PII: S0304-3975(12)00542-7
DOI: 10.1016/j.tcs.2012.06.006
Reference: TCS 8925

To appear in: *Theoretical Computer Science*

Received date: 19 July 2011
Revised date: 29 March 2012
Accepted date: 2 June 2012

An $O(nm)$-time Certifying Algorithm for Recognizing HHD-free Graphs

Stavros D. Nikolopoulos and Leonidas Palios
Department of Computer Science, University of Ioannina
P.O.Box 1186, GR-45110 Ioannina, Greece
{stavros, palios}@cs.uoi.gr

Abstract

In this paper, we consider the recognition problem on a class of perfectly orderable graphs, namely, the HHD-free graphs; such graphs do not contain any induced subgraph isomorphic to a house, a hole, or a domino. We prove properties of the HHD-free graphs which enable us to present an $O(n\,m)$-time and $O(n+m)$-space algorithm for determining whether a graph on n vertices and m edges is HHD-free; currently, this is the fastest algorithm for this problem. We also describe how the algorithm can be augmented to provide a certificate (an induced house, hole, or domino) whenever it decides that the input graph is not HHD-free, thus answering an open question posed by Hoàng and Sritharan (Theoretical Computer Science 259 (2001) 233-244). The certificate computation requires $O(n+m)$ additional time and $O(n)$ space.

Keywords: HHD-free graphs, perfectly orderable graphs, certifying algorithms, recognition.

1 Introduction

A linear order $\prec$ on the vertices of a graph G is *perfect* if the ordered graph $(G, \prec)$ contains no induced P_4 $abcd$ (i.e., a chordless path on the 4 vertices a, b, c, d) with $a \prec b$ and $d \prec c$; such a P_4 is called an *obstruction*. In the early 1980s, Chvátal [4] defined the class of graphs that admit a perfect order and called them *perfectly orderable* graphs.

Chvátal proved that if a graph G admits a perfect order $\prec$, then the greedy coloring algorithm applied to $(G, \prec)$ produces an optimal coloring using only $\omega(G)$ colors, where $\omega(G)$ is the clique number of G. This implies that the perfectly orderable graphs are perfect; a graph G is *perfect* if for each induced subgraph H of G, the chromatic number $\chi(H)$ equals the clique number $\omega(H)$ of the subgraph H. The class of perfect graphs was introduced in the early 1960s by Berge [1], who also conjectured that a graph is perfect if and only if it contains no induced subgraph isomorphic to an odd cycle of length at least five, or to the complement of such an odd cycle. This conjecture, known as the *strong perfect graph conjecture*, has been recently established due to the work of Chudnovsky *et al.* [3].

The interest in perfectly orderable graphs comes from the fact that several problems in graph theory, which are NP-complete in general graphs, have polynomial-time solutions in graphs that admit a perfect order [2, 7]; unfortunately, it is NP-complete to decide whether a graph admits a perfect order [16]. Since the recognition of perfectly orderable graphs is NP-complete, we are interested in characterizing graphs which form polynomially recognizable subclasses of perfectly orderable graphs. Many such classes of graphs, with very interesting structural and algorithmic properties, have been defined so far and shown to admit polynomial-time recognitions (see [2, 7]); note however that not all subclasses of perfectly orderable graphs admit polynomial-time recognition [10].

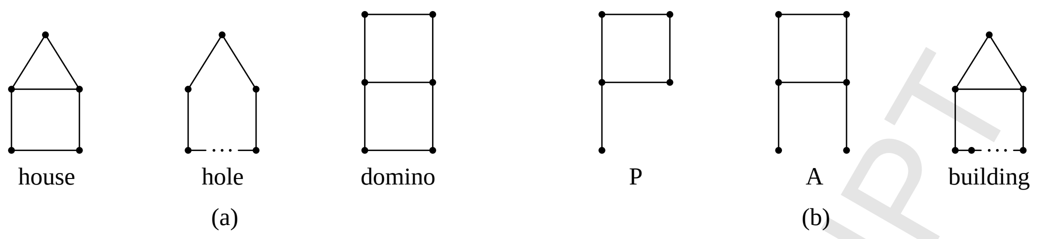

Figure 1: (a) The forbidden graps for the class of HHD-free graphs; (b) The graphs ‘P’, ‘A’, and the building.

Several subclasses of the class of perfectly orderable graphs have been extensively studied in the last decades due to their wide applicability in many fields of computer and engineering sciences; the most notable subclasses include the chordal, co-chordal, comparability, P_4-comparability, permutation, chordal bipartite, and distance-hereditary graphs [2, 7]. It is worth noting that many optimization problems, such as, coloring, max-clique, clique-cover, path-cover, domination set, independent set, cliquewidth, treewidth, and also the recognition problem admit polynomial solutions on theses classes of graphs.

An algorithm for the *recognition problem* is one that takes as input a graph G and decides whether G has a certain property; such an algorithm, which we call *recognition algorithm*, returns “yes” if the input graph has the property or “no” if it does not. A *certifying recognition algorithm* is a recognition algorithm that provides a certificate with each answer that it produces [9, 14]. The certificate is a piece of evidence that proves that the algorithm’s answer is correct; in particular, in case of membership (i.e., the input graph G belongs to a given graph class $\mathcal{G}$) a certifying algorithm usually provides as certificate a structure for the input graph that characterizes the class $\mathcal{G}$, while in case of non-membership it provides as certificate a forbidden induced subgraph of the class $\mathcal{G}$. For an extensive discussion on certificates, see [14].

In this paper, we study the recognition problem for the class of HHD-free graphs, which properly generalizes the well-known class of chordal graphs [7]: a graph is *HHD-free* if it contains no induced subgraph isomorphic to a house, a hole (i.e., a chordless cycle on ≥ 5 vertices), or a domino (see Figure 1(a)). In [11], Hoàng and Khouzam proved that the HHD-free graphs admit a perfect order, and thus are perfectly orderable. A superclass of the HHD-free graphs, which also properly generalizes the class of chordal graphs, is the class of HH-free graphs: a graph is *HH-free* if it contains no induced subgraph isomorphic to a house or a hole. Although an HH-free graph is not necessarily perfectly orderable, the complement of any HH-free graph is; this was conjectured by Chvátal and proved by Hayward [8].

Hoàng and Khouzam [11], while studying the class of brittle graphs (a class of perfectly orderable graphs which contains the HHD-free graphs), showed that HHD-free graphs can be recognized in $O(n^4)$ time, where n denotes the number of vertices of the input graph. An improved result was obtained by Hoàng and Sritharan [12] who presented an $O(n^3)$-time algorithm for recognizing HH-free graphs and showed that HHD-free graphs can be recognized in $O(n^3)$ time as well; for each vertex v of the input graph, their algorithm relies on computing the chordal completion of the (ordered) non-neighbors of v, and checking whether the resulting graph is chordal. A further improvement was achieved by Nikolopoulos and Palios [18]: based on properties characterizing the chordal completion of a graph, they were able to avoid performing the chordal completion step, which is the most time-consuming ingredient of the algorithm in [12], and described algorithms for recognizing HH-free and HHD-free graphs that require $O(n \min\{m\,\alpha(n,n),\, m{+}n\log n\})$ time and $O(n{+}m)$-space, where m is the number of edges of the input graph, and $\alpha(\,,\,)$ denotes the very slowly growing functional inverse of Ackermann’s function. On other related classes of perfectly orderable graphs, Eschen *et al.* [6] recently described recognition algorithms for several of them, among which a recognition algorithm for HHP-free graphs;

a graph is HHP-free if it contains no hole, no house, and no "P" as induced subgraphs (see Figure 1(b)). Their algorithm is based on the property that every HHP-free graph is HHDA-free graph (a graph with no induced hole, house, domino, or "A"), and thus a graph G is HHP-free graph if and only if G is HHDA-free and contains no "P" as an induced subgraph. The characterization of HHDA-free graphs due to Olariu [20] (a graph G is HHDA-free if and only if every induced subgraph of G either is chordal or contains a non-trivial module) and the use of modular decomposition [15] allowed Eschen *et al.* to present an $O(n\,m)$-time recognition algorithm for HHP-free graphs.

In this paper, we present a new, faster algorithm for recognizing HHD-free graphs. For each vertex v of a given graph G, our algorithm computes the partition of the non-neighbors of v into sets of vertices based on their common neighbors with v, and following that, the connected components of the subgraphs induced by these partition sets. We show that if G is HHD-free, the graph obtained from G by shrinking each of these connected components into a single vertex is "almost chordal." As a result, we obtain an $O(n\,m)$-time and $O(n{+}m)$-space algorithm for determining whether a graph on n vertices and m edges is HHD-free. Our approach is different from the approach of [12, 18], where the decision is based on properties of the graph obtained by considering the chordal completion of the subgraph induced by the non-neighbors of each vertex v; the chordal completion is explicitly computed in [12] whereas it is implicitly maintained by means of the array NextNeighbor in [18]. Our result improves upon the algorithms in [12] and [18]: our algorithm is no slower than the $O(n^3)$-time algorithm in [12] and achieves a better time complexity than [18] in the case where $m = o(n \log n)$ while it matches its time complexity of [18] otherwise (recall that the algorithm in [18]requires $O(n\ \min\{m\,\alpha(n,n),\ m + n \log n\})$ time).

An additional advantage of our recognition algorithm is that it is certifying since it provides a certificate in the case of non-membership. In particular, we describe an augmented version of our recognition algorithm which provides a forbidden induced subgraph of the class of HHD-free graphs (an induced house, hole, or domino) whenever it decides that the input graph is not HHD-free; the certificate computation requires $O(n + m)$ additional time and $O(n)$ space. This answers an open question posed by Hoàng and Sritharan [12]. An early version of this work has appeared in [19].

The paper is structured as follows. In Section 2, we review the terminology and the notation that we use throughout the paper. In Section 3, we establish properties that enable us to efficiently determine whether a given graph is HHD-free, describe the algorithm, and give its analysis. Section 4 presents the certificate computation while Section 5 summarizes our results and presents some open problems.

2 Terminology - Notation

We consider finite undirected graphs with no loops or multiple edges. Let G be such a graph; then, $V(G)$ and $E(G)$ denote the set of vertices and of edges of G respectively. The subgraph of G induced by a subset S of G's vertices is denoted by $G[S]$. The *neighbors* of a vertex x of G, i.e., the vertices adjacent to a x, form the *neighborhood* $N_G(x)$ of x; the cardinality of $N_G(x)$ is the *degree* of x. The *closed neighborhood* of x is defined as $N_G[x] := N_G(x) \cup \{x\}$. For simplicity, we denote the set of non-neighbors of x in G by $M_G(x)$, i.e., $M_G(x) = V(G) - N_G[x]$.

A *path* in a graph G is a sequence of vertices $v_0 v_1 \cdots v_k$ such that $v_{i-1}v_i \in E(G)$ for $i = 1, 2, \ldots, k$; we say that this is a path from v_0 to v_k and that its *length* is k. A path is called *simple* if none of its vertices occurs more than once; it is called *trivial* if its length is equal to 0. A path (simple path) $v_0 v_1 \cdots v_k$ is a *cycle* (*simple cycle*) of length $k+1$ if $v_0 v_k \in E(G)$. An edge connecting two non-consecutive vertices in a simple path (cycle) is called a *chord*; then, a simple path (cycle) $v_0 v_1 \cdots v_k$ of a graph G is *chordless* if G contains no chords of the path (cycle), i.e., $v_i v_j \notin E(G)$ for any two non-consecutive vertices v_i, v_j in the path (cycle). The chordless path (chordless cycle, respectively) on n vertices is commonly denoted by P_n (C_n, respectively).

A *connected component* of a graph G is a maximal set $A \subseteq V(G)$ such that the subgraph $G[A]$ is connected, i.e., there exists a path in G connecting any two vertices in A.

3 The Algorithm

3.1 Outline of the algorithm

Our algorithm works in a fashion similar to the algorithms in [12, 18] in that it processes each vertex v of the input graph G seeking evidence that G contains an induced house, hole, or domino. If no evidence is found, then the algorithm reports that G is HHD-free. Therefore, our algorithm follows the outline shown in Figure 2.

```
Algorithm Recognize-HHD-free(graph $G$)

    for each vertex $v$ of $G$ do
        Process_Vertex($G$, $v$);

    print("The graph $G$ is HHD-free");
```

Figure 2: Algorithm Recognize-HHD-free.

Again, similarly to the algorithms in [12, 18], our Procedure Process_Vertex works on the ordering of the non-neighbors of v in non-decreasing number of common-neighbors with v. From that point on, the three algorithms part ways. The algorithm in [12] performs chordal completion in the subgraph of G induced by the (ordered) non-neighbors of v. The algorithm in [18] maintains this chordal completion implicitly by means of the NextNeighbor array. The present algorithm works on a special *shrunk* graph G_v (in fact, it computes the above mentioned ordering of non-neighbors of v in that graph instead of working on G); the graph G_v possesses some very interesting properties: it maintains information on whether the input graph G contains an induced house, hole, or domino, while at the same time ensuring that if G does not contain an induced house, hole, or domino then the subgraph of G_v induced by the non-neighbors of v is "nearly" chordal (to become more precise in Section 3.3). For this reason, Procedure Process_Vertex applied on the pair (G, v) is a careful extension of the perfect-elimination-ordering (PEO) testing algorithm [21, 7] to handle the special structure of the graph G_v; note that the PEO testing algorithm is used to detect whether a graph is chordal if an ordering of its vertices produced by the algorithm LexBFS [21, 7] is given. More details on how Procedure Process_Vertex works are given in Section 3.4 after we have formally defined the graph G_v, established its properties, and shown how we can take advantage of these properties to be able to detect whether v participates in an induced house, hole, or domino.

3.2 The graph G_v

The motivation for the construction of the graph G_v comes from the observation that if a vertex v is the top vertex of a house, participates in a hole, or is a corner vertex of a domino, all these subgraphs include a path y_1uvwy_2 where y_1, y_2 are non-neighbors of v having different common neighbors with v. This suggests that it may be a good idea to partition the set of non-neighbors of v based on their common neighbors with v and then to work with the graph that results from shrinking each of the partition sets into a single super-vertex.

However, shrinking each of the different partition sets into a single vertex leads to error as the following example indicates: consider the graph G on the left of Figure 3 which contains no house, hole, or domino; the partition of the non-neighbors of v based on the common neighbors with v yields

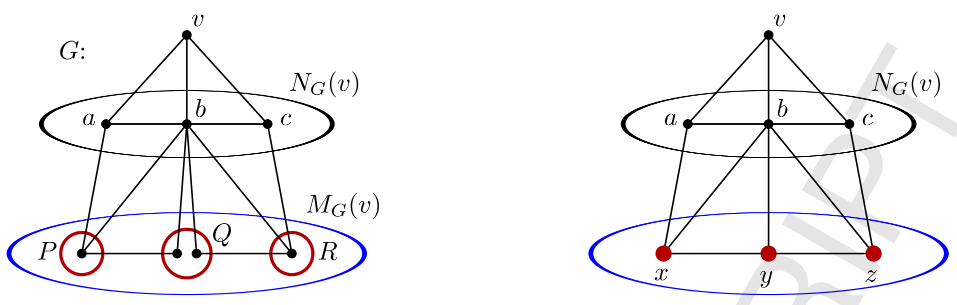

Figure 3: Shrinking each partition set into a single vertex may lead to error.

the sets P, Q, R; shrinking these sets into vertices x, y, z, respectively, yields the graph on the right of Figure 3, which contains the hole $vaxyzc$. That is, the shrunk graph cannot be used for the detection of holes in G because it may contain a hole even if G does not contain one.

A closer look at the example reveals that the error is due to the fact that the two connected components of the subgraph $G[Q]$ induced by the partition set Q in Figure 3 were shrunk into the same vertex. This suggests that if one intends to apply a shrinking mechanism, one needs to treat the connected components of any partition set as separate entities.

Therefore, we do the following:

1. we compute the partition of the non-neighbors $M_G(v)$ of vertex v in G based on the common neighbors of the vertices in $M_G(v)$ with v;

2. we order the partition sets by non-decreasing number of common neighbors with v (ties are broken arbitrarily); let $\mathcal{S}_v = (S_1, S_2, \ldots, S_\ell)$ be the resulting ordering;

3. for each set S_i, we compute the connected components of the subgraph $G[S_i]$;

4. we construct an auxiliary graph G_v by shrinking each of the connected components into a single vertex: namely, for each $i = 1, 2, \ldots, \ell$, let

$$Z_i = \{\, z_{C_1}, z_{C_2}, \ldots, z_{C_{t_i}} \mid C_1, C_2, \ldots, C_{t_i} \text{ are the conn. components of } G[S_i] \,\}; \qquad (1)$$

then
$V(G_v) = \{v\} \cup N_G(v) \cup \left(\bigcup_{i=1}^{\ell} Z_i\right)$
$E(G_v) = \{\, u\,w \mid u, w \in \{v\} \cup N_G(v) : \; uw \in E(G) \,\}$
$\cup \{\, u\, z_C \mid u \in N_G(v),\, \exists\, x \in \text{conn. component } C \text{ of } G[S_i] : \; ux \in E(G) \,\}$
$\cup \{\, z_C\, z_{C'} \mid \exists\, x, y \in \text{conn. components } C \text{ and } C' \text{ of } G[S_i] \text{ and } G[S_j], \text{ resp., where } i \neq j : \; xy \in E(G) \,\}.$

Note that when a component C is shrunk into a vertex z_C, then

(i) z_C is adjacent to a vertex $u \in N_G(v)$ iff there exists a vertex $x \in C$ such that $ux \in E(G)$, and

(ii) z_C is adjacent to vertex $z_{C'}$ that resulted from the shrinking of a component $C' \neq C$ iff there exist vertices $x \in C$ and $y \in C'$ such that $xy \in E(G)$.

As an example, Figures 4(b) and (c) show the graphs G_v and G_u for the graph shown (in two different ways) in Figure 4(a).

Notation: Since for vertex v, $N_G(v) = N_{G_v}(v)$, for simplicity, in the following we will write $N(v)$ instead.

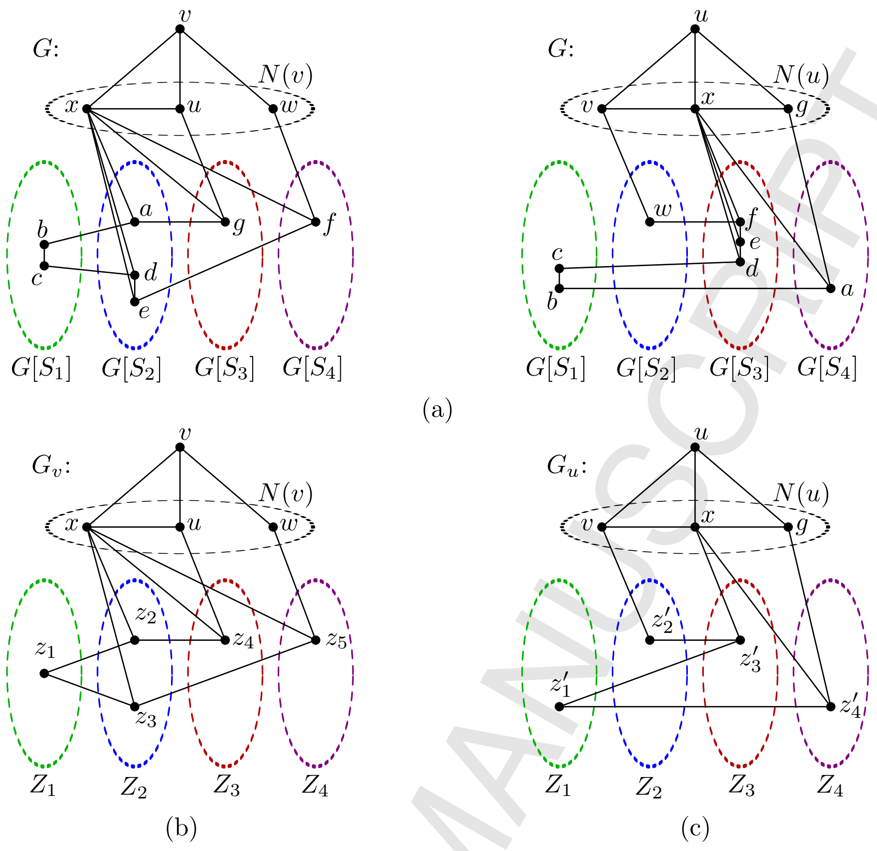

Figure 4: (a) The same graph shown in two different ways (with respect to vertices v and u); (b) the graph G_v; (c) the graph G_u.

3.3 Properties of the graph G_v

In this subsection, we prove some properties that a graph G_v corresponding to vertex v of G possesses.

It follows from the definition of the graph G_v and of the sets Z_i ($1 \le i \le \ell$), that the ordering $(Z_1, Z_2, \ldots, Z_\ell)$ contains the sets ordered by non-decreasing number of common neighbors with v. Then, the following lemma holds (the lemma holds for the ordering $\mathcal{S}_v$ of the given graph g as well):

Lemma 3.1 *Let v be a vertex of a graph G and G_v the corresponding graph. Consider the partition of the non-neighbors of vertex v in G_v based on their common neighbors with v and let $(Z_1, Z_2, \ldots, Z_\ell)$ be the ordering of the partition sets ordered by non-decreasing number of common neighbors with v (ties are broken arbitrarily). Then, for any two sets Z_i and Z_j where $i < j$, the following hold:*

(i) There exists a neighbor y of v such that y is adjacent to all the vertices in Z_j and to no vertex in Z_i.

(ii) Let $z \in Z_i$ and $z' \in Z_j$ such that $zz' \in E(G_v)$. If z has a common neighbor with v that is not a neighbor of z', then the graph G_v contains an induced house (with v at its top) or C_5.

Proof: (i) Let n_i and n_j be the number of common neighbors of the vertices in Z_i and in Z_j, respectively, with v. The ordering of the partition sets implies that $n_i \le n_j$, since $i < j$. If $n_i < n_j$, then clearly there exists a vertex y as described in the statement of the lemma. Such a vertex also exists if $n_i = n_j$, because the sets Z_i, Z_j are different and hence their sets of common neighbors with v differ.

(ii) Since $i < j$, statement (i) of this lemma holds and hence there exists a neighbor y of v such that y is adjacent to z' but is not adjacent to z. Suppose that there exists a vertex x that is a neighbor of both z and v but not of z'. Then, the vertices v, x, z, z', y induce a house (with v at its top) if $xy \in E(G_v)$ or a C_5 if $xy \notin E(G_v)$. ∎

Additionally, the existence of chordless cycles in the graph G_v for some vertex v of the given graph G implies that G contains a chordless cycle of at least equal length. In particular:

Lemma 3.2 *Let G be a graph. Consider the graph G_v as defined above w.r.t. a vertex v of G and let $O_{G_v} = z_{A_1} z_{A_2} \cdots z_{A_k}$ ($k \geq 3$) be a* chordless *cycle in G_v. If z_{A_i} is v or a neighbor of v in G_v, let $A_i = \{z_{A_i}\}$, whereas if z_{A_i} is a non-neighbor of v, let A_i be the connected component of a subgraph $G[S_{t_i}]$ of G which was shrunk into vertex z_{A_i}. Then, G contains a* chordless *cycle*

$$O_G \;=\; x_{1,1} \cdots x_{1,j_1} \; x_{2,1} \cdots x_{2,j_2} \quad \cdots \quad x_{k-1,1} \cdots x_{k-1,j_{k-1}} \; x_{k,1} \cdots x_{k,j_k},$$

where $\forall\, i = 1, \ldots, k$, $x_{i,1} \cdots x_{i,j_i}$ is a (chordless) path in $G[A_i]$ and $j_i \geq 1$; the length of O_G is equal to $\left(\sum_{i=1}^{k} j_i\right)$ which is no less than the length of the cycle O_{G_v} $(= k)$.

Proof: Since $z_{A_i} z_{A_{i+1}} \in E(G_v)$ $(1 \leq i < k)$ there exist vertices $q_i \in A_i$ and $p_{i+1} \in A_{i+1}$ such that $q_i p_{i+1} \in E(G)$ (note that if z_{A_i} is v or a neighbor of v then $q_i = z_{A_i}$; similarly for p_{i+1}). Then, for $1 \leq i \leq k$, since $p_i, q_i \in A_i$ and since the subgraph $G[A_i]$ is connected (no matter whether z_{A_i} is a neighbor of v or not), there exists a (possibly trivial) path, say, ρ_i, in $G[A_i]$ from p_i to q_i. Therefore, the vertices of the paths $\rho_1, \rho_2, \ldots, \rho_k$ in order form a (not necessarily chordless) cycle O'_G in G, from which we can obtain a chordless cycle O_G. Since the cycle O_{G_v} is chordless, no vertex in A_i $(1 \leq i \leq k)$ is adjacent to vertices of the cycle O'_G other than vertices in the sets A_j preceding and following the set A_i around the cycle. Therefore, the chordless cycle O_G has to pass from vertices in each of A_i and is of the form given in the statement on the lemma; consequently its length is at least equal to the length of the cycle O_{G_v}. ∎

The above property is crucial in showing the following very important result.

Corollary 3.1 *Let G be a graph and consider the graph G_v as defined above w.r.t. any vertex v of G. If G_v contains a house, hole, or domino then so does G.*

Proof: Lemma 3.2 readily implies that the existence of an induced hole in G_v implies the existence of an induced hole in G.

Now, consider that G_v contains a house induced by the vertices z_A, z_B, z_C, z_D, z_E so that $z_A z_B z_C z_D z_E$ is a cycle in G_v that has the chord $z_B z_E$, i.e., z_A is the top vertex of the house with neighbors and z_B and z_E. As in the statement of Lemma 3.2, let us associate the vertices z_A, z_B, z_C, z_D, z_E with the subsets A, B, C, D, E of $V(G)$, respectively, i.e., if z_X is a neighbor of v, then $X = \{z_X\} \subseteq V(G)$, otherwise, X is the connected component of G which was shrunk into vertex z_X (note that, in either case, $G[X]$ is connected). In accordance with Lemma 3.2, the existence of the chordless cycle $z_B z_C z_D z_E$ in G_v implies that G contains a chordless cycle O_G of length ≥ 4. If the length of O_G is ≥ 5 then G contains an induced hole; so, suppose that the length of the cycle O_G is 4. Then, as proved in Lemma 3.2, the cycle is $b'cde'$ where $b' \in B$, $c \in C$, $d \in D$, and $e' \in E$ (see Figure 5(a)). Similarly, the existence of the chordless cycle $z_A z_B z_E$ in G_v implies that G contains a chordless cycle O'_G of length ≥ 3. If the length of O'_G is ≥ 5 then G contains an induced hole; so, suppose that the length of the cycle O'_G is 3 or 4. In either case, the cycle contains an edge be where $b \in B$ and $e \in E$ (see Figure 5(a)).

Since $b, b' \in B$ and since $G[B]$ is connected, let $\rho_B = x_0 x_1 \cdots x_k$ be a (possibly trivial) shortest path in $G[B]$ connecting $b' = x_0$ to $b = x_k$. Similarly, let $\rho_E = y_0 y_1 \cdots y_\ell$ be a (possibly trivial) shortest path in $G[E]$ connecting $e' = y_0$ to $e = y_\ell$. Next, among the vertices of ρ_B that are adjacent

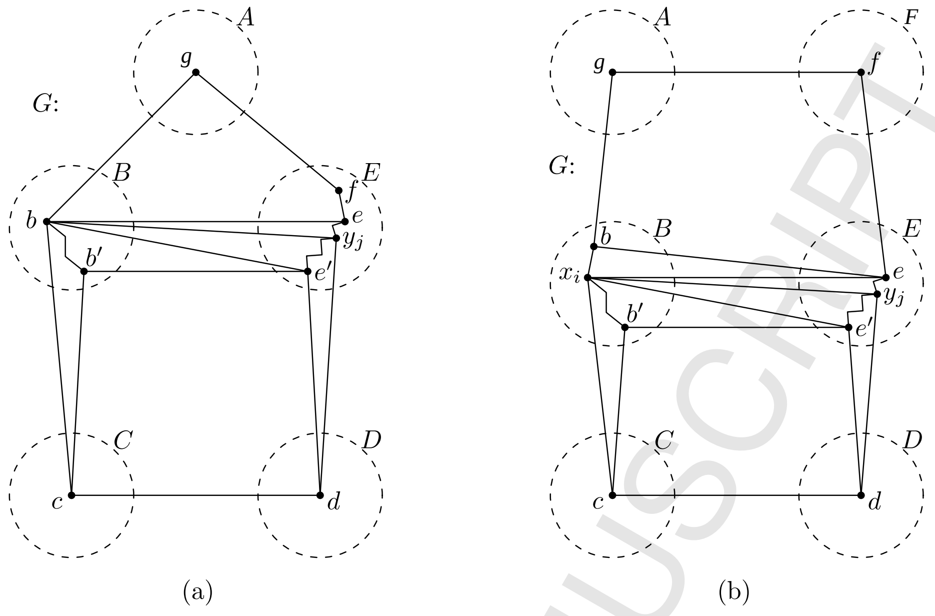

Figure 5: For the proof of Corollary 3.1; (a) for the case of a house (x_i coincides with b); (b) for the case of a domino.

to both c and e' (note that $b' = x_0$ is such a vertex), select the vertex that is closest to b along ρ_B; let x_i be that vertex (in Figure 5(a), $x_i = b$). Then, no vertex $x_{i+1}, \ldots, x_k$ is adjacent to both c and e'. If there exist vertices among $x_{i+1}, \ldots, x_k$ that are adjacent to c, let $x_{i'}$ be the vertex that is closest to x_i along ρ_B. Then: if $i' = i + 1$, then the vertices $v_i, v_{i'}, c, d, e'$ induce a house in G; if $i' = i + 2$, then the vertices $v_i, v_{i+1}, v_{i'}, c, d, e'$ induce a domino in G; finally, if $i' \geq i + 3$, then the vertices $v_i, v_{i+1}, \ldots, v_{i'}$ induce a hole in G. In a similar fashion, if there exist vertices among $x_{i+1}, \ldots, x_k$ that are adjacent to e', then G contains an induced house, hole, or domino. So, in the following, we assume that no vertex among $x_{i+1}, \ldots, x_k$ is adjacent to c or e'.

Let us now work similarly on the path ρ_E. More specifically, let y_j be the vertex of ρ_E that is adjacent to both x_i and d, and is closest to e along ρ_E (see Figure 5(a)). If any of the vertices $y_{j+1}, \ldots, y_\ell$ is adjacent to x_i or d, then, as in the previous paragraph, we conclude that the graph G contains an induced house, hole, or domino. So, in the following, we assume that no vertex among $y_{j+1}, \ldots, y_\ell$ is adjacent to x_i or d.

If $x_i = b$ and $y_j = e$, then the subgraph of G induced by the vertices c, d, and the vertices of the cycle O'_G (of length 3 or 4) contains an induced house or domino: If O'_G is of length 3, then c, d, and the vertices of O'_G induce a house. Next, suppose that O'_G is of length 4; in particular, let $O'_G = befg$. If $f, g \in A$, then c, d, and the vertices of O'_G induce a domino. If $f \in E$, then $g \in A$. If vertex f is adjacent to both b and d, then g, b, c, d, f induce a house (with g at its top). If f is adjacent to exactly one of b and d, then f, b, c, d, e induce a house (with f at its top). If f is not adjacent to b or d, then b, c, d, e, f, g induce a domino. Similarly if $g \in B$.

Now, if $x_i \neq b$ or $y_j \neq e$, consider the shortest path connecting x_i to y_j in the subgraph of G induced by the vertices $x_i, x_{i+1}, \ldots, x_{k-1}, b, e, y_{\ell-1}, \ldots, y_{j+1}, y_j$ in which we have removed the edge $x_i y_j$. If the shortest path is of length ≥ 4, then its vertices induce a hole in G. If its length is 2 or 3, then its vertices along with c and d induce a house or a domino, respectively.

The proof for a domino is similar to that for the house. Suppose that G_v contains a domino induced by the vertices $z_A, z_B, z_C, z_D, z_E, z_F$ so that $z_A z_B z_C z_D z_E z_F$ is a cycle in G_v that has the chord $z_B z_E$. As in the case of the house, we associate the vertices $z_A, z_B, z_C, z_D, z_E, z_F$ with the subsets A, B, C, D, E, F of $V(G)$, respectively (see Figure 5(b)). In accordance with Lemma 3.2, the existence of the chordless cycles $z_B z_C z_D z_E$ and $z_A z_B z_E z_F$ in G_v implies that G contains corresponding chordless cycles O_G and O'_G, respectively, each of length ≥ 4. If the length of any of these cycles is ≥ 5 then G contains an induced hole; so, suppose that the length of both O_G and O'_G is 4 and let them be $O_G = b'cde'$ and $O'_G = befg$ where $g \in A$, $b, b' \in B$, $c \in C$, $d \in D$, $e, e' \in E$, and $f \in F$ (see Figure 5(b)). From then on, the proof proceeds in the same way as in the case of the house under the conditions that the length of O'_G is 4, and that f belongs to a separate set F and thus is not adjacent to b or d. ∎

On the other hand, due to the way the graph G_v is defined, if v is a (in some cases, special) vertex in a house, hole, or domino in G then the graph G_v contains a house, hole, or domino. More specifically:

Lemma 3.3 *If a graph G contains an induced house, hole, or domino then there exists a vertex $v \in V(G)$ such that in the graph G_v (as defined in Subsection 3.2) w.r.t. vertex v:*

(i) v is the top vertex of an induced house in G_v or

(ii) v belongs to an induced hole in G_v or

(iii) v is a corner vertex of an induced domino in G_v.

Proof: We assume that G contains an induced house, hole, or domino. We consider the following cases:

(i) *Suppose that G contains an induced house*: Let v be the vertex at the top of the house and let the remaining vertices be a, b, c, d such that $vabcd$ is a cycle of G. For the non-neighbors b and c of v, let $b \in S_i$ and $c \in S_j$. Clearly $i \neq j$ since b and c have different common neighbors with v. Consider the graph G_v w.r.t. vertex v. Let z_B (z_C, resp.) be the vertex of G_v which resulted from the shrinking of the connected component B (C, resp.) of $G[S_i]$ ($G[S_j]$, resp.) containing b (c, resp.). Then, the vertices v, a, z_B, z_C, d induce a house in G_v having v as its top vertex.

(ii) *Suppose that G contains an induced hole*: Let v be any vertex of the hole and let the hole be $vabp_1p_2 \cdots p_k cd$, i.e., the vertices $b, p_1, \ldots, p_k, c$ are all non-neighbors of v in G. Consider the graph G_v w.r.t. vertex v. The path $bp_1 \cdots p_k c$ in the subgraph of G induced by the non-neighbors of v implies that there exists a non-simple path in the subgraph of G_v induced by the non-neighbors of v connecting z_B to z_C, where z_B, z_C are the vertices of G_v that resulted from the shrinking of the connected components containing b and c, respectively. From this path, we can obtain a chordless path ρ connecting z_B to z_C (it suffices to apply breadth-first-search on this path starting from z_B until z_c is reached, and then collect the BFS-tree edges from z_B to z_C). Then, the vertices v, a, d and the vertices of the path ρ induce a hole in G_v containing v (even when the length of ρ is equal to 1).

(iii) *Suppose finally that G contains an induced domino*: Let v be a corner vertex of such a domino and let $vabcde$ be the Hamilton cycle of the domino. Consider the graph G_v w.r.t. vertex v. Let z_B, z_C, z_D be the vertices of G_v corresponding to the connected components B, C, and D, respectively, of G containing b, c, and d, respectively; z_B, z_C, z_D are different as b, c, d have different common neighbors with v. Due to the domino, the graph G_v contains the cycle $vaz_Bz_Cz_De$. Next, if there exist vertices $b' \in B$ and $d' \in D$ such that $b'd' \in E(G)$, then the vertices b', d', e, v, a induce a house (with b' at its top) in G, and according to case (i) the graph $G_{b'}$ corresponding to b' would contain an induced house with b' at its top. Suppose now that no vertex in B is adjacent to a vertex in D. Then, $z_Bz_D \notin G_v$ and the vertices v, a, z_B, z_C, z_D, e induce a domino in G_v having v as a corner vertex. ∎

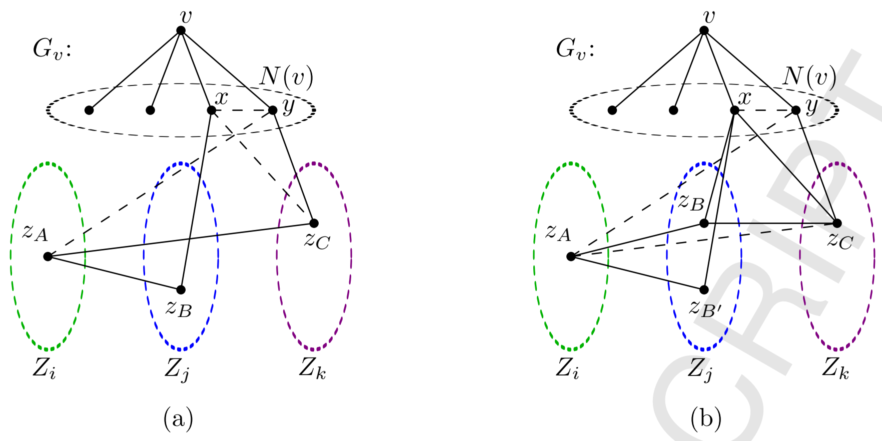

Figure 6: For the proofs of: (a) Lemmas 3.4 and 3.6; (b) Lemmas 3.5 and 3.7 (edges that may or may not exist are shown dashed).

The above result readily implies a way to check if an input graph G is HHD-free: process the graph G_v for each vertex v of G checking whether v is the top vertex of a house, or belongs to a hole, or is a corner vertex of a domino in G_v; if it is/does so, then G is not HHD-free. Of course, if there exists another induced house, hole, or domino in G_v, then, according to Corollary 3.1, again G is not HHD-free.

In this way, we reduced the problem of deciding whether a graph G is HHD-free into the problem of deciding whether any of the graphs G_v for the vertices of G is HHD-free. It seems that we are back at the beginning. However, this is not true because a graph G_v has the important property that if G is HHD-free then its subgraph induced by the non-neighbors of v is "nearly" chordal, or the contrapositive: if its subgraph induced by the non-neighbors of v is not "nearly" chordal then G is not HHD-free; the latter property is proved in the next lemma and is crucial in achieving a faster algorithm.

Lemma 3.4 *Let v be a vertex of a graph G and z_A, z_B, z_C three vertices of the graph G_v belonging to the partition sets Z_i, Z_j, Z_k, respectively, and assume that $i < j < k$. Suppose further that $z_A z_B \in E(G_v)$ and $z_A z_C \in E(G_v)$. Then if $z_B z_C \notin E(G_v)$, the graph G contains an induced house, hole, or domino.*

Proof: Since $i < j$, in accordance with Lemma 3.1 (statement (i)), there exists a vertex $x \in N(v)$ such that $x z_B \in E(G_v)$ and $x z_A \notin E(G_v)$. Similarly, since $j < k$, there exists a vertex $y \in N(v)$ such that $y z_C \in E(G_v)$ and $y z_B \notin E(G_v)$; see Figure 6(a).

Now, if $y z_A \in E(G)$ then the vertices v, x, y, z_A, z_B induce a house (with v at its top) or a C_5 in G_v depending on whether $xy \in E(G_v)$ or not, respectively. So suppose next that $y z_A \notin E(G_v)$. If $x z_C \in E(G)$ then the vertices x, z_B, z_A, z_C induce a C_4, and thus G_v contains a house induced by x, y, z_A, z_B, z_C (with y as its top vertex) if $xy \in E(G_v)$ or a domino induced by v, x, y, z_A, z_B, z_C otherwise; if $x z_C \notin E(G)$ then G_v contains a C_5 induced by x, y, z_A, z_B, z_C if $xy \in E(G_v)$ or a C_6 induced by v, x, y, z_A, z_B, z_C otherwise.

In all cases, the graph G_v contains an induced house, hole or domino; then, by Corollary 3.1, G contains an induced house, hole, or domino. ∎

Therefore, in the case in which each non-neighbor of v, belonging to a set Z_i, is adjacent to at most 1 element of Z_j with $j > i$, this lemma has the following interesting implication: if the subgraph of G_v induced by the non-neighbors of v is not chordal then the graph G is not HHD-free. Then, we could apply a (linear-time) chordal recognition algorithm in that subgraph. If the algorithm returned "no", that is, the subgraph was not chordal, then we could safely answer that G is not HHD-free. If the algorithm returned "yes", then we could apply the algorithm of [12] without having to execute the chordal completion step, or the algorithm of [18] without having to compute the array NextNeighbor, and we could answer whether G contains an induced house or building (see Figure 1(b)) with v at its top. If any graph G_v contained a house or a building, we could correctly report that G was not HHD-free; if none of them did, then G would not contain an induced house or hole. Then, all that would be needed, would be to test for an induced domino. By avoiding the chordal completion and the computation of the array NextNeighbor, we would have reduced the time complexity of the overall algorithm, as all the other steps in the algorithms of [12] and [18] can be done in $O(nm)$ time.

The above method relies on the assumption that no non-neighbor of v in a set, say, Z_i, is adjacent to more than 1 element of sets Z_j with $j > i$, which is not true in general. Nevertheless, even if this assumption is not true, we can still detect houses, holes, or dominoes, if there exist in G_v, by taking advantage of the following property.

Lemma 3.5 *Let v be a vertex of a graph G and z_A, z_B, z_C three vertices of the graph G_v belonging to the partition sets Z_i, Z_j, Z_k, respectively, where $i < j < k$, such that $z_A z_B \in E(G_v)$ and $z_B z_C \in E(G_v)$. Suppose further that there exists a vertex $z_{B'} \in Z_j$ such that $z_A z_{B'} \in E(G_v)$ and $z_{B'} z_C \notin E(G_v)$. Then, the graph G contains an induced house, hole, or domino.*

Proof: Since $i < j$, in accordance with Lemma 3.1 (statement (i)), there exists a vertex $x \in N(v)$ such that $x z_B \in E(G_v)$, $x z_{B'} \in E(G_v)$, and $x z_A \notin E(G_v)$. Additionally, in accordance with Lemma 3.1 (statement (ii)) due to the edges $z_B z_C$ and $x z_B$, if $x \notin N_{G_v}(z_c)$ then G_v contains an induced house or C_5. So, next consider that $x z_C \in E(G_v)$; see Figure 6(b).

If $z_A z_C \in E(G)$ then the conditions of Lemma 3.4 for z_A, $z_{B'}$, and z_C would be met and then the graph G_v would contain an induced house, hole, or domino. If $z_A z_C \notin E(G)$, the vertices $x, z_A, z_B, z_{B'}, z_C$ induce a house with z_C at its top; note that $z_B z_{B'} \notin E(G_v)$ since the vertices $z_B, z_{B'}$ resulted from shrinking different connected components of the subgraph $G[S_j]$.

In all cases, the graph G_v contains an induced house, hole or domino; then, by Corollary 3.1, G contains an induced house, hole, or domino. ∎

The lemma implies that in G_v whenever a non-neighbor z of v with $z \in Z_i$ is adjacent to two other non-neighbors z' and z'' belonging to the same set, say, Z_j with $j > i$, then we check whether z' and z'' have the same neighbors "to the right"; if they don't, then the graph G is not HHD-free.

3.4 Description of the Algorithm

Based on the results in Lemmas 3.4 and 3.5, in Figure 7 we give a detailed description of Procedure Process_Vertex with parameters the input graph G and one of its vertices v. From G, we construct the auxiliary graph G_v by shrinking each of the connected components of each of the subgraphs $G[S_i]$, $i = 1, 2, \ldots, \ell$, into a single vertex; the components in $G[S_i]$ yield the vertices in the set Z_i (Step 2). Then, for each $i = 1, 2, \ldots, \ell$, we process all the vertices in the set Z_i together (note that, by construction, in the graph G_v there are no edges between any two vertices in Z_i).

For any two vertices z', z'' in each set Z_i that are neighbors of a vertex in some set Z_t with $t < i$, we intend to check that z' and z'' have the same neighbors "to the right" and if not, to report that G is not HHD-free in accordance with Lemma 3.5. This checking is delayed until the processing of the set Z_i and thus, we construct a partition $\mathcal{P}_{Z_i}$ of each set Z_i into sets of vertices that should have the same neighbors "to the right" if G is HHD-free: Initially, each vertex belongs to a separate partition set (Step 3). Whenever, a non-neighbor z of v in G_v has more than one *immediately next* neighbors (i.e., neighbors in the minimum-index set, say, Z_k, among all the sets containing neighbors of z "to the right"), then the partition sets containing these neighbors are unioned (Step 4.3); in fact, checking only the immediately next neighbors of a *representative* of each partition set, instead of each vertex z, suffices since this is done after all the vertices in a partition set have been found to have the same neighbors "to the right" (Step 4.2).

The rest of the algorithm is an extension of the linear-time perfect-elimination-ordering test algorithm of Rose, Tarjan, and Lueker [21, 7] applied on the graph G_v. With each non-neighbor z of vertex v in the graph G_v, we associate an (initially empty) set $A(z)$ (Step 3) which collects all the vertices that will eventually have to be checked for adjacency with z. This checking is also delayed until the partition set, say, P_j, containing z gets processed; in fact, the elements of $A(z)$ are checked for adjacency with a *representative* of the set P_j (Step 4.4), which is correct since all the vertices in P_j have been checked to have the same neighbors "to the right" (and of course they have the same common neighbors with v).

For example, when applied on the graph G_v in Figure 4(b), we have: the set Z_1 contains a single vertex z_1, whose processing in Step 4.3 produces $W = \{z_2, z_3\}$ and thus these two vertices are unioned in a single partition set; when Z_2 gets processed, the partition set $\{z_2, z_3\}$ is processed and in Step 4.2, we find out that the two vertices do not have the same neighbors "to the right" (no matter which one is picked as the representative z_R), and the algorithm correctly concludes that the given graph G is not HHD-free. If the algorithm is applied on the graph G_u in Figure 4(c), then we have: the set Z_1 contains a single vertex z'_1, whose processing in Step 4.3 produces $W = \{z'_3\}$ and $X' = \{z'_3, z'_4\}$, and thus the set $A(z'_3)$ becomes $\{z'_4\}$; next, during the processing of Z_2 and its element z'_2, we have that $W = \{z'_3\}$ and $X' = \{z'_3, v\}$, and thus the vertex v is added to $A(z'_3)$; finally, when Z_3 gets processed, in Step 4.4 of the processing of z'_3, we find that $A(z'_3)$ cobtains v and z'_4, none of which is a neighbor of z'_3, and thus the algorithm correctly concludes that the given graph G is not HHD-free.

3.5 Correctness

The correctness of Procedure Process_Vertex, and consequently Algorithm Recognize-HHD-free, is established in Theorem 3.1 with the help of Lemmas 3.6-3.8 through which we establish that Procedure Process_Vertex reports that the given graph is not HHD-free under certain conditions that are often met in Theorem 3.1. In particular, these 3 lemmas prove the afore mentioned result for Procedure Process_Vertex if the conditions of Lemmas 3.4, 3.5, and 3.1(ii) respectively, hold.

Lemma 3.6 *Let G_v be the graph corresponding to a vertex v of a graph G as defined in Section 3.2 and suppose that there exist $z_A \in Z_i$, $z_B \in Z_j$, and $z_C \in Z_k$ with $i < j < k$ such that $z_A z_B \in E(G_v)$, $z_B z_C \in E(G_v)$, and $z_B z_C \notin E(G_v)$; see Figure 6(a). Then Procedure Process_Vertex, when run on the pair (G, v), reports that the graph G is not HHD-free.*

Proof: Among all triples of vertices z_A, z_B, z_C meeting the conditions of the lemma, consider a triple such that $j - i$ is minimum.

Consider the processing of the graph G_v and in particular the processing of the partition set containing z_A. Clearly, $X' \neq \emptyset$, since $z_B, z_C \in X'$; let z_X be the vertex chosen at the end of Step 4.3 in order that the set $X - W$ be concatenated to $A(z_X)$. Then, z_X belongs to Z_j (as does z_B). Suppose for contradiction that z_X belongs to Z_q where $q < j$. Then $z_X z_B \in E(G_v)$ otherwise the triple z_A, z_X, z_B would contradict the minimality of the triple z_A, z_B, z_C. Similarly, $z_X z_C \in E(G_v)$. But then again, the triple z_X, z_B, z_C contradicts the minimality of the triple z_A, z_B, z_C. Therefore, $z_X \in Z_j$ (as does z_B) and the vertices z_X and z_B both end up belonging to the set W in Step 4.3 during the processing of the partition set containing z_A; due to that, the partition sets to which z_X and z_B belong get unioned (if they have not got unioned earlier). Later the vertices in $X - W$ are concatenated to $A(z_X)$ and thus z_C is concatenated to $A(z_X)$.

Consider now the processing of the partition set containing z_X (and z_B) and let z_R be the arbitrary vertex selected in Step 4.1. If $z_C \in N_{G_v}(z_R)$ then Procedure Process_Vertex reports that the graph G is not HHD-free in Step 4.2 because z_B is not adjacent to z_C. If $z_C \notin N_{G_v}(z_R)$ then Procedure

Procedure Process_Vertex(graph G, vertex v)

```
1. Compute the set $N(v)$ of neighbors and the set of non-neighbors of $v$ in $G$;
   partition the non-neighbors of $v$ based on their common neighbors with $v$ in $G$, and order the
   partition sets by non-decreasing number of such common neighbors; let $\mathcal{S}_v = (S_1, S_2, \ldots, S_\ell)$ be
   the resulting ordering;

2. Construct the auxiliary graph $G_v$ from a copy of $G$ by shrinking each connected component $C$
   of the subgraphs $G[S_i]$, $i = 1, 2, \ldots, \ell$, into a single vertex $z_C$; let $Z_i$ $(i = 1, 2, \ldots, \ell)$ be the set
   of vertices of $G_v$ obtained from shrinking the connected components of $G[S_i]$;

3. for $i \leftarrow 1$ to $\ell$ do
      form a partition $\mathcal{P}_{Z_i}$ of the set $Z_i$ by placing each element of $Z_i$ in a separate partition set;
      for each vertex $z_C \in Z_i$ do
         associate with $z_C$ an initially empty set $A(z_C)$;

4. for $i \leftarrow 1$ to $\ell$ do
      let the partition $\mathcal{P}_{Z_i}$ of $Z_i$ be $\mathcal{P}_{Z_i} = \{P_1, P_2, \ldots, P_t\}$;
      for $j \leftarrow 1$ to $t$ do
      4.1 let $z_R$ be any vertex contained in the set $P_j$;     {a representative of $P_j$}
          $X' \leftarrow N_{G_v}(z_R) \cap \left(\bigcup_{r=i+1}^{\ell} Z_r\right)$;     {neighbors of $z_R$ “to the right”}
      4.2 if $P_j$ is not a singleton set
          then   {check that all vertices in $P_j$ have the same neighbors “to the right”}
                if there exists a vertex in $P_j$ that is not adjacent in $G_v$ to a vertex in $X'$ or
                   is adjacent to a vertex in $\left(\bigcup_{r=i+1}^{\ell} Z_r\right) - X'$
                then print(“The graph $G$ is not HHD-free”); exit;
      4.3 if $X' \neq \emptyset$
          then let $Z_k$ be the minimum-index set such that $X' \cap Z_k \neq \emptyset$;
                $W \leftarrow X' \cap Z_k$;     {closest neighbors of $z_R$ “to the right”}
                if $|W| > 1$
                then union the sets of the partition $\mathcal{P}_{Z_k}$ (of $Z_k$) containing the vertices in $W$;
                $X \leftarrow X' \cup \left(N_{G_v}(z_R) \cap N(v)\right)$;     {include neighbors in $N(v)$}
                choose any $z_X \in W$ and concatenate the set $X - W$ to $A(z_X)$;
      4.4 if $\left(\bigcup_{z \in P_j} A(z)\right) - N_{G_v}(z_R) \neq \emptyset$
          then print(“The graph $G$ is not HHD-free”); exit;
```

Figure 7: Procedure Process_Vertex.

Process_Vertex again reports that the graph G is not HHD-free; in Step 4.4, the set $\bigcup_{z \in P_s} A(z)$ includes $A(z_X)$ which contains z_C whereas z_C is not adjacent to z_R. ∎

Lemma 3.7 *Let G_v be the graph corresponding to a vertex v of a graph G as defined in Section 3.2 and suppose that there exist $z_A \in Z_i$, $z_B, z_{B'} \in Z_j$, and $z_C \in Z_k$ with $i < j < k$ such that $z_A z_B \in E(G_v)$, $z_A z_{B'} \in E(G_v)$, $z_B z_C \in E(G_v)$, and $z_{B'} z_C \notin E(G_v)$; see Figure 6(b). Then, Procedure Process_Vertex, when run on the pair (G, v), reports that the graph G is not HHD-free.*

Proof: Fix vertices z_B and $z_{B'}$; among all possible vertices z_A as described in the lemma, select a z_A such that $j - i$ is minimum.

Consider the processing of the graph G_v and in particular the processing of the partition set containing z_A. Clearly, $X' \neq \emptyset$, since $z_B, z_{B'} \in X'$; let z_X be the vertex chosen at the end of Step 4.3

in order that the set $X - W$ be concatenated to $A(z_X)$. Suppose that $z_X \in Z_q$ where $q < j$. The minimality of the choice of z_A implies that z_X is not adjacent to at least one of z_B and $z_{B'}$ thus, suppose without loss of generality that $z_X z_B \notin E(G_v)$. Then, the triple z_A, z_X, z_B would satisfy the conditions of Lemma 3.6 and Procedure Process_Vertex reports that the graph G is not HHD-free. Now suppose that $z_X \in Z_j$ (as do z_B and $z_{B'}$). Therefore, all three vertices z_X, z_B, and $z_{B'}$ belong to the set W in Step 4.3 during the processing of the partition set containing z_A, and the partition sets to which they belong get unioned (if they have not got unioned earlier).

Consider now the processing of the partition set of Z_j containing z_X, z_B, and $z_{B'}$, and let z_R be the arbitrary vertex selected in Step 4.1. Clearly, the partition set is not a singleton and in Step 4.2 it is checked whether any member of the set has different neighborhood "to the right" from that of z_R. Since $z_C \in Z_k$ (where $k > j$) is adjacent to z_B but not to $z_{B'}$, and both z_B and $z_{B'}$ belong to the partition set containing z_R, the neighborhood "to the right" of z_R will disagree to at least one of the neighborhoods of z_B or $z_{B'}$, and therefore Procedure Process_Vertex reports that the graph G is not HHD-free. ∎

Lemma 3.8 *Let G_v be the graph corresponding to a vertex v of a graph G as defined in Section 3.2 and suppose that $z_A z_B \in E(G_v)$ where $z_A \in Z_i$ and $z_B \in Z_j$ are two non-neighbors of v in G_v with $i < j$. If there exists a vertex x that is a common neighbor of z_A and v in G_v, which is not a neighbor of z_B (see the left graph in Figure 8(b)), then Procedure Process_Vertex when run on the pair (G, v) reports that the graph G is not HHD-free.*

Proof: Among all pairs of vertices z_A, z_B meeting the conditions of the lemma, consider a pair such that $j - i$ is minimum.

Consider the processing of the graph G_v and in particular the processing of the partition set containing z_A; let z_X be the vertex chosen at the end of Step 4.3 in order that the set $X - W$ be concatenated to $A(z_X)$. Then, since $x \in N_{G_v}(z_A) \cap N(v)$, the vertex x will be concatenated to $A(z_X)$. Suppose that z_X belongs to Z_k; clearly, $i < k \leq j$.

Consider the case where $k < j$. Since $z_A z_X \in E(G_v)$, if $x \notin N_{G_v}(z_X)$. then the pair z_X, z_B would contradict the minimality of the pair z_A, z_B. Thus $x \in N_{G_v}(z_X)$; then $z_X z_B \notin E(G_v)$ otherwise the pair z_X, z_B would contradict the minimality of the pair z_A, z_B. Since $z_A z_X \in E(G_v)$, $z_A z_B \in E(G_v)$, and $z_X z_B \notin E(G_v)$, the vertices z_A, z_X, z_B satisfy the conditions of Lemma 3.6 and we have that Procedure Process_Vertex reports that the graph G is not HHD-free.

Consider now the case where $k = j$. Since $k = j$ and since both z_X and z_B are neighbors of z_A, the vertices z_X and z_B both belong to the set W computed in Step 4.3 while processing the partition set containing z_A, and hence the partition sets containing these vertices (if different) are unioned. During the processing of the partition set P_s containing z_X and z_B, in Step 4.1 an arbitrary vertex z_R is selected from P_s; since $z_R \in Z_j$ (as does z_B), $x \notin N_{G_v}(z_R)$ as well. Next, in Step 4.2, the set $\bigcup_{z \in P_s} A(z)$ includes $A(z_X)$ which contains x. On the other hand, x is not adjacent to z_R and Procedure Process_Vertex reports that the graph G is not HHD-free. ∎

Now we are ready to prove the theorem establishing the correctness of our algorithm.

Theorem 3.1 *When Algorithm Recognize-HHD-free is run on a graph G, it reports that G is not HHD-free if and only if G is indeed not HHD-free.*

Proof: ($\Longrightarrow$) Suppose that the algorithm prints that G is not HHD-free while Procedure Process_Vertex runs on the pair (G, v). This may happen either in Step 4.2 or in Step 4.4 while processing a partition set P_h of a set Z_j of vertices of G_v; let z_R be the vertex selected in Step 4.1 of the processing of P_h. Then:

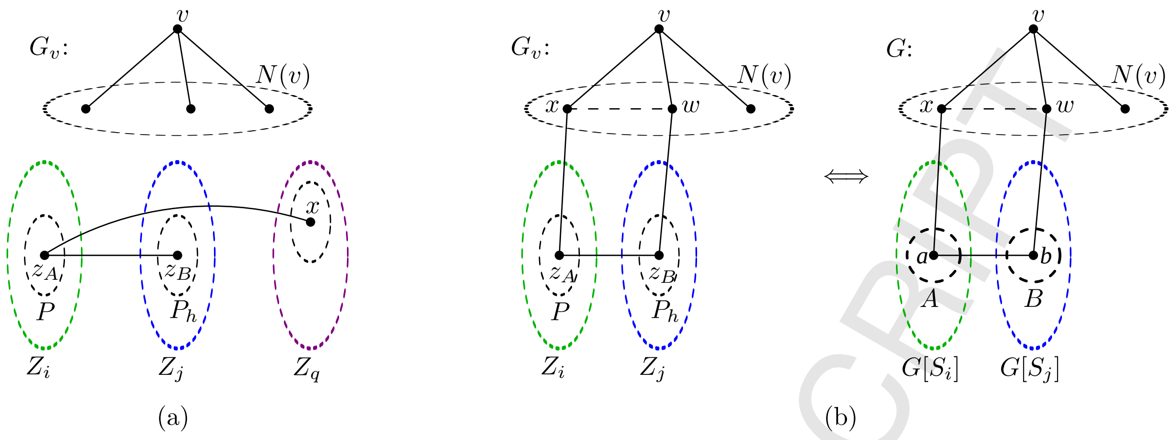

Figure 8: For the proof of Theorem 3.1.

- Step 4.2: there exists a vertex in P_h whose neighborhood "to the right" differs from that of vertex z_R, i.e., there exists a vertex $z_C \in \left(\bigcup_{r=i+1}^{\ell} Z_r\right) - X'$ such that some vertices in P_j are adjacent to z in the graph G_v while the remaining vertices in P_h are not adjacent to z.
  The set P_h has been formed by unions of disjoint subsets of the set Z_j; a union of such subsets is done in Step 4.4 during the processing of a partition set which is a subset of a set Z_i (with $i < j$) and contains a vertex adjacent in G_v to elements of these subsets that belong to the set W. Thus, in the history of the formation of the set P_h, there is a moment when a subset of P_h is formed while unioning a subset P_{q_1} of vertices of Z_j all of which are adjacent to z_C with a subset P_{q_2} of vertices of Z_j none of which is adjacent to z_C. This union has been performed at an earlier iteration of the for-loop in Step 4, i.e while processing a subset of a set Z_i with $i < j$ because the vertex selected in Step 1.4, say, z_A, is adjacent to a vertex z_B in P_{q_1} and to a vertex $z_{B'}$ in P_{q_2}. But then the vertices $z_A, z_B, z_{B'}, z_C$ meet the conditions of Lemma 3.5, and thus G contains an induced house, hole, or domino.

- Step 4.4: there exists a vertex $x \in \left(\bigcup_{z \in P_h} A(z)\right) - N_{G_v}(z_R)$, where z_R is the vertex selected from the set P_h in Step 4.1 when processing P_h. The vertex x belongs to $\bigcup_{z \in P_h} A(z)$ because it has been added to some $A(z_B)$, where $z_B \in P_h$; this addition was done earlier in Step 4 for a value i of the index of the for-loop (i.e. $i < j$) while processing a subset P of the set $Z_{i'}$ of vertices of G_v; let z_A be the vertex selected from P in Step 4.1 during its processing. Then, $z_A z_B \in E(G_v)$ and $z_A x \in E(G_v)$. Since $\bigcup_{z \in P_h} A(z) \subseteq \left(\bigcup_{r=i+1}^{\ell} Z_r\right) \cup N(v)$, we distinguish the following two cases:
  Suppose that $x \in \bigcup_{r=i+1}^{\ell} Z_r$: Let $x \in Z_q$ where $q > j$ (see Figure 8(a)). Since the processing of the set P_h did not stop at Step 4.2, then all the vertices in P_h have the same neighbors "to the right" as z_R; since x is not a neighbor of z_R, then x is not a neighbor of z_B either (recall $z_R, z_B \in P_h$). But then the vertices z_A, z_B, x meet the conditions of Lemma 3.4, and thus G contains an induced house, hole, or domino.
  Next, suppose that $x \in N(v)$ (see Figure 8(b)). Since $z_A \in Z_i$ and $z_B \in Z_j$, there exist connected components A of $G[S_i]$ and B of $G[S_j]$ such that A, B were shrunk into vertices z_A and z_B, respectively. Since $x z_A \in E(G_v)$ and $x z_B \notin E(G_v)$, in G, vertex x is adjacent to all the vertices in A and to no vertex in B. Additionally, because $z_A z_B \in E(G_v)$, there exist vertices $a \in A$ and $b \in B$ such that $ab \in E(G)$. Moreover, since $i < j$, there exists a vertex $w \in N(v)$ such that $w \in N_G(b) - N_G(a)$. Then, the vertices v, x, w, a, b induce a house or a C_5 in G.

($\Longleftarrow$) Now, suppose that the graph G is not HHD-free; we will show that Algorithm Recognize-HHD-free will report that.

Suppose that G contains a house or a C_5 induced by the vertices v, x, w, a, b that form a cycle $vxabw$; in case of a house, let v be its top vertex (see Figure 8(b)). Without loss of generality suppose that in the partition of the non-neighbors of v, the vertices a, b belong to the connected components A and B of the sets S_i and S_j, respectively, with $i < j$. Consider the graph G_v and let z_A, z_B be the vertices of G_v to which the components A, B shrunk. Then, in G_v, $z_A \in Z_i$, $z_B \in Z_j$ with $i < j$, and the neighbor x of v is adjacent to z_A but not to z_B. Then, from Lemma 3.8, we conclude that Procedure Process_Vertex when run on the pair (G, v) reports that the graph G is not HHD-free.

Suppose that G contains a hole $vua_1a_2 \cdots a_h w$, where $h \geq 3$. Again, if Algorithm Recognize-HHD-free does not stop early (and reports that the graph G is not HHD-free), it eventually executes the body of the for-loop of Step 1 for the vertex v. Note that the vertices $a_1, a_2, \ldots, a_h$ are all non-neighbors of v; let $A_1, A_2, \ldots, A_h$, respectively, be the connected components of the subgraphs of G induced by the sets of the partition $\mathcal{S}_v$ to which the vertices $a_1, a_2, \ldots, a_h$ belong (note that the components A_1 and A_h differ from each other and from all other components, whereas $A_2, A_3, \ldots, A_{h-1}$ are not necessarily distinct). Then, the graph G_v contains vertices $z_{A_1}, z_{A_2}, \ldots, z_{A_h}$ corresponding to the above connected components, and because G contains the path $a_1a_2 \cdots a_h$, the subgraph $G_v[\{z_{A_1}, z_{A_2}, \ldots, z_{A_h}\}]$ is connected; let ρ be a chordless path in this subgraph from z_{A_1} to z_{A_h}. If ρ is of length 1, then as in the case of a house, we show that Procedure Process_Vertex when run on (G, a_1) reports that the graph G is not HHD-free. So, let $\rho = z_{A_1} z_{B_2} \cdots z_{A_h}$. Further, without loss of generality suppose that $z_{A_1} \in Z_i$ and $z_{A_h} \in Z_j$ where $i < j$. Then, if $z_{B_2} \in Z_p$ with $p > i$, Lemma 3.8 implies that Procedure Process_Vertex when run on (G, v) reports that the graph G is not HHD-free because of the edge $z_{A_1} z_{B_2}$ and the fact that u is adjacent to z_{A_1} but not to z_{B_2}. So, suppose that $p < i$, that is, walking from z_{A_1} to z_{B_2} we walk towards the left in the ordered set of non-neighbors of v (based on the number of their common neighbors with v). From z_{A_1} keep walking along the path ρ for as long as you go left. Since z_{A_h} belongs to Z_j with $j > i$, eventually we will find a vertex z_{B_t} such that both $z_{B_{t-1}}$ (which may be z_{A_1}) and $z_{B_{t+1}}$ are to the right of z_{B_t}. If $z_{B_{t-1}}$ and $z_{B_{t+1}}$ do not belong to the same set Z_q, then the vertices z_{B_t}, $z_{B_{t-1}}$, and $z_{B_{t+1}}$ meet the conditions of Lemma 3.6 and therefore Procedure Process_Vertex when run on (G, v) reports that the graph G is not HHD-free. If $z_{B_{t-1}}$ and $z_{B_{t+1}}$ belong to the same set Z_q then none of them is z_{A_1} since no vertex in ρ other than z_{A_1} is adjacent to u. Then $z_{B_{t-2}}$ (which may be z_{A_1}) belongs to a set $Z_{q'}$ with $q' > q$ and $z_{B_{t-2}} z_{B_{t+1}} \notin E(G_v)$; then the vertices z_{B_t}, $z_{B_{t-1}}$, $z_{B_{t+1}}$, and $z_{B_{t-2}}$ meet the conditions of Lemma 3.7 and therefore Procedure Process_Vertex when run on (G, v) reports that the graph G is not HHD-free.

Finally, suppose that G contains a domino D induced by the cycle $vudefw$ with a single chord uf (i.e., v is a corner vertex of D). Again, if Algorithm Recognize-HHD-free does not stop early (in which case, it reports that the input graph G is not HHD-free), it will eventually execute the body of the for-loop of Step 4 for the vertex v. The vertex adjacencies in the domino D imply that the vertices d, e, f belong to distinct sets of the partition $\mathcal{S}_v$; let D, E, F be the connected components of the subgraphs $G[S_j], G[S_i], G[S_k]$ to which d, e, f belong, respectively, where $i \neq j$, $i \neq k$, and $j \neq k$. Because $de \in E(G)$ and $ef \in E(G)$, then $\{z_D z_E, z_E z_F\} \subseteq E(G_v)$. If $z_D z_F \in E(G_v)$, then there exist vertices $x \in D$ and $y \in E$ such that $xy \in E(G)$; but then, the vertices v, u, x, y, w induce a house in G with x at its top, and as proved earlier for the case of an induced house or C_5, Procedure Process_Vertex when run on (G, x) reports that the graph G is not HHD-free. So let us assume that $z_D z_F \notin E(G_v)$. If $i > j$, Lemma 3.8 applies to the edge $z_D z_E$ of G_v, and because u is adjacent to z_D but not to z_E, it implies that Procedure Process_Vertex when run on (G, v) reports that G is not HHD-free. Similarly, if $i > k$ due to the edge $z_E z_F$. So, suppose that $i < \min\{j, k\}$. But then the vertices z_E, z_D, z_F meet the conditions of Lemma 3.6 which again implies that Procedure Process_Vertex when run on (G, v) reports that the graph G is not HHD-free.

Since Algorithm Recognize-HHD-free calls Procedure Process_Vertex for all pairs (G, x), where x is any vertex of G, then in all the above cases Algorithm Recognize-HHD-free reports that the graph G is not HHD-free. ∎

3.6 Time and Space Complexity

Let n be the number of vertices and m be the number of edges of the graph G. Since each of the forbidden subgraphs that we are looking for (a house, a hole, or a domino) is connected, we may assume that G is connected, otherwise we work on G's connected components which we can compute in $O(n+m)$ time [5]; thus, $n = O(m)$. Below, we give the time and space complexity of each step of Procedure Process_Vertex when applied on a graph G and a vertex v from which we obtain the time and space complexity of Algorithm Recognize-HHD-free.

The neighbors and non-neighbors of vertex v in the graph G can be stored in $O(n)$-size arrays for constant-time access; this takes $O(n)$ time. The partition of the non-neighbors of v based on their common neighbors with v can be computed in $O(m + n\, deg(v))$ time and $O(n)$ space, where $deg(v)$ denotes the degree of v in G; see [17][1]. After having computed for a vertex of each of the partition sets the number of its common neighbors with v, which can be done in $O(n+m)$ time, we can form the ordered sequence $(S_1, S_2, \ldots, S_\ell)$ in $O(\ell + deg(v)) = O(n)$ time and $O(n)$ space using bucket sorting. Thus, Step 1 of Procedure Process_Vertex takes $O(m + n\, deg(v))$ time and $O(n)$ space in total.

Adjacency-list representations of the subgraphs $G[S_i]$, $i = 1, 2, \ldots, \ell$, can be obtained in $O(n+m)$ time and space by appropriate partitioning of a copy of an adjacency-list representation of the graph G and removal of unneeded records; then, computing the connected components of all these subgraphs takes a total of $O(n+m)$ time and space, from which the graph G_v can be constructed in $O(n+m)$ additional time and space. Thus, Step 2 of Procedure Process_Vertex takes a total of $O(n+m)$ time and space. It is important to note that the graph G_v has $O(n)$ vertices and $O(m)$ edges.

Crucial for Steps 3 and 4 of Procedure Process_Vertex is the construction and processing of the partitions $\mathcal{P}_{Z_i}$, $i = 1, 2, \ldots, \ell$. These are maintained by means of an auxiliary multi-graph H_v represented by means of adjacency lists: members of the same partition set belong to the same connected component of H_v. The graph H_v has one vertex for each non-neighbor of v in G_v and, with a slight abuse of notation, we can write that $V(H_v) = \bigcup_{i=1}^{\ell} Z_i$; thus, $|V(H_v)| = O(n)$. Initially, the graph H_v has no edges. Then

- ▷ whenever, for a non-neighbor z_R of v in G_v, we need to union the sets of a partition $\mathcal{P}_{Z_k}$ that contain the vertices in a set W in Step 4, we pick a vertex, say, $z \in W$, and add edges (in an adjacency list representation of H_v) connecting z to all the other vertices in W; this takes $O(|W|) = O(|X'|) = O(deg_{G_v}(z_R))$ time and space.

Since each non-neighbor of v in G_v can be the representative of a partition set at most once, then

$$|E(H_v)| \;=\; O\left(\sum_{z_R \notin N[v]} deg_{G_v}(z_R) \right) \;=\; O\big(|E(G_v)|\big) \;=\; O(m).$$

Then, when the time comes in Step 4 to process the partition $\mathcal{P}_{Z_i}$ of the set Z_i,

- ▷ we compute the connected component of the graph H_v to which each vertex in Z_i belongs; graph traversal algorithms, such as, depth-first search and breadth-first search, can be used on H_v to yield the connected components in time linear in the number of vertices and edges of $H_v[Z_i]$.

In summary, constructing H_v without any edges, or equivalently, forming the initial partitions $\mathcal{P}_{Z_i}$ ($i = 1, 2, \ldots, \ell$) where each vertex in Z_i is placed in a separate partition set, takes $O(n)$ time and space; computing the partition $\mathcal{P}_{Z_i}$ in Step 4 for all $i = 1, 2, \ldots, \ell$ takes $O(n+m)$ time; the space required for the multi-graph H_v is also $O(n+m)$.

[1] An algorithm to construct a partition of a set L_2 in terms of adjacency to elements of a set L_1 is given in Section 3.2 of [17] with a stated time complexity of $O(m + n\,|L_2|)$; yet, it can be easily seen that the algorithm has a time complexity of $O(m + |L_1| \cdot |L_2|)$, which in our case gives $O(m + n\, deg(v))$ since $|L_1| = deg(v)$ and $|L_2| = O(n)$.

Since, initializing the sets $A(\,)$ for all the vertices in $\bigcup_{i=1}^{\ell} Z_i$ takes $O(n)$ time and space, Step 3 of Procedure Process_Vertex takes $O(n)$ time and space.

Next, let us consider the processing of a set P_j of the partition $\mathcal{P}_{Z_i}$ in Step 4. Computing the set X' takes $O(deg_{G_v}(z_R))$ time and space, where $deg_{G_v}(z_R)$ denotes the degree of vertex z_R in the graph G_v. Checking if P_j is a singleton set takes $O(1)$ time, and checking if all the vertices in P_j are adjacent to exactly the vertices in X' among the vertices in $\bigcup_{i=i+1}^{\ell} Z_i$ takes $O(\sum_{z \in P_j} deg_{G_v}(z))$ time. Next, checking whether X' is non-empty takes $O(1)$ time and doing all the processing if $X' \neq \emptyset$ takes $O(deg_{G_v}(z_R))$ time and space; note that $|W| \leq |X'| \leq deg_{G_v}(z_c)$. Finally, checking whether $\big(\bigcup_{z \in P_j} A(z)\big) - N_{G_v}(z_R) \neq \emptyset$ takes $O\big(\bigcup_{z \in P_j} |A(z)|\big)$ time. In summary, processing the set P_j takes $O\big(\sum_{z \in P_j} \big(deg_{G_v}(z) + |A(z)|\big)\big)$ time and $O(deg_{G_v}(z_R))$ space. Since the sets of each partition $\mathcal{P}_{Z_i}$ of the set Z_i are disjoint and the sets Z_i are disjoint, we have that $O\big(\sum_{i=1}^{\ell} \sum_{P_j \in \mathcal{P}_{Z_i}} \sum_{z \in P_j} deg_{G_v}(z)\big) = O\big(|V(G_v)| + |E(G_v)|\big) = O(n+m)$. Additionally, since the sets $A(\,)$ are formed by concatenating some of the neighbors in G_v of one vertex z_R from each set P_j, we have that $O\big(\sum_{i=1}^{\ell} \sum_{P_j \in \mathcal{P}_{Z_i}} \sum_{z \in P_j} |A(z)|\big) = O\big(\sum_{i=1}^{\ell} \sum_{P_j \in \mathcal{P}_{Z_i}} \sum_{z \in P_j} deg_{G_v}(z)\big) = O\big(|V(G_v)| + |E(G_v)|\big) = O(n+m)$ as well. Thus, in total, Step 4 of Procedure Process_Vertex takes $O(n+m)$ time and space.

Since Steps 1-4 of Procedure Process_Vertex are executed for each vertex v of the input graph G, we have that the overall time complexity of Algorithm Recognize-HHD-free is:

$$\Big[\sum_{v \in V(G)} \Big(O(m + n\, deg(v)) \Big) \;+\; O(n+m) \Big] \;+\; O(1) \;\;=\;\; O(n\,m).$$

Therefore, we obtain the following result.

Theorem 3.2 *Let G be an undirected graph on n vertices and m edges. Then, Algorithm Recognize-HHD-free determines whether G is an HHD-free graph in $O(nm)$ time and $O(n+m)$ space.*

4 Providing a Certificate

Algorithm Recognize-HHD-free can be made to provide a certificate (a house, a hole, or a domino) whenever it decides that the input graph G is not HHD-free. The algorithm reports that the graph G is not HHD-free in two occasions, in Step 4.2 and in Step 4.4 of Procedure Process_Vertex.

In order to be able to efficiently produce a certificate when Procedure Process_Vertex reports that the input graph G is not HHD-free, we do the following additional work:

W_1: Whenever, during the processing of a set P_j of a partition $\mathcal{P}_{Z_i}$, we need to union the sets of a partition $\mathcal{P}_{Z_k}$ containing the vertices in the set W, which is done by adding edges in the auxiliary multi-graph H_v (as explained in Section 3.6), we associate with each such edge the selected vertex z_R of P_j.

W_2: When processing a set P_j, we store with each element of the set $X - W$, which is added to $A(z_X)$ for $z_X \in W$, a reference to the selected vertex z_R of P_j; in this way, for each vertex z, each element of the set $A(z)$ carries a reference to a vertex of the set during whose processing this element was added to $A(z)$.

Note that this additional work does not increase asymptotically the time and space complexity of the algorithm.

Next, let us see what we do in order to compute a certificate in the two cases that Procedure Process_Vertex reports that the graph G is not HHD-free:

1. Step 4.2: There exists a vertex in a set P_h of a partition $\mathcal{P}_{Z_j}$ that is not adjacent to a vertex in X' or is adjacent to a vertex in $\big(\bigcup_{r=j+1}^{\ell} Z_r\big) - X'$. In any case, there exist vertices $z_A, z_B, z_{B'}, z_C$ of the graph G_v where $z_A \in Z_i$, $z_B, z_{B'} \in P_h$ (and thus $z_B, z_{B'} \in Z_j$), and $z_C \in Z_k$ with $i < j < k$, such that $\{z_A z_B, z_A z_{B'}, z_B z_C\} \subseteq E(G_v)$, and $z_{B'} z_C \notin E(G_v)$. Additionally, because $i < j < k$, Lemma 3.1 statement (i) implies that there exist vertices $x, y \in N(v)$ such that x is adjacent to all the vertices in Z_j and to no vertex in Z_i, and y is adjacent to all the vertices in Z_k and to no vertex in Z_j; see Figure 6(b).

We can find the vertices $z_A, z_B, z_{B'}, z_C$ by processing the edges of the tree that was traversed when computing the connected component of the graph H_v that produced the partition set P_h: for each edge in the tree, we test whether its endpoints have the exact same neighbors in $\bigcup_{r=j+1}^{\ell} Z_r$; note that (i) if this is true for each edge, then clearly all the vertices in P_j have the same neighbors in $\bigcup_{r=j+1}^{\ell} Z_r$, and (ii) this test takes $O\big(\sum_{z \in P_h} deg_{G_v}(z)\big)$ time as in the algorithm's analysis. In our case, we will be able to find an edge in the tree with endpoints z_1 and z_2, and a vertex $z_3 \in \bigcup_{r=j+1}^{\ell} Z_r$ such that z_1 is adjacent to z_3 in G_v whereas z_2 is not; then, $z_B = z_1$, $z_{B'} = z_2$, $z_C = z_3$, while z_A is the vertex associated with the tree edge $z_1 z_2$ (see W_1).

Let A, B, B', C be the connected components whose shrinking produced the vertices $z_A, z_B, z_{B'}, z_C$ of the graph G_v. We consider the following two cases depending on whether z_A, z_C are adjacent in G_v or not:

1a. $z_A z_C \notin E(G_v)$: We traverse the adjacency lists of the vertices in B' until we find a vertex $b \in B'$ which is adjacent to a vertex in A. Next, we run breadth-first search in the subgraph $G[\{b\} \cup A \cup B \cup C]$ starting at b until a vertex $c \in C$ is encountered; let ρ be the path in the breadth-first search tree connecting b to c. Clearly, ρ is chordless and because $z_A z_C \notin E(G_v)$ and $z_B z_{B'} \notin E(G_v)$, it is of the form $\rho = b a_1 \cdots a_r b_1 \cdots b_s c$ where $r, s \geq 1$, $a_1, \ldots, a_r \in A$, and $b_1, \ldots, b_s \in B$; see Figure 9(a). If $r > 1$, then the vertices $x, b, a_1, \ldots, a_r, b_1$ induce a hole in G. If $r = 1$ and $s > 1$, the vertices x, b, a_1, b_1, b_2 induce a house in G with b_2 at its top. Finally, if $r = 1$ and $s = 1$, then if $xc \in E(G)$, the vertices x, b, a_1, b_1, c induce a house with c at its top, whereas if $xc \notin E(G)$, the vertices v, x, b_1, c, y induce a house (with v at its top) or a C_5 in G depending on whether $xy \in E(G)$ or not.

1b. $z_A z_C \in E(G_v)$: See Figure 8(a). As in the previous case, we traverse the adjacency lists of the vertices in B' until we find a vertex $b \in B'$ that is adjacent to a vertex in A. Next, we run breadth-first search in the subgraph $G[\{b\} \cup A \cup C]$ starting at b until a vertex $c \in C$ is encountered; let ρ be the path in the breadth-first search tree connecting b to c; see Figure 9(b). Again, ρ is chordless and because $z_{B'} z_C \notin E(G_v)$, it is of the form $\rho = b a_1 \cdots a_r c$ where $r \geq 1$ and $a_1, \ldots, a_r \in A$. Consider first that $xc \in E(G)$. If $r > 1$, the vertices $x c a_1 \cdots a_r b$ induce a hole in G. If $r = 1$, then if $y a_1 \in E(G)$, the vertices v, x, b, a_1, y induce a house or a C_5 in G depending on whether $xy \in E(G)$ or not, whereas if $y a_1 \notin E(G)$, G contains a house induced by the vertices x, b, a_1, c, y or a domino induced by v, x, b, a_1, c, y depending on whether $xy \in E(G)$ or not. Suppose next that $xc \notin E(G)$. Then, if $y a_1 \in E(G)$, the vertices v, x, b, a_r, y induce a house or a C_5 depending on whether $xy \in E(G)$, whereas if $y a_1 \notin E(G)$, G contains the hole $c a_1 \cdots a_r b x y$ or $v y c a_1 \cdots a_r b x$ depending on whether $xy \in E(G)$.

2. Step 4.4: For a set P_h of a partition $\mathcal{P}_{Z_j}$, we have that $\big(\bigcup_{z \in P_h} A(z)\big) - N_{G_v}(z_R) \neq \emptyset$, i.e., there exist vertices $z_B \in P_h$ (z_B has resulted from the shrinking of a component B of $G[S_j]$) and $z' \in \big(\bigcup_{r=j+1}^{\ell} Z_r\big) \cup N(v)$ such that $z' \in A(z_B)$ and $z' \notin N_{G_v}(z_R)$; note that since the algorithm has not stopped earlier, z_B and z_R have the exact same neighbors in $\bigcup_{r=j+1}^{\ell} Z_r$, which implies that $z' \notin N_{G_v}(z_B)$. Since $z' \in A(z_B)$, by means of W_2, z' is associated with a vertex $z_A \in \bigcup_{r=1}^{j-1} Z_r$ such that $z_B, z' \in N_{G_v}(z_A)$. If $z' \in N(v)$, we locate two vertices $a \in A$ and $b \in B$ such that $ab \in E(G)$; because $z' \in N_{G_v}(z_A) - N_{G_v}(z_B)$, z' is adjacent to a in G but is not adjacent to b (see Figure 8(b) with

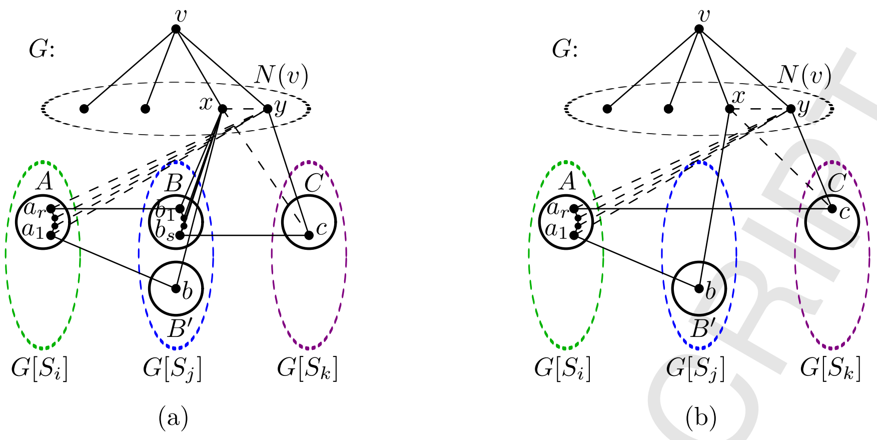

Figure 9: For the certificate computation at Step 4.2.

$z' = x$). Moreover, there exists a neighbor w of v such that $w \in N_G(b) - N_G(a)$; then, the vertices v, z', w, a, b induce a house or a C_5 in G depending on whether $z'w \in E(G)$ or not. Now, if $z' \notin N(v)$, then $z' = z_C$ corresponding to a component C of a subgraph $G[S_k]$ with $k > j$. This case is identical to Case 1b above for the vertices z_A, z_B, z_C.

It is not difficult to see that doing the work described above and checking the adjacencies in each of the aforementioned cases can be performed in $O(n + m)$ time using $O(n)$ space. Thus, we have:

Theorem 4.1 *Let G be an undirected graph on n vertices and m edges. Then, Procedure process_Vertex can be augmented so that Algorithm Recognize-HHD-free produces a house, a hole, or a domino whenever it decides that G is not an HHD-free graph in $O(n + m)$ additional time and $O(n)$ additional space.*

5 Concluding Remarks

We have presented a recognition algorithm for the class of HHD-free graphs that runs in $O(n\,m)$ time and requires $O(n + m)$ space, where n is the number of vertices and m is the number of edges of the input graph. Moreover, we show how our algorithm can be augmented to yield, in $O(n + m)$ time and $O(n)$ space, a certificate (a house, a hole, or a domino) whenever it decides that the input graph is not HHD-free.

Despite the close relation between HHD-free and HH-free graphs, our results do not lead to an improvement in the recognition time complexity for HH-free graphs; therefore, we leave as an open problem the design of an $O(n\,m)$-time algorithm for recognizing HH-free graphs. Additionally, it would be interesting to obtain faster recognition algorithms for other related classes of graphs, such as, the brittle and the semi-simplicial graphs.

Acknowledgment. The authors would like to thank the referees whose suggestions helped improve the readability and presentation of the paper.

References

[1] C. Berge, Färbung von Graphen deren sämtliche bzw deren ungerade Kreisse starr sind, *Wiss. Z. Martin-Luther-Univ. Hale-Wittenberg Mathe.-Natur.*, 114-115, 1961.

[2] A. Brandstädt, V.B. Le, and J.P. Spinrad, *Graph classes: A survey*, SIAM Monographs on Discrete Mathematics and Applications, 1999.

[3] M. Chudnovsky, N. Robertson, P. Seymour, and R. Thomas, The strong perfect graph theorem, *Ann. of Math.* **164**, 51-229, 2006.

[4] V. Chvátal, Perfectly ordered graphs, *Annals of Discrete Math.* **21**, 63–65, 1984.

[5] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, *Introduction to Algorithms* (2nd edition), MIT Press, Inc., 2001.

[6] E.M. Eschen, J.L. Johnson, J.P. Spinrad, and R. Sritharan, Recognition of some perfectly orderable graph classes, *Discrete Appl. Math.* **128**, 355–373, 2003.

[7] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, Inc., 1980.

[8] R. Hayward, Meyniel weakly triangulated graphs I: co-perfect orderability, *Discrete Appl. Math.* **73**, 199–210, 1997.

[9] P. Hell and J. Huang, Certifying LexBFS recognition algorithms for proper interval graphs and proper interval bigraphs, *SIAM J. Discrete Math.* **18**, 554–570, 2004.

[10] C.T. Hoàng, On the complexity of recognizing a class of perfectly orderable graphs, *Discrete Appl. Math.* **66**, 219–226, 1996.

[11] C.T. Hoàng and N. Khouzam, On brittle graphs, *J. Graph Theory* **12**, 391–404, 1988.

[12] C.T. Hoàng and R. Sritharan, Finding houses and holes in graphs, *Theoret. Comput. Sci.* **259**, 233–244, 2001.

[13] B. Jamison and S. Olariu, On the semi-perfect elimination, *Adv. Appl. Math.* **9**, 364–376, 1988.

[14] D. Kratsch, R. M. McConnell, K. Mehlhorn, and J. Spinrad, Certifying Algorithms for Recognizing Interval Graphs and Permutation Graphs, *SIAM J. Computing* **36**, 326–353, 2006.

[15] R.M. McConnell and J. Spinrad, Linear-time modular decomposition and efficient transitive orientation, *Proc. 5th Annual ACM-SIAM Symp. on Discrete Algorithms (SODA'94)*, 536–545, 1994.

[16] M. Middendorf and F. Pfeiffer, On the complexity of recognizing perfectly orderable graphs, *Discrete Math.* **80**, 327–333, 1990.

[17] S.D. Nikolopoulos and L. Palios, Algorithms for P_4-comparability graph recognition and acyclic P_4-transitive orientation, *Algorithmica* **39**, 95–126, 2004.

[18] S.D. Nikolopoulos and L. Palios, Recognizing HH-free, HHD-free, and Welsh-Powell opposition graphs, *Discrete Math. and Theoret. Comput. Sci.* **8**, 65–82, 2006.

[19] S.D. Nikolopoulos and L. Palios, An $O(nm)$-Time Certifying Algorithm for Recognizing HHD-free graphs, *Proc. 1st Annual Frontiers in Algorithmics Workshop (FAW'07)*, 281–292, 2007.

[20] S. Olariu, Weak bipolarizable graphs, *Discrete Math.* **74**, 159–171, 1989.

[21] D.J. Rose, R.E. Tarjan, and G.S. Lueker, Algorithmic aspects of vertex elimination on graphs, *SIAM J. Comput.* **5**, 266–283, 1976.