CS762: Graph-Theoretic Algorithms
Lecture 5: Perfect elimination orders
January 16, 2002

Scribe: Philip Tilker

Abstract

The topic of this lecture is about chordal graphs and perfect elimination orders. We have already seen in a previous lecture that if a graph G has a perfect elimination order then G is chordal. Today we will show that if G is chordal, then G has a perfect elimination order.

To prove this result, we will introduce the concept of a simplicial vertex and show that every chordal has a simplicial vertex. During the proofs, we will also see that every induced subgraph of a chordal graph is chordal.

Two algorithms are presented that can compute a perfect elimination in linear time. Full examples are included.

As a consequence, independent set, coloring, clique are solvable in linear time on chordal graphs since we can compute a perfect elimination order easily.

1 Introduction

In a previous lecture, we introduced chordal graphs, that is, graphs for which every cycle of length ≥ 4 has a chord. Chordal graphs are known under many names. Chordal graphs are at times also referred to as rigid-circuit graphs, monotone transitive graphs, triangulated graphs or perfect elimination graphs [Gol80].

Some of these names suggest the very nature of the graph. We have been studying perfect eliminiation orders in graphs and the term "perfect elimination graph" suggests that being a chordal graph is equivalent to having a perfect elimination order. Recall that a perfect elimination order is an ordering of the vertices $(v_1, v_2, \ldots, v_n)$, such that for every v_i, the predecessors of v_i form a clique.

We have already studied interval graphs and it turns out that every interval graph is a chordal graph.

2 Chordal Graphs and Perfect Elimination Orders

Our goal is to show that G is chordal if and only if G has a perfect elimination order. On Jan 14th, we showed that if G has a perfect elimination order then G is chordal. Our goal for today is to show that if G is chordal then G has a perfect elimination order.

To do so, we need the concept of a simplicial vertex. A vertex v is said to be simplicial if all the neighbours of v form a clique.

In 1962, Lekkerkerker and Boland showed that chordal graphs always have at least 2 simplicial vertices [LB62]. We will prove this fact below.

Theorem 1 *Every chordal graph G has a simplicial vertex. If G is not a complete graph, then it has 2 simplicial vertices that are not adjacent.*

Proof: The case when G is the complete graph is trivial since all vertices of a complete graph are simplicial. For the remainder of this proof, we will deal with the case where G is not the complete graph. We proceed by induction n (the number of vertices in G).

Base Case: $n = 1$. Trivial. If G has just one vertex, then this vertex is simplicial in G.

Inductive Hypothesis: Assume that for $n \leq k$, that G has 2 simplicial vertices that are not adjacent.

Inductive Step: Assume $n = k + 1$. We will show that G has 2 simplicial vertices that are not adjacent.

Since G is not complete, we can choose an edge $(a,b) \notin E$.

Recall that the notation $G[V - S]$ means the subgraph of G induced on the vertices $V - S$. Let S be a minimal set of $V - \{a, b\}$ such that $G[V - S]$ has a and b in different components A and B respectively. We call S a minimal separator of a and b.

Note: S exists since we could choose all neighbours of a and b to be in S.

Goal: We want to find a simplicial vertex in A and a simplicial vertex in B. We will show how to find a simplicial vertex in A. Finding a simplicial vertex in B is analagous.

Let G_{A+S} be the graph induced by $A \cup S$.

Case 1: G_{A+S} is complete. Therefore a is a simplicial vertex in G_{A+S}. Since $a \in A$, all neighbours of a are in G_{A+S} and a is also simplicial in G.

Case 2: G_{A+S} is not complete.

By induction, there are 2 nonadjacent simplicial vertices x,$y \in G_{A+S}$.

Since x and y are not adjacent, we have $(x,y) \notin E$.

Case 2a: One of x,y is in A, say $x \in A$. Similar to Case 1, since x is simplicial in G_{A+S} and all neighbours of x are in G_{A+S} then x is also simplicial in G.

Case 2b: $x \in S$ and $y \in S$. We will show that this can not happen. Assume for a contradiction that $x \in S$ and $y \in S$.

Claim: Both x and y have neighbours in both A and B.

Proof: Withour loss of generality, suppose x did not have a neighbour in A. We could then remove x from S to get a smaller set S' that separates components A and B. This is a contradiction to the minimality of S. Since S is minimal, we can not get a smaller set S'!

So, suppose x is connected to $a_x \in A$ and $b_x \in B$ and y is connected to $a_y \in A$ and $b_y \in B$. Because A is a connected component, there exists a path from x to y using vertices in A. Similarily for B, there exists a path from y to x using vertices in B. See also Figure 1.

Let the shortest path from x to y using vertices in A be be denoted by $(x, a_x, a_1, \ldots, a_k, a_y, y)$. Let the shortest path from y to x using vertices in B be denoted by $(y, b_y, b_1, \ldots, b_m, b_x, x)$.

Then $(x, a_x, a_1, \ldots, a_k, a_y, y, b_y, b_1, \ldots, b_m, b_x, x)$ is a cycle. This cycle has length ≥ 4 and it is the shortest cycle since we chose the shortest paths in components A and B.

Note: If $a_x = a_y$ and $b_x = b_y$ then our cycle is (x, a_x, y, b_x, x). This cycle has length 4.

In chordal graphs, all cycles of length ≥ 4 must have a chord. Since we have a cycle of length ≥ 4, then we must be able to find the chord contained within this cycle.

There is no chord in the cycle from x to y in either A or B since we chose the shortest paths from x to y in each component. Neither is there an edge from A to B since A and B are two different components. The only other possibility is for there to be a chord between x and y. But, the edge $(x,y) \notin E$!

Therefore, there is no chord in the cycle. Therefore x and y can not both be in S.

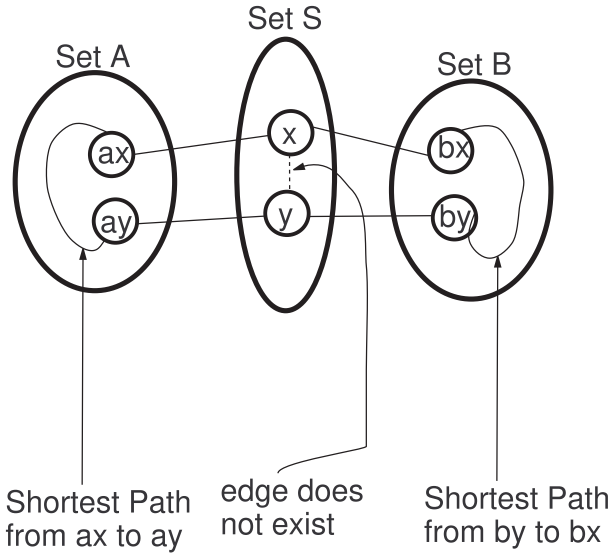


Figure 1:

Summary: We have showed how to find a simplicial vertex in A. In case 1, any vertex in A is simplicial. In Case 2a, x is simplicial in A.

The proof is symmetrical for finding the simplicial vertex in B.

Note that during the proof, we have used the fact that the graph induced by $A \cup S$ is chordal. This indeed holds because of the following observation: □

Corollary 1 *Every induced subgraph of a chordal graph is chordal*

Proof: Suppose we have a chordal graph G. Let H be any induced subgraph of G. Then, $V[H] \subseteq V[G]$ and $E[H] \subseteq E[G]$. Since we are not adding any more vertices or edges to H, we are not creating any new cycles in H.

Thus, if all cycles of length ≥ 4 have chords in G, then the same applies to all cycles of length ≥ 4 in H since we are not deleting any chords.

Using Theorem 1, we can now show the desired result. □

Corollary 2 *Every chordal graph has a perfect elimination order.*

Proof: Let v_n be a simplicial vertex of G, we know this exists by Theorem 1.

$G[V - v_n]$ is an induced subgraph of a chordal graph and therefore chordal and so it also has a perfect elimination order. This is essentially a proof by induction on the number of vertices in G.

Suppose that $(v_1, v_2, \ldots, v_{n-1})$ is a perfect elimination order of $G[V - v_n]$, then $(v_1, v_2, \ldots, v_{n-1}, v_n)$ is a perfect elimination order of G. This is true because v_n is a simplicial vertex and so all neighbours of v_n form a clique.

This proof suggests an iterative approach for finding a perfect elimination order.

This iterative algorithm was first suggested by Fulkerson and Gross in 1965 [FG65]. Fulkerson and Gross proposed that by running the algorithm, one of two things can happen:

- The algorithm runs until there are no vertices left. In this case, the vertex ordering is a perfect elimination order and the graph G is chordal.
- At some stage, no simplicial vertex exists. In this case, the original graph G is not chordal.

□

Corollary 3 *Independent Set, Colouring, Clique can be done in linear time for chordal graphs.*

Proof: The first step is to find a perfect elimination order. We will see in the next section how to do this in linear time. Once a perfect elimination order is found, solving Independent Set, Colouring, Clique can be done easily, as seen in previous lectures. □

3 Algorithms

In this section, we will see how to find a perfect elimination order in linear time. (i.e., $O(m + n)$ time)

3.1 Maximum Cardinality Search

Below is an algorithm for finding a perfect elimination order that was presented by Tarjan in 1976 [Tar76]. The algorithm works by numbering vertices from 1 to n. Vertices are chosen by choosing an unnumbered vertex that is adjacent to the most numbered vertices.

for i = 1, . . . , n
    Let v_i be the vertex such that $v_i \notin \{v_1, v_2, ..., v_{i-1}\}$ and
        v_i has the most neighbours in $\{v_1, v_2, ..., v_{i-1}\}$

One can show that the vertex order $(v_1, v_2, ..., v_n)$ found by this algorithm is a perfect elimination order. The proof is omitted.

3.2 Lexicographical BFS

Below is an algorithm for finding a perfect elimination order based on a lexicographical ordering of the nodes. The algorithm was presented by Lueker, Rose and Tarjan in 1976 [RL76]. As the name implies, this algorithm uses a lexicographical BFS to find the vertex ordering. The algorithm works by assigning labels to vertices and then choosing labels that are the "largest" lexicographical labels. The labels are used only for the selection process. Intuitively, a label L_a is bigger than label L_b if L_a would appear after L_b in a phone book.

For all vertices v, let $L(v) = \emptyset$ (label)
for i = n, . . . , 1
    Let v_i be the vertex such that $v_i \notin \{v_n, v_{n-1}, ..., v_{i+1}\}$ and
        v_i has the lexicographically largest label $L(v_i)$
    For all neighbours v of v_i, set $L(v) = L(v) \circ i$

One can show that the vertex order $(v_n, v_{n-1}, ..., v_1)$ found by this algorithm is a perfect elimination order. The proof will be given in the next lecture.

4 Algorithm Examples

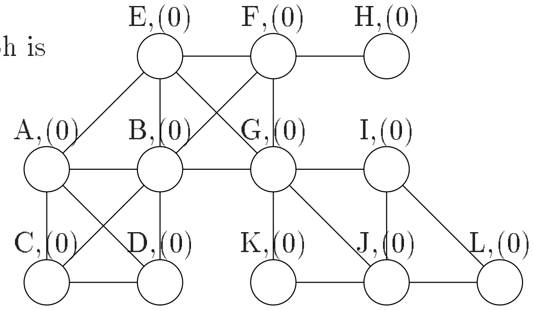
In this section, we will give full examples of the two algorithms given above. In the pictures, a grey vertex indicates a vertex that has already been selected. A vertex with a solid black line indicates the vertex currently being selected.

4.1 Maximum Cardinality Search (MCS)

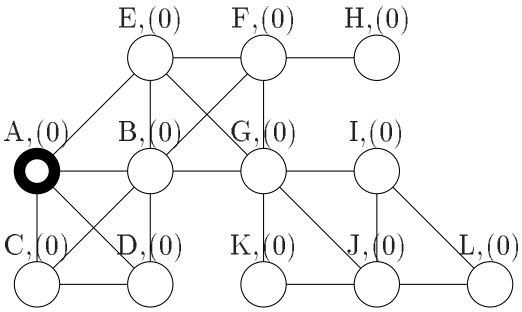
After each iteration i, the values of $v_1, v_2, \ldots, v_i$ are shown.

Node labels are [Node,(Number of neighbours already chosen)]. At each iteration, we are interested only in vertices that have not yet been chosen. Therefore, we do not need to track the number of neighbours for vertices already selected.

The initial graph is

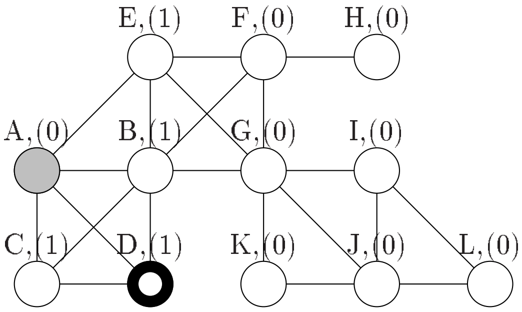


Iteration 1



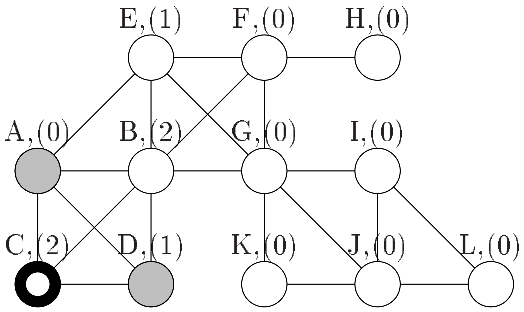
Possible choices for v_1 are {A,B,C,D,E,F,G,H,I,J,K,L}.
Choose v_1=A
Vertex order so far: {A}

Iteration 2



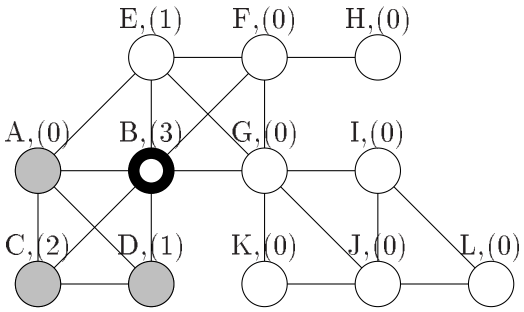
Possible choices for v_2 are {B,C,D,E}.
Choose v_2=D
Vertex order so far: {A,D}

Iteration 3



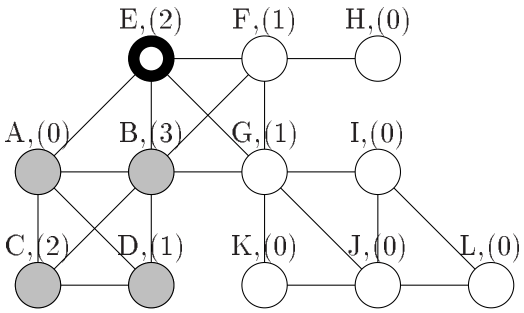
Possible choices for v_3 are {B,C}.
Choose v_3=C
Vertex order so far: {A,D,C}

Iteration 4



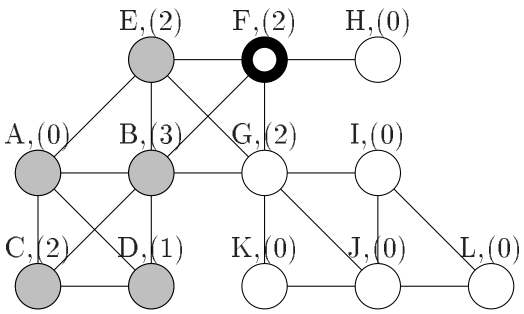
Possible choices for v_4 are {B}.
Choose v_4=B
Vertex order so far: {A,D,C,B}

Iteration 5



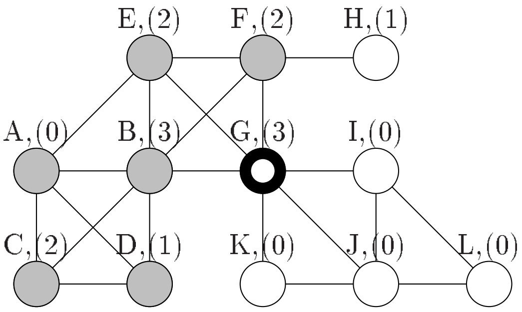
Possible choices for v_5 are {E}.
Choose v_5=E
Vertex order so far: {A,D,C,B,E}

Iteration 6



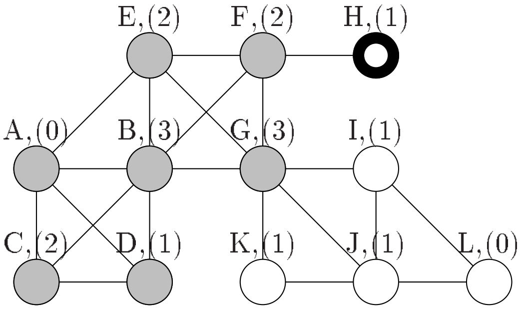
Possible choices for v_6 are {F,G}.
Choose v_6=F
Vertex order so far: {A,D,C,B,E,F}

Iteration 7



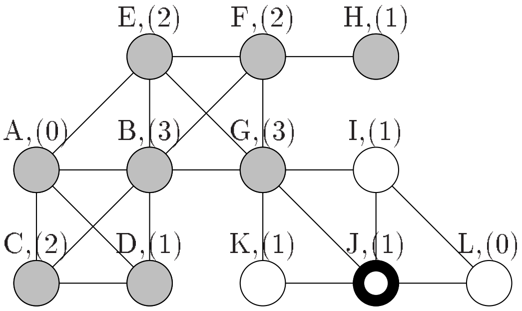
Possible choices for v_7 are {G}.
Choose v_7=G
Vertex order so far: {A,D,C,B,E,F,G}

Iteration 8



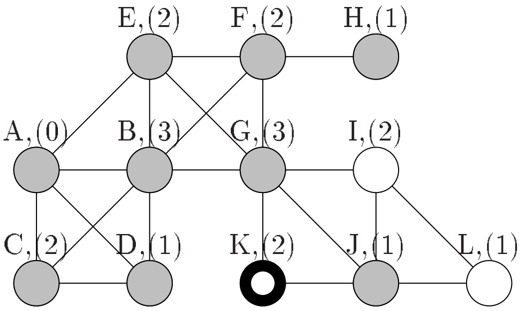
Possible choices for v_8 are {H,I,J,K}.
Choose v_8=H
Vertex order so far: {A,D,C,B,E,F,G,H}

Iteration 9



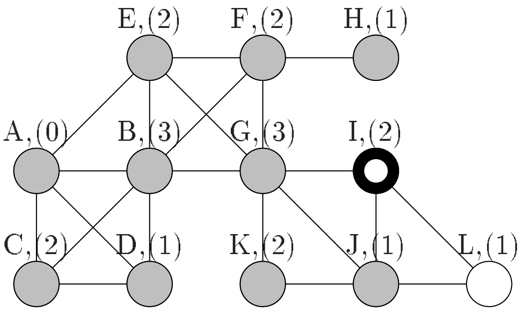
Possible choices for v_9 are {I,J,K}.
Choose v_9=J
Vertex order so far: {A,D,C,B,E,F,G,H,J}

Iteration 10



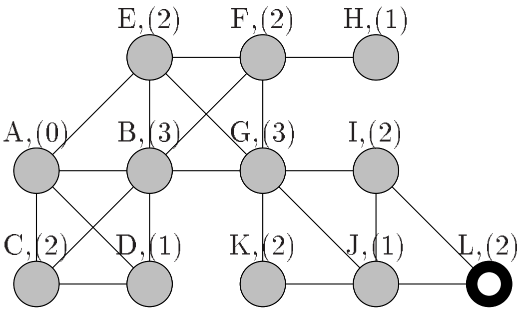
Possible choices for v_{10} are {I,K}.
Choose v_{10}=K
Vertex order so far: {A,D,C,B,E,F,G,H,J,K}

Iteration 11



Possible choices for v_{11} are {I}.
Choose v_{11}=I
Vertex order so far: {A,D,C,B,E,F,G,H,J,K,I}

Iteration 12



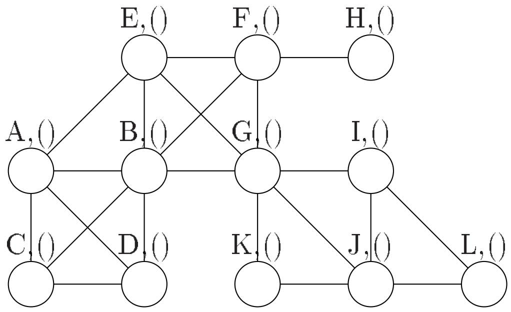
Possible choices for v_{12} are {L}.
Choose v_{12}=L
Vertex order so far: {A,D,C,B,E,F,G,H,J,K,I,L}

4.2 Lexicographical BFS

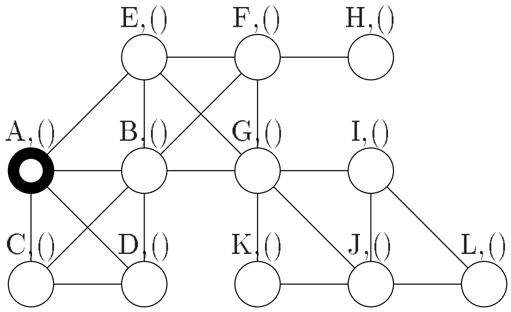
After each iteration i, the values of $\{v_n, v_{n-1}, ..., v_{n-i+1}\}$ are shown

Node labels are [Node,(Label)]. At each iteration, we are interested only in vertices that have not yet been chosen. Therefore, we do not need to track the label for vertices already selected.

The initial graph is

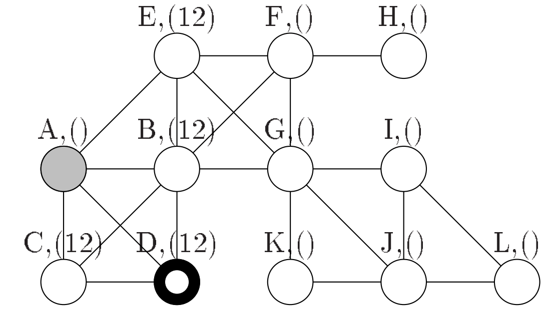


Iteration 12



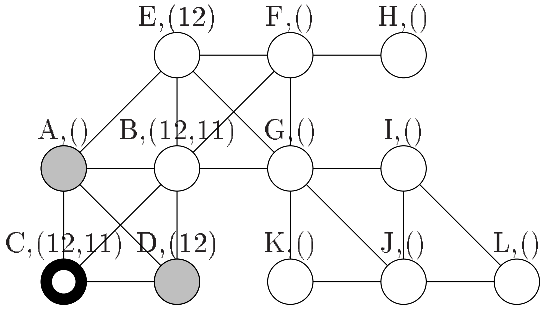
Possible choices for v_{12} are {A,B,C,D,E,F,G,H,I,J,K,L}.
Choose v_{12}=A
Vertex order so far: {A}

Iteration 11



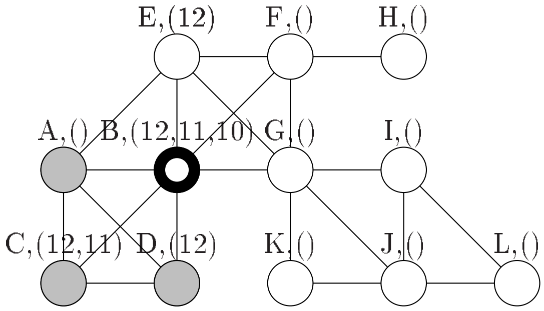
Possible choices for v_{11} are {B,C,D,E}.
Choose v_{11}=D
Vertex order so far: {A,D}

Iteration 10



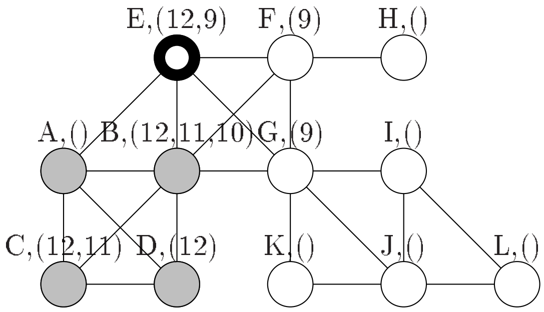
Possible choices for v_{10} are {B,C}.
Choose v_{10}=C
Vertex order so far: {A,D,C}

Iteration 9



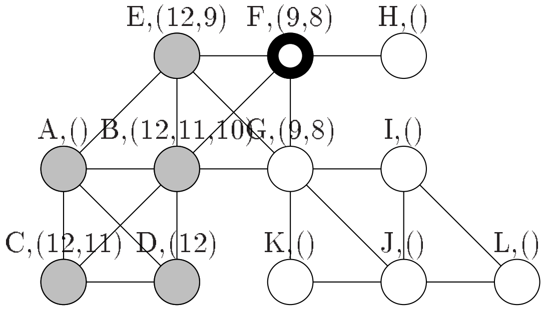
Possible choices for v_9 are {B}.
Choose v_9=B
Vertex order so far: {A,D,C,B}

Iteration 8



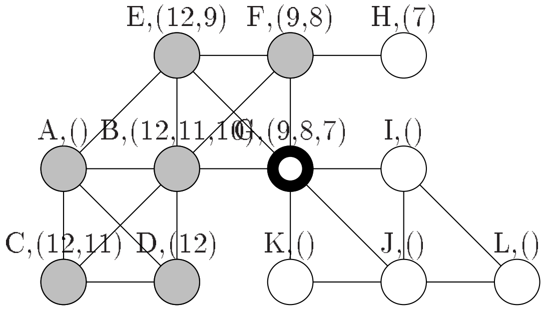
Possible choices for v_8 are {E}.
Choose v_8=E
Vertex order so far: {A,D,C,B,E}

Iteration 7



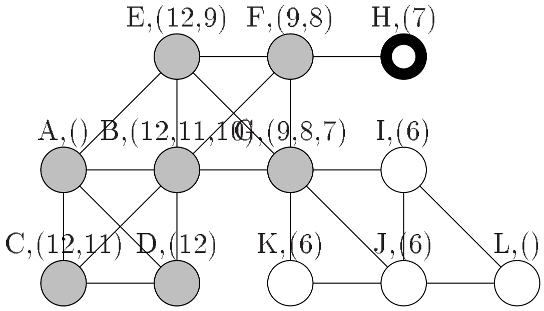
Possible choices for v_7 are {F,G}.
Choose v_7=F
Vertex order so far: {A,D,C,B,E,F}

Iteration 6



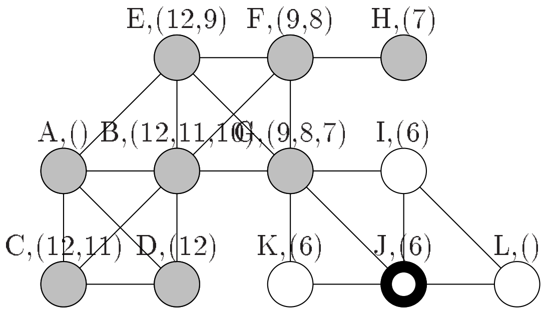
Possible choices for v_6 are {G}.
Choose v_6=G
Vertex order so far: {A,D,C,B,E,F,G}

Iteration 5



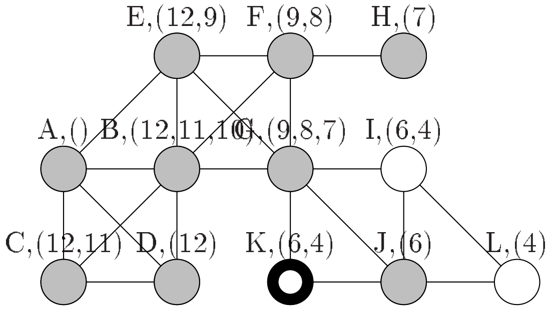
Possible choices for v_5 are {H}.
Choose v_5=H
Vertex order so far: {A,D,C,B,E,F,G,H}

Iteration 4



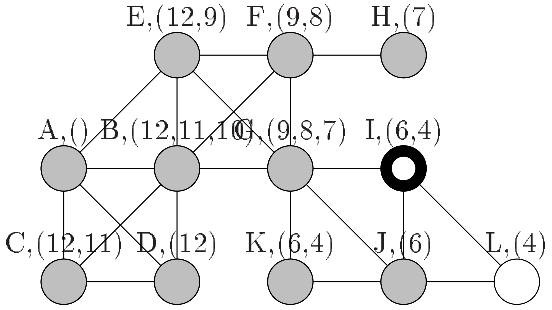
Possible choices for v_4 are {I,J,K}.
Choose v_4=J
Vertex order so far: {A,D,C,B,E,F,G,H,J}

Iteration 3



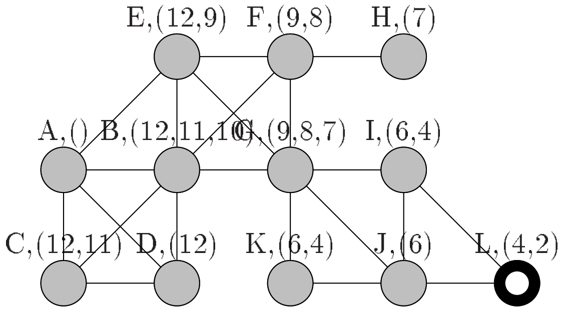
Possible choices for v_3 are {I,K}.
Choose v_3=K
Vertex order so far: {A,D,C,B,E,F,G,H,J,K}

Iteration 2



Possible choices for v_2 are {I}.
Choose v_2=I
Vertex order so far: {A,D,C,B,E,F,G,H,J,K,I}

Iteration 1



Possible choices for v_1 are {L}.
Choose v_1=L
Vertex order so far: {A,D,C,B,E,F,G,H,J,K,I,L}

Note that in these two examples the two perfection elimination orders are the same, but this is not always the case.

Also, one can show that there are perfect elimination orders which are obtained with neither MCS nor lex BFS. This is left as the homework for today.

References

[FG65] D. R. Fulkerson and O. A. Gross. *Incidence matrices and interval graphs.* Pacific J. Math, 1965.

[Gol80] Martin Charles Golumbic. *Algorithmic graph theory and perfect graphs.* Academic Press, New York, 1980.

[LB62] C. G. Lekkerkerker and J. Ch. Boland. *Representation of a finite graph by a set of intervals on the real line.* Fund, Math, 1962.

[RL76] Tarjan Robert Endre Rose, Donald J. and George S. Lueker. *Algorithmic aspects of vertex elimination on graphs.* SIAM J. Comput, 1976.

[Tar76] Robert Endre Tarjan. *Maximum cardinality search and chordal graphs. Unpublished Lecture Notes CS 259.* Stanford Univ., 1976.