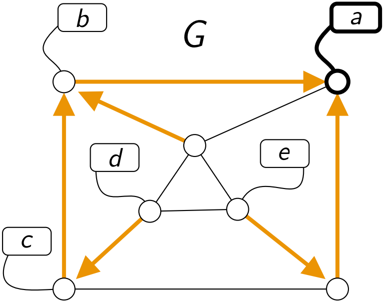
Question : what about the certification of the complementary property (H-freeness)?

Example 3: how to certify that a graph contains H as induced subgraph ?

- Fix a name for each vertex of H.
- Tell every vertex of G to which vertex of H it corresponds (if any).
- For every vertex of H, certify that there is exactly one vertex of G which corresponds to it. $\longrightarrow$ use a spanning-tree !

$H = P_5$

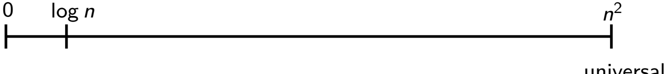




Size of the certificates :
$O(\log n)$

Question : what about the certification of the complementary property (H-freeness)?

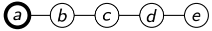
We will show : linear lower bounds and subquaratic upper bounds.

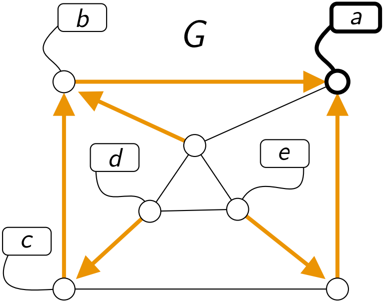


Example 3: how to certify that a graph contains H as induced subgraph ?

- Fix a name for each vertex of H.
- Tell every vertex of G to which vertex of H it corresponds (if any).
- For every vertex of H, certify that there is exactly one vertex of G which corresponds to it. $\longrightarrow$ use a spanning-tree !

$H = P_5$

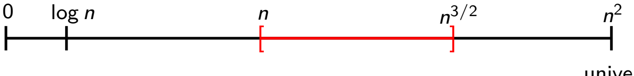




Size of the certificates :
$O(\log n)$

Question : what about the certification of the complementary property (H-freeness)?

We will show : linear lower bounds and subquaratic upper bounds.



Lower bounds

Certification of P_k-freeness: lower bound

Certification of P_k-freeness: lower bound

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\Omega(n)$ bits are necessary to certify that a graph is P_7-free.

Certification of P_k-freeness: lower bound

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\Omega(n)$ bits are necessary to certify that a graph is P_7-free.

H, H' bipartite graphs with n vertices on each side

Certification of P_k-freeness: lower bound

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\Omega(n)$ bits are necessary to certify that a graph is P_7-free.

H, H' bipartite graphs with n vertices on each side

↳ size $= \Theta(n^2)$

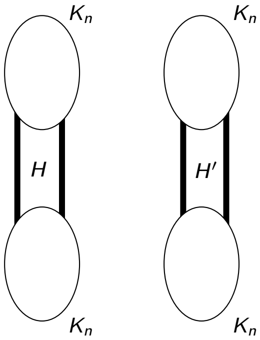
Certification of P_k-freeness: lower bound

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\Omega(n)$ bits are necessary to certify that a graph is P_7-free.

H, H' bipartite graphs with n vertices on each side

↳ size $= \Theta(n^2)$



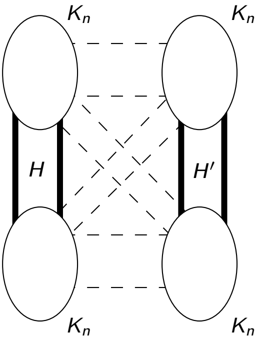
Certification of P_k-freeness: lower bound

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\Omega(n)$ bits are necessary to certify that a graph is P_7-free.

H, H' bipartite graphs with n vertices on each side

↳ size $= \Theta(n^2)$



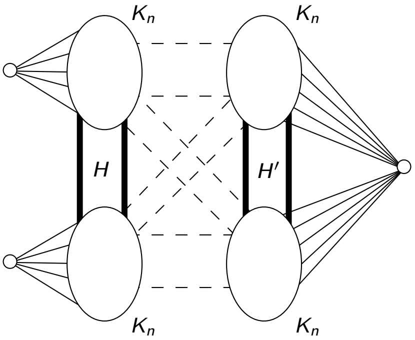
Certification of P_k-freeness: lower bound

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\Omega(n)$ bits are necessary to certify that a graph is P_7-free.

H, H' bipartite graphs with n vertices on each side

↳ size $= \Theta(n^2)$



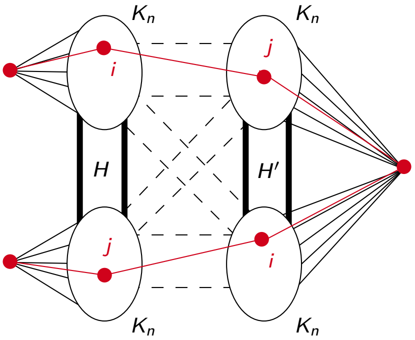
Certification of P_k-freeness: lower bound

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\Omega(n)$ bits are necessary to certify that a graph is P_7-free.

H, H' bipartite graphs with n vertices on each side

↳ size $= \Theta(n^2)$



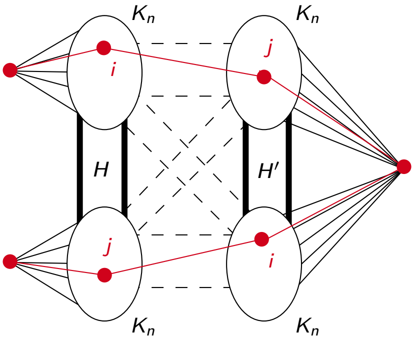
Certification of P_k-freeness: lower bound

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\Omega(n)$ bits are necessary to certify that a graph is P_7-free.

H, H' bipartite graphs with n vertices on each side

size $= \Theta(n^2)$



$\exists P_7 \iff H$ and H' have a common non-edge

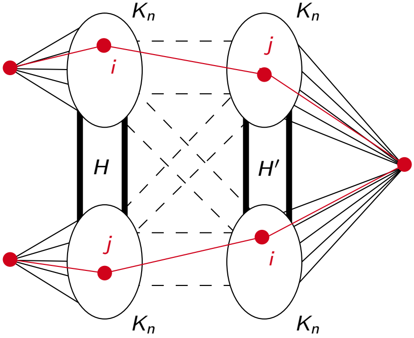
Certification of P_k-freeness: lower bound

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\Omega(n)$ bits are necessary to certify that a graph is P_7-free.

H, H' bipartite graphs with n vertices on each side

↳ size $= \Theta(n^2)$



$\exists P_7 \iff H$ and H' have a common non-edge

P_7-free $\iff \overline{H} \cap \overline{H'} = \emptyset$

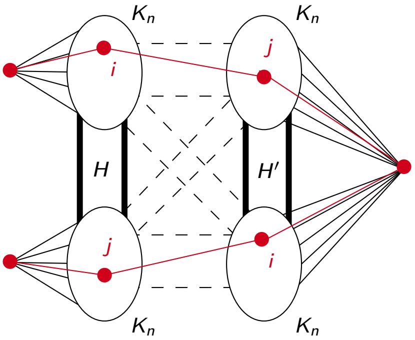
Certification of P_k-freeness: lower bound

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\Omega(n)$ bits are necessary to certify that a graph is P_7-free.

H, H' bipartite graphs with n vertices on each side

↪ size $= \Theta(n^2)$



$\exists P_7 \iff H$ and H' have a common non-edge

P_7-free $\iff \overline{H} \cap \overline{H'} = \emptyset$

In the certificates, $\Theta(n^2)$ bits of information have to be transmitted through $O(n)$ vertices

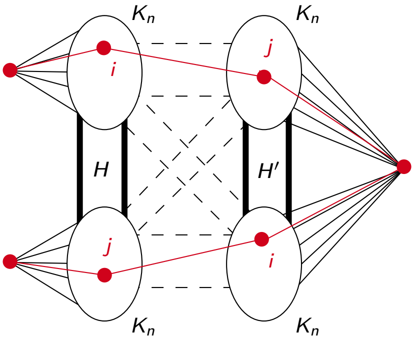
Certification of P_k-freeness: lower bound

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\Omega(n)$ bits are necessary to certify that a graph is P_7-free.

H, H' bipartite graphs with n vertices on each side

↳ size $= \Theta(n^2)$



$\exists P_7 \iff H$ and H' have a common non-edge

P_7-free $\iff \overline{H} \cap \overline{H'} = \emptyset$

In the certificates, $\Theta(n^2)$ bits of information have to be transmitted through $O(n)$ vertices $\implies$ certificates of size $\Omega(n)$

Certification of P_k-freeness: lower bound

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\Omega\left(\frac{n}{d}\right)$ bits are necessary to certify that a graph is P_{4d+3}-free, if vertices can see at distance d.

Certification of P_k-freeness: lower bound

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\Omega\left(\frac{n}{d}\right)$ bits are necessary to certify that a graph is P_{4d+3}-free, if vertices can see at distance d.

View of a vertex = all the information available at distance $\leqslant d$:

- vertices (and their identifiers)
- edges
- certificates

Certification of P_k-freeness: lower bound

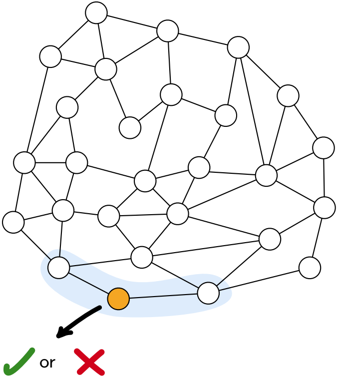
Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\Omega\left(\frac{n}{d}\right)$ bits are necessary to certify that a graph is P_{4d+3}-free, if vertices can see at distance d.

View of a vertex = all the information available at distance $\leqslant d$:

- vertices (and their identifiers)
- edges
- certificates

$d = 1$



Certification of P_k-freeness: lower bound

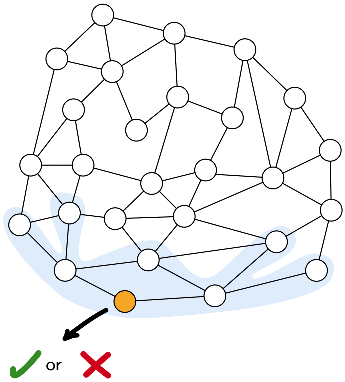
Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\Omega\left(\frac{n}{d}\right)$ bits are necessary to certify that a graph is P_{4d+3}-free, if vertices can see at distance d.

View of a vertex = all the information available at distance $\leqslant d$:

- vertices (and their identifiers)
- edges
- certificates

$d = 2$



Certification of P_k-freeness: lower bound

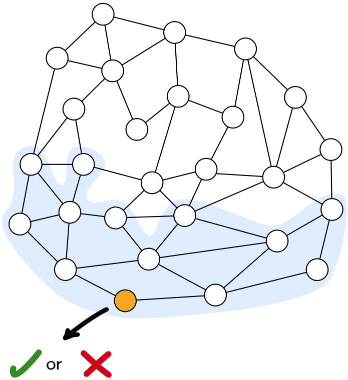
Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\Omega\left(\frac{n}{d}\right)$ bits are necessary to certify that a graph is P_{4d+3}-free, if vertices can see at distance d.

View of a vertex = all the information available at distance $\leqslant d$:

- vertices (and their identifiers)
- edges
- certificates

$d = 3$



Certification of P_k-freeness: lower bound

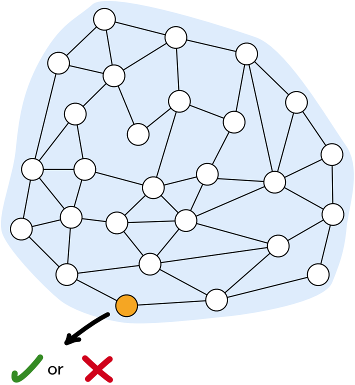
Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\Omega\left(\frac{n}{d}\right)$ bits are necessary to certify that a graph is P_{4d+3}-free, if vertices can see at distance d.

View of a vertex = all the information available at distance $\leqslant d$:

- vertices (and their identifiers)
- edges
- certificates

$d = 6$



Upper bounds

Certification in graphs of minimum degree $O(n^{\delta})$

Certification in graphs of minimum degree $O(n^\delta)$

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

Let $\delta < 1$. Any property can be certified with certificates of size $O(n^{2-\delta} \log n)$ in graphs of minimum degree n^δ, if vertices can see at distance 2.

Certification in graphs of minimum degree $O(n^\delta)$

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

Let $\delta < 1$. Any property can be certified with certificates of size $O(n^{2-\delta} \log n)$ in graphs of minimum degree n^δ, if vertices can see at distance 2.

Idea of the proof:

- cut the information of the graph in n^δ pieces of size $O(n^{2-\delta})$

Certification in graphs of minimum degree $O(n^\delta)$

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

Let $\delta < 1$. Any property can be certified with certificates of size $O(n^{2-\delta} \log n)$ in graphs of minimum degree n^δ, if vertices can see at distance 2.

Idea of the proof:

- cut the information of the graph in n^δ pieces of size $O(n^{2-\delta})$
- give well-chosen $O(\log n)$ pieces to every vertex

Certification in graphs of minimum degree $O(n^\delta)$

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

Let $\delta < 1$. Any property can be certified with certificates of size $O(n^{2-\delta} \log n)$ in graphs of minimum degree n^δ, if vertices can see at distance 2.

Idea of the proof:

- cut the information of the graph in n^δ pieces of size $O(n^{2-\delta})$
- give well-chosen $O(\log n)$ pieces to every vertex

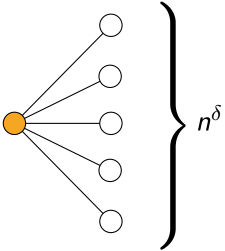
Certification in graphs of minimum degree $O(n^\delta)$

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

Let $\delta < 1$. Any property can be certified with certificates of size $O(n^{2-\delta} \log n)$ in graphs of minimum degree n^δ, if vertices can see at distance 2.

Idea of the proof:

- cut the information of the graph in n^δ pieces of size $O(n^{2-\delta})$
- give well-chosen $O(\log n)$ pieces to every vertex



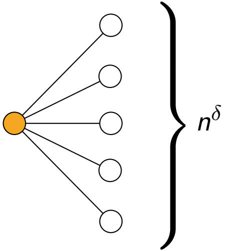
Certification in graphs of minimum degree $O(n^\delta)$

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

Let $\delta < 1$. Any property can be certified with certificates of size $O(n^{2-\delta} \log n)$ in graphs of minimum degree n^δ, if vertices can see at distance 2.

Idea of the proof:

- cut the information of the graph in n^δ pieces of size $O(n^{2-\delta})$
- give well-chosen $O(\log n)$ pieces to every vertex
- each vertex checks that it sees all the pieces in its neighborhood, and reconstructs the graph



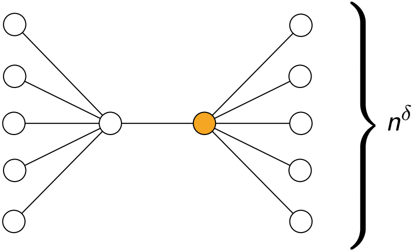
Certification in graphs of minimum degree $O(n^\delta)$

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

Let $\delta < 1$. Any property can be certified with certificates of size $O(n^{2-\delta} \log n)$ in graphs of minimum degree n^δ, if vertices can see at distance 2.

Idea of the proof:

- cut the information of the graph in n^δ pieces of size $O(n^{2-\delta})$
- give well-chosen $O(\log n)$ pieces to every vertex
- each vertex checks that it sees all the pieces in its neighborhood, and reconstructs the graph



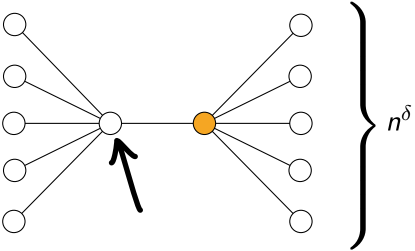
Certification in graphs of minimum degree $O(n^\delta)$

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

Let $\delta < 1$. Any property can be certified with certificates of size $O(n^{2-\delta} \log n)$ in graphs of minimum degree n^δ, if vertices can see at distance 2.

Idea of the proof:

- cut the information of the graph in n^δ pieces of size $O(n^{2-\delta})$
- give well-chosen $O(\log n)$ pieces to every vertex
- each vertex checks that it sees all the pieces in its neighborhood, and reconstructs the graph



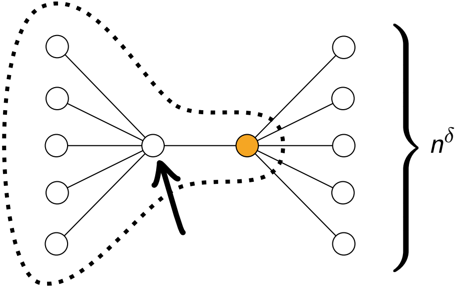
Certification in graphs of minimum degree $O(n^\delta)$

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

Let $\delta < 1$. Any property can be certified with certificates of size $O(n^{2-\delta} \log n)$ in graphs of minimum degree n^δ, if vertices can see at distance 2.

Idea of the proof:

- cut the information of the graph in n^δ pieces of size $O(n^{2-\delta})$
- give well-chosen $O(\log n)$ pieces to every vertex
- each vertex checks that it sees all the pieces in its neighborhood, and reconstructs the graph



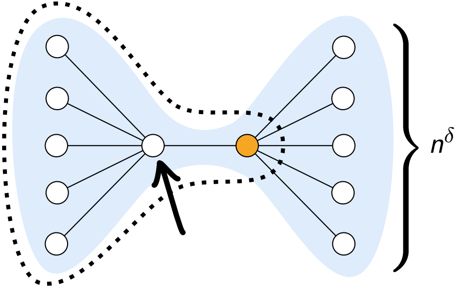
Certification in graphs of minimum degree $O(n^\delta)$

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

Let $\delta < 1$. Any property can be certified with certificates of size $O(n^{2-\delta} \log n)$ in graphs of minimum degree n^δ, if vertices can see at distance 2.

Idea of the proof:

- cut the information of the graph in n^δ pieces of size $O(n^{2-\delta})$
- give well-chosen $O(\log n)$ pieces to every vertex
- each vertex checks that it sees all the pieces in its neighborhood, and reconstructs the graph



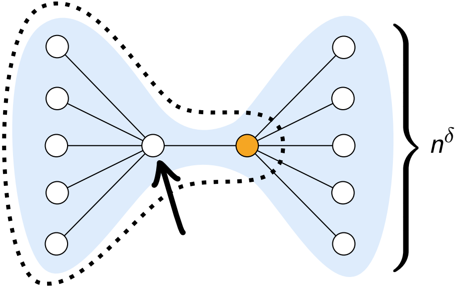
Certification in graphs of minimum degree $O(n^{\delta})$

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

Let $\delta < 1$. Any property can be certified with certificates of size $O(n^{2-\delta} \log n)$ in graphs of minimum degree n^{δ}, if vertices can see at distance 2.

Idea of the proof:

- cut the information of the graph in n^{δ} pieces of size $O(n^{2-\delta})$
- give well-chosen $O(\log n)$ pieces to every vertex
- each vertex checks that it sees all the pieces in its neighborhood, and reconstructs the graph
- each vertex checks that it is the same reconstructed graph for all its neighbors



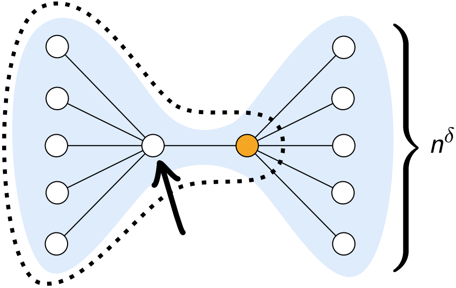
Certification in graphs of minimum degree $O(n^\delta)$

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

Let $\delta < 1$. Any property can be certified with certificates of size $O(n^{2-\delta} \log n)$ in graphs of minimum degree n^δ, if vertices can see at distance 2.

Idea of the proof:

- cut the information of the graph in n^δ pieces of size $O(n^{2-\delta})$
- give well-chosen $O(\log n)$ pieces to every vertex
- each vertex checks that it sees all the pieces in its neighborhood, and reconstructs the graph
- each vertex checks that it is the same reconstructed graph for all its neighbors
- each vertex checks that its neighborhood is correctly written in this graph



Certification in graphs of minimum degree $O(n^\delta)$

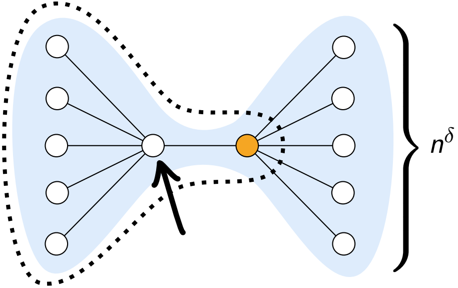
Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

Let $\delta < 1$. Any property can be certified with certificates of size $O(n^{2-\delta} \log n)$ in graphs of minimum degree n^δ, if vertices can see at distance 2.

Idea of the proof:

- cut the information of the graph in n^δ pieces of size $O(n^{2-\delta})$
- give well-chosen $O(\log n)$ pieces to every vertex
- each vertex checks that it sees all the pieces in its neighborhood, and reconstructs the graph
- each vertex checks that it is the same reconstructed graph for all its neighbors
- each vertex checks that its neighborhood is correctly written in this graph

$\implies$ every vertex knows G



Upper bound for path-freeness certification

Upper bound for path-freeness certification

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\tilde{O}(n^{3/2})$ bits are sufficient to certify that a graph is P_{4d-1}-free, if vertices can see at distance d.

Upper bound for path-freeness certification

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\tilde{O}(n^{3/2})$ bits are sufficient to certify that a graph is P_{4d-1}-free, if vertices can see at distance d.

- if all vertices have degree $\geqslant \sqrt{n} \longrightarrow$ ok by previous Theorem

Upper bound for path-freeness certification

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\tilde{O}(n^{3/2})$ bits are sufficient to certify that a graph is P_{4d-1}-free, if vertices can see at distance d.

- if all vertices have degree $\geqslant \sqrt{n} \longrightarrow$ ok by previous Theorem
- if all vertices have degree $\leqslant \sqrt{n} \longrightarrow$ ok because G has at most $\leqslant n^{3/2}$ edges

Upper bound for path-freeness certification

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\tilde{O}(n^{3/2})$ bits are sufficient to certify that a graph is P_{4d-1}-free, if vertices can see at distance d.

- if all vertices have degree $\geqslant \sqrt{n} \longrightarrow$ ok by previous Theorem
- if all vertices have degree $\leqslant \sqrt{n} \longrightarrow$ ok because G has at most $\leqslant n^{3/2}$ edges

$V^- :=$ vertices of degree $< \sqrt{n}$ $\qquad$ $V^+ :=$ vertices of degree $\geqslant \sqrt{n}$

Upper bound for path-freeness certification

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\tilde{O}(n^{3/2})$ bits are sufficient to certify that a graph is P_{4d-1}-free, if vertices can see at distance d.

- if all vertices have degree $\geqslant \sqrt{n} \longrightarrow$ ok by previous Theorem
- if all vertices have degree $\leqslant \sqrt{n} \longrightarrow$ ok because G has at most $\leqslant n^{3/2}$ edges

$V^- :=$ vertices of degree $< \sqrt{n}$ $\qquad$ $V^+ :=$ vertices of degree $\geqslant \sqrt{n}$

- give $G[V^-]$ to every vertex

Upper bound for path-freeness certification

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\tilde{O}(n^{3/2})$ bits are sufficient to certify that a graph is P_{4d-1}-free, if vertices can see at distance d.

- if all vertices have degree $\geqslant \sqrt{n} \longrightarrow$ ok by previous Theorem
- if all vertices have degree $\leqslant \sqrt{n} \longrightarrow$ ok because G has at most $\leqslant n^{3/2}$ edges

$V^- :=$ vertices of degree $< \sqrt{n}$ $\qquad$ $V^+ :=$ vertices of degree $\geqslant \sqrt{n}$

- give $G[V^-]$ to every vertex
- cut G in $\sqrt{n}$ pieces of size $n^{3/2}$ and give $O(\log n)$ pieces to every vertex

Upper bound for path-freeness certification

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\tilde{O}(n^{3/2})$ bits are sufficient to certify that a graph is P_{4d-1}-free, if vertices can see at distance d.

- if all vertices have degree $\geqslant \sqrt{n} \longrightarrow$ ok by previous Theorem
- if all vertices have degree $\leqslant \sqrt{n} \longrightarrow$ ok because G has at most $\leqslant n^{3/2}$ edges

$V^- :=$ vertices of degree $< \sqrt{n}$ $\qquad$ $V^+ :=$ vertices of degree $\geqslant \sqrt{n}$

- give $G[V^-]$ to every vertex
- cut G in $\sqrt{n}$ pieces of size $n^{3/2}$ and give $O(\log n)$ pieces to every vertex

} size $\tilde{O}(n^{3/2})$

Upper bound for path-freeness certification

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\tilde{O}(n^{3/2})$ bits are sufficient to certify that a graph is P_{4d-1}-free, if vertices can see at distance d.

- if all vertices have degree $\geqslant \sqrt{n} \longrightarrow$ ok by previous Theorem
- if all vertices have degree $\leqslant \sqrt{n} \longrightarrow$ ok because G has at most $\leqslant n^{3/2}$ edges

$V^- :=$ vertices of degree $< \sqrt{n}$ $\qquad$ $V^+ :=$ vertices of degree $\geqslant \sqrt{n}$

- give $G[V^-]$ to every vertex
- cut G in $\sqrt{n}$ pieces of size $n^{3/2}$ and give $O(\log n)$ pieces to every vertex

} size $\tilde{O}(n^{3/2})$

$u \in V^-$	$u \in V^+$
$\downarrow$	$\downarrow$
u knows $G[V^-]$	u knows $G[V^-]$

Upper bound for path-freeness certification

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\tilde{O}(n^{3/2})$ bits are sufficient to certify that a graph is P_{4d-1}-free, if vertices can see at distance d.

- if all vertices have degree $\geqslant \sqrt{n} \longrightarrow$ ok by previous Theorem
- if all vertices have degree $\leqslant \sqrt{n} \longrightarrow$ ok because G has at most $\leqslant n^{3/2}$ edges

$V^- :=$ vertices of degree $< \sqrt{n}$ $\qquad$ $V^+ :=$ vertices of degree $\geqslant \sqrt{n}$

- give $G[V^-]$ to every vertex
- cut G in $\sqrt{n}$ pieces of size $n^{3/2}$ and give $O(\log n)$ pieces to every vertex

} size $\tilde{O}(n^{3/2})$

$u \in V^-$	$u \in V^+$
$\downarrow$	$\downarrow$
u knows $G[V^-]$	u knows $G[V^-]$ and u sees all the pieces of the G in its neighborhood, so it can reconstruct G

Upper bound for path-freeness certification

Theorem (Bousquet, Cook, Feuilloley, Pierron, Z.)

$\tilde{O}(n^{3/2})$ bits are sufficient to certify that a graph is P_{4d-1}-free, if vertices can see at distance d.

- if all vertices have degree $\geqslant \sqrt{n} \longrightarrow$ ok by previous Theorem
- if all vertices have degree $\leqslant \sqrt{n} \longrightarrow$ ok because G has at most $\leqslant n^{3/2}$ edges

$V^- :=$ vertices of degree $< \sqrt{n}$ $V^+ :=$ vertices of degree $\geqslant \sqrt{n}$

- give $G[V^-]$ to every vertex
- cut G in $\sqrt{n}$ pieces of size $n^{3/2}$ and give $O(\log n)$ pieces to every vertex

} size $\tilde{O}(n^{3/2})$

$u \in V^-$	$u \in V^+$
$\downarrow$	$\downarrow$
u knows $G[V^-]$	u knows $G[V^-]$ and u sees all the pieces of the G in its neighborhood, so it can ~~reconstruct G~~

Upper bound for path-freeness certification

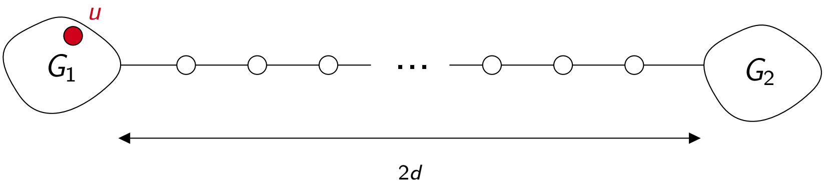
Main challenge : if $u \in V^+$, is it possible for u to verify that it reconstructed the correct graph G ?

Upper bound for path-freeness certification

Main challenge : if $u \in V^+$, is it possible for u to verify that it reconstructed the correct graph G ? $\longrightarrow$ **in general : no.**

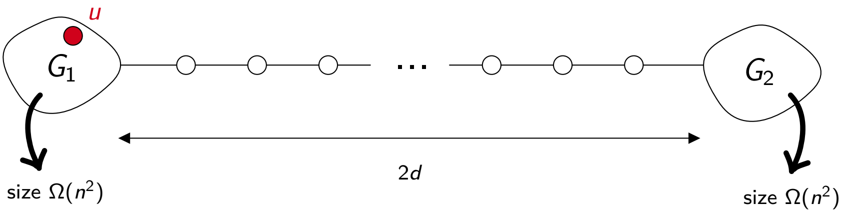
Upper bound for path-freeness certification

<u>Main challenge</u> : if $u \in V^+$, is it possible for u to verify that it reconstructed the correct graph G ? $\longrightarrow$ **in general : no.**



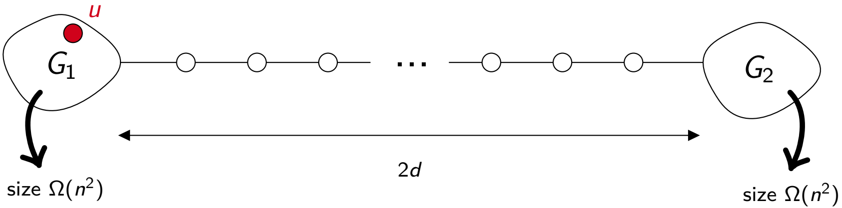
Upper bound for path-freeness certification

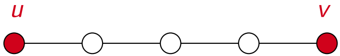
Main challenge : if $u \in V^+$, is it possible for u to verify that it reconstructed the correct graph G ? $\longrightarrow$ **in general : no.**



Upper bound for path-freeness certification

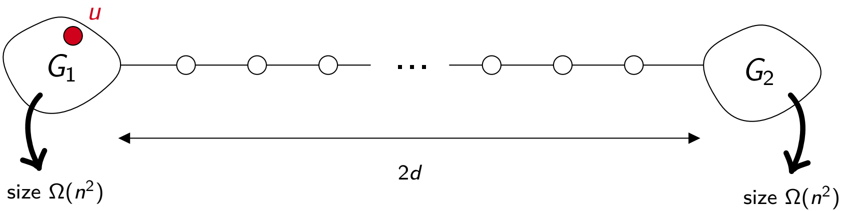
Main challenge : if $u \in V^+$, is it possible for u to verify that it reconstructed the correct graph G ? $\longrightarrow$ **in general : no.**

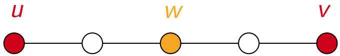




Upper bound for path-freeness certification

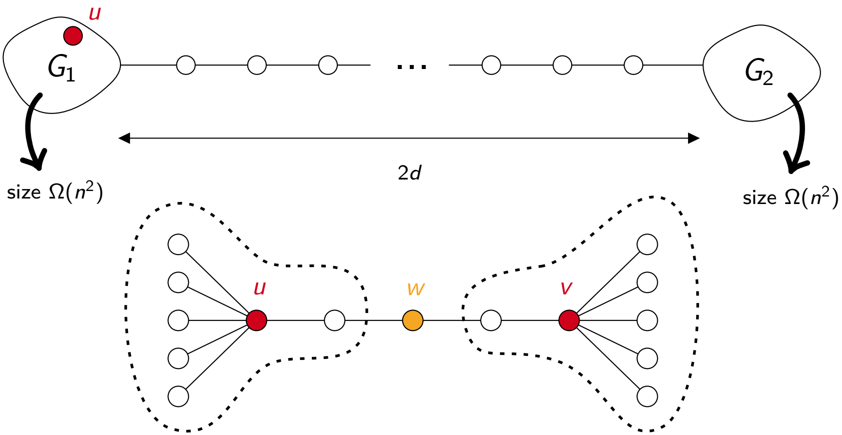
Main challenge : if $u \in V^+$, is it possible for u to verify that it reconstructed the correct graph G ? $\longrightarrow$ **in general : no.**





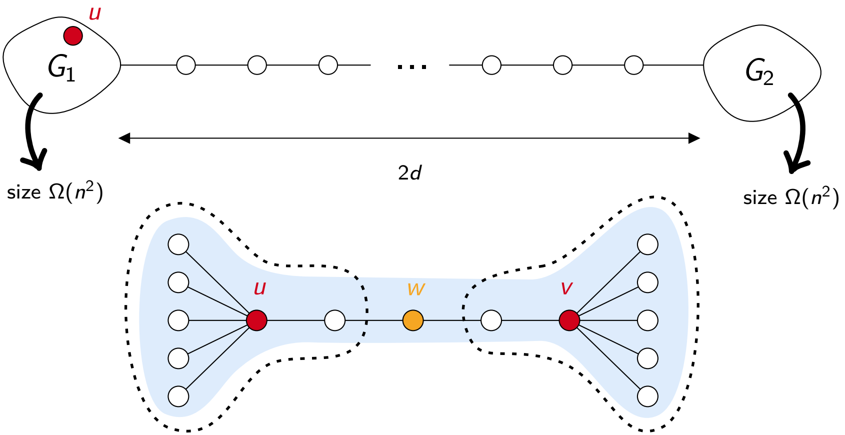
Upper bound for path-freeness certification

Main challenge : if $u \in V^+$, is it possible for u to verify that it reconstructed the correct graph G ? $\longrightarrow$ **in general : no.**



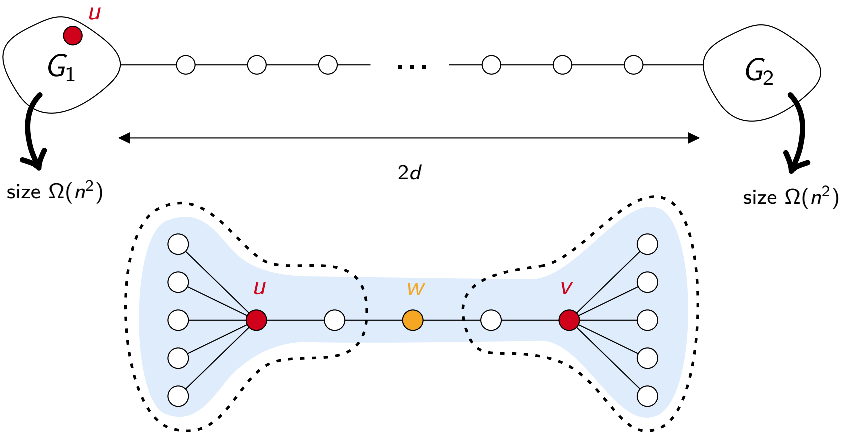
Upper bound for path-freeness certification

Main challenge : if $u \in V^+$, is it possible for u to verify that it reconstructed the correct graph G ? $\longrightarrow$ **in general : no.**



Upper bound for path-freeness certification

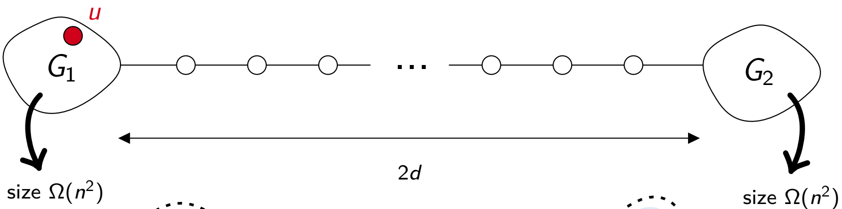
Main challenge : if $u \in V^+$, is it possible for u to verify that it reconstructed the correct graph G ? $\longrightarrow$ **in general : no.**

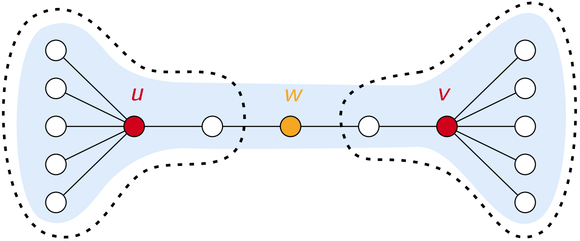


$d(u, v) \leqslant 2d - 2 \Longrightarrow u$ and v reconstruct the same graph

Upper bound for path-freeness certification

Main challenge : if $u \in V^+$, is it possible for u to verify that it reconstructed the correct graph G ? $\longrightarrow$ **in general : no.**



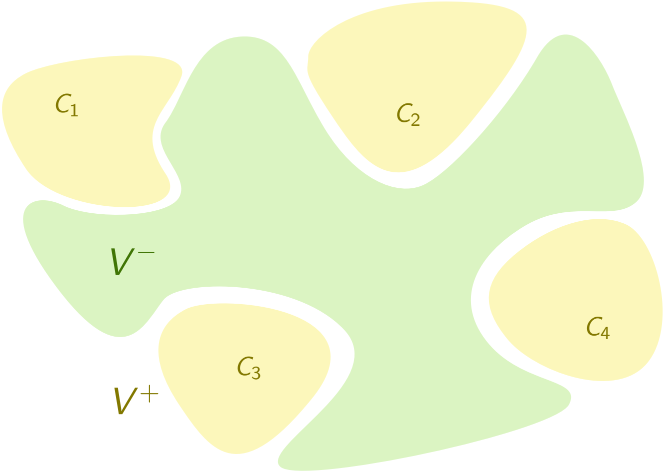


$d(u, v) \leqslant 2d - 2 \Longrightarrow u$ and v reconstruct the same graph

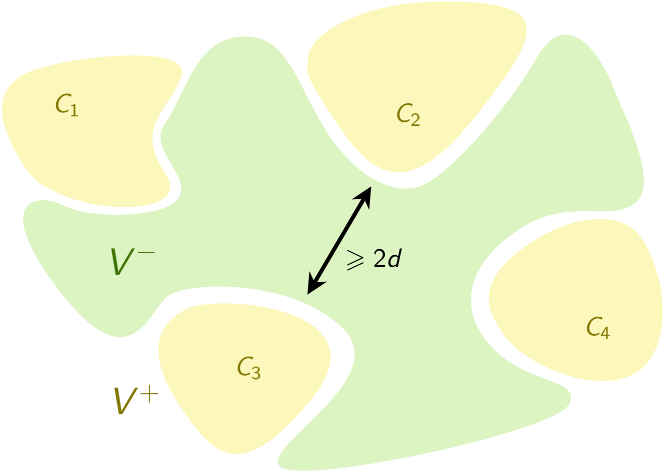
Partition V^+ into components: set of vertices which reconstruct the same graph

Upper bound for path-freeness certification

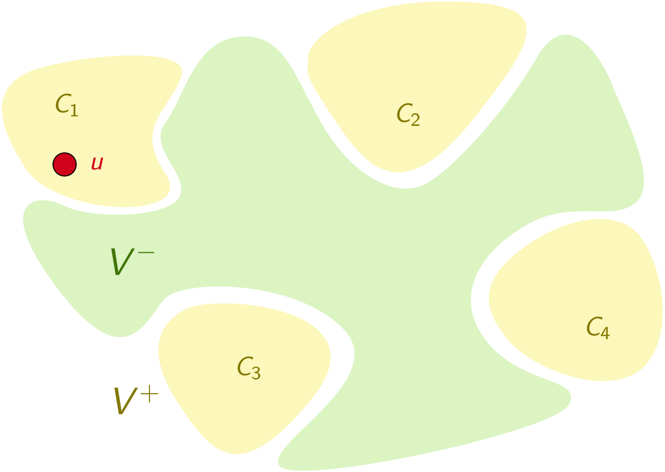
Upper bound for path-freeness certification



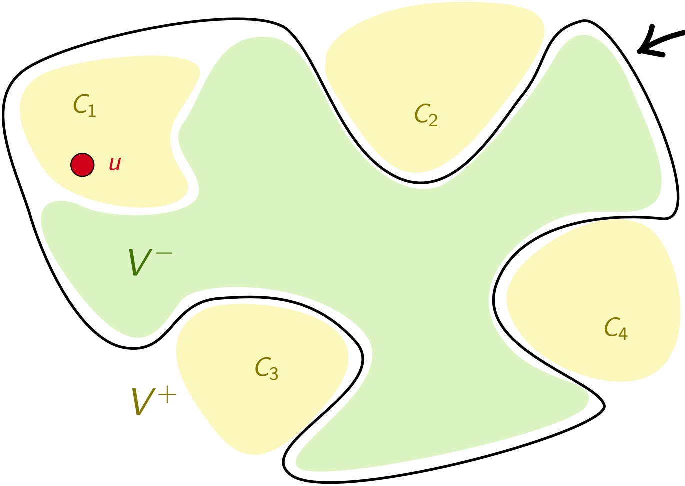
Upper bound for path-freeness certification



Upper bound for path-freeness certification

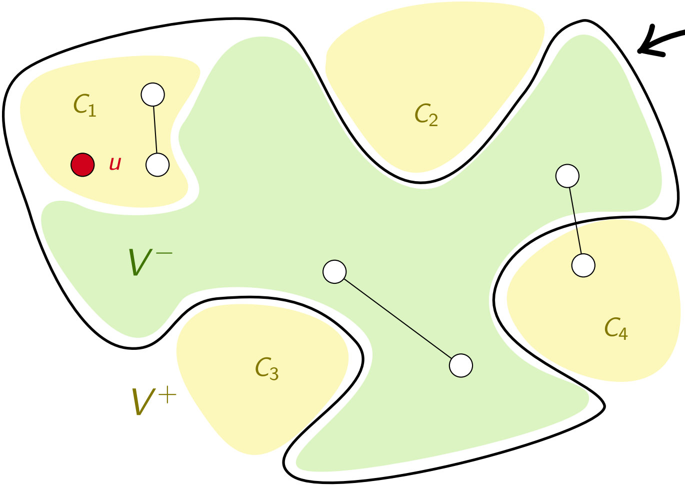


Upper bound for path-freeness certification



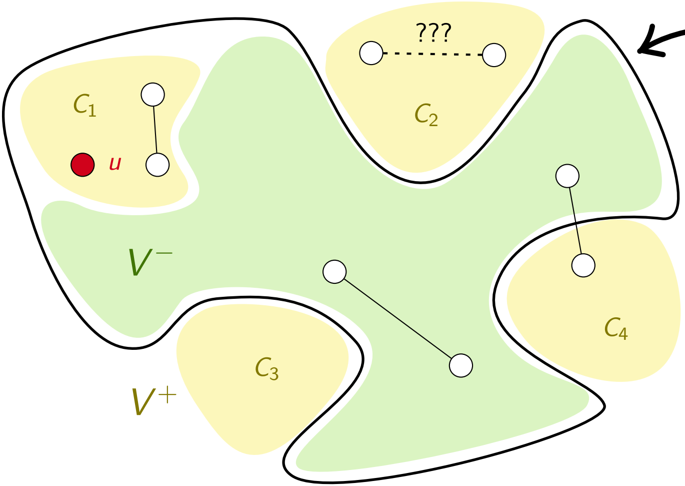
u knows all the edges and non-edges having an endpoint here

Upper bound for path-freeness certification



u knows all the edges and non-edges having an endpoint here

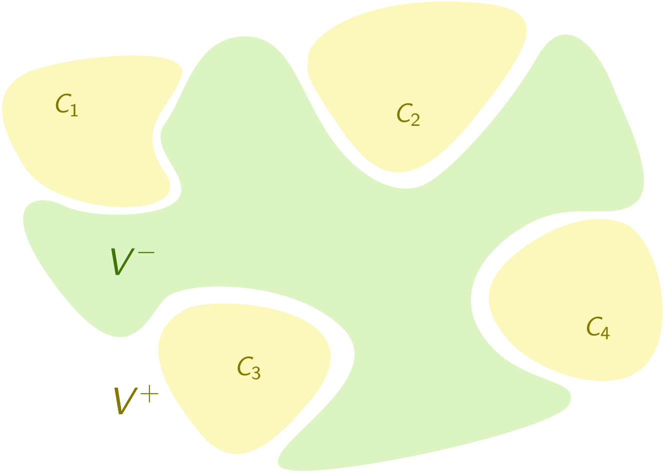
Upper bound for path-freeness certification



u knows all the edges and non-edges having an endpoint here

Upper bound for path-freeness certification

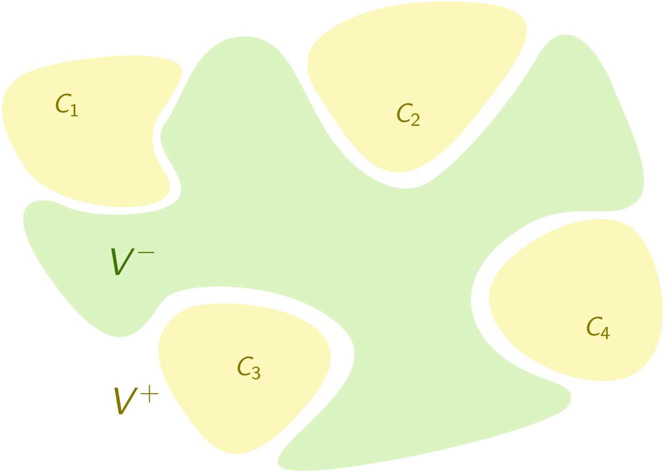
If there is a P_{4d-1}, which vertex detects it ?



Upper bound for path-freeness certification

If there is a P_{4d-1}, which vertex detects it ?

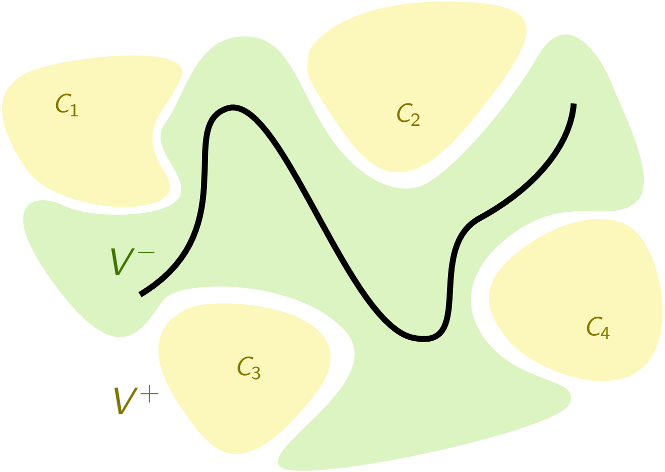
Case 1: P_{4d-1} is included in V^-.



Upper bound for path-freeness certification

If there is a P_{4d-1}, which vertex detects it ?

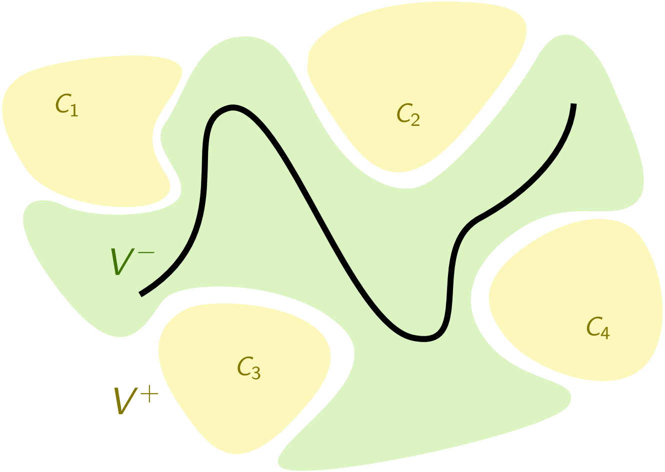
Case 1: P_{4d-1} is included in V^-.



Upper bound for path-freeness certification

If there is a P_{4d-1}, which vertex detects it ?

Case 1: P_{4d-1} is included in V^-.

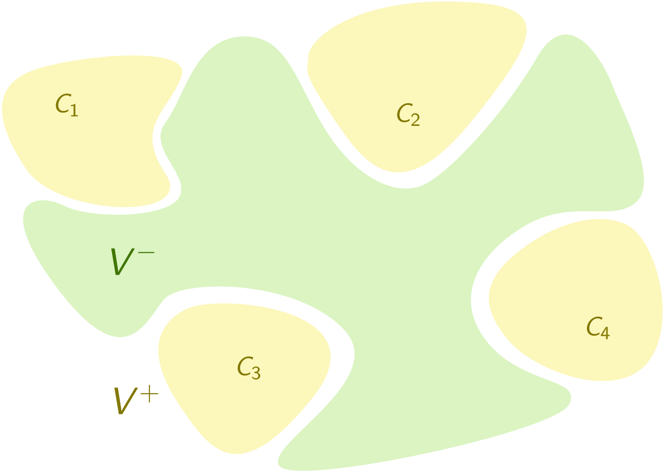


Every vertex detects it !

Upper bound for path-freeness certification

If there is a P_{4d-1}, which vertex detects it ?

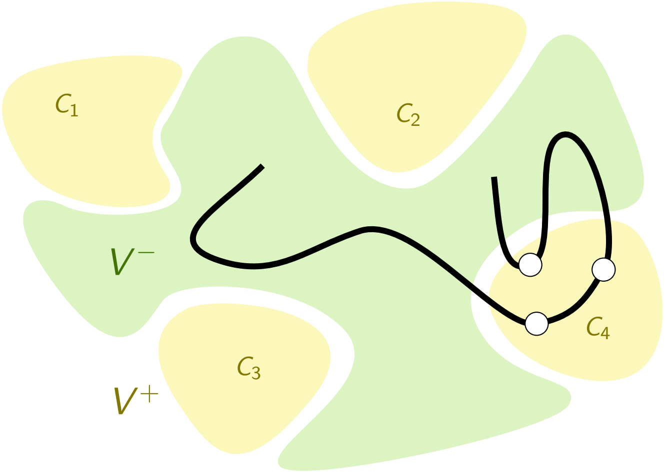
Case 2: exactly one component intersects P_{4d-1}.



Upper bound for path-freeness certification

If there is a P_{4d-1}, which vertex detects it ?

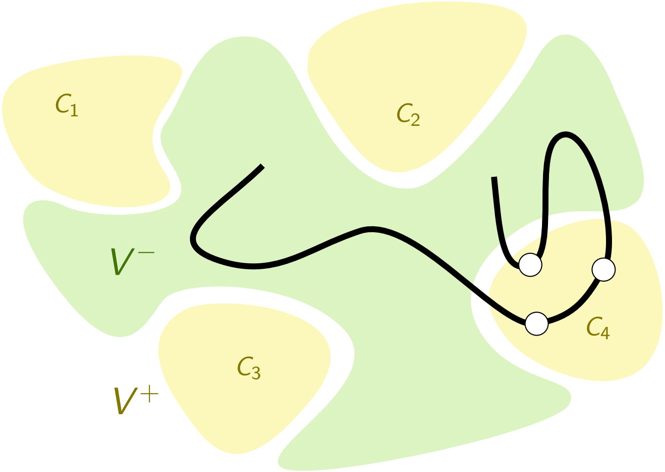
Case 2: exactly one component intersects P_{4d-1}.



Upper bound for path-freeness certification

If there is a P_{4d-1}, which vertex detects it ?

<u>Case 2</u>: exactly one component intersects P_{4d-1}.

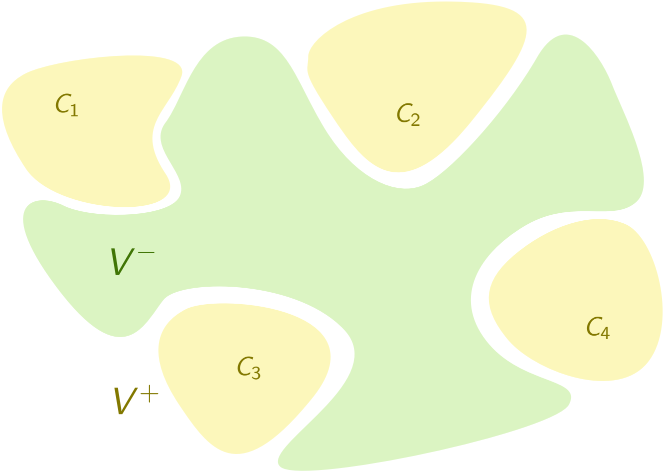


Every vertex in C_4 detects it !

Upper bound for path-freeness certification

If there is a P_{4d-1}, which vertex detects it ?

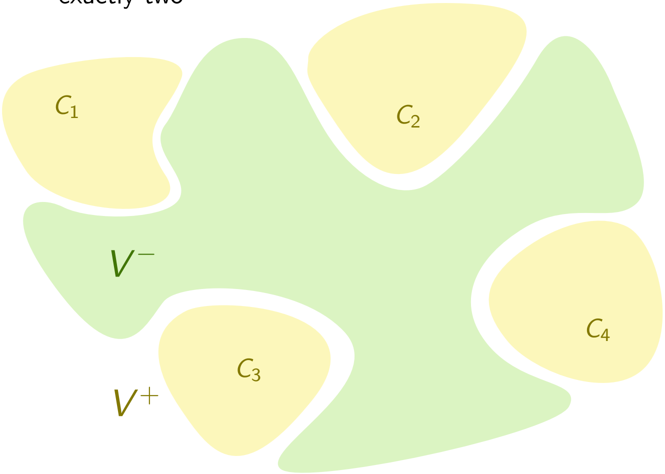
Case 3: at least two components intersect P_{4d-1}.



Upper bound for path-freeness certification

If there is a P_{4d-1}, which vertex detects it ?

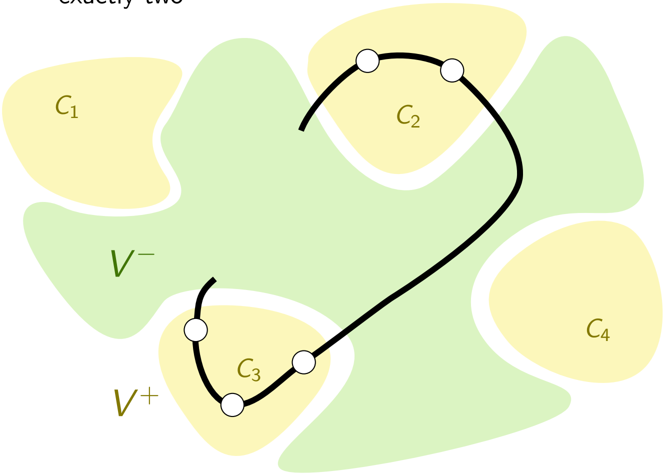
Case 3: ~~at least two~~ exactly two components intersect P_{4d-1}.



Upper bound for path-freeness certification

If there is a P_{4d-1}, which vertex detects it ?

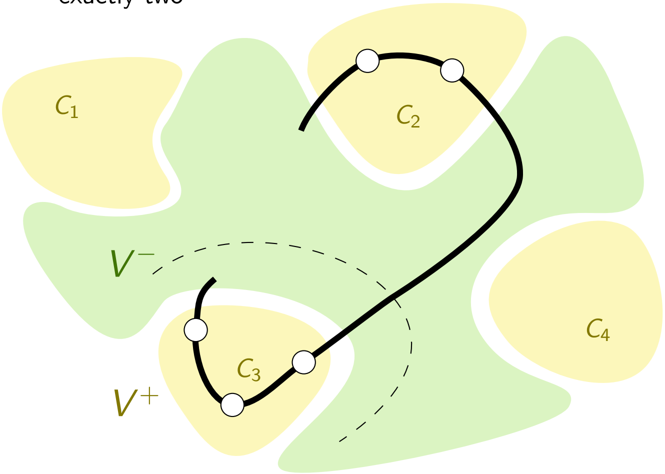
Case 3: ~~at least two~~ exactly two components intersect P_{4d-1}.



Upper bound for path-freeness certification

If there is a P_{4d-1}, which vertex detects it ?

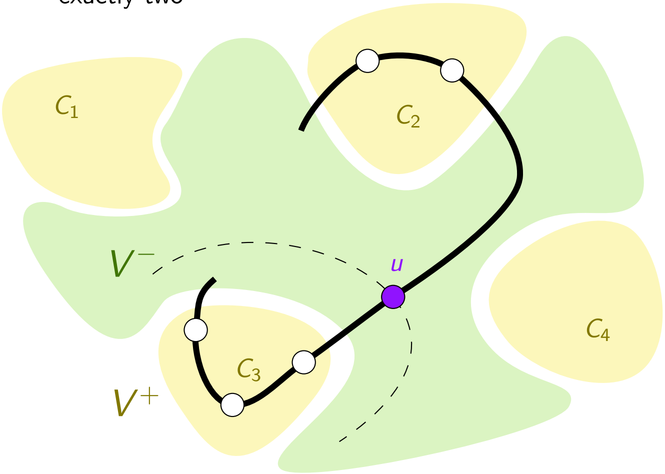
Case 3: ~~at least two~~ exactly two components intersect P_{4d-1}.



Upper bound for path-freeness certification

If there is a P_{4d-1}, which vertex detects it ?

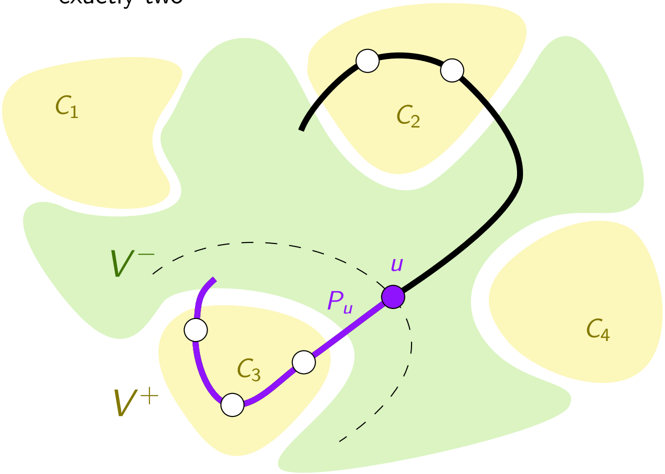
Case 3: ~~at least two~~ exactly two components intersect P_{4d-1}.



Upper bound for path-freeness certification

If there is a P_{4d-1}, which vertex detects it ?

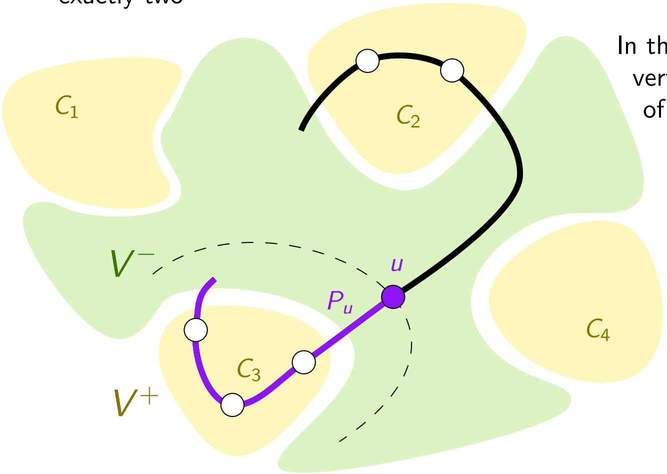
Case 3: ~~at least two~~ exactly two components intersect P_{4d-1}.



Upper bound for path-freeness certification

If there is a P_{4d-1}, which vertex detects it ?

Case 3: ~~at least two~~ exactly two components intersect P_{4d-1}.

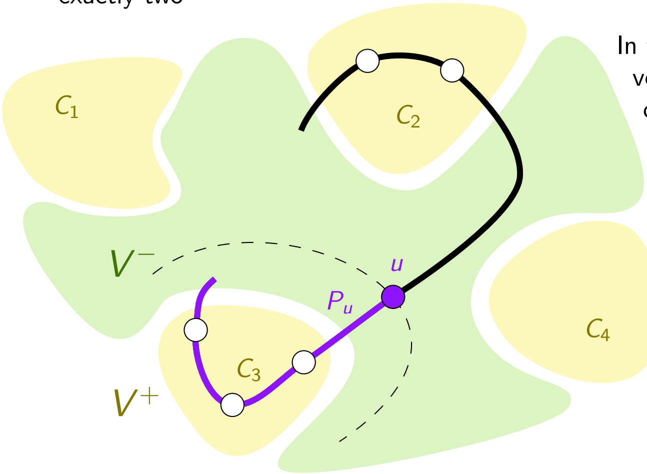


In the certificate of every vertex, add the length of P_u.

Upper bound for path-freeness certification

If there is a P_{4d-1}, which vertex detects it ?

Case 3: ~~at least two~~ exactly two components intersect P_{4d-1}.



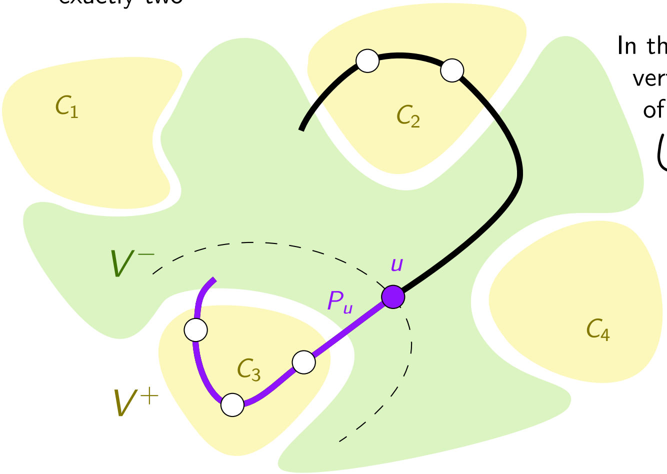
In the certificate of every vertex, add the length of P_u.

↪ size $O(n \log n)$ in total.

Upper bound for path-freeness certification

If there is a P_{4d-1}, which vertex detects it ?

Case 3: ~~at least two~~ exactly two components intersect P_{4d-1}.



In the certificate of every vertex, add the length of P_u.

↳ size $O(n \log n)$ in total.

Every vertex in C_2 detects it !

Conclusion: overview of our results for H-freeness and open questions

Conclusion: overview of our results for H-freeness and open questions

Graph H	Bound

Conclusion: overview of our results for H-freeness and open questions

Graph H	Bound
P_{4d+3}	$\Omega(n)$

Conclusion: overview of our results for H-freeness and open questions

Graph H	Bound
P_{4d+3}	$\Omega(n)$
P_{4d-1}	$\tilde{O}(n^{3/2})$

Conclusion: overview of our results for H-freeness and open questions

Graph H	Bound
P_{4d+3}	$\Omega(n)$
P_{4d-1}	$\tilde{O}(n^{3/2})$
$\lvert V(H)\rvert \leqslant 4d-1$	$\tilde{O}(n^{3/2})$

Conclusion: overview of our results for H-freeness and open questions

Graph H	Bound
P_{4d+3}	$\Omega(n)$
P_{4d-1}	$\tilde{O}(n^{3/2})$
$\|V(H)\| \leqslant 4d-1$	$\tilde{O}(n^{3/2})$
$P_{\lceil 14d/3 \rceil -1}$	$\tilde{O}(n^{3/2})$

Conclusion: overview of our results for H-freeness and open questions

Graph H	Bound
P_{4d+3}	$\Omega(n)$
P_{4d-1}	$\tilde{O}(n^{3/2})$
$\|V(H)\| \leqslant 4d-1$	$\tilde{O}(n^{3/2})$
$P_{\lceil 14d/3 \rceil -1}$	$\tilde{O}(n^{3/2})$
P_{3d-1}	$\tilde{O}(n)$

Conclusion: overview of our results for H-freeness and open questions

Graph H	Bound
P_{4d+3}	$\Omega(n)$
P_{4d-1}	$\tilde{O}(n^{3/2})$
$\|V(H)\| \leqslant 4d-1$	$\tilde{O}(n^{3/2})$
$P_{\lceil 14d/3 \rceil -1}$	$\tilde{O}(n^{3/2})$
P_{3d-1}	$\tilde{O}(n)$

Open questions:

- what if $d = 1$?

Conclusion: overview of our results for H-freeness and open questions

Graph H	Bound
P_{4d+3}	$\Omega(n)$
P_{4d-1}	$\tilde{O}(n^{3/2})$
$\lvert V(H)\rvert \leqslant 4d-1$	$\tilde{O}(n^{3/2})$
$P_{\lceil 14d/3 \rceil -1}$	$\tilde{O}(n^{3/2})$
P_{3d-1}	$\tilde{O}(n)$

Open questions:

- what if $d = 1$? $\longrightarrow \tilde{O}(n^{3/2})$ for P_5

Conclusion: overview of our results for H-freeness and open questions

Graph H	Bound
P_{4d+3}	$\Omega(n)$
P_{4d-1}	$\tilde{O}(n^{3/2})$
$\|V(H)\| \leqslant 4d-1$	$\tilde{O}(n^{3/2})$
$P_{\lceil 14d/3 \rceil -1}$	$\tilde{O}(n^{3/2})$
P_{3d-1}	$\tilde{O}(n)$

Open questions:

- what if $d = 1$? $\longrightarrow \tilde{O}(n^{3/2})$ for P_5
- Conjecture: for every $\alpha > 0$, there exists $\varepsilon > 0$ such that we can certify $P_{\alpha d}$-free graphs with certificates of size $O(n^{2-\varepsilon})$.

Thanks for your attention !