Applied and Computational Mathematics
2013; 2(3): 86-91
Published online July 20, 2013 (http://www.sciencepublishinggroup.com/j/acm)
doi: 10.11648/j.acm.20130203.13

A note on self complementary brittle and self complementary quasi chordal graphs

Parvez Ali[1], Merajuddin[2], Syed Ajaz Kareem Kirmani[3]

[1]Department of Mathematics, Maharana Pratap Engineering College, Mandhana , Kanpur, INDIA
[2]Department of Applied Mathematics, Faculty of Engineering, Aligarh Muslim University, Aligarh, INDIA
[3]College of Engineering Unayzah, Qassim University, KINGDOM OF SAUDI ARABIA

Email address:
parvezamu@rediffmail.com(P. Ali), meraj1957@rediffmail.com(Merajuddin), ajazkirmani@rediffmail.com(S. A. K. Kirmani)

To cite this article:
Parvez Ali, Merajuddin, Syed Ajaz Kareem Kirmani. A Note on Self Complementary Brittle and Self Complementary Quasi Chordal Graphs. *Applied and Computational Mathematics.* Vol. 2, No. 3, 2013, pp. 86-91. doi: 10.11648/j.acm.20130203.13

Abstract: In this paper we deal with some classes of self-complementary (sc) perfectly orderable graphs namely sc brittle, sc quasi chordal graphs and propose algorithms for these classes. We obtain some results on these classes and an algorithm is proposed based on these results that recognize these classes. We also compile a catalogue for these classes up to 17 vertices.

Keywords: Self Complementary, Brittle, Quasi Chordal, No Mid, No End

1. Introduction

A graph G is self-complementary (sc) if it is isomorphic to its complement. A sc graph exists on $4k$ or $4k + 1$ vertices, where k is a positive integer. A vertex x is simplicial if its neighborhood $N(x)$ induces a complete subgraph and it is co-simplicial if its non-neighbors form an independent subset of vertices. An induced P_4 with vertices a, b, c, d and edges ab, bc, cd then we refer to the vertices a and d as the endpoints, and the vertices b and c as the midpoints of the P_4. A graph G is brittle if each induced subgraph H of G contains a vertex that is not a midpoint of any P_4 or not an endpoint of any P_4 [1]. A graph is quasi-chordal if each of its induced subgraphs contains a simplicial or co-simplicial vertex. A self-complementary graph which is also quasi-chordal (brittle) is known as sc quasi-chordal (brittle) graph. For standard definition not mentioned here refer [03], [10] and [16].

Recognition Algorithm for brittle graphs was discussed by many authors, Hoàng and Khouzam [06] recognized it in $O(n^3m)$ time. Hoàng and Reed [05] recognizes brittle graph in $O(n^5)$ time using the concept of brittle order. After that Schäffer [11] dealt specifically with the recognition problem for brittle graphs and gave $O(m^2)$ time but complicated recognition algorithm. In a technical report [12], Spinrad and Johnson also designed an $O(n^3log^2n)$ algorithm for brittle graphs. Later Eschen et al. [02] presented two algorithms for recognition of brittle graphs by direct application of the definition yields an $O(n^3log^2n)$ time deterministic or $O(n^3)$ time randomized recognition algorithm for brittle graphs.

Quasi-chordal graphs were introduced by Voloshin [15] as a generalization of chordal graphs. Hoàng and Mahadev also gave the similar concept in [07], where they called quasi-chordal graphs as good graphs. However, no subgraph characterization of quasi-chordal graphs is known [10]. The recognition problem of quasi-chordal graphs was first studied by Voloshin [14], he also proposed an algorithm for this. Later Spinrad [13], Hoàng [08] and Gorgos et al. [03] improved the time complexity of the recognition algorithm for quasi-chordal graphs.

In this note, we consider sc brittle graphs and sc quasi-chordal graphs. Both are subclasses of sc perfectly orderable graphs.

Section 2 is devoted to sc brittle graphs where a recognition algorithm for sc brittle graphs is propose. In section 3, we study sc quasi-chordal graphs and present an algorithm for the recognition of sc quasi-chordal graphs.

2. Self-Complementary Brittle Graphs

In this section, we proposed recognition algorithm for sc brittle graphs that employs the idea similar to those used in above mentioned algorithms. Sc brittle graphs can be recognized using the vertex elimination scheme i.e. if all its vertices eliminated by successive deletion of no-mid and no-end vertices, where a vertex of a graph G is called no-mid if it is not the midpoint of any P_4 in G. Similarly, a

vertex of a graph G is called no-end if it is not the endpoint of any P_4 in G. If a graph G does not contain any induced P_4 then all the vertices should be treated as no-mid as well as no-end vertices. No-mid and no-end vertices in sc brittle graphs have same structure. To ensure this we give following Theorem, which relates no-mid vertex to no-end vertex in a sc brittle graph.

Theorem 1. Let G be a sc graph, then if there exists any no-mid vertex in G, then there also exists a no-end vertex in G. Converse is also true.

Proof. Let G be a sc graph, suppose a vertex v be a no-mid in G, then by definition of no-mid vertex it is not the middle vertex of any P_4 in G. Now in the complement of the graph G, the same vertex v becomes no-end vertex because the complement of a P_4 is also a P_4. Since G is sc graph, therefore if there exists any no-mid vertex in G then there also exists a no-end vertex in G. The same argument is also true for the converse Hence the Theorem.

The following corollary is immediate.

Corollary 2. Let G be a sc graph, if there exists no no-mid vertex in G then there will be no no-end vertex in G. Converse is also true.

To recognize sc brittle graphs, we have to find first a vertex which is either a no-mid or no-end by using algorithm-1

Algorithm 1: *An Algorithm for no-mid and no-end vertices.*

Input: A sc graph G.

Output: Vertex set of no-mid and no-end vertices.

Step-1: no-mid set = ϕ, no-end set = ϕ and R_v = ϕ, List all the induced P_4's of G.
Step-2: Select arbitrary vertex 'u' if it is not a middle vertex of any P_4 in G, then put the vertex 'u' in no-mid set. Else if it is not a end-vertex of any P_4 in G, then put the vertex 'u' in no-mid set. Else Put the vertex 'u' in R_v
Step-3: If all the vertices scanned then Stop.
Else goto step-2
End.

*where R_v is the vertex set which are neither no-mid nor no-end.

Complexity Since all the P_4's are generated in $O(n^2m)$ time [11] and there can be at most $O(n^2m)$ P_4's in G, so step-1 can be done $O(n^2m)$ time. From step-3 to step-4, we require one more iteration of size n, each of which requires $O(n^2m)$ time. So overall time complexity is $O(n^3m)$.

After computing no-mid vertex set and no-end vertex set, we are in a position to discuss algorithm for recognition of sc brittle graphs. Algorithm-2, which recognizes whether a given sc graph is sc brittle or not works as follows: first it computes no-mid and no-end vertex set in step-1 by using algorithm-1. Now if both the sets i.e. no-mid and no-end vertex sets are empty then algorithm terminates and gives output that the given sc graph is not sc brittle, otherwise algorithm proceeds to the next step i.e. step-3. In this step, it deletes vertex either from no-mid or no-end vertex set (note that the choice of no-mid or no-end vertex to delete is arbitrary since the deletion of a vertex cannot cause another vertex to become the midpoint or endpoint of a P_4). Then the algorithm repeats the procedure from step-1 to step-3. In this way if all the vertices are eliminated then algorithm-2 decides that the given sc graph is sc brittle. Algorithm-2 is as follows.

Algorithm 2: *An Algorithm for recognition of sc brittle graph.*

Input: A sc graph G.

Output: "G is sc brittle graph" or "G is not sc brittle graph".

Step-1: Compute the no-mid and no-end vertex set and elimination order = ϕ
Step-2: If no-mid set = ϕ = no-end set, then return "G is not sc brittle".Stop.
Step-3: eliminate vertex 'u' either from no-mid set or no-end set. Put 'u' in elimination order.
Step-5: update the graph and goto step-1.
Step-6: If all the vertices of G are eliminated in this way, then "G is sc brittle" and print the vertices of the elimination order. Else"G is not sc brittle".
End.

Complexity. In algorithm-2, the bottleneck is step-1, which requires $O(n^3m)$ time. All the other steps have lower complexity. So the overall time complexity is $O(n^3m)$.

The following result justifies the claim of algorithm-2.

Theorem 3. Algorithm-2 checks whether an input sc graphs is sc brittle or not correctly.

Proof. By the definition of brittle graphs, its each induced subgraph contains either a no-mid or no-end vertex. So we start with a sc graph G and look for no-mid or no-end vertex, if found, remove that vertex (let it be u). Now by definition of brittle graphs, G - u again contains no-mid or no-end vertex, if found, remove that vertex again. Clearly if all the vertices are removed in this manner, then we are left with no vertex and the graph G is sc brittle. If while deleting the vertices we found that there does not exist any no-mid or no-end vertex in any subgraph of G, then at that stage we decide that the graph is not sc brittle. Hence the Theorem. □

To illustrate algorithm-2, we consider the following sc graphs G_1 shown in figure-1.

Let graph G_1 be the input to algorithm-2. Step-1 of algorithm-2 computes its no-mid set as $\{v_3,v_4\}$ and no-end set as $\{v_7,v_8\}$, by calculating all induced P_4's as $[v_1,v_5,v_2,v_6]$, $[v_1,v_8,v_6,v_2]$, $[v_1,v_8,v_6,v_3]$, $[v_2,v_5,v_1,v_4]$, $[v_2,v_5,v_8,v_4]$, $[v_2,v_6,v_8,v_4]$, $[v_2,v_7,v_1,v_4]$, $[v_2,v_7,v_8,v_4]$, $[v_3,v_6,v_2,v_5]$, $[v_3,v_6,v_8,v_4]$, $[v_3,v_6,v_8,v_5]$, $[v_3,v_7,v_1,v_4]$, $[v_3,v_7,v_1,v_5]$, $[v_3,v_7,v_2,v_5]$, $[v_3,v_7,v_8,v_4]$, $[v_3,v_7,v_8,v_5]$, $[v_4,v_1,v_7,v_6]$ and $[v_5,v_1,v_7,v_6]$ using algorithm-1. Since both the no-mid and no-end sets are not empty thus algorithm proceeds to step-3. In step-3 any vertex from either no-mid or no-end vertex set can be deleted, let it be v_3. The graph is updated in step-5 and then algorithm repeats the procedure. In this way algorithm-2 successfully eliminates all the vertices of G_1

and produces output as G_1 is a sc brittle graph. The overall procedure of elimination of vertices from no-mid and no-end set can be seen in figure-2, where a vertex which is enclosed by dotted line is deleted vertex.

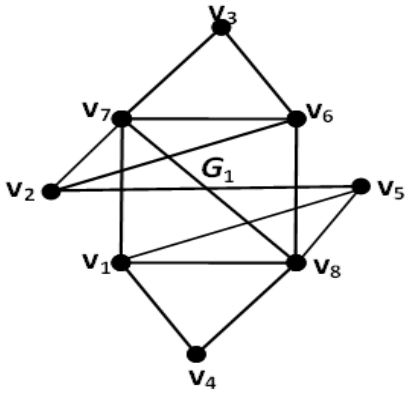


Figure 1. *sc brittle graph on 8 vertices*

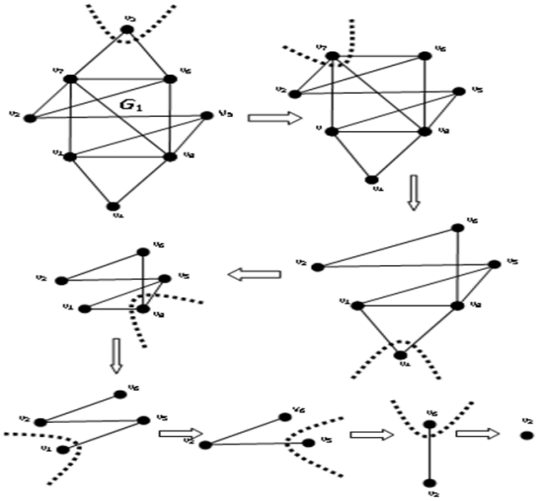


Figure 2. *Illustration of algorithm-2*

3. Self-Complementary Quasi-Chordal Graphs

The following Theorem was known to researchers and referred in the literature but its proof had never been published. Recently Gorgos et al. [04] proved the Theorem.

Theorem 4 [04]. For a graph G, the following conditions are equivalent:

(i) G is quasi-chordal.

(ii) G does not contain a latticed subgraph as an induced subgraph.

(iii) G admits a good order.

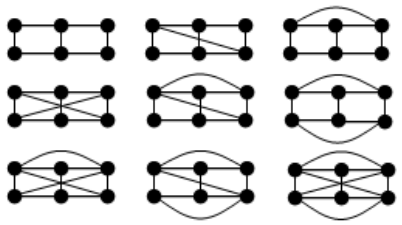
Where an order $v_1<v_2<,\ldots,<v_n$ on a graph G is good if, for any induced subgraph H of G, either the largest vertex of $(H, <)$ is simplicial or the smallest vertex of $(H, <)$ is co-simplicial and a graph with each vertex belonging to some hole and some antihole is called latticed.

Recently, in [09] Hoàng et al. reported the following result.

Theorem 5 [09]. If G is a weakly chordal graph such that every pair of squares meets in a non-edge, then G is a quasi-chordal graph.

Next we investigate the case when at least one pair of squares meets in an edge and and obtain the following

Theorem 6. Let G be a sc graph, such that at least one pair of squares meets in an edge then G contains one or more of the following graphs as induced subgraphs.



Proof. Let G be sc graph, suppose there exists at least one pair of squares which meets in an edge. Now whenever two squares meet in an edge then the graph formed in this way will always have six vertices only, however they may have seven vertices but in this case the graph has no edge common as shown in figure-3, so we do not consider this graph ,

Now consider a six cycle graph with vertices v_1, v_2, v_3, v_4, v_5,v_6 with edges v_1v_2, v_2v_3, v_3v_4, v_4v_5, v_5v_6, v_6v_1. We add an edge between the vertices v_2 and v_5 as v_2v_5. This obtained graph is shown in figure-4 as graph D_1. The graph D_1 has clearly two squares as $\{v_1,v_2,v_5,v_6\}$, $\{v_2,v_3,v_4,v_5\}$ and they meet on edge v_2v_5.

If we add edges on graph D_1 one by one between the non-adjacent pair of vertices then we get following graphs D_2, D_3, D_4, D_5, D_6, D_7, D_8 and D_9 as follows;

The graphs D_2 and D_3 as shown in figure-5 are obtained by just adding a single edge on graph D_1 between the vertices v_1,v_4 and v_1,v_3 respectively.

The graphs D_4, D_5 and D_6 as shown in figure-6 are obtained by adding edges on D_1 as (v_1,v_4), (v_3,v_6) for D_4, edges (v_1,v_4), (v_1,v_3) for D_5 and edges (v_1,v_3), (v_4,v_6) for D_6.

Similarly graphs D_7, D_8 and D_9 as shown in figure-7, can be obtained from D_1 by adding the edges as follows; (v_1,v_4), (v_1,v_3), (v_3,v_6) for D_7, (v_1,v_4), (v_1,v_3) (v_4,v_6) for D_8 and (v_1,v_4), (v_1,v_3), (v_3,v_6), (v_4,v_6) for D_9.

Clearly graphs D_1 to D_9 are the only possible graphs on six vertices such that two squares meet in an edge. Hence the Theorem.

We note that the graph D_3 given in figure-5(b) contains an induced cycle C_5: v_1, v_3, v_4, v_5, v_6 and v_1 and the graph D_6 is the complement of induced cycle C_6 as given in figure-6(c). This observation shows that both the graphs D_3 and D_6 are not weakly chordal, so we have the following result.

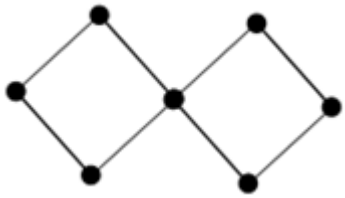


Figure 3. *Pair of squares meeting at vertex*

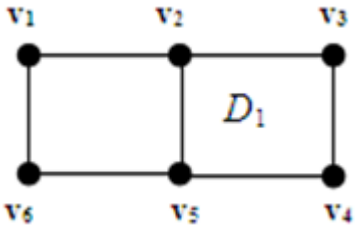


Figure 4. *Pair of squares meeting at edge*

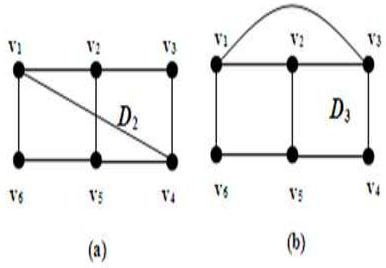


Figure 5. *Graphs obtained from D_1 (Fig-4) by adding an edge*

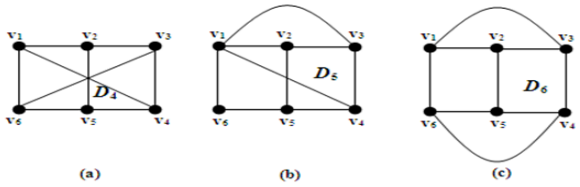


Figure 6. *Graphs obtained from D_1 by adding two edges*

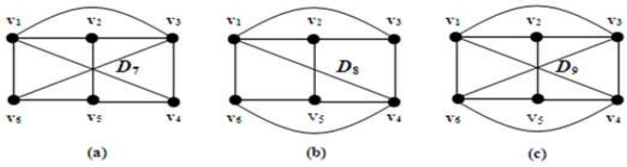
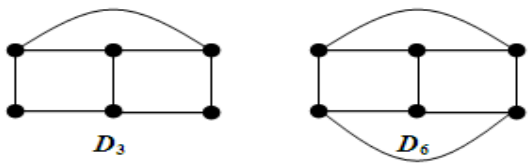


Figure 7. *Graphs obtained from D_1 by adding three edges*

Corollary 7. Let G be a sc weakly chordal graph such that at least one pair of squares meet in an edge then G does not contain the following graphs as induced subgraphs.



Many graph classes are defined or characterized in terms of an elimination scheme. For example chordal graphs are defined as having no induced cycles of length greater than 3. They are characterized as those graphs which have an elimination scheme with the property that neighbors of v_i induce a clique. Trees can be characterized as those graphs such that every eliminated vertex (except for v_n) has degree 1 in the remaining graph. Similarly for quasi-chordal graph there also exists an elimination scheme, which is ensured by the following result given by Voloshin [14].

Theorem 8 [14]. Let G be a graph on n vertices. Then G is quasi-chordal if and only if each of its induced subgraph contains a simplicial or co-simplicial vertex.

Based on the above result, recognition of quasi-chordal was first studied by Voloshin in [14] and he proposed an $O(n^4)$ time algorithm. Later Spinrad [13] proposed an $O(n^{2.77})$ time algorithm. Hoàng [08] also independently proposed an $O(nm)$ time algorithm. Much recently Gorgos et al. [04] also proposed an $O(nm)$ time algorithm for recognizing quasi-chordal graphs.

Quasi-chordal graph may admit many different elimination schemes i.e. if it has eliminated scheme only on the basis of simplicial vertices then the graph is chordal and if the eliminated vertices are only co-simplicial then the graph is co-chordal. Now if the vertices of quasi-chordal graphs are eliminated by first removing simplicial and then co-simplicial vertices then the graph is called semi-chordal. The following lemma shows the connection between the simplicial vertices and no-mid vertices as well as co-simplicial vertices and no-end vertices.

Lemma 9. Every simplicial vertex is no-mid vertex and every co-simplicial vertex is no-end vertex. Converse need not to be true.

Proof. Since simplicial vertex cannot be middle vertex of any induced P_3 and every induced P_4 always contains two, P_3 as a induced subgraphs. This implies that the middle vertices of both P_3 are always mid vertices of P_4, therefore simplicial vertices cannot be middle vertex of any P_4. Hence they are always no-mid vertex. Now suppose x is any end vertex of an induced P_4 then the non-neighbors of x will never form stable set as one of its non-neighbors is an end-vertex and the other is mid-vertex, therefore end vertices of any P_4 cannot be co-simplicial. Hence co-simplicial vertex is always no-end vertex.

Now for converse consider a chordless cycle of length 4 (i.e. C_4), then each vertex of this chordless C_4 is no-mid but not simplicial. Similarly in the complement of this C_4(i.e. $2K_2$), every vertex is no-end but not co-simplicial.

Hence the result.

The following result shows the relation between simplicial and co-simplicial vertices in sc graphs.

Theorem 10. Let G be a sc graph, if there exists any simplicial vertex in G then there also exists a co-simplicial vertex, converse is also true.

Proof. Let G be a sc graph, suppose a vertex v be a simplicial in G, then by definition of simplicial vertex, all the neighborhood vertices of v are adjacent to each other. Now, in the complement of the G, same vertex v and

adjacent vertices make an independent set of vertices i.e. v becomes co-simplicial in $\overline{G}$. Since G and $\overline{G}$ are isomorphic, thus in G, there also exist a co-simplicial vertex. The same argument also holds for the converse. □

The following corollary is immediate from the above Theorem.

Corollary 11. Let G be a sc graph, if there exist no simplicial vertex in G then there exist no co-simplicial vertex, converse is also true.

For recognizing whether a sc graph is quasi-chordal graph, we use a different method as compare to algorithms discussed in [13], [04]. The basic difference between the algorithms in [13], [04] and the one given here is that we find the simplicial and co-simplicial vertices within the set of no-mid and no-end vertices while the other algorithms find simplicial and co-simplicial vertices in whole graph. So to decide whether a vertex is simplicial or co-simplicial or not first we present an algorithm-3 which is also used as subroutine in algorithm-4. The algorithm-3 is as follows.

Algorithm 3: *An Algorithm for recognizing simplicial and co-simplicial vertices.*

Input: A graph G and a vertex 'u'.
Output: Return either simplicial or co-simplicial vertex.

Step-1: Compute neighborhood of 'u' i.e. $N(u)$.
Step-2: If $N(u)$induces a complete subgraph of G, then return " 'u' is a simplicial vertex" Stopelse compute its non-neighbors $N'(u)$
If $N'(u)$ induces a stable set of G, then return "'u' is co-simplicial vertex" Stop.
Else
return "'u' is neither simplicial nor co-simplicial ".
End.
Complexity. The time complexity of the algorithm-3 is $O(n^2)$ by [09].

The algorithm-4 as given here mainly depends on finding no-mid and no-end vertex set in the input graph G. As soon as it computes no-mid and no-end vertex set, algorithm goes for the search of simplicial and co-simplicial vertices within the set of no-mid and no-end vertices. If no-mid and no-end vertex sets do not contain any simplicial and co-simplicial vertices respectively then at the initial stage of algorithm it is possible to get the output i.e. the input graph is not quasi-chordal.

Algorithm 4: *Algorithm for recognizing sc quasi-chordal graph.*

Input: A sc graph G.
Output: "G is sc quasi-chordal graph" or "G is not sc quasi-chordal graph".

Step-1: Compute set of no-mid, no-end, R_v (using algorithm-1)and elimination order = ϕ
Step-2: If all the vertices of no-mid and no-end set are simplicial and co-simplicial respectively and no-mid $\cup$ no-end = vertex set of graph, then print " G is sc quasi-chordal graph" and print "vertex set of G" Stop. Else if no-mid and no-end set contains no simplicial and no co-simplicial vertices respectively, then print "G is not sc quasi-chordal graph" Stop.
Step-3: select either a simplicial vertex from no-mid set or co-simplicial vertex from no-end set (let this vertex be *u)* remove vertex 'u' and put 'u' in elimination order.
Step-4: update the graph and goto step-1.
Step-5: If all the vertices are eliminated in this way then print "G is sc quasi-chordal graph" and print "elimination order set"
Else print "G is not sc quasi-chordal graph"
End.
Complexity. Algorithm-4 first uses algorithm-1 in step-1, which has time complexity $O(n^3m)$. All the other steps require lesser time. Hence the overall time complexity is $O(n^3m)$.
The correctness of algorithm-4 is as follows.
Theorem 12. Algorithm-4 checks whether an input sc graphs is quasi-chordal or not correctly.
Proof. Simplicial vertices are no-mid vertices and co-simplicial vertices are no-end vertices. So while running the algorithm-4, when we compute set of no-mid and no-end vertices simplicial vertex always lies in no-mid and co-simplicial vertex lies in no-end set. Now from the statements (*iii*) and (*i*) of Theorem-4, it is clear that if we eliminate all the vertices in such a way then the resulting graph is always a quasi-chordal. Hence the Theorem.

To illustrate algorithm 4 we consider the sc graphs G_1 and G_2 on 9 vertices as shown in figure-8(a) and figure-8(b) respectively.

Let the sc graph G_1 as shown in figure-8(a) be input to the algorithm-4. Step-1 finds no-mid set, no-end set and R_v as follows. no-mid = $\{v_3,v_4\}$, no-end = $\{v_7,v_8\}$, R_v = $\{v_1,v_2,v_5,v_6,v_9\}$, this is done by using algorithm-3. Now step-2 of algorithm-4 finds that vertices v_3, v_4 from no-mid set and vertices v_7, v_8 from no-end set. These are simplicial and co-simplicial vertices respectively (this is done by using algorithm-3). Although all the vertices of no-mid and no-end sets are simplicial and co-simplicial respectively, but no-mid $\cup$ no-end $\neq$ vertex set of graphs i.e. $\{v_3,v_4\} \cup \{v_7,v_8\} \neq \{v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8v_9\}$ therefore algorithm-4 proceeds to step-3. In this step let vertex v_3 be eliminated first. Then step-4 updates the graph and repeats the process again. Eventually algorithm-4 successfully eliminates all the vertices of G_1 one by one and produces the elimination order set = $\{v_3,v_4,v_7,v_1,v_8,v_6,v_5,v_2,v_9\}$.

Thus algorithm-4 decides that the input graph G_1 is sc quasi-chordal graph. The overall procedure of elimination of vertices can be seen in figure-9.

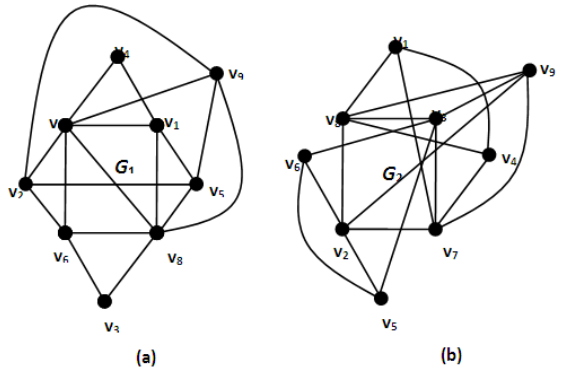


Figure 8a. *sc quasi-chordal graph on 9 vertices;* ***Figure 8b.*** *sc graph on 9 vertices but not quasi chordal*

In the similar fashion It can be decided that Graph G_2 in figure-8(b) is not quasi chordal graph.

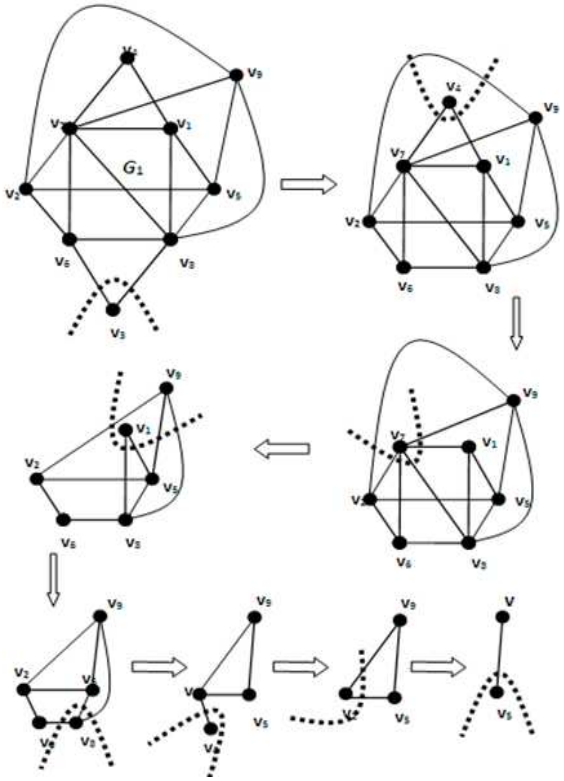


Figure 9. *Illustration of algorithm-4*

4. Catalogue Compilation

Using algorithm-2 and algorithm-4 we compile the catalogue of sc brittle graphs and sc quasi-chordal graphs with at most 17 vertices from the available catalogue of sc graphs with at most 17 vertices.

Table 1. *catalogue of sc brittle and sc quasi chordal graphs*

Vertices(n)	4	5	8	9	12	13	16	17
Sc Graphs	1	2	10	36	720	5600	703760	11220000
Sc quasi chordal Graphs	1	1	5	5	62	62	2406	2406
Sc brittle graphs	1	1	6	6	82	82	5912	5912

References

[1] V. Chvátal, *Perfect graph seminar*, McGill university, Montreal, (1983).

[2] E.M. Eschen, J.L. Johnson, J.P. Spinrad and R. Sritharan, *Recognition of some perfectly orderable graph classes*, Discrete Applied Mathematics, 128 (2003) 335-373.

[3] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, (1980).

[4] I. Gorgos, C.T. Hoang and V. Voloshin, *A note on quasi-triangulated graphs*, SIAM Journal of Discrete Mathematics, 20 (2006) 597-602.

[5] C.T. Hoàng and B.A. Reed, *Some classes of perfectly orderable graphs*, J. Graph Theory, 13 (1989) 445-463.

[6] C.T. Hoàng and N. Khouzam, *On brittle graphs*, J. Graph Theory, 12 (1988) 391-404

[7] C.T. Hoàng and N.V.R. Mahadev, *A note on perfect orders*, Discrete Mathematics, 74 (1989) 77-84.

[8] C.T. Hoàng, Recognizing quasi-triangulated graphs in O(nm) time, Manuscript, (unpublished).

[9] C.T. Hoàng, S. Hougardy, F, Maffray and N.V.R. Mahadev, *On simplicial and Co-simplicial vertices in graphs*, Discrete Applied Mathematics, 138 (2004) 117-132.

[10] J.L. Ramirez and B.A. Reed, *Perfect graphs*, John Wiley and Sons, (2000).

[11] A.A. Schaffer, *Recognizing brittle graphs: remarks on a paper of Hoang and Khouzam*, Discrete Applied Mathematics, 31 (1991) 29-35.

[12] J.P. Spinrad and J. Johnson, *Brittle and Bipolarizable graph recognition*, Vanderbilt Uni. Comp. Sci. Deptt., Technical Report (1998).

[13] J.P. Spinrad, *Recognizing quasi-triangulated graphs*, Disc. App. Math., 138 (2004) 203-213.

[14] V.I. Voloshin, *Quasi-triangulated graphs recognition program*, Algorithms and programs, P006124, Moscow, Russian, 1983 (in Russian).

[15] V.I. Voloshin, *Quasi-triangulated graphs*, Preprint, 5569-81, Kishinev state university, Kishinev, Moldova, 1981(in Russian).

[16] J. Yellen and J Gross, *Graph Theory and its Applications*, CRC Press (USA), 1999.