Advanced Graph Algorithms

Spring 2003

Gabriel Valiente

valiente@lsi.upc.es

Technical University of Catalonia

Schedule

1.	Introduction	13 May 2003	
2.	Intersection Graph Theory	27 May 2003	
3.	Algorithms on Permutation Graphs		28 May 2003
4.	Algorithms on Circle Graphs		28 May 2003
5.	Algorithms on Interval Graphs	03 June 2003	
6.	Algorithms on Chordal Graphs		04 June 2003
7.	Algorithms on Circular-Arc Graphs		04 June 2003
8.	Student Presentations	10 June 2003	11 June 2003

Introduction

Introduction

- Most graph problems of theoretical interest and practical relevance are intractable, even hard to approximate.
- However, NP-hard problems become polynomial-time solvable on special graph classes.
- A closer look will be taken in this course at some fundamental graph problems (clique, independent set, coloring, isomorphism).
  - Understanding the structure of the graph classes that make them polynomial-time solvable.
  - Drawing the boundary between P and NP-hard for them.
  - Studying robust (certifying) algorithms.
  - Identifying open problems.
- Graph classes will be studied from the point of view of intersection graph theory.

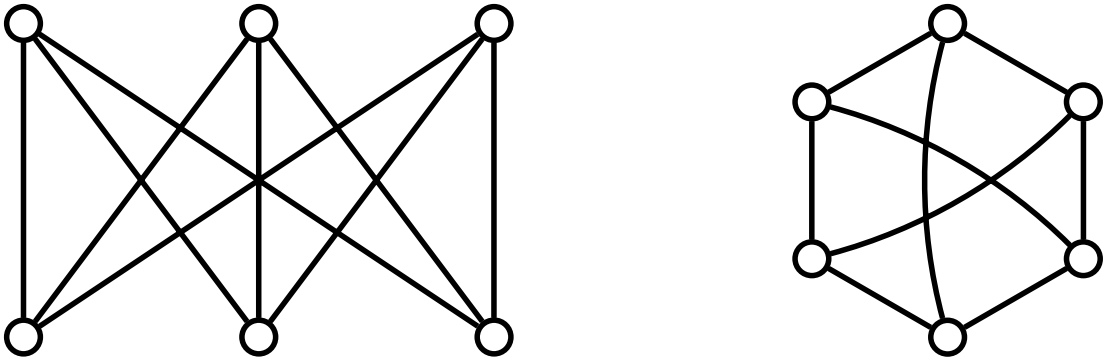
Introduction

Isomorphism expresses what is meant when two graphs are said to be the same graph.

Definition *Two graphs G and H are isomorphic, denoted by $G \cong H$, if there is a bijection $h : V(G) \to V(H)$ such that, for every pair of vertices $u, v \in V(G)$, $\{u,v\} \in E(G)$ if and only if $\{h(u), h(v)\} \in E(H)$.*

Two isomorphic graphs may be depicted in such a way that they look very different.

Example *The following two graphs are isomorphic.*



Introduction

Nonisomorphism of graphs is not usually hard to prove, because several invariants or necessary conditions for isomorphism are not difficult to compute. These are properties that do not depend on the presentation or labeling of a graph.

Definition *An graph isomorphism invariant is a necessary condition for two graphs to be isomorphic.*

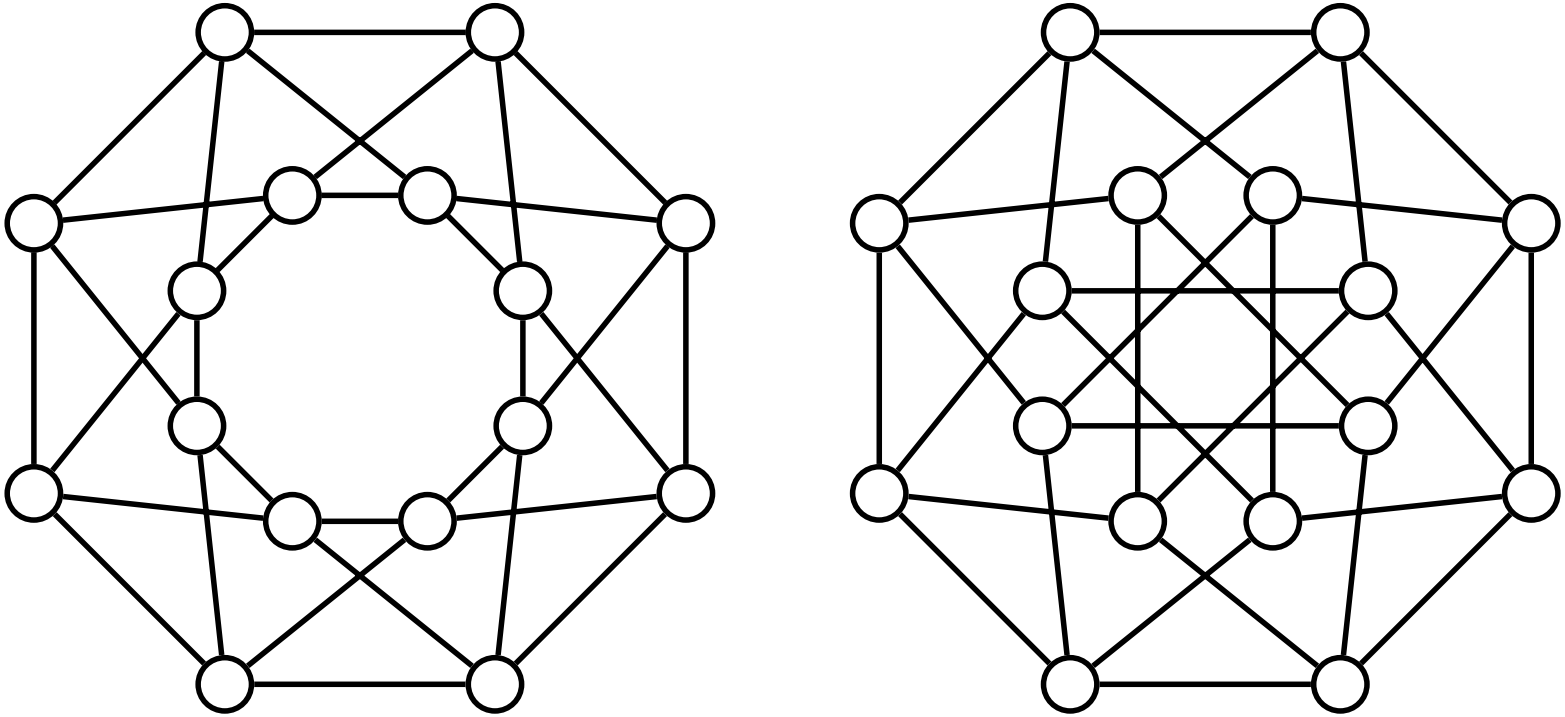
Example *Two graphs cannot be isomorphic if they differ in their order, size, or degree sequence.*

Remark *For input graphs G and H with $V(G) = \{u_1, \ldots, u_n\}$ and $V(H) = \{v_1, \ldots, v_n\}$, a necessary condition for $G \cong H$ is that the multisets $\{\Gamma(u_i) \mid 1 \leqslant i \leqslant n\}$ and $\{\Gamma(v_i) \mid 1 \leqslant i \leqslant n\}$ be equal.*

Introduction

Invariants are not sufficient conditions for isomorphism. Nothing can be concluded about two graphs which share an invariant.

Example *The following two graphs are not isomorphic, although they have the same number of vertices, the same number of edges, and are both* 4*-regular.*



Introduction

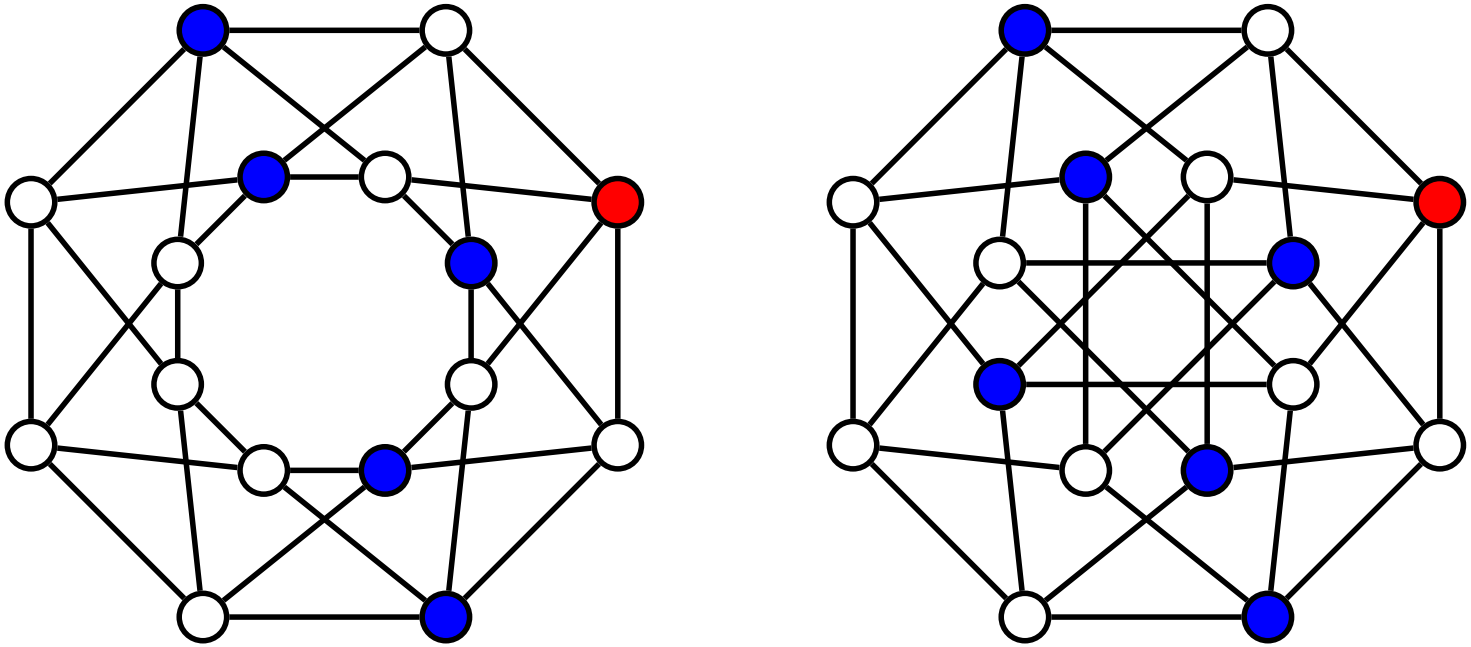
Example *The following two graphs are not isomorphic, although they have the same number of vertices, the same number of edges, and are both 4-regular.*

$$E(G) = \left\{ \begin{array}{l} \{u_0,u_1\},\{u_0,u_2\},\{u_0,u_4\},\{u_0,u_6\},\{u_1,u_3\},\{u_1,u_5\},\{u_1,u_7\}, \\ \{u_2,u_3\},\{u_2,u_8\},\{u_2,u_{10}\},\{u_3,u_4\},\{u_3,u_6\},\{u_4,u_5\},\{u_4,u_7\}, \\ \{u_5,u_9\},\{u_5,u_{13}\},\{u_6,u_8\},\{u_6,u_{10}\},\{u_7,u_9\},\{u_7,u_{13}\}, \\ \{u_8,u_{11}\},\{u_8,u_{14}\},\{u_9,u_{12}\},\{u_9,u_{15}\},\{u_{10},u_{11}\},\{u_{10},u_{14}\}, \\ \{u_{11},u_{12}\},\{u_{11},u_{15}\},\{u_{12},u_{13}\},\{u_{12},u_{14}\},\{u_{13},u_{15}\},\{u_{14},u_{15}\} \end{array} \right\}$$

$$E(H) = \left\{ \begin{array}{l} \{v_0,v_1\},\{v_0,v_2\},\{v_0,v_4\},\{v_0,v_6\},\{v_1,v_3\},\{v_1,v_5\},\{v_1,v_7\}, \\ \{v_2,v_3\},\{v_2,v_8\},\{v_2,v_{10}\},\{v_3,v_9\},\{v_3,v_{11}\},\{v_4,v_5\},\{v_4,v_8\}, \\ \{v_4,v_{12}\},\{v_5,v_9\},\{v_5,v_{13}\},\{v_6,v_7\},\{v_6,v_{10}\},\{v_6,v_{12}\}, \\ \{v_7,v_{11}\},\{v_7,v_{13}\},\{v_8,v_9\},\{v_8,v_{14}\},\{v_9,v_{15}\},\{v_{10},v_{11}\}, \\ \{v_{10},v_{14}\},\{v_{11},v_{15}\},\{v_{12},v_{13}\},\{v_{12},v_{14}\},\{v_{13},v_{15}\},\{v_{14},v_{15}\} \end{array} \right\}$$

Introduction

Example *The following two graphs are not isomorphic, although they have the same number of vertices, the same number of edges, and are both 4-regular.*



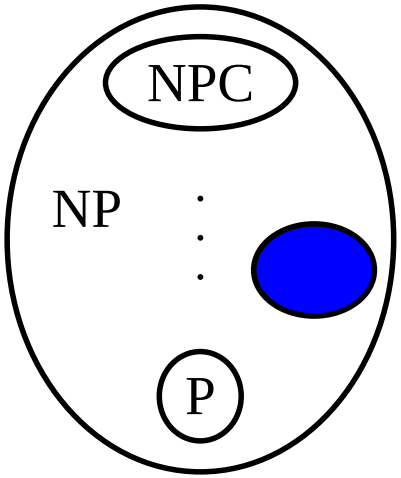
There are, in fact, only five vertices at distance 2 of any given vertex of the graph to the left-hand side, but six vertices at distance 2 of any given vertex of the graph to the right-hand side.

Introduction

Isomorphism of graphs is usually much harder to prove than nonisomorphism of graphs, because all known certificates or necessary and sufficient conditions for graph isomorphism are as difficult to compute as graph isomorphism itself.

Definition *An graph isomorphism certificate is a necessary and sufficient condition for two graphs to be isomorphic.*

From a complexity-theoretical point of view, graph isomorphism is one of the few NP problems believed neither to be in P nor to be NP-complete.



Introduction

Definition *A subgraph isomorphism of a graph G into a graph H is an injection $h : V(G) \to V(H)$ such that, for every pair of vertices $u, v \in V(G)$, $\{h(u), h(v)\} \in E(H)$ if $\{u, v\} \in E(G)$. In an induced subgraph isomorphism, $\{u, v\} \in E(G)$ if and only if $\{h(u), h(v)\} \in E(H)$.*

Definition *The subgraph isomorphism problem is to determine, given two input graphs G and H, whether H has a subgraph which is isomorphic to G.*

Remark *Subgraph isomorphism is a common generalization of many important graph problems.*

- *Clique (K_n)*
- *Independent set (nK_1)*
- *Hamiltonian cycle (C_n)*
- *Matching (nK_2)*
- *Girth (P_n)*
- *Shortest path (P_n)*

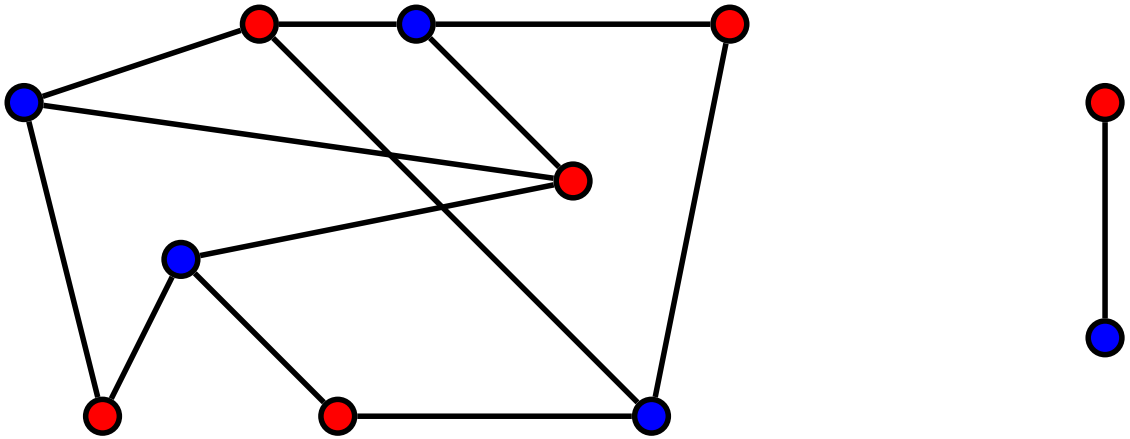
Introduction

Definition *Two graphs G and H are homomorphic, denoted by $G \preccurlyeq H$, if there is a mapping $h : V(G) \to V(H)$ such that, for every pair of vertices $u, v \in V(G)$, $\{h(u), h(v)\} \in E(H)$ if $\{u, v\} \in E(G)$.*

Definition *The subgraph homomorphism problem is to determine, given two input graphs G and H, whether H has a subgraph which is homomorphic to G.*

Remark *Graph homomorphism is a generalization of graph coloring, in which adjacent vertices obtain adjacent colors.*

Example *$G \preccurlyeq K_n$ if and only if G is n-colorable.*



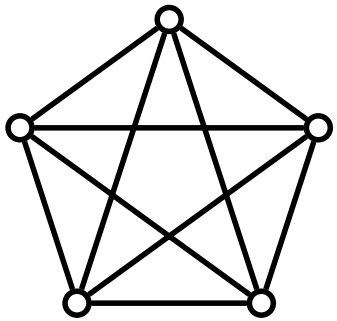
Introduction

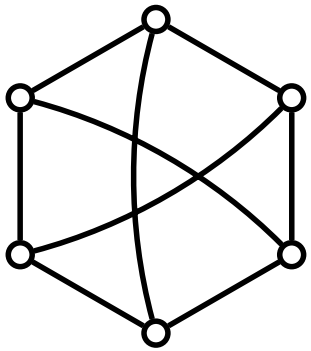
Definition *A graph G is homeomorphic to a graph H if G can be obtained from H by deleting degree-two vertices.*

Definition *The subgraph homeomorphism problem is to determine, given two input graphs G and H, whether H has a subgraph which is homeomorphic to G.*

Remark *Subgraph homeomorphism is the recognition problem for classes of graphs characterized by the absence of forbidden structures.*

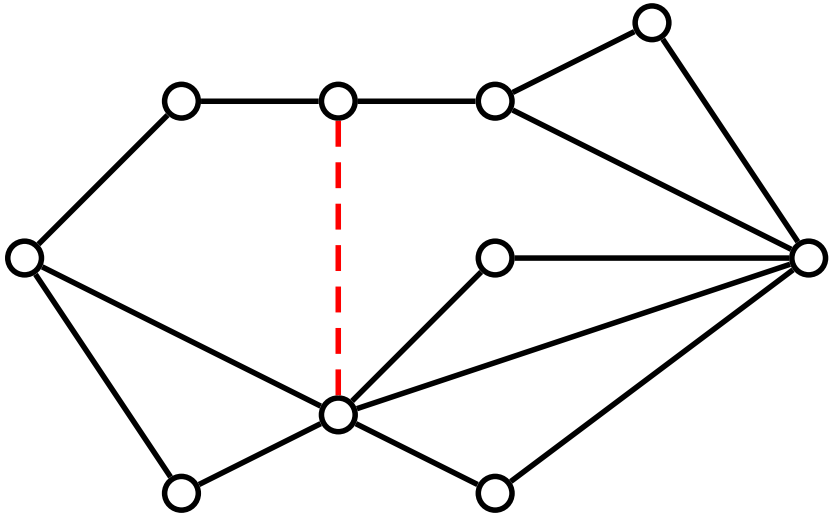
Example (Kuratowski, 1930) *Planar graphs are characterized by the absence of a subdivision of either K_5 or $K_{3,3}$.*





Introduction

Example *Series-parallel graphs are characterized by the absence of a subdivision of* K_4.



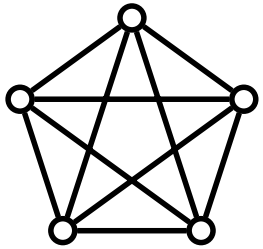
Introduction

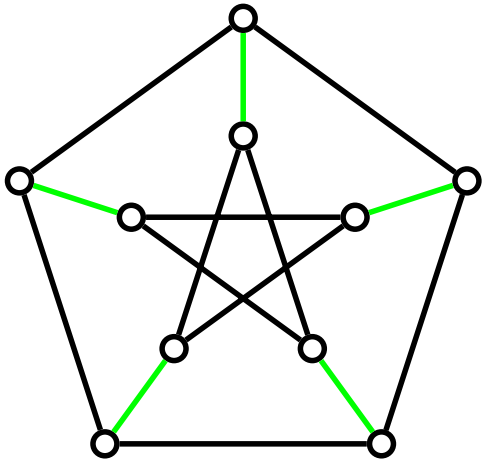
Definition *A graph G is a minor of a graph H if G can be obtained from a subgraph of H by contracting edges.*

Definition *The minor containment problem is to determine, given two input graphs G and H, whether H (has a subgraph which) is a minor of G.*

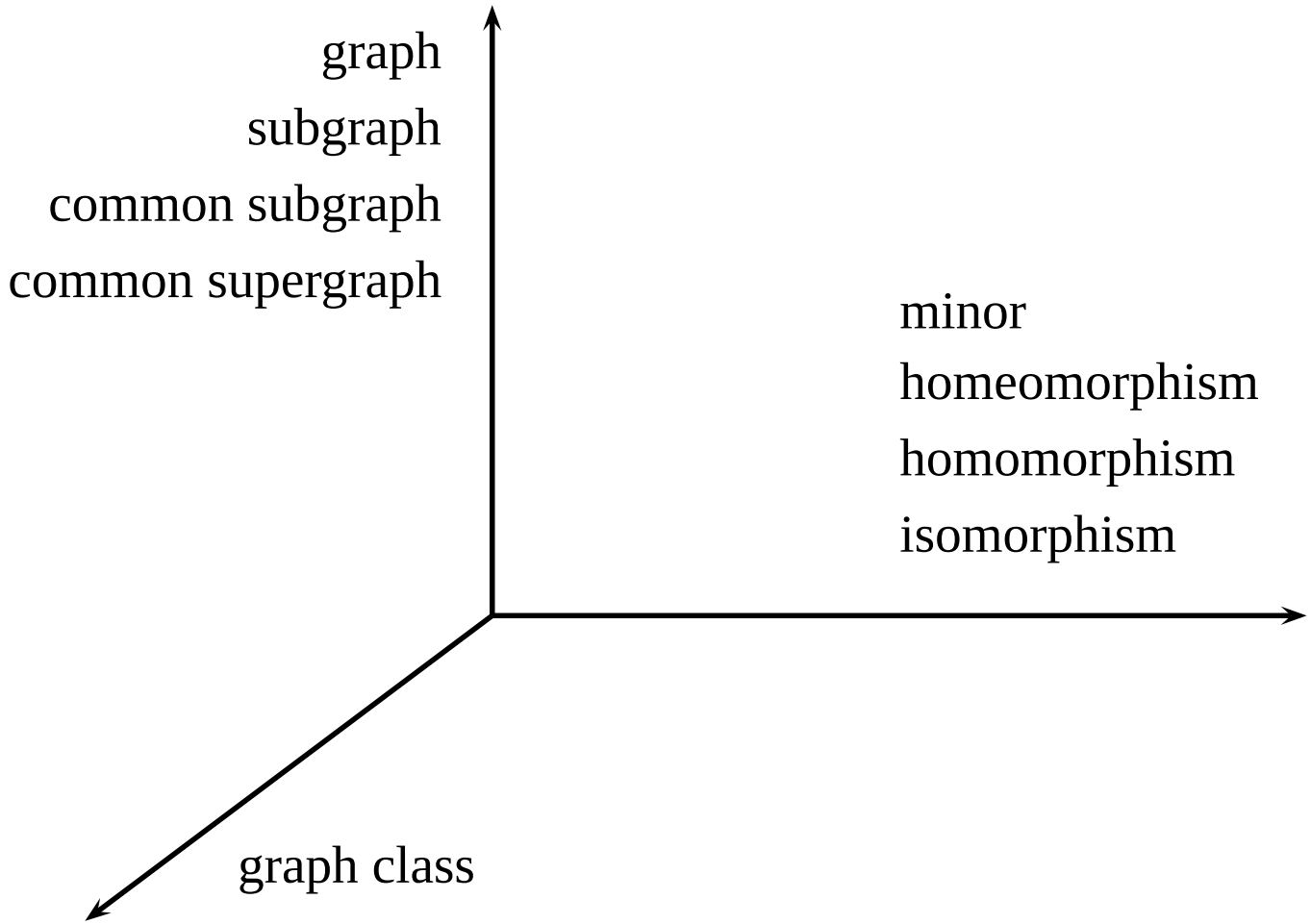
Remark *Minor containment is the recognition problem for classes of graphs characterized by the absence of forbidden minors.*

Example *K_5 is a minor of the Petersen graph.*





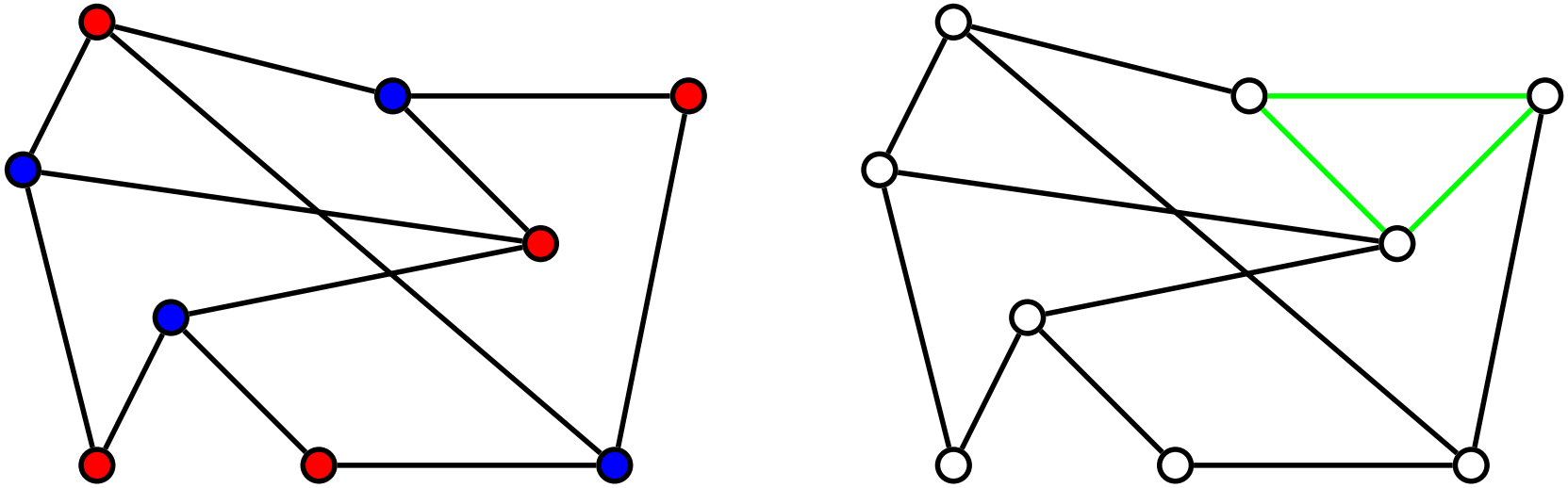
Introduction



Introduction

Definition *A certifying algorithm for a decision problem is an algorithm that provides a certificate with each answer that it produces.*

Example *Consider the problem of recognizing bipartite graphs. Given an input graph G, a 2-coloring of G is an acceptance certificate, while an odd cycle in G is a rejection certificate.*



Remark *Certifying algorithms are also called robust.*

Introduction

- Martin C. Golumbic, Algorithmic Graph Theory and Perfect Graphs, New York: Academic Press, 1980.
- Klaus Simon, Effiziente Algorithmen für perfekte Graphen, Stuttgart: B. G. Teubner, 1992.
- Johannes Köbler, Uwe Schöning, and Jacobo Turán. The Graph Isomorphism Problem: its Structural Complexity. Boston: Birkhäuser, 1993.
- Ron Shamir. Advanced Topics in Graph Algorithms. Technical Report, Tel-Aviv University, 1994.
- Terry A. McKee and Fred R. McMorris, Topics in Intersection Graph Theory, Philadelphia: SIAM, 1999.
- Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad, Graph Classes: A Survey, Philadelphia: SIAM, 1999.
- Gabriel Valiente. Algorithms on Trees and Graphs. Berlin: Springer-Verlag, 2002.

Intersection Graph Theory

Intersection Graph Theory

Definition *The intersection graph of a multiset of sets $\mathcal{F} = \{S_1, \ldots, S_n\}$, denoted by $\Omega(\mathcal{F})$, is the graph having $\mathcal{F}$ as vertex set with S_i adjacent to S_j if and only if $i \neq j$ and $S_i \cap S_j \neq \emptyset$. A graph G is called an intersection graph if there exists a multiset of sets $\mathcal{F}$ such that $G \cong \Omega(\mathcal{F})$.*

Example *Let $\mathcal{F} = \{S_1, S_2, S_3, S_4\}$, where $S_1 = \{x_1\}$, $S_2 = \{x_1, x_2, x_3\}$, $S_3 = \{x_4\}$, and $S_4 = \{x_1, x_3, x_4, x_5\}$. Then, $G \cong \Omega(\mathcal{F})$ is depicted next.*

Theorem (Marczewski, 1945) *Every graph is an intersection graph.*

Theorem *Every graph is the intersection graph of a family of subgraphs of a graph.*

Intersection Graph Theory

Definition *A chordal graph is the intersection graph of a finite set of subtrees of a tree.*

Definition *A circular-arc graph is the intersection graph of a finite set of arcs along a circle.*

Definition *An interval graph is the intersection graph of a finite set of intervals along a line.*

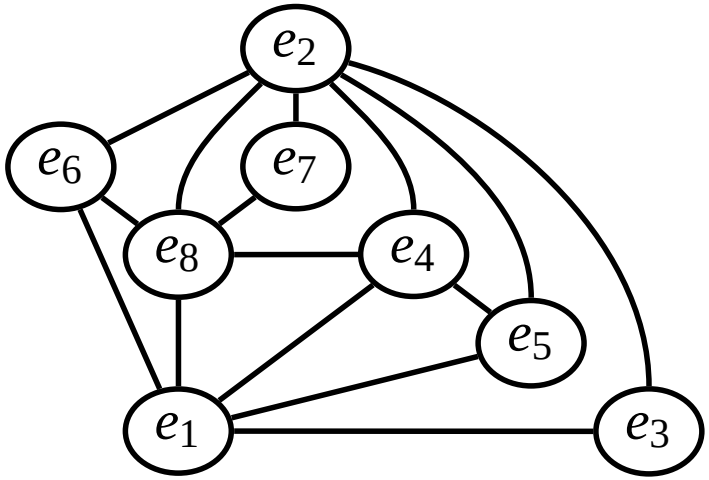
Definition *A circle graph is the intersection graph of a finite set of chords of a circle.*

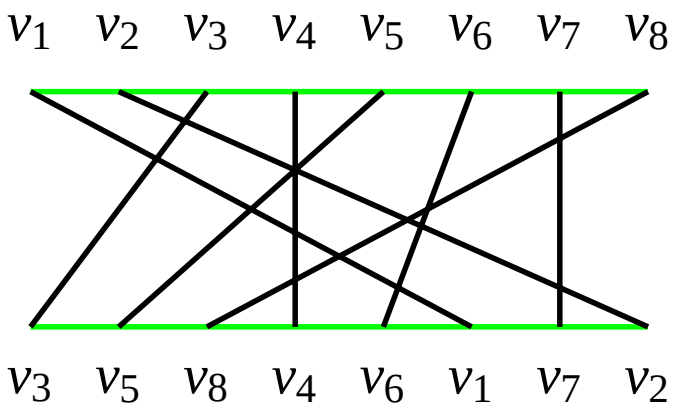
Definition *A permutation graph is the intersection graph of a finite set of chords between an ordered set of vertices and a permutation of them.*

Intersection Graph Theory

Definition *A permutation graph is a graph that is isomorphic to the intersection graph of a finite set of chords between an ordered set of vertices and a permutation of them.*

Example *The following finite set of chords between an ordered set of vertices and a permutation of them is a permutation model or matching diagram of the permutation graph shown to the left.*



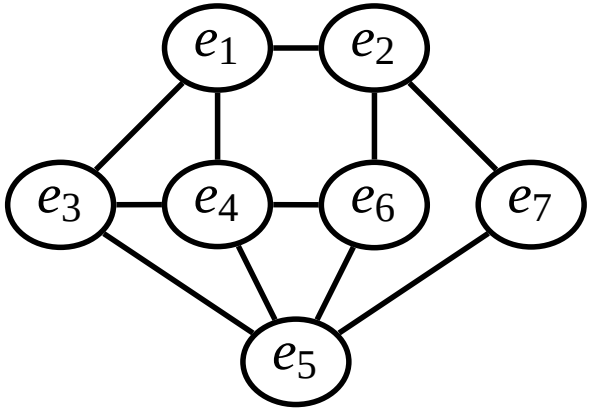


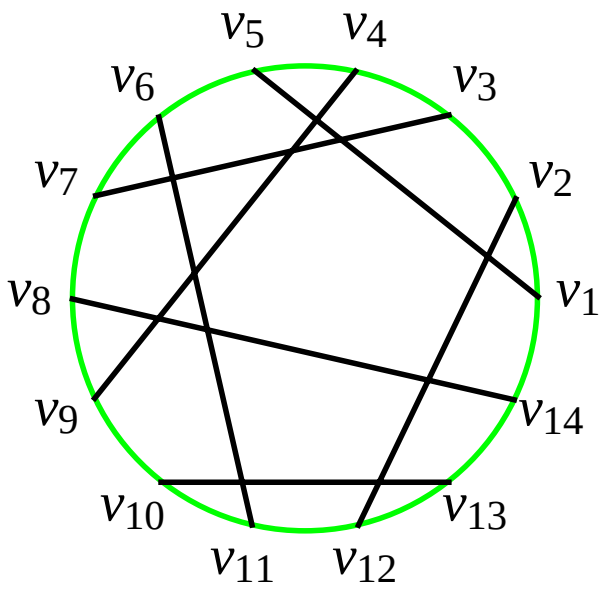
Remark *A permutation graph is the graph of inversions in a permutation.*

Intersection Graph Theory

Definition *A circle graph is a graph that is isomorphic to the intersection graph of a finite set of chords of a circle.*

Example *The following finite set of chords of a circle is a chord model of the circle graph shown to the left.*

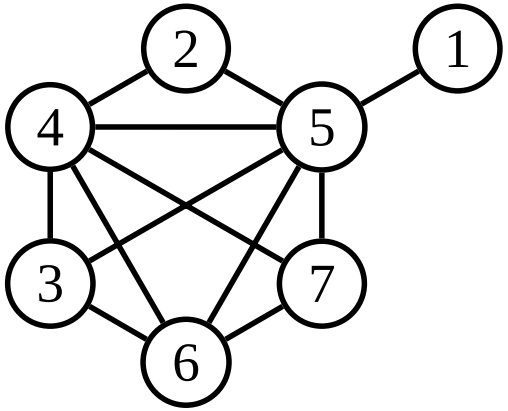


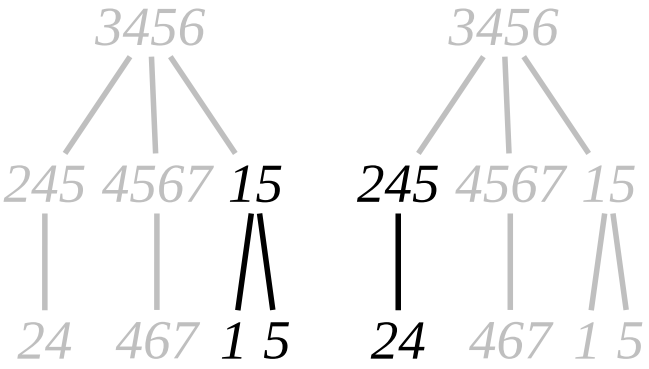


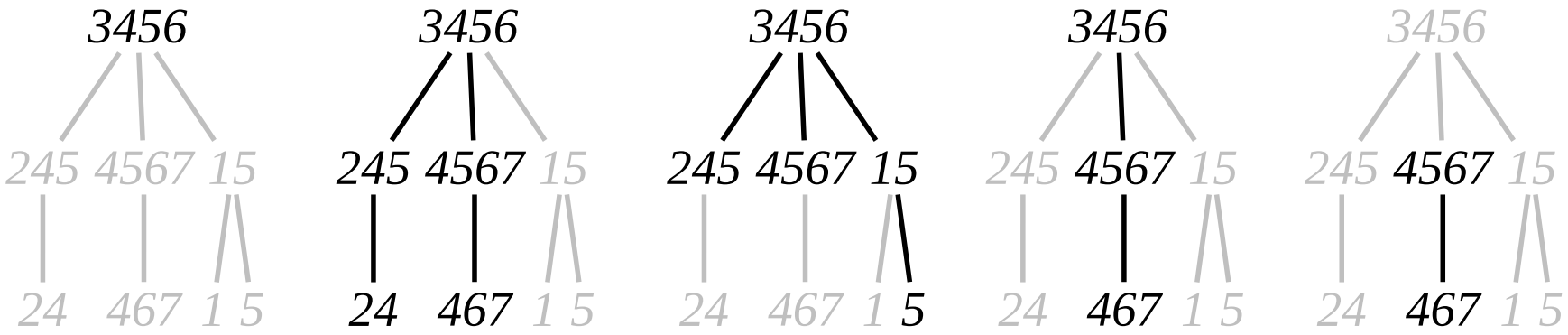
Intersection Graph Theory

Definition *A chordal graph is a graph that is isomorphic to the intersection graph of a finite set of subtrees of a tree.*

Example *The following family of subtrees of a tree is a subtree model of the chordal graph shown to the left.*



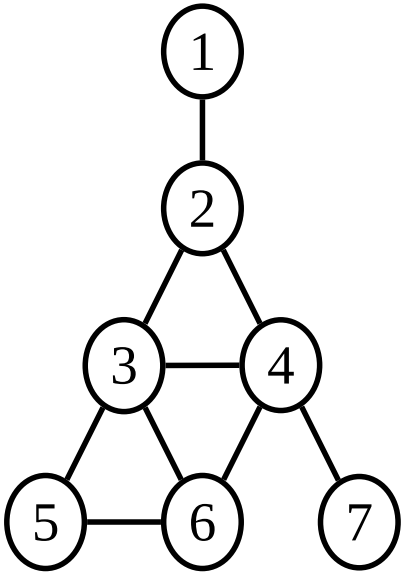


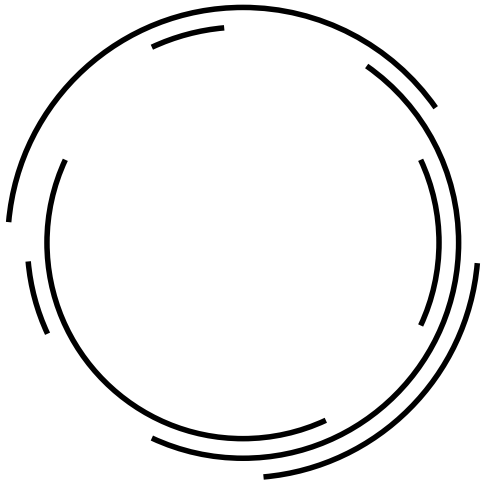


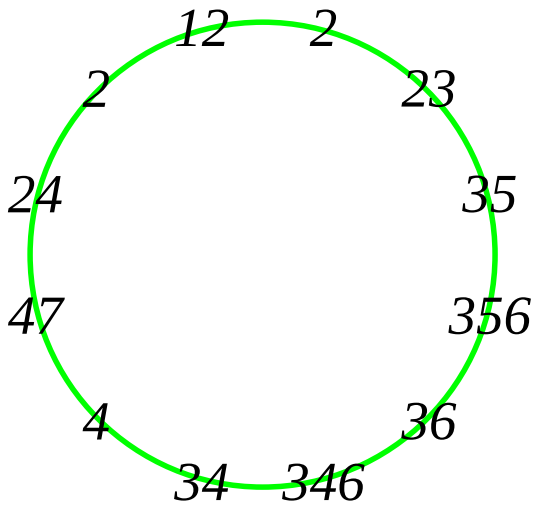
Intersection Graph Theory

Definition *A circular-arc graph is a graph that is isomorphic to the intersection graph of a finite set of arcs along a circle.*

Example *The following finite set of arcs along a circle is an arc model of the circular-arc graph shown to the left.*



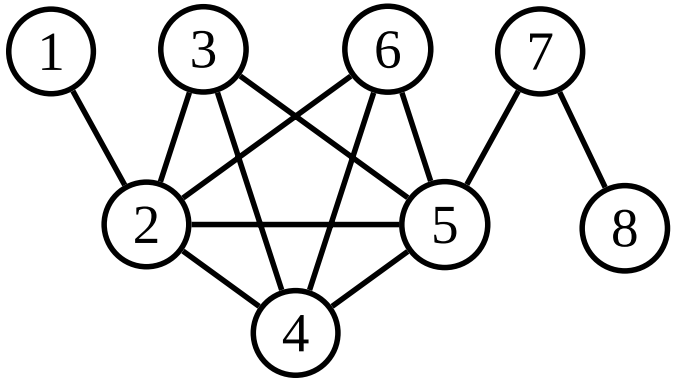


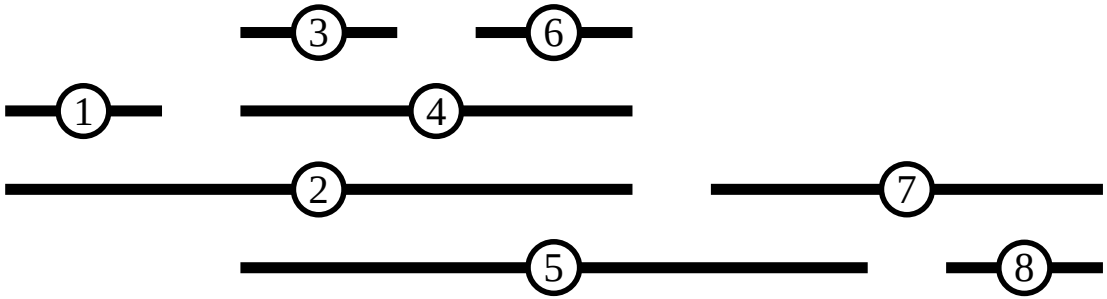


Intersection Graph Theory

Definition *An interval graph is a graph that is isomorphic to the intersection graph of a finite set of intervals along a line.*

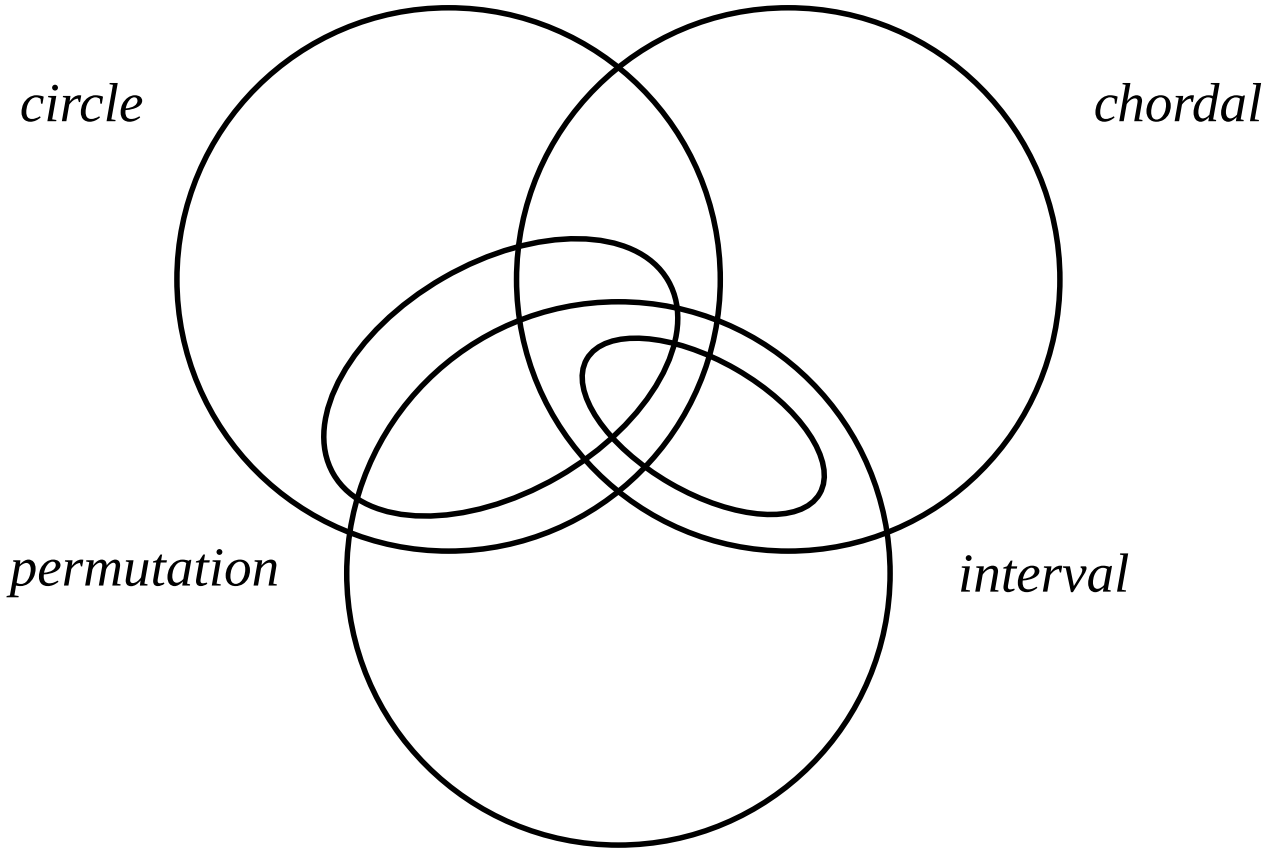
Example *The following finite set of intervals along a line is an interval model of the interval graph shown to the left.*





Intersection Graph Theory

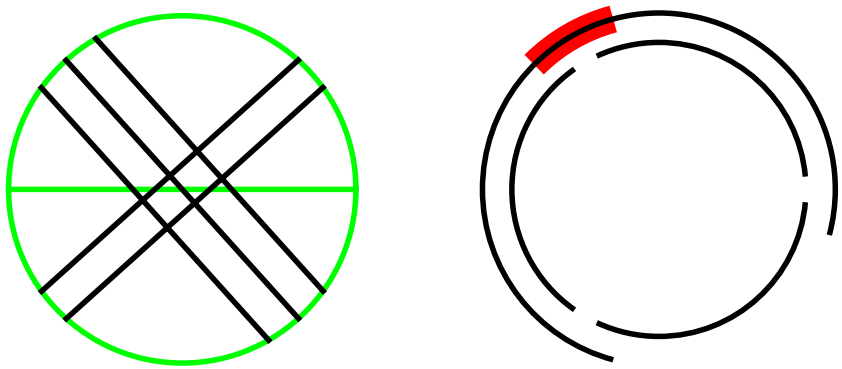
Lemma *The following diagram summarizes the relationships that hold between the previous five graph classes.*



Intersection Graph Theory

Exercise *Prove the previous lemma. Show that the claimed relationships between the five graph classes hold, or give at most 32 examples of graphs that belong to some of the previous five graph classes.*

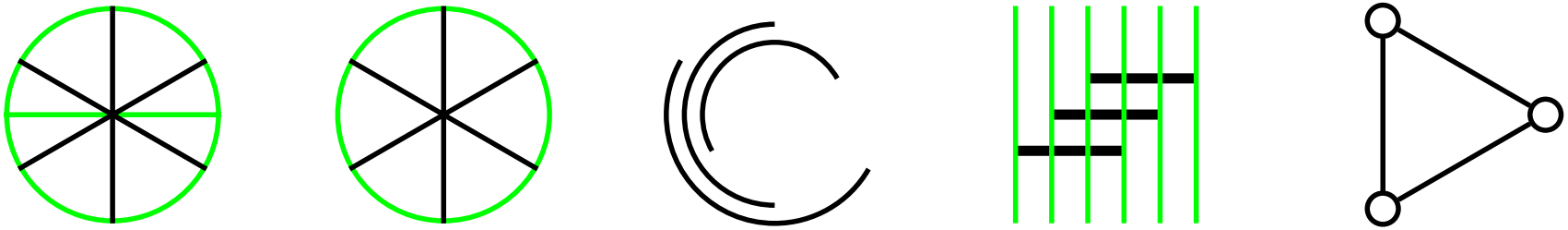
Example *A permutation graph is a circle graph, but it need not be a circular-arc graph. Consider, for instance, the intersection graph of three noncrossing chords which all cross two noncrossing chords along the circle graph with equator. This permutation graph is isomorphic to* $K_{2,3}$ *and has no circular-arc model, because two arcs along the circle cannot both intersect three nonintersecting arcs along the circle without intersecting themselves.*



Intersection Graph Theory

Example (Hajós, 1957) *Interval graphs are chordal. As a matter of fact, Let* $G = (V, E)$ *be an interval graph having a chordless cycle* $[v_0, v_1, v_2, \ldots, v_{\ell-1}, v_0]$ *with* $\ell > 3$*, and let* I_k *denote the interval corresponding to vertex* v_k*. Choose a point* $p_i \in I_{i-1} \cap I_i$*, for* $i = 1, 2, \ldots, \ell - 1$*. Since* I_{i-1} *and* I_{i+1} *do not overlap, the points* p_i *constitute a strictly increasing or strictly decreasing sequence. Therefore, it is impossible for* I_0 *and* $I_{\ell-1}$ *to intersect, contradicting the assumption that* $\{v_0, v_{\ell-1}\} \in E$*.*

Example K_n *has a permutation model, a circle model, a circular-arc model, an interval model, and is chordal.*



Student Presentations

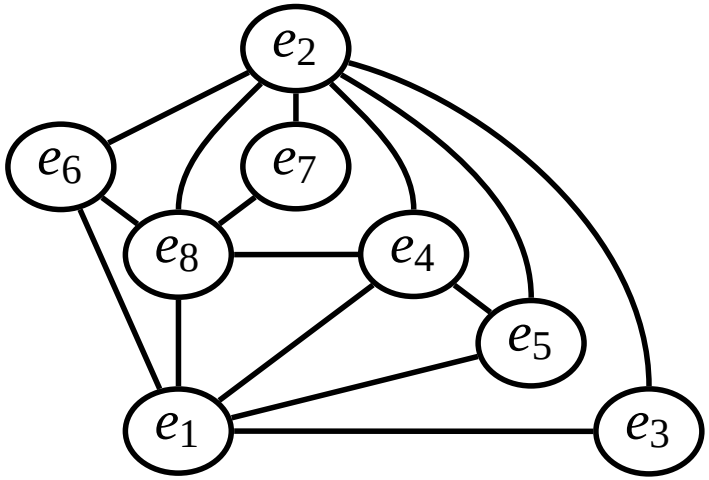
- Prove the claimed relationships between the five classes of intersection graphs (chordal, circle, circular-arc, interval, and permutation graphs).
- Wen-Lian Hsu, Ross M. McConnell. PC Trees and Circular-Ones Arrangements. Theoretical Computing Science, 296(1):99–116, 2003.
- Wen-Lian Hsu. A Simple Test for the Consecutive Ones Property. Journal of Algorithms, 43(1):1–16, 2002.
- Emmanuel Gasse. A Proof of a Circle Graph Characterization. Discrete Mathematics, 173(1–3):277-283, 1997.
- Stephan Olariu. An Optimal Greedy Heuristic to Color Interval Graphs. Information Processing Letters, 37(1):21–25, 1991.
- Wei-Kuan Shih, T. C. Chern, Wen-Lian Hsu. An $O(n^2 \log n)$ Algorithm for the Hamiltonian Cycle Problem on Circular-Arc Graphs. SIAM Journal on Computing, 21(6):1026–1046, 1992.

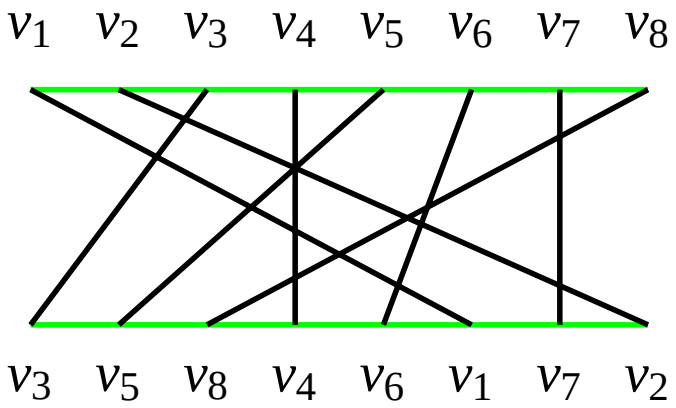
Algorithms on Permutation Graphs

Algorithms on Permutation Graphs

Definition *A permutation graph is a graph that is isomorphic to the intersection graph of a finite set of chords between an ordered set of vertices and a permutation of them.*

Example *The following finite set of chords between an ordered set of vertices and a permutation of them is a permutation model or matching diagram of the permutation graph shown to the left.*





Remark *A permutation graph is the graph of inversions in a permutation.*

Algorithms on Permutation Graphs

- Recognition of Permutation Graphs

  $O(n^3)$ Paul C. Gilmore, A. J. Hoffman. A Characterization of Comparability Graphs and of Interval Graphs. Canadian Journal of Mathematics, 16(3):539–548, 1964.

  $O(n^3)$ Shimon Even, Amir Pnueli, Abraham Lempel. Permutation Graphs and Transitive Graphs. Journal of the ACM, 19(3):400–410, 1972.

  $O(n^3)$ Martin C. Golumbic. The Complexity of Comparability Graph Recognition and Coloring. Computing, 18(4): 199–208, 1977.

  $O(n^2)$ Jeremy Spinrad. On Comparability and Permutation Graphs. SIAM Journal on Computing, 14(3):658–670, 1985.

  $O(n+m)$ Dieter Kratsch, Ross M. McConnell, Kurt Mehlhorn, Jeremy P. Spinrad. Certifying Algorithms for Recognizing Interval Graphs and Permutation Graphs. Proc. 14th Annual ACM-SIAM Symp. Discrete Algorithms, 2003, pages 158–167.

Algorithms on Permutation Graphs

- Maximum Independent Set of Permutation Graphs

  $O(n \log n)$ Haklin Kim. Finding a Maximum Independent Set in a Permutation Graph. Information Processing Letters, 36(1):19–23, 1990.

  $O(n \log n)$ D. T. Lee, Majid Sarrafzadeh. Maximum Independent Set of a Permutation Graph in k Tracks. International Journal of Computational Geometry and Applications, 3(3):291–304, 1993.

  $O(n \log n)$ Peter Widmayer, C. K. Wong. An Optimal Algorithm for the Maximum Alignment of Terminals. Information Processing Letters, 20(2):75–82, 1985.

Algorithms on Permutation Graphs

- Maximum Independent Set of Permutation Graphs

  $O(n \log\log n)$ Maw-Shang Chang, Fu-Hsing Wang. Efficient Algorithms for the Maximum Weight Clique and Maximum Weight Independent Set Problems on Permutation Graphs. Information Processing Letters, 43(6):293–295, 1992.

  $O(n \log\log n)$ Erkki Mäkinen. On the Longest Subsequence Problem for Permutations. International Journal of Computer Mathematics, 77(1):45–53, 2001.

  $O(n \log\log n)$ Frederico Malucelli, Thomas Ottmann, Daniele Pretolani. Efficient Labelling Algorithms for the Maximum Noncrossing Matching Problem. Discrete Applied Mathematics, 47(2):175–179, 1993.

  $O(n \log\log n)$ Ming-Shing Yu, Lin Yu Tseng, Shoe-Jane Chang. Sequential and Parallel Algorithms for the Maximum-Weight Independent Set Problem on Permutation Graphs. Information Processing Letters, 46(1):7–11, 1993.

Algorithms on Permutation Graphs

- Isomorphism of Permutation Graphs

  $O(n^2)$ Charles J. Colbourn. On Testing Isomorphism of Permutation Graphs. Networks, 11(1):13–21, 1981.

Algorithms on Permutation Graphs

Lemma (Gries, 1981) *The longest nondecreasing sequence problem can be solved in $O(n \log n)$ time.*

Proof *Scan the sequence $(v_1, v_2, \ldots, v_n)$ from left to right and maintain the minimum values $m_1, m_2, \ldots, m_k$ which end the nondecreasing sequences of length $1, 2, \ldots, k$ found so far. Then, k is length of a longest nondecreasing sequence.*

Each value v_i in the sequence is compared with the minimum value m_1 of the shortest nondecreasing sequence and with the minimum value m_k of the longest nondecreasing sequence.

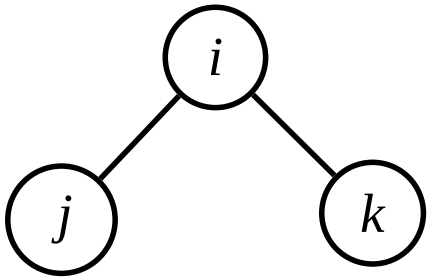
- *If $v_i < m_1$, then m_1 is set to v_i.*
- *If $v_i \geqslant m_k$, then a nondecreasing sequence of length $k+1$ is found, and m_{k+1} is set to v_i.*
- *If $m_1 \leqslant v_i < m_k$, then an index j is found such that $m_{j-1} \leqslant v_i < m_j$, and m_j is set to v_i.*

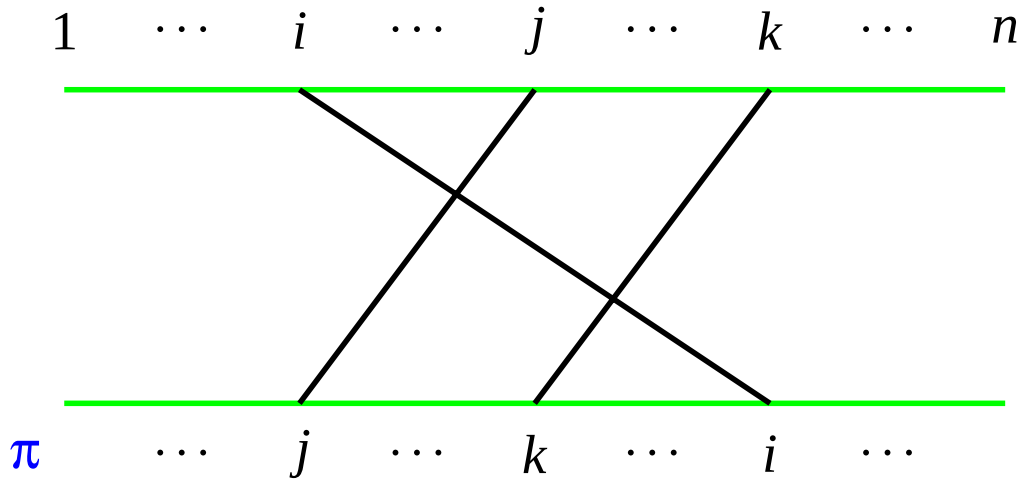
Finding the index j takes $O(\log k)$ time, yielding the $O(n \log n)$ time bound.

Algorithms on Permutation Graphs

Lemma (Golumbic, 1980) *Let G be the (permutation) graph of inversions of a permutation π. Then, the decreasing sequences of π are in one-to-one correspondence with the cliques of G, and the increasing sequences of π are in one-to-one correspondence with the independent sets of G.*

Proof *Follows from two vertices v_i and v_j in a permutation graph G being adjacent if and only if the corresponding chords in a matching diagram of G intersect if and only if v_i and v_j are inverted in the permutation π.*





Algorithms on Permutation Graphs

Lemma *The maximum independent set problem can be solved on permutation graphs in* $O(n \log\log n)$ *time.*

Proof *Follows from the problem of finding the successor of an element in a restricted universe being solvable in* $O(n \log\log n)$ *time.*

Algorithms on Circle Graphs

Algorithms on Circle Graphs

- Recognition of Circle Graphs

  $O(n^7)$ Walid Naji. Reconnaissance des Graphes de Cordes. Discrete Mathematics, 54(3):329–337, 1985.

  $O(n^5)$ André Bouchet. Reducing Prime Graphs and Recognizing Circle Graphs. Combinatorica, 7(3):243–254, 1987.

  $O(mn)$ Csaba P. Gabor, Kenneth J. Supowit, Wen-Lian Hsu. Recognizing Circle Graphs in Polynomial Time. Journal of the ACM, 36(3):435–473, 1989.

  $O(mn)$ Wen-Lian Hsu. $O(mn)$ Algorithms for the Recognition and Isomorphism Problems on Circular-Arc Graphs. SIAM Journal on Computing, 24(3):411–439, 1995.

  $O(n^2)$ Jeremy Spinrad. Recognition of Circle Graphs. Journal of Algorithms, 16(2):264–282, 1994.

Algorithms on Circle Graphs

- Maximum Independent Set of Circle Graphs

  $O(n^3)$ Fanica Gavril. Algorithms for a Maximum Clique and a Maximum Independent Set of a Circle Graph. Networks, 3(3):261–273, 1973.

  $O(n^3)$ Robin Liu, Simeon C. Ntafos. On Decomposing Polygons into Uniformly Monotone Parts. Information Processing Letters, 27(2):85–88, 1988.

Algorithms on Circle Graphs

- Maximum Independent Set of Circle Graphs

  $O(n^2)$ Takao Asano, Tetsuo Asano, Hiroshi Imai. Partitioning a Polygonal Region into Trapezoids. Journal of the ACM, 33(2):290–312, 1986.

  $O(n^2)$ Takao Asano, Hiroshi Imai, Akira Mukaiyama. Finding a Maximum Weight Independent Set of a Circle Graph. IEICE Transactions, E74(4):681–683, 1991.

  $O(n^2)$ Olivier Goldschmidt, Alexan Takvorian. An Efficient Algorithm for Finding a Maximum Weight Independent Set of a Circle Graph. IEICE Transactions, E77-A(10):1672–1674, 1994.

  $O(n^2)$ Ronald C. Read, Doron Rotem, Jorge Urrutia. Orientations of Circle Graphs. Journal of Graph Theory, 6(3):325–341, 1982.

  $O(n^2)$ Kenneth J. Supowit. Finding a Maximum Planar Subset of a Set of Nets in a Channel. IEEE Transactions on Computer-Aided Design, 6(1):93–94, 1987.

Algorithms on Circle Graphs

- Maximum Independent Set of Circle Graphs

  $O(nd)$ Alberto Apostolico, Mikhail J. Atallah, Susanne E. Hambrusch. New Clique and Independent Set Algorithms for Circle Graphs. Discrete Applied Mathematics, 36(1):1–24, 1992; 41(2):179–180, 1993.

Algorithms on Circle Graphs

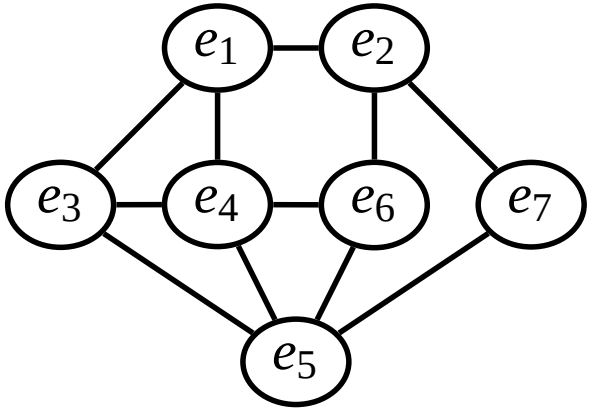
- Isomorphism of Circle Graphs

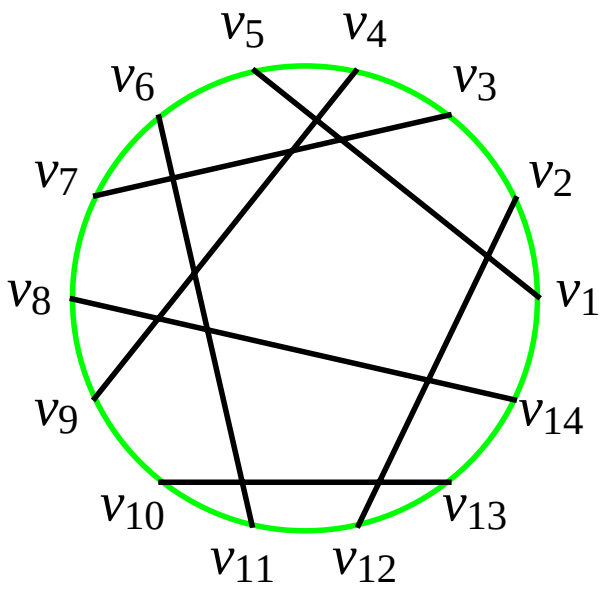
  $O(mn)$ Wen-Lian Hsu. $O(mn)$ Algorithms for the Recognition and Isomorphism Problems on Circular-Arc Graphs. SIAM Journal on Computing, 24(3):411–439, 1995.

Algorithms on Circle Graphs

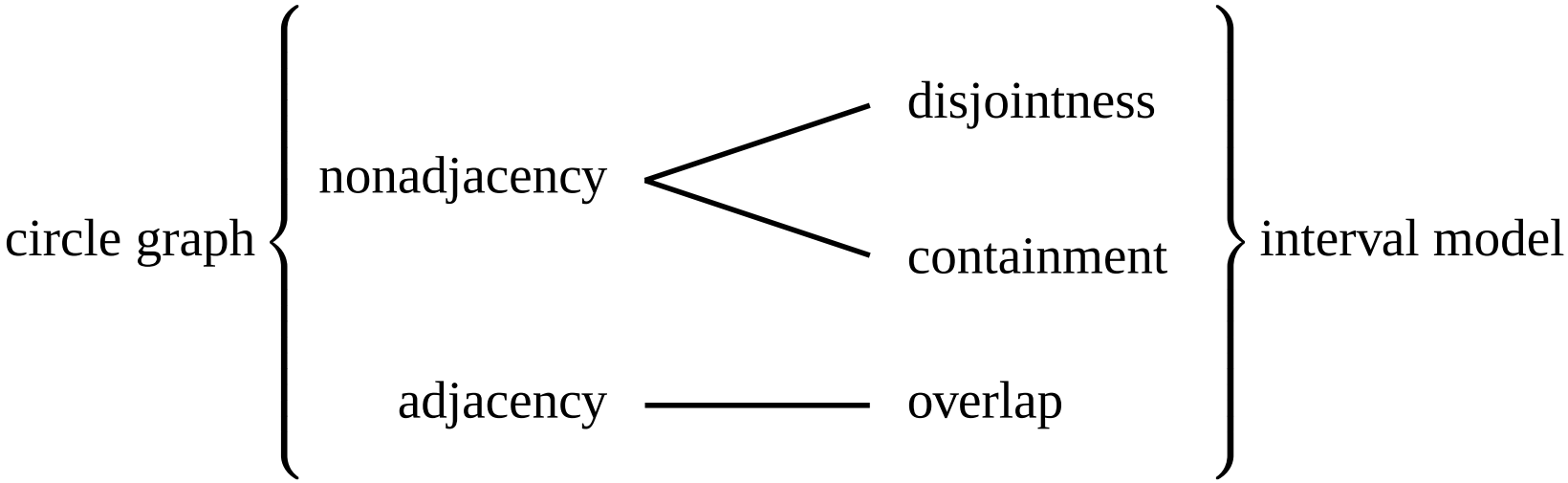
Definition *A circle graph is a graph that is isomorphic to the intersection graph of a finite set of chords of a circle.*

Example *The following finite set of chords of a circle is a chord model of the circle graph shown to the left.*



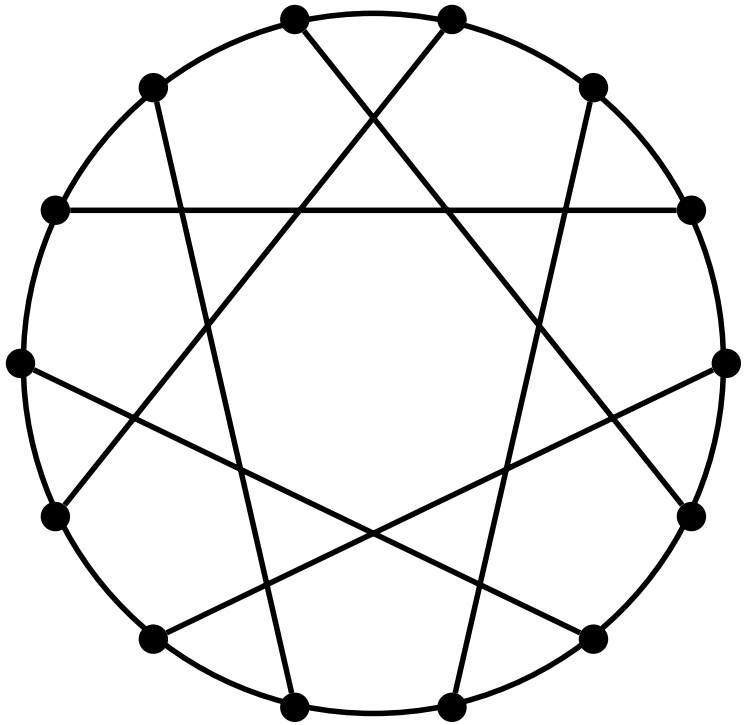


Algorithms on Circle Graphs



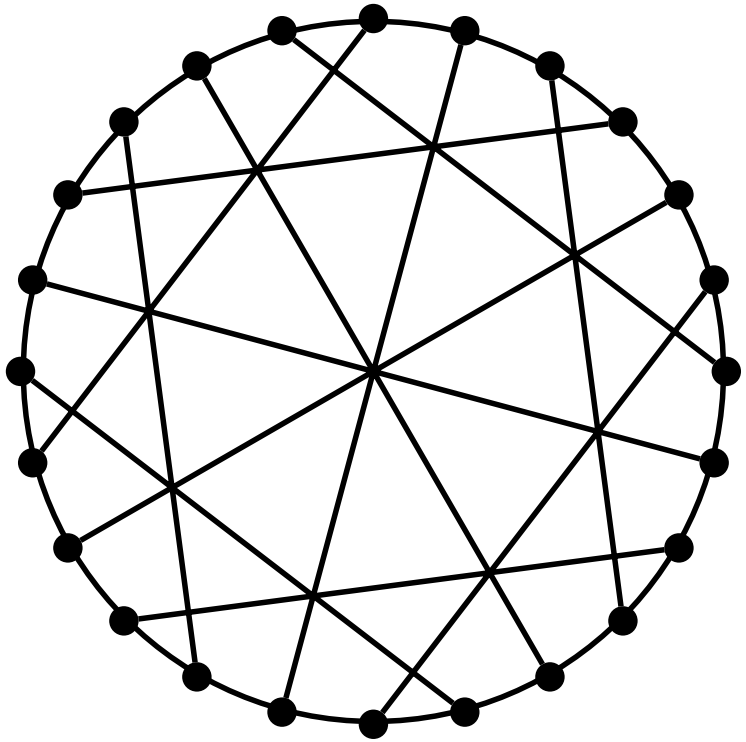
- A set of mutually disjoint, or contained in each other, intervals of $I(G)$ models an independent set of G.
- A set of mutually overlapping intervals of $I(G)$ models a clique of G.

$(3, 6)$-cage



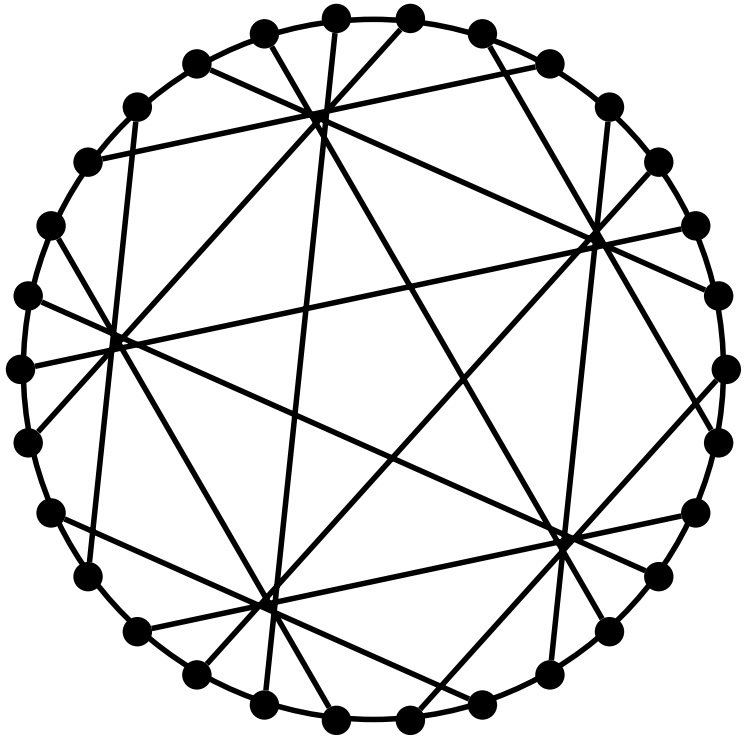
P. J. Heawood (1861–1955)

$(3, 7)$-cage



W. F. McGee (1937–)

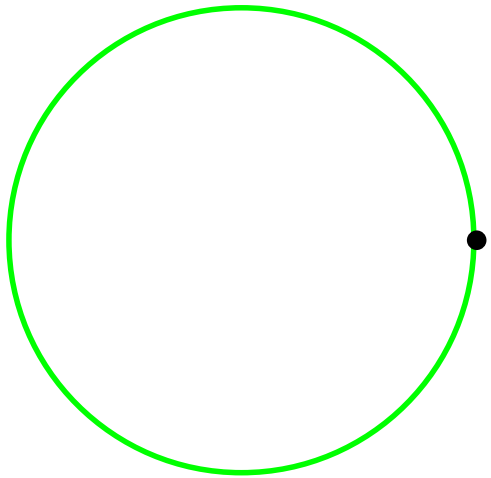
$(3, 8)$-cage



W. T. Tutte (1917–2002) and H. S. Coxeter (1907–)

Algorithms on Circle Graphs

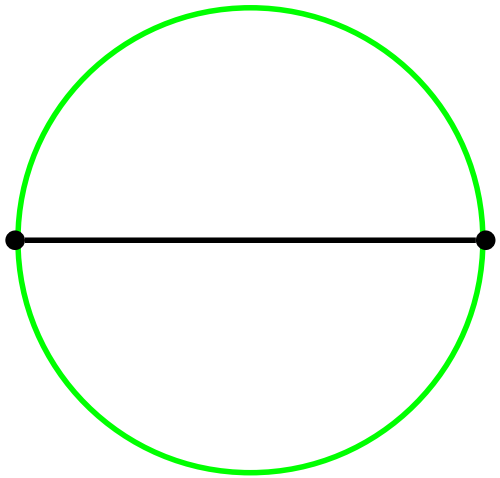
Exercise *Maximum number of non-overlapping regions r that can be obtained by joining n points on the circle with chords*



$n = 1$ $r = 1$

Algorithms on Circle Graphs

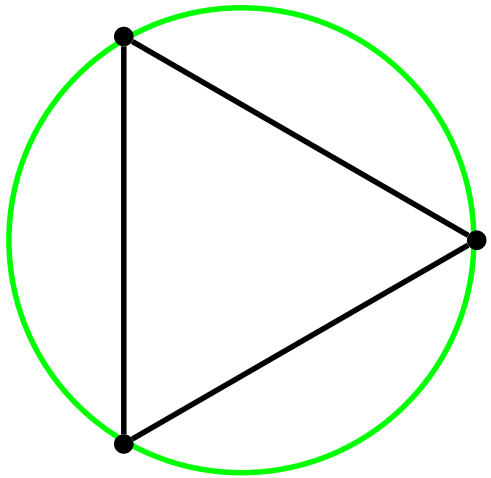
Exercise *Maximum number of non-overlapping regions r that can be obtained by joining n points on the circle with chords*



$n = 2$ $r = 2$

Algorithms on Circle Graphs

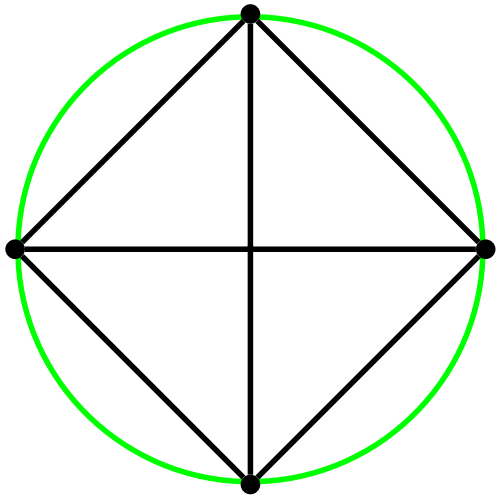
Exercise *Maximum number of non-overlapping regions r that can be obtained by joining n points on the circle with chords*



$n = 3$ $r = 4$

Algorithms on Circle Graphs

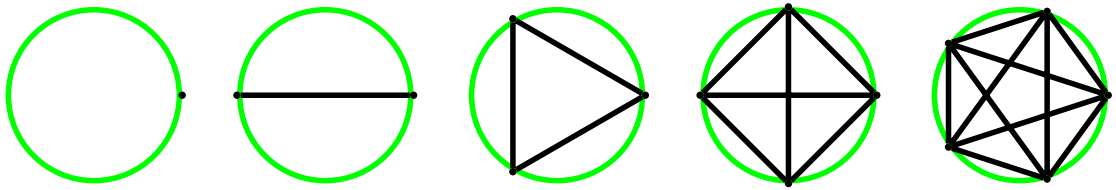
Exercise *Maximum number of non-overlapping regions r that can be obtained by joining n points on the circle with chords*



$n = 4$ $r = 8$

Algorithms on Circle Graphs

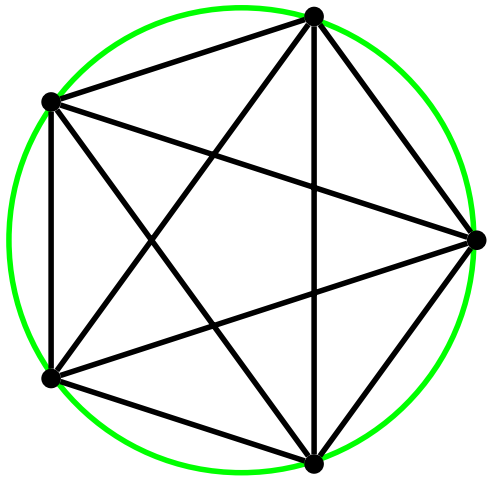
Exercise *Maximum number of non-overlapping regions r that can be obtained by joining n points on the circle with chords*



n	1	2	3	4	5
r	1	2	4	8	...

Algorithms on Circle Graphs

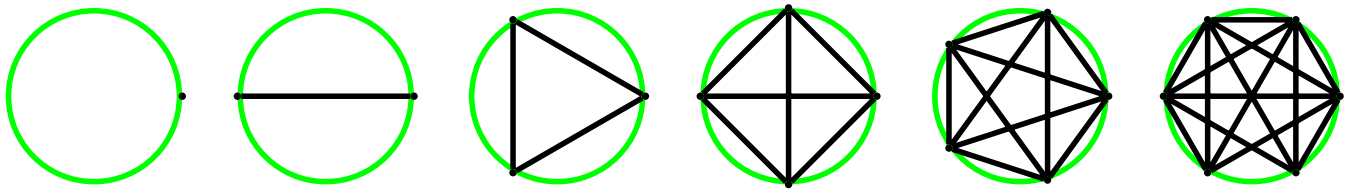
Exercise *Maximum number of non-overlapping regions r that can be obtained by joining n points on the circle with chords*



$n = 5$ $r = 16$

Algorithms on Circle Graphs

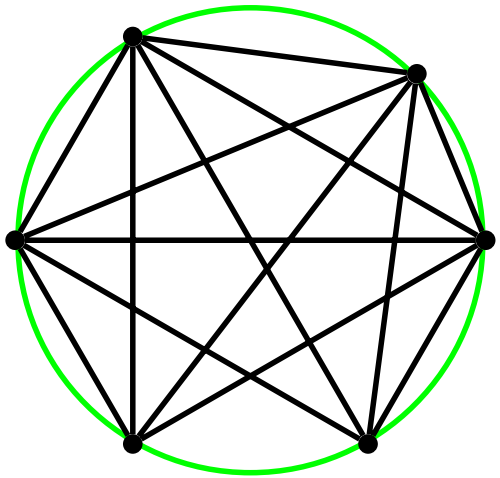
Exercise *Maximum number of non-overlapping regions r that can be obtained by joining n points on the circle with chords*



n	1	2	3	4	5	6
r	1	2	4	8	16	...

Algorithms on Circle Graphs

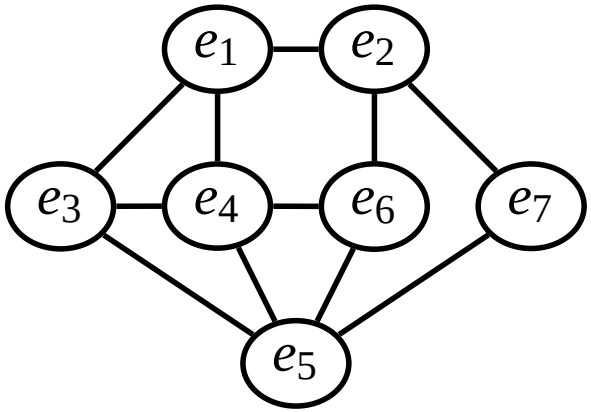
Exercise *Maximum number of non-overlapping regions r that can be obtained by joining n points on the circle with chords*

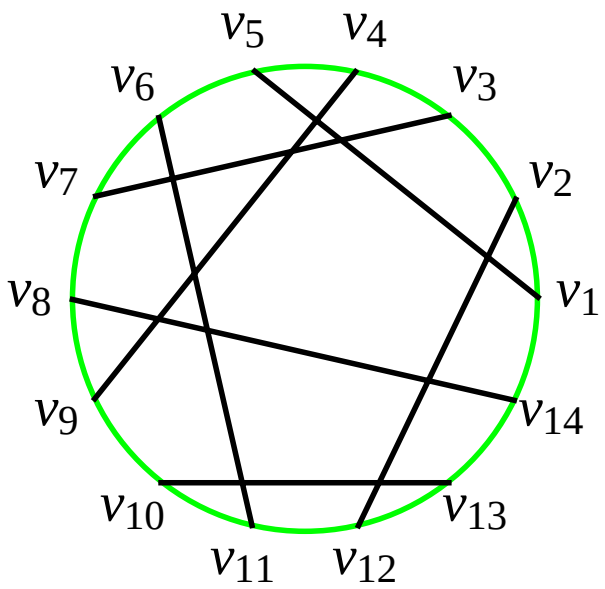


$n = 6$ $r = 31$

Algorithms on Circle Graphs

Definition *The total chord length of a circle graph is the sum of the length of all chords in the chord model of the graph*

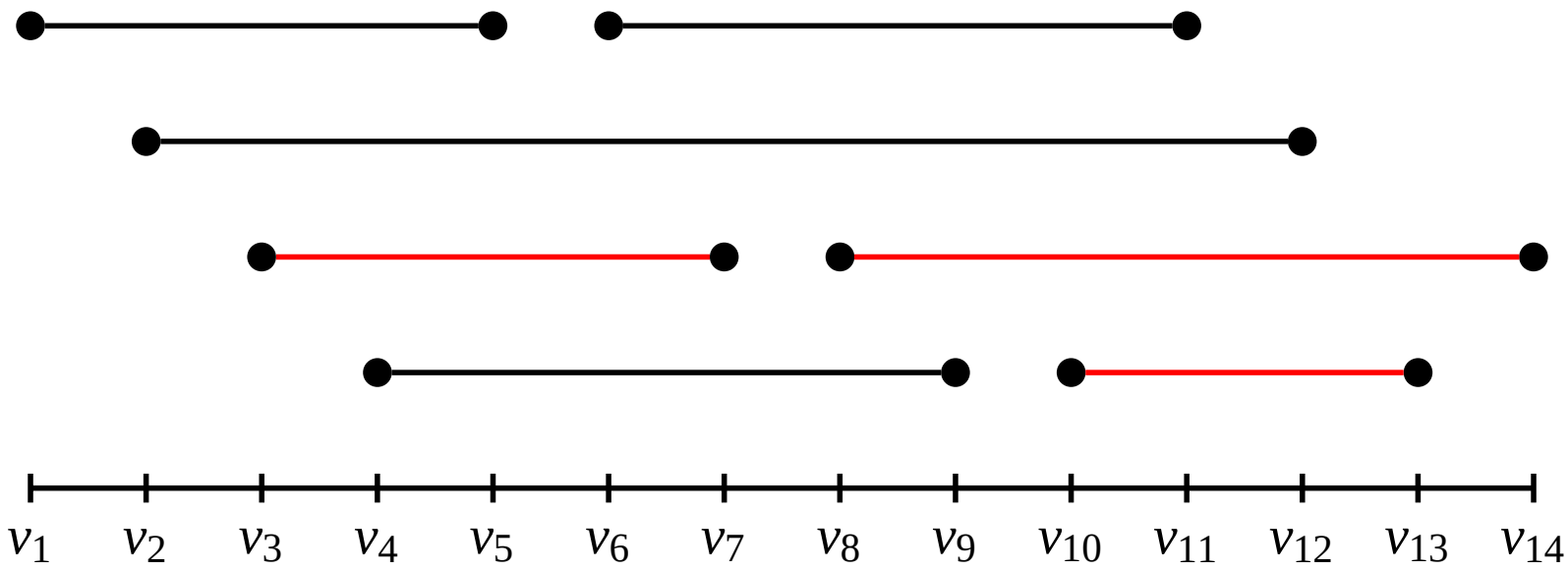




Example *The total chord length of the previous circle graph is 37*

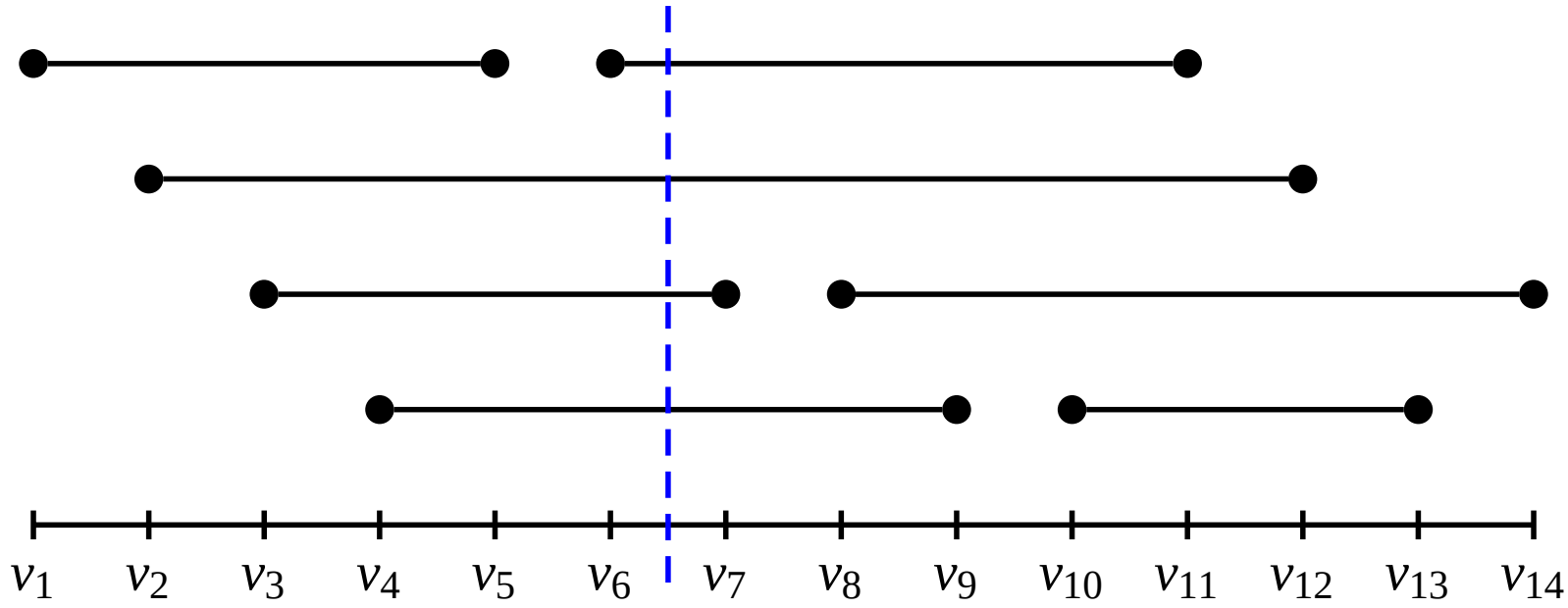
Algorithms on Circle Graphs

In the interval model of a circle graph, the total order on the vertex set along the circumference of a circle is replaced by a total order along the line



Algorithms on Circle Graphs

Definition *The density of a circle graph is the maximum number of intervals crossing any position on the line in the interval model of the graph*



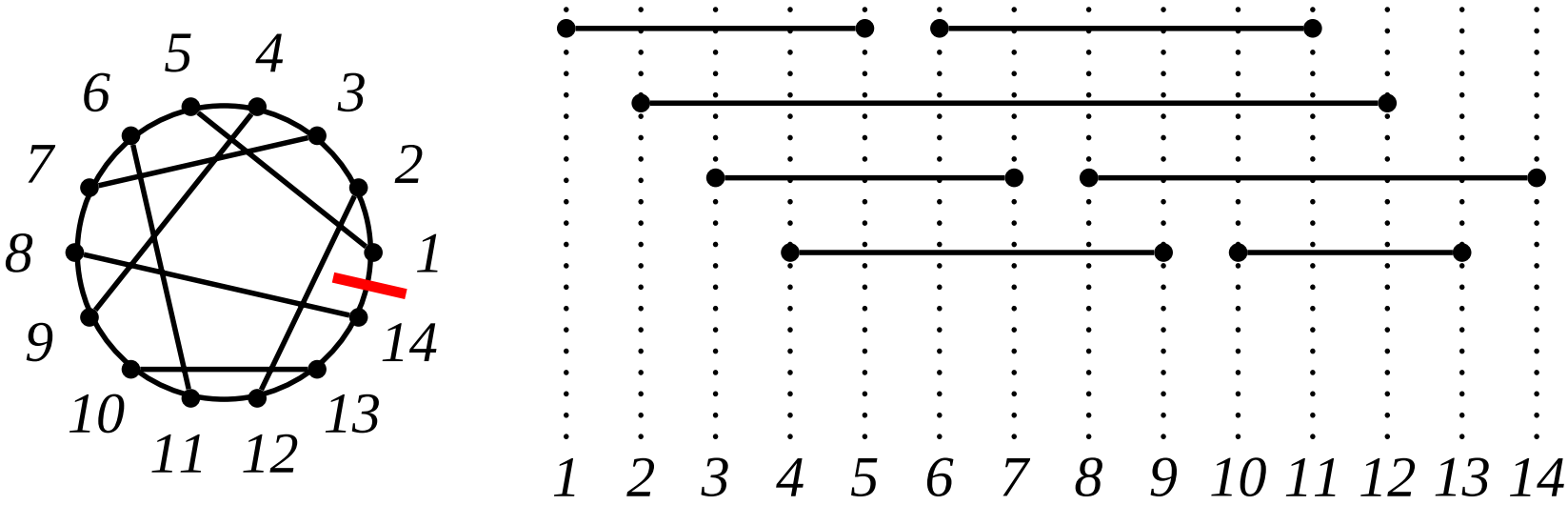
Example *The density of the previous circle graph is 4*

Algorithms on Circle Graphs

- An interval model of a circle graph can be obtained from the chord model of the circle graph by a simple transformation, consisting in cutting the circumference of the circle at some point p which is not an endpoint of a chord and unfolding it at point p
- The chord model can be reconstructed by wrapping around the circle the collection of intervals on the line
- Given a chord model of a circle graph, for each choice of point p in the previous transformation a different interval model is obtained, and both the density and the total chord length of the circle graph depend on the particular chord or interval model chosen, that is, on the choice of point p for the transformation between the chord model and the interval model of the circle graph

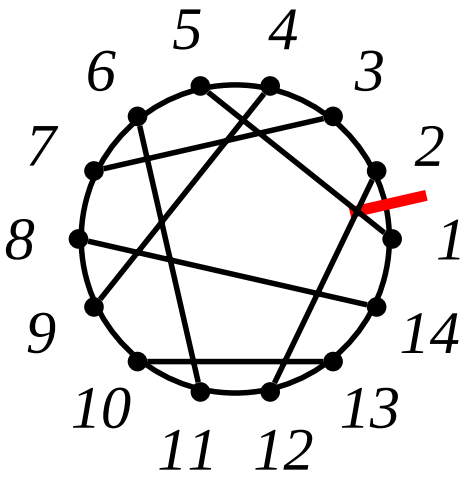
Algorithms on Circle Graphs

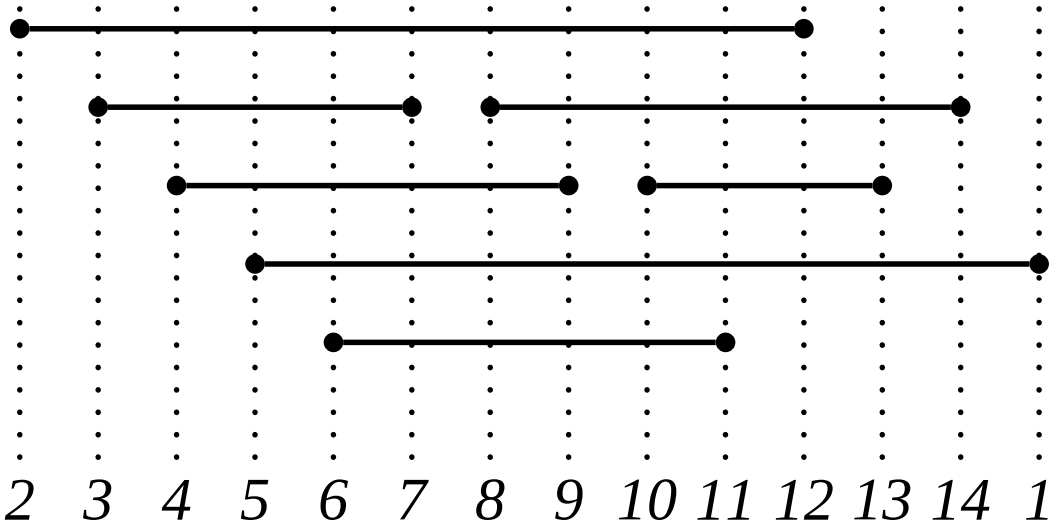
Example *The following chord model and corresponding interval model for the previous circle graph have total chord length 37 and density 4*



Algorithms on Circle Graphs

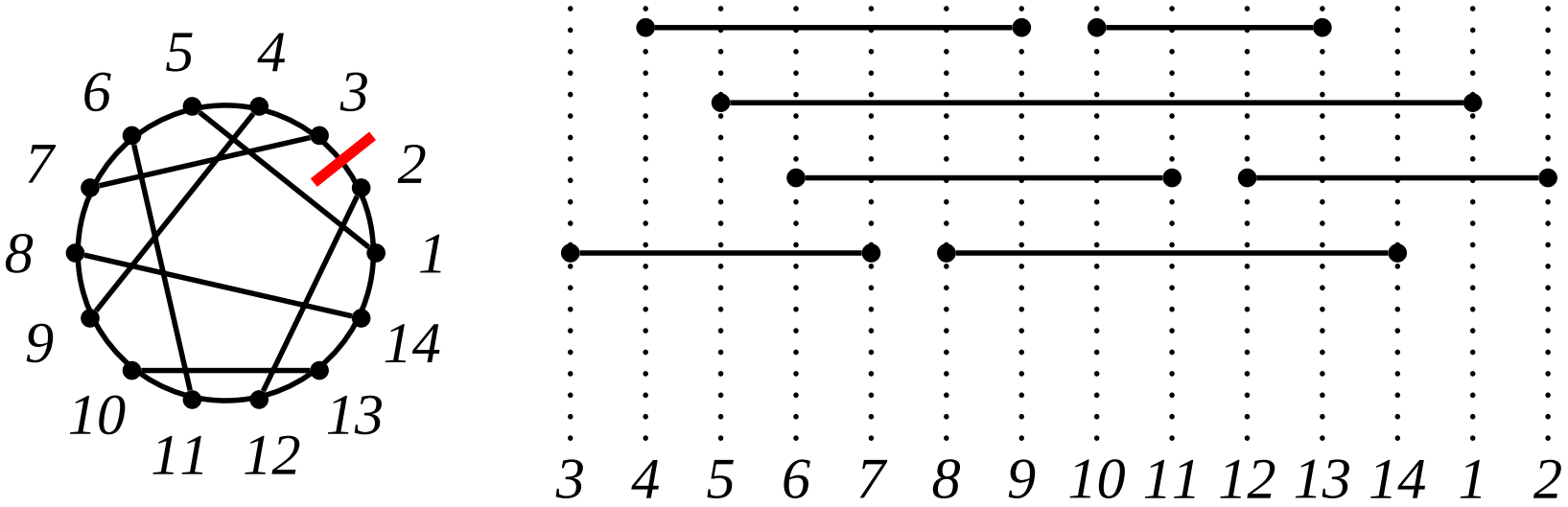
Example *The following chord model and corresponding interval model for the previous circle graph have total chord length 43 and density 5*





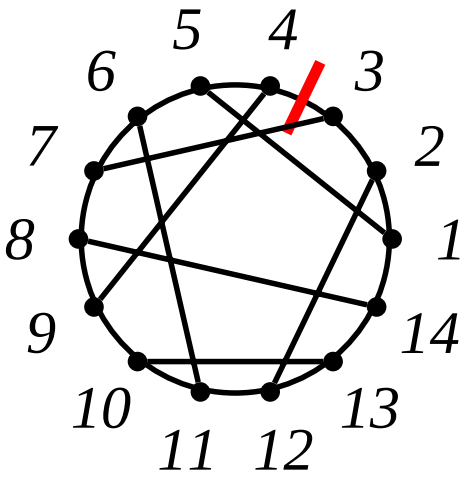
Algorithms on Circle Graphs

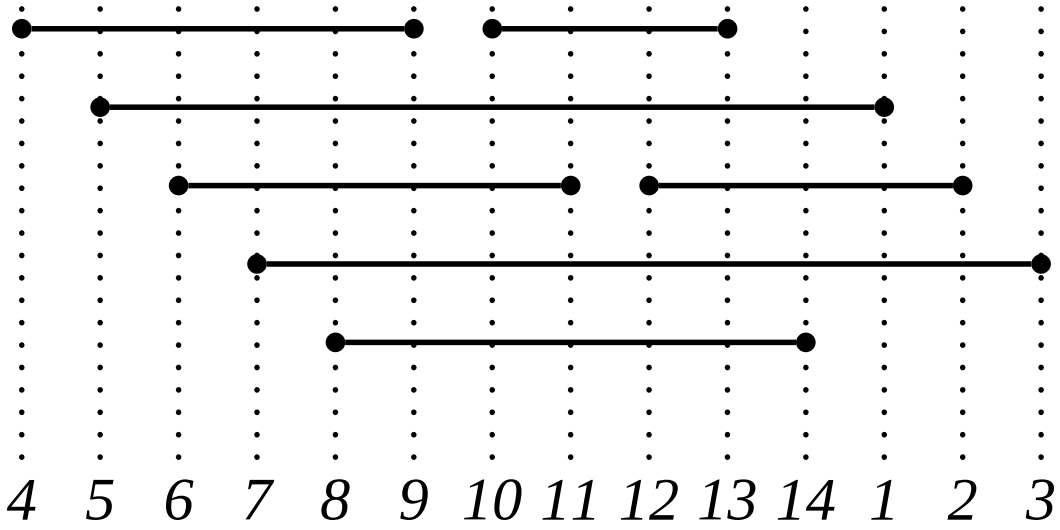
Example *The following chord model and corresponding interval model for the previous circle graph have total chord length 37 and density 4*



Algorithms on Circle Graphs

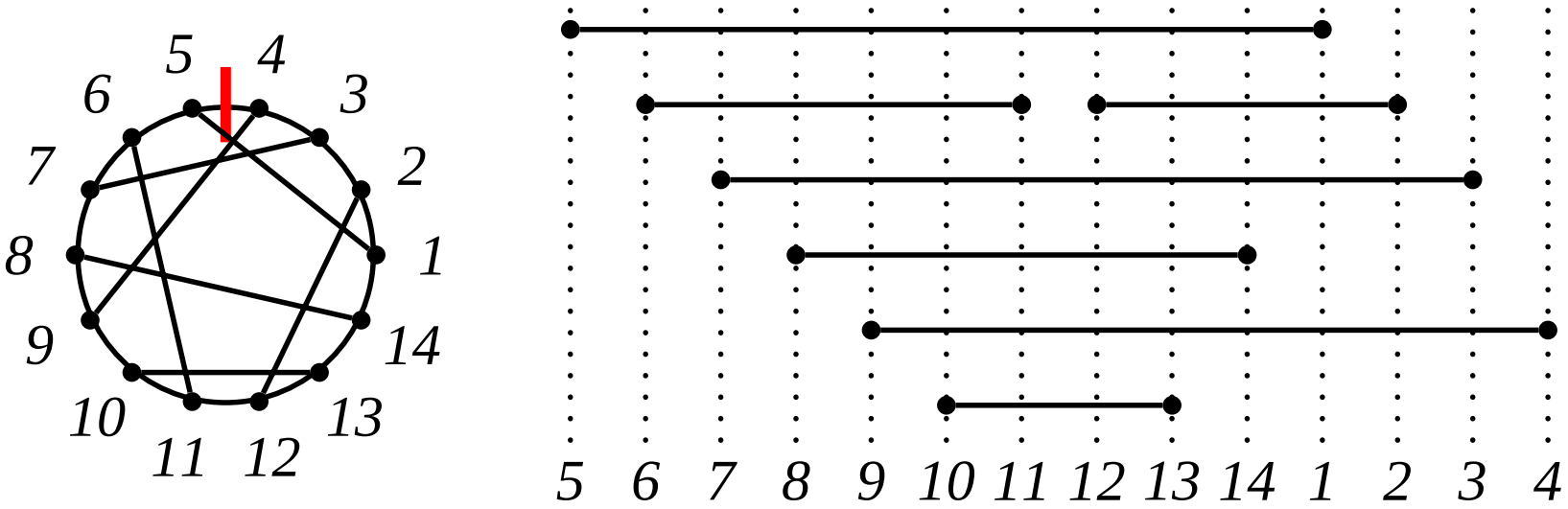
Example *The following chord model and corresponding interval model for the previous circle graph have total chord length 43 and density 5*





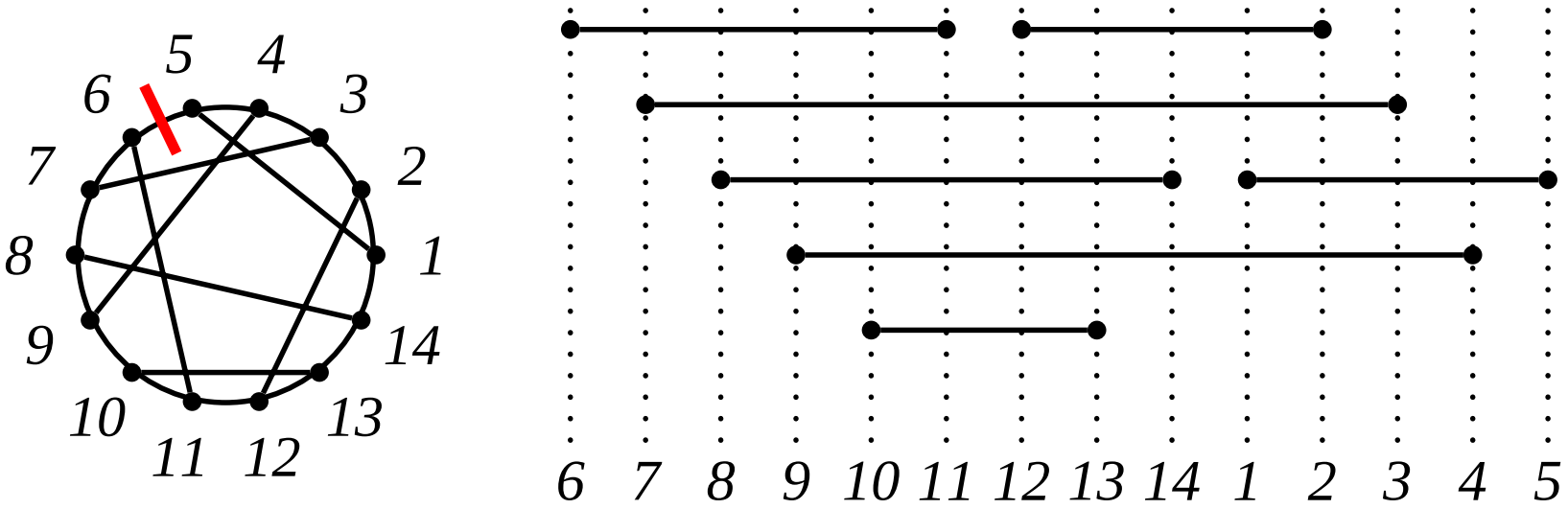
Algorithms on Circle Graphs

Example *The following chord model and corresponding interval model for the previous circle graph have total chord length 47 and density 6*



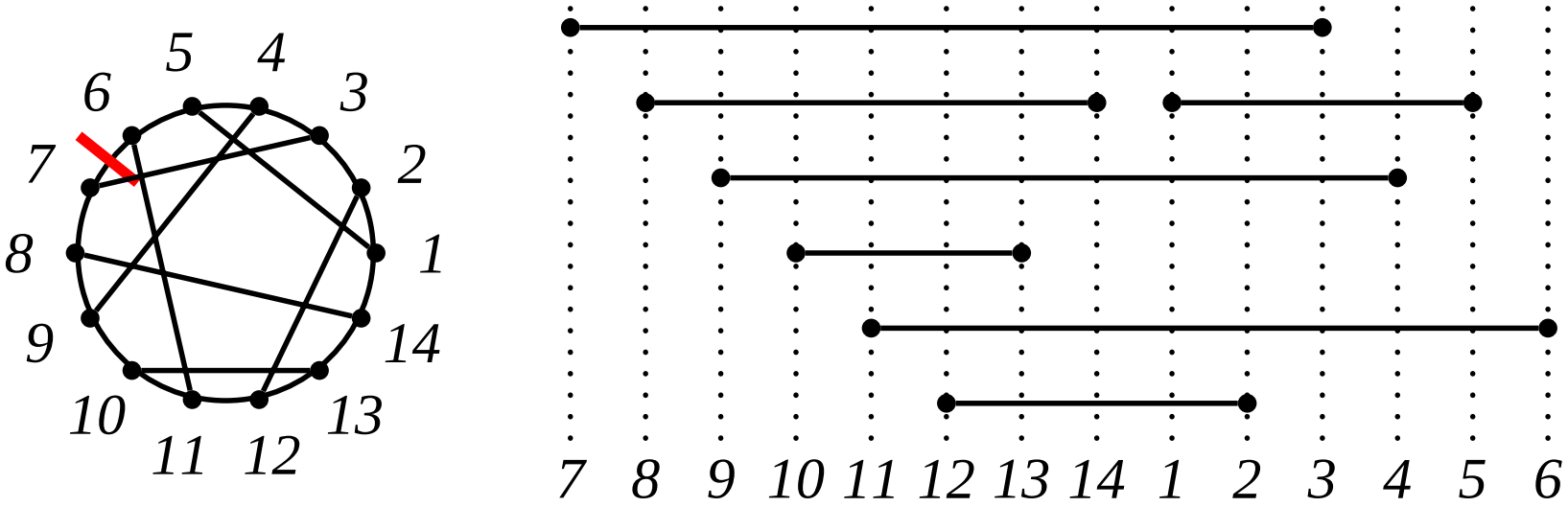
Algorithms on Circle Graphs

Example *The following chord model and corresponding interval model for the previous circle graph have total chord length 41 and density 5*



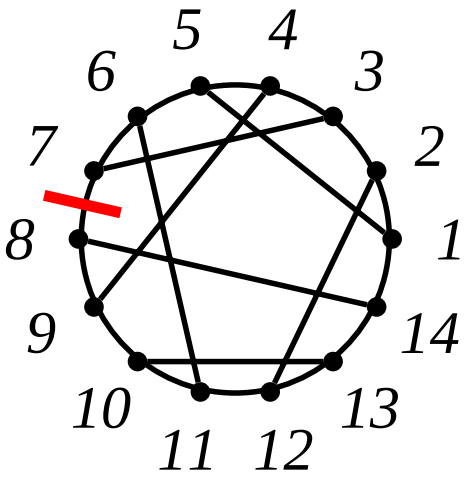
Algorithms on Circle Graphs

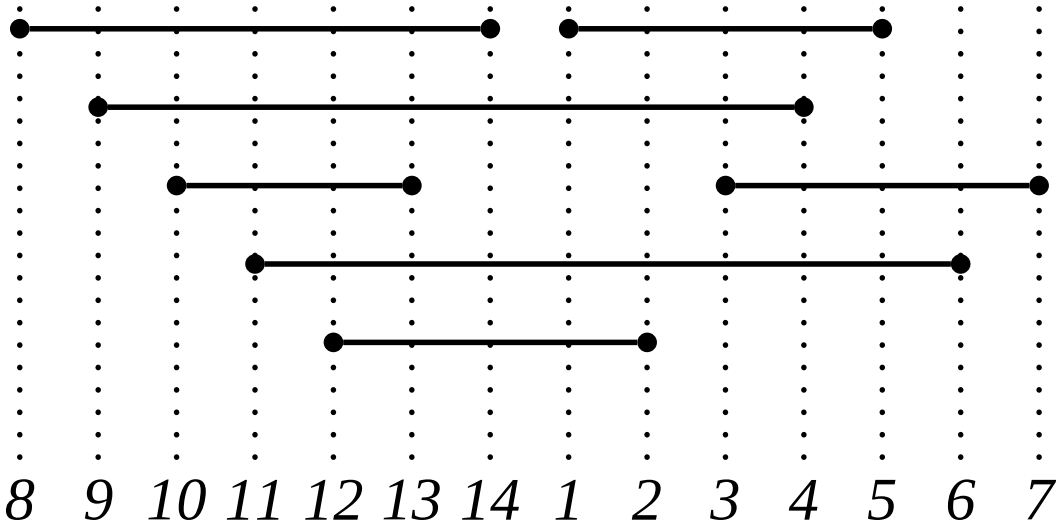
Example *The following chord model and corresponding interval model for the previous circle graph have total chord length 45 and density 6*



Algorithms on Circle Graphs

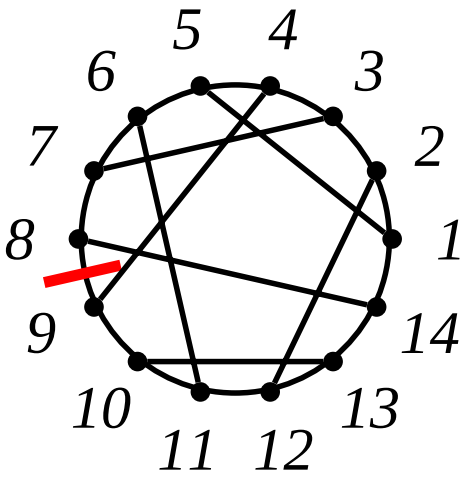
Example *The following chord model and corresponding interval model for the previous circle graph have total chord length 39 and density 5*

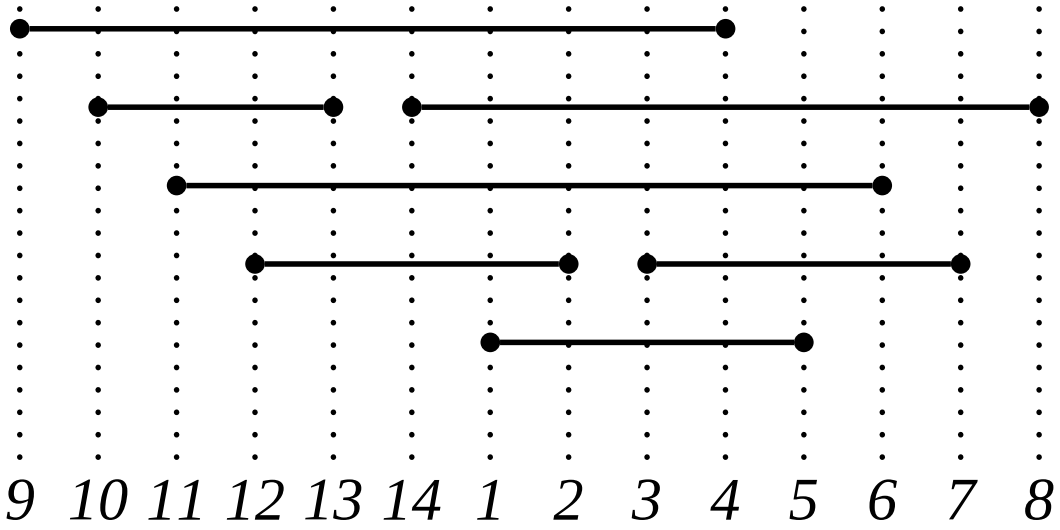




Algorithms on Circle Graphs

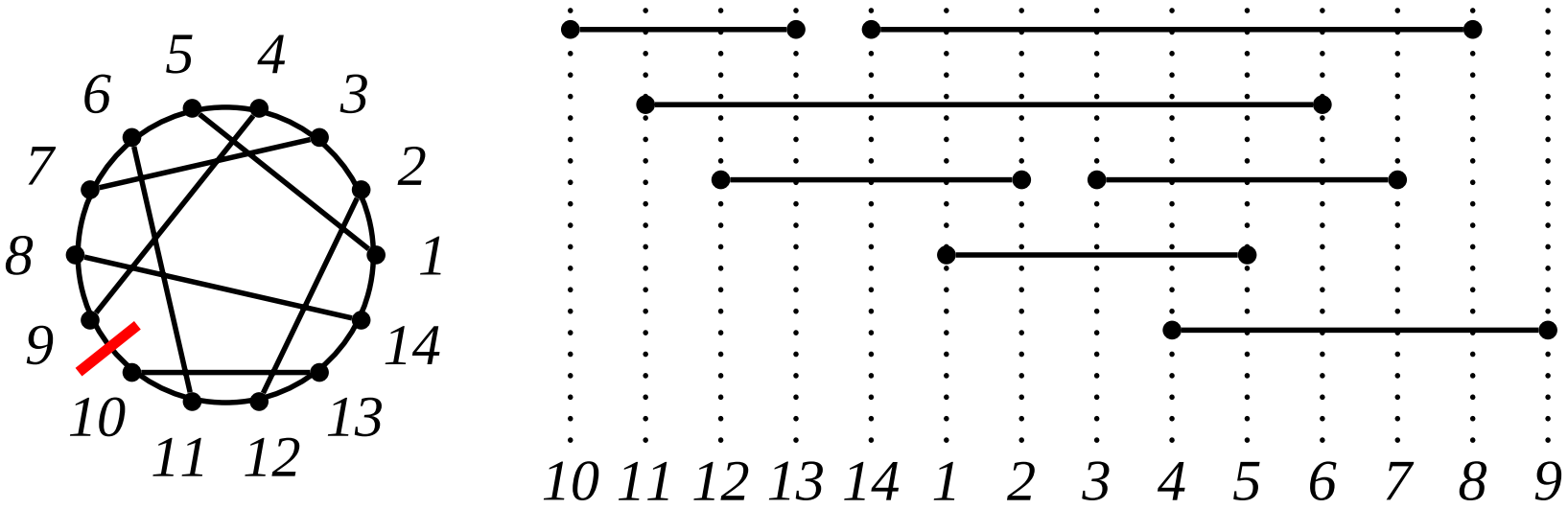
Example *The following chord model and corresponding interval model for the previous circle graph have total chord length 41 and density 5*





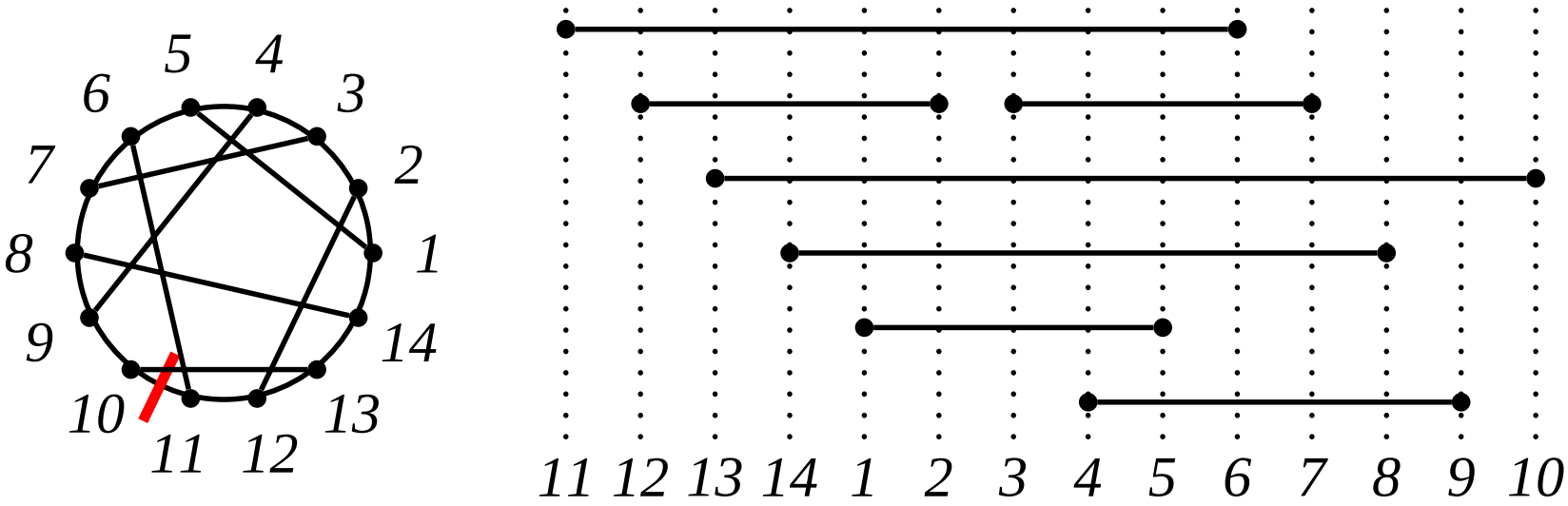
Algorithms on Circle Graphs

Example *The following chord model and corresponding interval model for the previous circle graph have total chord length 37 and density 5*



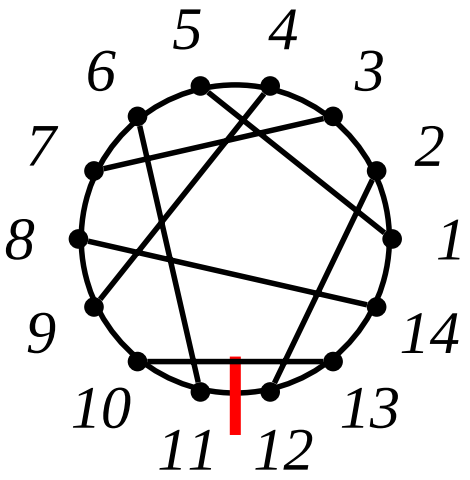
Algorithms on Circle Graphs

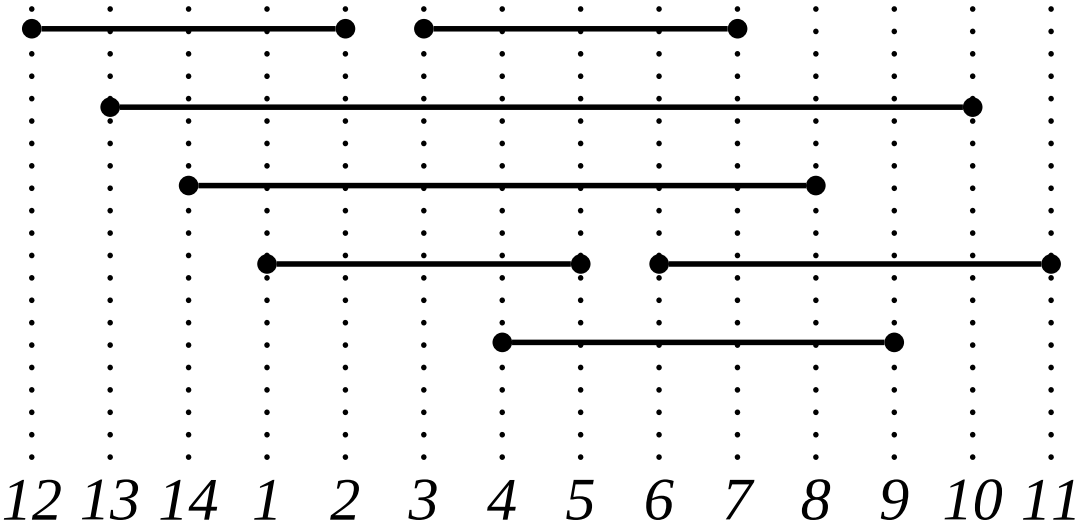
Example *The following chord model and corresponding interval model for the previous circle graph have total chord length 45 and density 6*



Algorithms on Circle Graphs

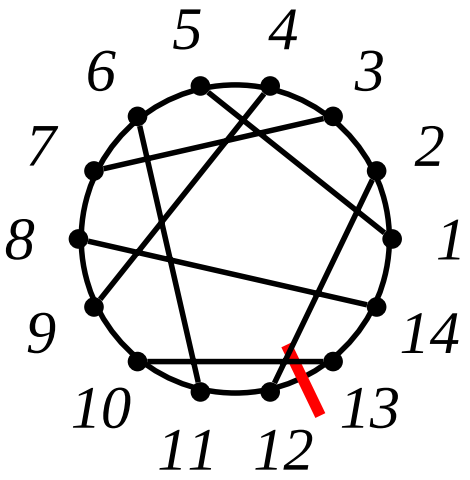
Example *The following chord model and corresponding interval model for the previous circle graph have total chord length 41 and density 5*

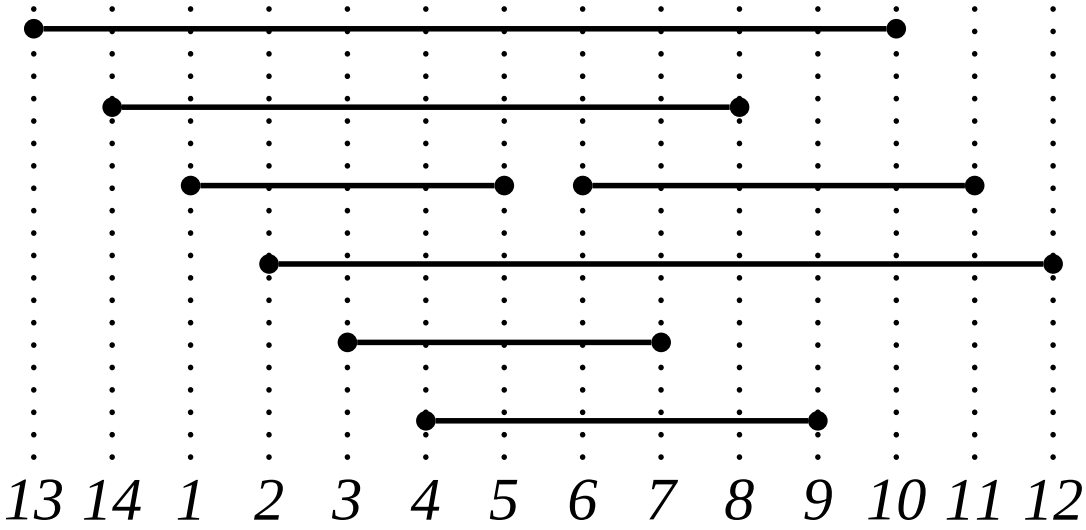




Algorithms on Circle Graphs

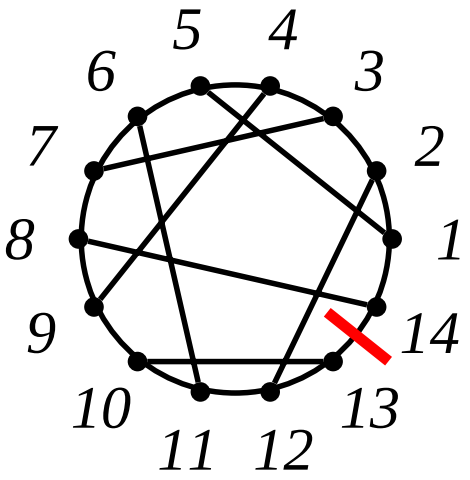
Example *The following chord model and corresponding interval model for the previous circle graph have total chord length 47 and density 6*

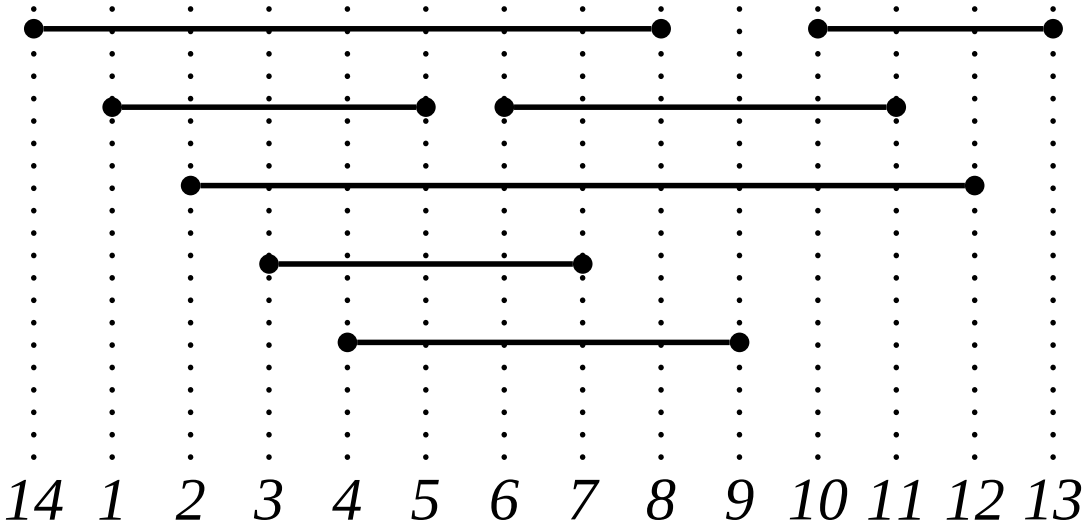




Algorithms on Circle Graphs

Example *The following chord model and corresponding interval model for the previous circle graph have total chord length 39 and density 5*





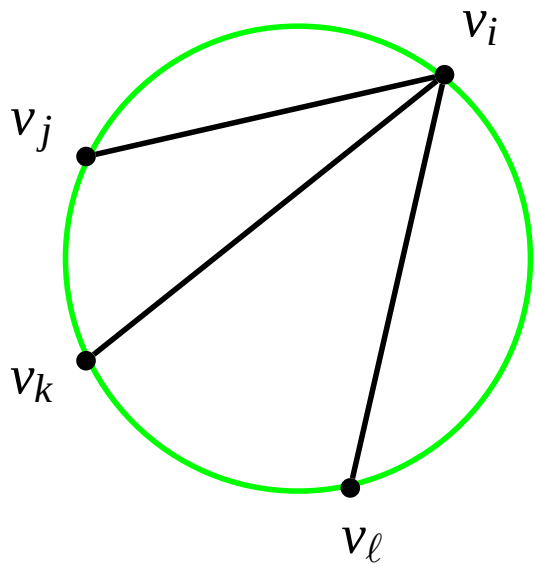
Algorithms on Circle Graphs

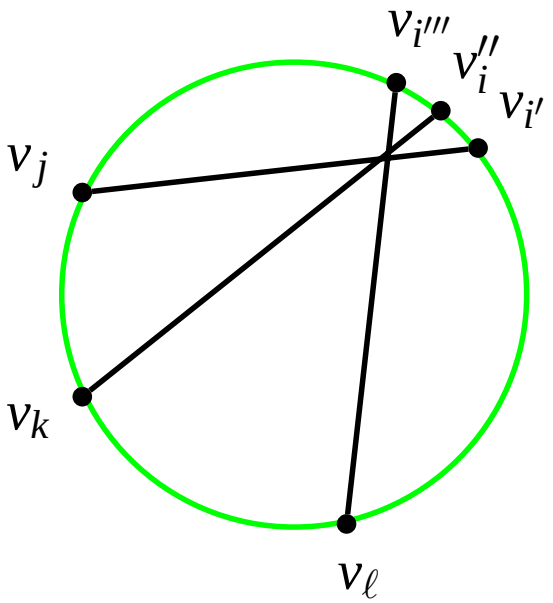
It will be assumed, without loss of generality, that all chords are ordered from lower to higher vertex number, that is, $i < j$ for all chords $(v_i, v_j) \in E$ in a circle graph $G = (V, E)$

- The length of chord $e = (v_i, v_j) \in E$ is $len(e) = j - i$
- $MIS(G)$ denotes a maximum independent set of G
- For all chords $(v_i, v_j) = e \in E$, $MIS(e)$ denotes a maximum independent set of the subgraph of G induced by $\{v_k \in V \mid i < k < j\}$

Algorithms on Circle Graphs

It can be assumed, without loss of generality, that chords in a circle model of a circle graph are vertex-disjoint





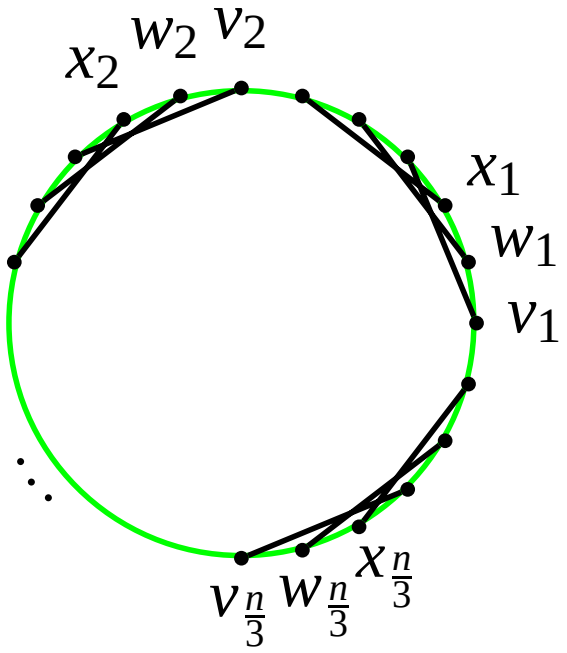
Algorithms on Circle Graphs

- A maximum independent set of the line graph $L(G)$ of a graph G corresponds to a maximum matching of G
- A maximum independent set of a permutation graph corresponds to a noncrossing bipartite matching
- A maximum independent set of a circle graph corresponds to a maximum planar subgraph (maximum planar matching) of a general graph with a fixed vertex ordering

Algorithms on Circle Graphs

Remark (Gavril, 1973) *There are families of circle graphs whose number of independent sets grow exponentially with the number of vertices*

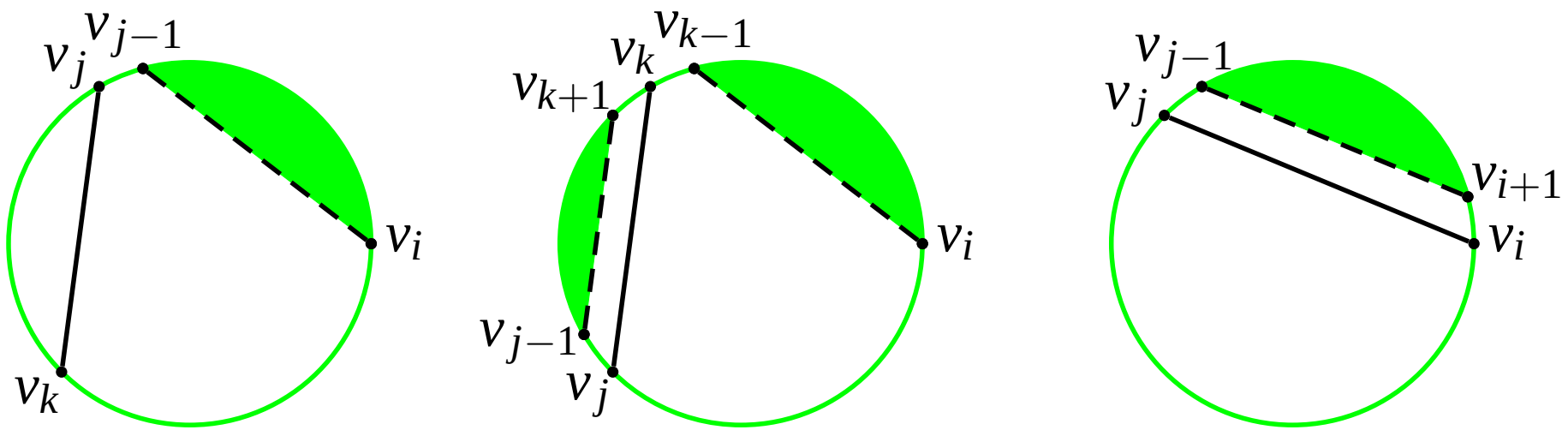
Example *Family of graphs with n chords and* $3^{n/3}$ *maximum independent sets*



Algorithms on Circle Graphs

Lemma (Supowit, 1987) *Let $G = (V, E)$ be a circle graph with n chords. Then, $MIS(G)$ can be found in $O(n^2)$ time using $O(n^2)$ space*

Proof *By dynamic programming*



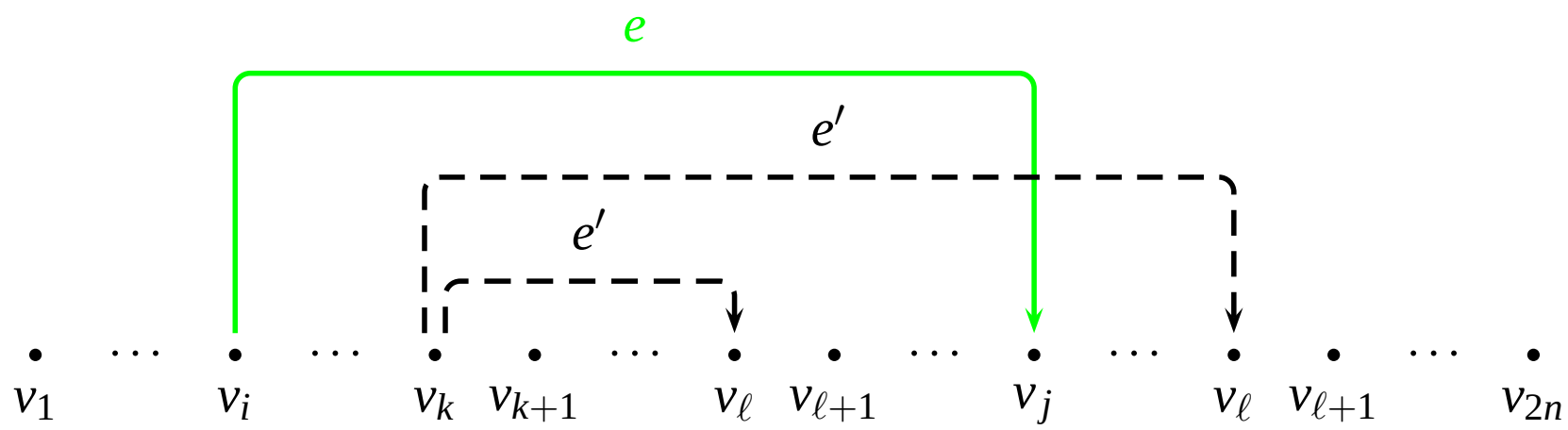
Algorithms on Circle Graphs

- The dynamic programming solution involves subproblems that do not contribute to the solution of any larger problem
- The problem of finding a MIS of a circle graph can be decomposed along the chords of the graph
- Unnecessary computation can be avoided by solving subproblems in nondecreasing order of chord length, because a chord can only contain shorter chords

Algorithms on Circle Graphs

Lemma *Let $e = (v_i, v_j) \in E$, and assume $MIS(e')$ is known for all chords $e' \in E$ with $len(e') < len(e)$. Then, $MIS(e)$ can be found in $O(len(e))$ time using $O(len(e))$ space*

Proof *For all vertices $v_k \in V$ with $i < k < j$, let $T(k)$ be a MIS of the subgraph of G induced by $\{v_\ell \in V \mid k \leqslant \ell < j\}$*

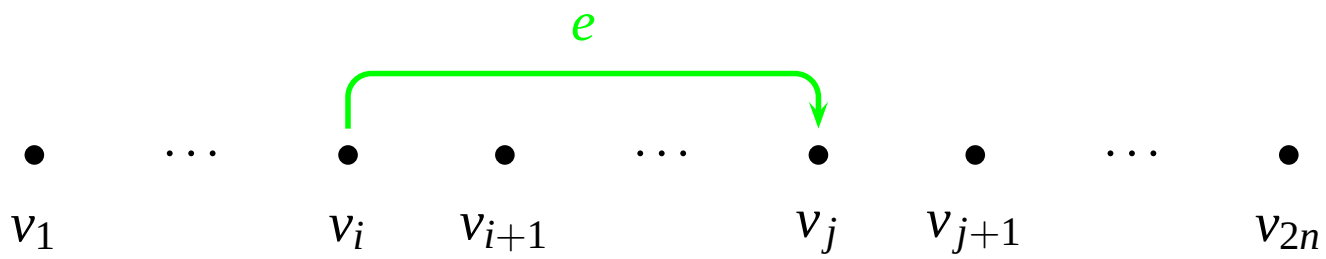


- $T(k) = T(k+1)$
- $T(k) = \max\{MIS(e') \cup T(\ell+1), T(k+1)\}$

Algorithms on Circle Graphs

Lemma *Let $G = (V,E)$ be a circle graph with n chords, and assume $MIS(e)$ is known for all chords $e \in E$. Then, $MIS(G)$ can be found in $O(n)$ time using $O(n)$ space*

Proof *Let $T(i)$ be a MIS of the subgraph of G induced by $\{v_j \in V \mid i \leqslant j \leqslant 2n\}$*



- $T(i) = \max\{MIS(e) \cup T(j+1), T(i+1)\}$.
- $T(1) = MIS(G)$.

Algorithms on Circle Graphs

Lemma *Let $G = (V, E)$ be a circle graph with n chords, and let ℓ be the total chord length of G. Then, $MIS(G)$ can be found in $O(\ell)$ time using $O(n)$ space.*

Proof *Immediate.*

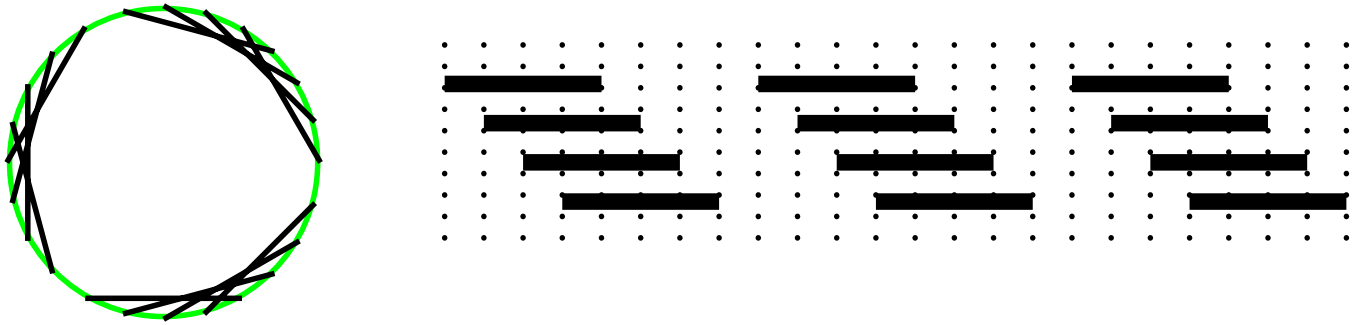
- The chords of G can be oriented in $O(n)$ time using $O(1)$ space, and can be bucket sorted in nondecreasing order of chord length in $O(n)$ time using $O(n)$ space.
- For each chord $e \in E$ in nondecreasing order of chord length, $MIS(e)$ can be found in $O(len(e))$ time using $O(len(e))$ space.
- $MIS(G)$ can be found in $O(\ell)$ time using $O(\ell + n)$ space
- Independent sets need not be stored explicitly and can be represented by reference to the sets they directly contain only.

Algorithms on Circle Graphs

Theorem *Let G be a circle graph with m chords and $n = 2m$ vertices, let ℓ be the total chord length of G, and let d be the density of G. Then, $\ell \leqslant dn$.*

Proof *For any fixed density d, the circle graph on $n = 2m = 2kd$ vertices with the largest total chord length ℓ has $k = m/d = n/(2d)$ identical blocks, each of them with d chords of length d each. The total chord length is thus $\ell = kdd = nd$.*

Example *The circle graph on $n = 24$ vertices with the largest total chord length $\ell = 84$, for a fixed density $d = 4$, has $n/(2d) = 3$ identical trapezoidal blocks of d identical chords each. No other circle graph of density $d = 4$ and $n = 24$ vertices can have a larger total chord length, because the chords, which must be vertex-disjoint, already fill up the space along d line segments.*



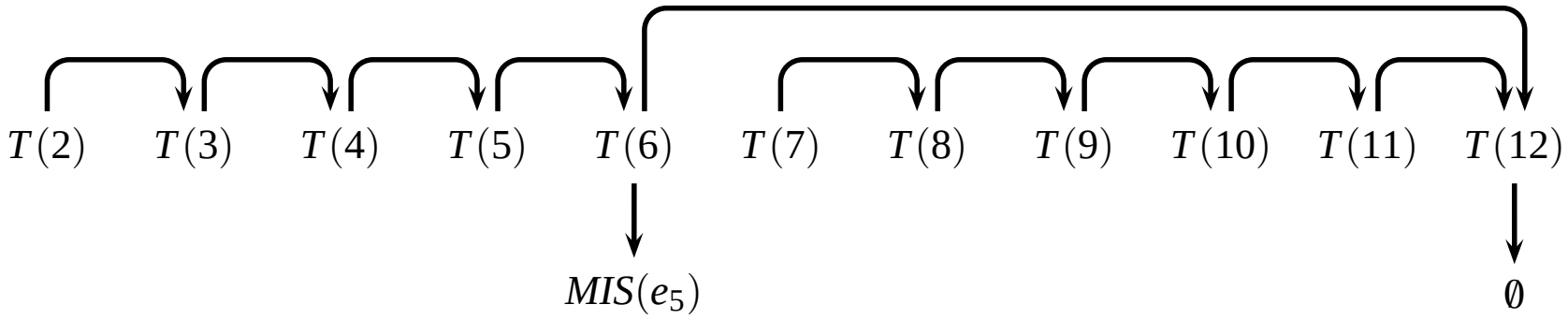
Algorithms on Circle Graphs

Corollary *Let $G = (V, E)$ be a circle graph with n chords. Then, $MIS(G)$ can be found in $O(dn)$ time using $O(n)$ space.*

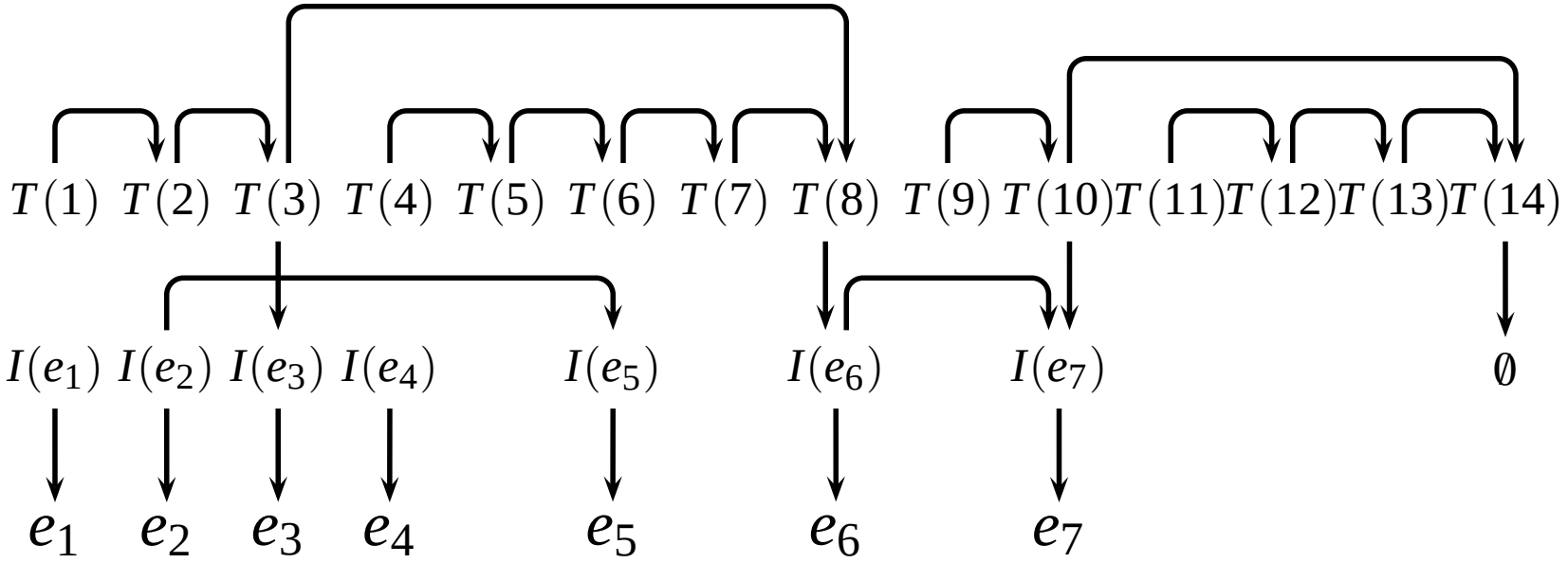
Proof *Immediate.*

- Orient all chords $e \in E$ from lower to higher-numbered vertices.
- Orient all chords $e \in E$ and bucket sort E in nondecreasing order of chord length.
- Compute $MIS(e)$ for all chords $e \in E$.
- Compute $T(i)$ for all vertices $v_i \in V$.
- $MIS(G) = T(1)$.

Example *Shared representation of each set $T(k)$ with $2 < k < 12$ by reference to a maximum independent set (for some shorter chord) and/or to a previous set, when computing $MIS(e_2)$.*



Example *Shared representation of each set $MIS(e)$ with $e \in E$ by reference to zero or more previous maximum independent sets (for shorter chords) and to chord e itself, together with the shared representation of $MIS(G) = T(1)$.*



Algorithms on Circle Graphs

The following choice of simple data structures allows to meet the time and space bounds. Given a circle graph $G = (V, E)$ with n chords and $2n$ vertices,

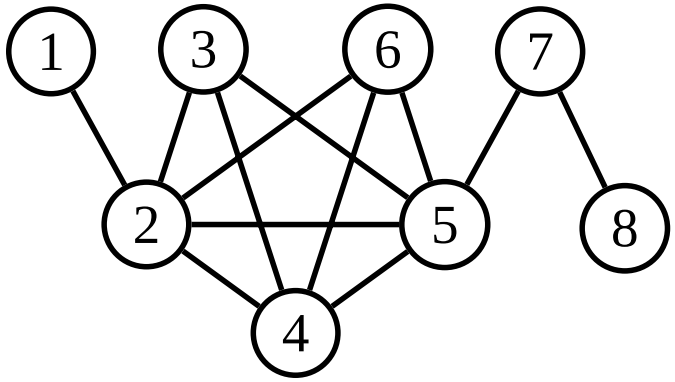
- For each chord $e \in E$, $MIS(e)$ is represented by a *MIS* structure, consisting of a list of pointers to *MIS* structures, a pointer to a chord, and a weight.
- For each vertex $v_i \in V$, $T(i)$ is represented by a T structure, consisting of a pointer to a T structure, a pointer to a *MIS* structure, and a weight.
- $MIS(G) = T(1)$ can be collected in $O(n)$ time by traversing the pointer structure.

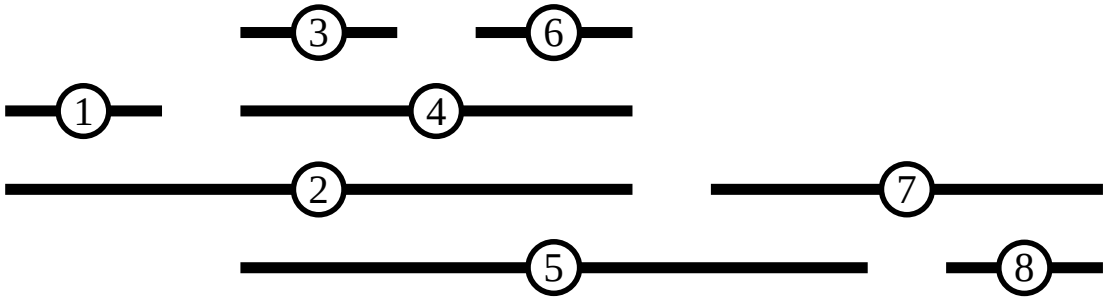
Algorithms on Interval Graphs

Algorithms on Interval Graphs

Definition *An interval graph is a graph that is isomorphic to the intersection graph of a finite set of intervals along a line.*

Example *The following finite set of intervals along a line is an interval model of the interval graph shown to the left.*





Algorithms on Interval Graphs

- Recognition of Interval Graphs

  $O(n^4)$ D. Fulkerson, O. Gross. Incidence Matrices and Interval Graphs. Pacific Journal of Mathematics, 15(3):835–855, 1965.

  $O(n+m)$ Kellogg S. Booth, George S. Lueker. Testing for the Consecutive Ones Property, Interval Graphs and Graph Planarity using PQ-Tree Algorithms. Journal of Computing and System Sciences, 13(3):335–379, 1976.

  $O(n+m)$ Norbert Korte, Rolf H. Möhring. An Incremental Linear-Time Algorithm for Recognizing Interval Graphs. SIAM Journal on Computing, 18(1):68–81, 1989.

  $O(n+m)$ Wen-Lian Hsu. A Simple Test for Interval Graphs. Proc. 18th Int. Workshop Graph-Theoretic Concepts in Computer Science. Lecture Notes in Computer Science, 657, 11–16, 1992.

  $O(n+m)$ Klaus Simon. A New Simple Linear Algorithm to Recognize Interval Graphs. Proc. Int. Workshop on Computational Geometry. Lecture Notes in Computer Science, 553, 289–308, 1991.

Algorithms on Interval Graphs

- Maximum Independent Set of Interval Graphs

  $O(n \log n)$ U. I. Gupta, D. T. Lee, Joseph Y.-T. Leung. Efficient Algorithms for Interval Graphs and Circular-Arc Graphs. Networks, 12(4):459–467, 1982.

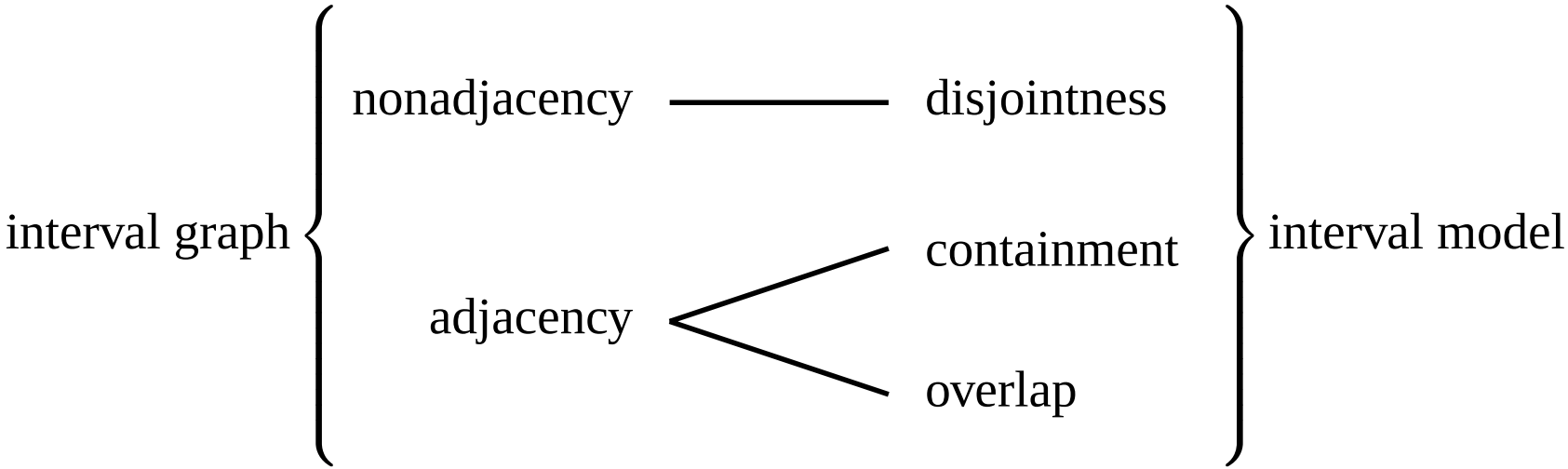
  $O(n)$ Ju Yuan Hsiao, Chuan Yi Tang. An Efficient Algorithm for Finding a Maximum Weight 2-Independent Set on Interval Graphs. Information Processing Letters, 43(5):229–235, 1992.

Algorithms on Interval Graphs

- Isomorphism of Interval Graphs

  $O(n+m)$ George S. Lueker, Kellogg S. Booth. A Linear Time Algorithm for Deciding Interval Graph Isomorphism. Journal of the ACM, 26(2):183–195, 1979.

Algorithms on Interval Graphs



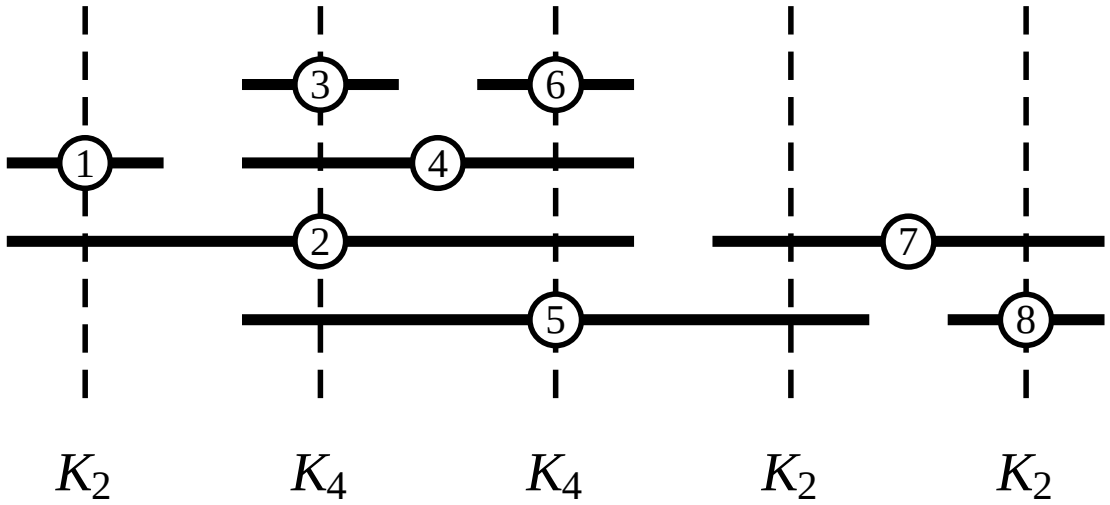
- A set of mutually disjoint intervals of $I(G)$ models an independent set of G.
- A set of mutually overlapping, or contained in each other, intervals of $I(G)$ models a clique of G.

Algorithms on Interval Graphs

Theorem (Gilmore, Hoffman, 1964) *A graph G is an interval graph if and only if the maximal cliques of G can be linearly ordered such that, for every vertex v of G, the maximal cliques containing vertex v occur consecutively.*

- Paul C. Gilmore, A. J. Hoffman. A Characterization of Comparability Graphs and of Interval Graphs. Canadian Journal of Mathematics, 16(3):539–548, 1964.

Example *Consider the following interval representation of the interval graph of the previous example.*



Algorithms on Interval Graphs

Let $C(G)$ be the set of maximal cliques of a graph G, let F be a transitive orientation of the complement of G and, for maximal cliques A_1, A_2 of $C(G)$, let $A_1 < A_2$ if and only if there is an edge of F connecting A_1 with A_2 which is oriented toward A_2.

Proof *Let $G = (V, E)$ be a graph whose maximal cliques can be linearly ordered such that, for every vertex v of G, the maximal cliques containing vertex v occur consecutively.*

- *For each vertex $v \in V$, let $I(v)$ denote the set of all maximal cliques of G which contain v.*
- *The sets $I(v)$, for $v \in V$, are intervals of the linearly ordered set $(C(G), <)$.*
- *Now, for all vertices $u, v \in V$, it holds that $\{u, v\} \in E$ if and only if $I(u) \cap I(v) \neq \emptyset$, because two vertices are adjacent if and only if they are both contained in some maximal clique.*

Therefore, G is an interval graph. (See (Golumbic, 1980) for a proof of the reverse implication.)

The Gilmore–Hoffman theorem has an interesting matrix formulation.

Definition *A matrix whose entries are zeros and ones, is said to have the consecutive ones property for columns if its rows can be permuted in such a way that the ones in each column occur consecutively.*

Example *The following matrix has the consecutive ones property for columns.*

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{pmatrix} \rightarrow \begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

Example *The following matrix does not have the consecutive ones property for columns.*

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Definition *The clique matrix of an undirected graph is an incidence matrix having maximal cliques as rows and vertices as columns.*

Algorithms on Interval Graphs

Corollary *A graph G is an interval graph if and only if the clique matrix of G has the consecutive ones property for columns.*

- D. R. Fulkerson, O. A. Gross. Incidence Matrices and Interval Graphs. Pacific Journal of Mathematics, 15(3):835–855, 1965.

Proof *Let G be an undirected graph and M the clique matrix of G. An ordering of the maximal cliques of G corresponds to a permutation of the rows of M. The corollary follows from the Gilmore-Hoffman theorem.*

Example *The clique matrix of the interval graph of the previous example can be permuted as follows.*

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix}$$

Algorithms on Interval Graphs

Theorem *Interval graphs can be recognized in $O(n+m)$ time.*

- George S. Lueker, Kellogg S. Booth. Testing for the Consecutive Ones Property, Interval Graphs, and Graph Planarity using PQ-Tree Algorithms. Journal of Computing and System Sciences, 13(3):335–379, 1976.

Sketch of recognition algorithm. Given an undirected graph $G = (V, E)$,

1. Verify that G is chordal and, if so, enumerate its maximal cliques.
2. Test whether or not the cliques can be ordered so that those which contain vertex v occur consecutively, for each $v \in V$.

- Step 1 takes $O(n+m)$ time and produces at most n maximal cliques. [Details will be given in the lecture on chordal graphs.]
- Step 2 also takes $O(n+m)$ time, because the clique matrix of an interval graph has $O(n+m)$ nonzero entries, and PQ-trees allow to test a zero-one matrix with r rows, c columns, and f nonzero entries for the consecutive ones property for columns in $O(r+c+f)$ time.

Algorithms on Interval Graphs

Definition *Given a finite set X and a collection F of subsets of X, the consecutive arrangement problem is to determine whether or not there exists a permutation π of X in which the elements of each subset $S \in F$ appear as a consecutive subsequence of π.*

- X is the set of maximal cliques of G.
- $F = \{S(v) \mid v \in V\}$, where $S(v)$ is the set of all maximal cliques of G containing vertex v.

The consecutive arrangement problem (over a finite set X) and the consecutive ones problem (over a zero-one matrix M) are equivalent.

- Each row of M corresponds to an element of X.
- Each column of M corresponds to a subset of X consisting of those rows of M containing a one in the specified column.

Algorithms on Interval Graphs

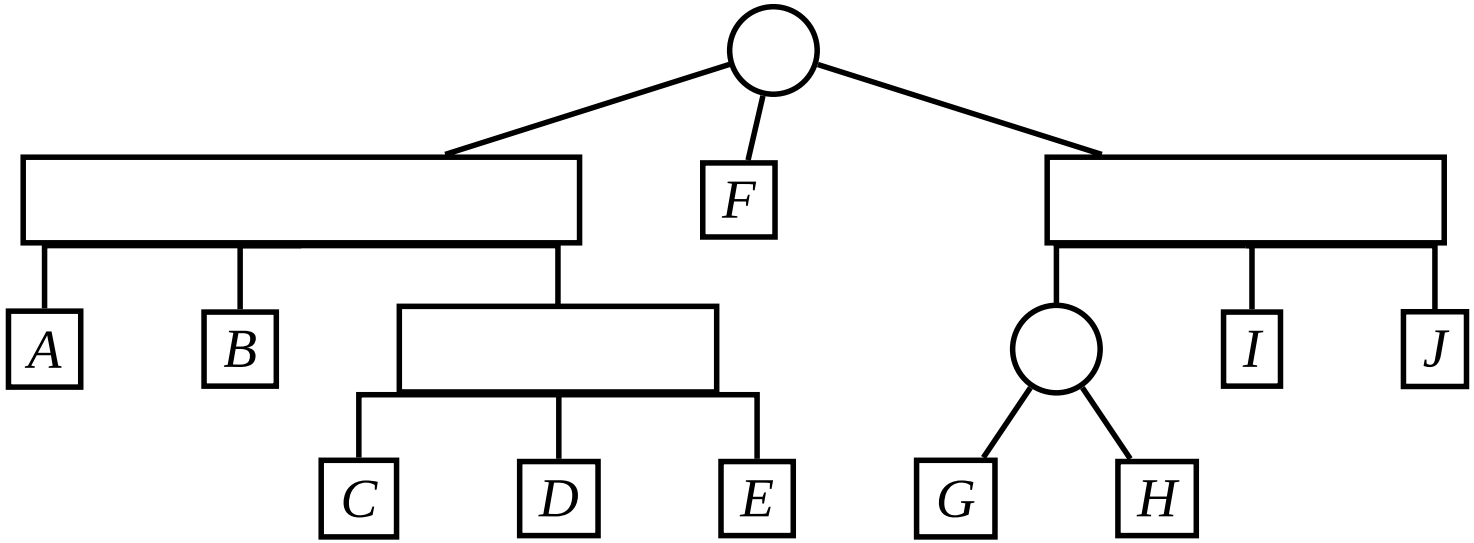
The *PQ*-tree is a data structure allowing to represent in a small amount of space all the permutations of X which are consistent with the constraints of consecutivity determined by F.

Definition *A PQ-tree T is a rooted, ordered tree whose nonterminal nodes fall into two classes, namely P-nodes and Q-nodes.*

- The children of a *P*-node occur in no particular order, while those of a *Q*-node appear in an order which must be locally preserved.
- P-nodes are designated by circles and Q-nodes by wide rectangles.
- The leaves of T are labeled bijectively by the elements of set X.

Definition *The frontier of a PQ-tree is the permutation of X obtained by reading the labels of the leaves from left to right. The frontier of a node is the frontier of the subtree rooted at the node.*

Example *The frontier of the following PQ-tree is* $[A,B,C,D,E,F,G,H,I,J]$.



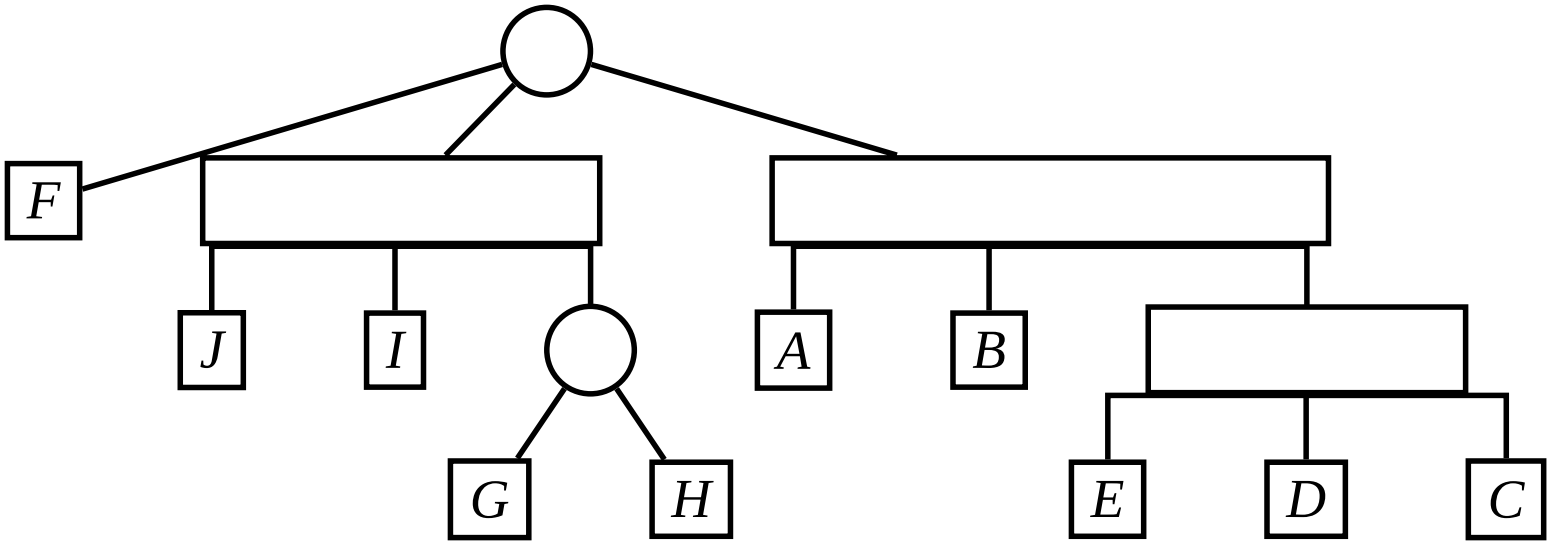
Definition *A PQ-tree is proper if each P-node has at least two children, and each Q-node has at least three children.*

All *PQ*-trees to be considered henceforth are assumed to be proper.

Definition *Two PQ-trees T_1 and T_2 are equivalent, denoted $T_1 \equiv T_2$, if one can be obtained from the other by applying a sequence of the following transformation rules:*

1. *Arbitrarily permute the children of a P-node.*
2. *Reverse the children of a Q-node.*

Example *The following PQ-tree is equivalent to the one of the previous example.*



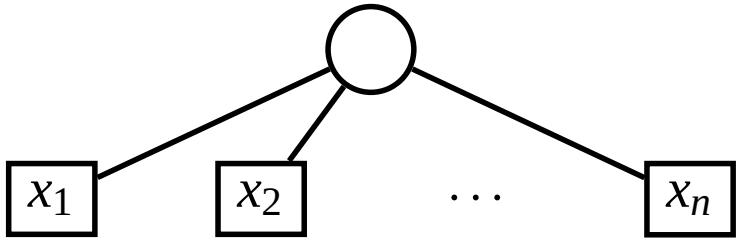
Algorithms on Interval Graphs

Definition *An ordering of the leaves of a PQ-tree T is consistent with T if it is the frontier of a PQ-tree equivalent to T. The set of all orderings consistent with T is called the consistent set of T, and is denoted consistent(T).*

Let $X = \{x_1, x_2, \ldots, x_n\}$. The class of consistent permutations of PQ-trees over X forms a lattice.

- The null tree T_0 has no nodes and $consistent(T_0) = \emptyset$.

- The universal tree T_n has one internal P-node (the root) and a leaf for every element of X, and $consistent(T_n)$ includes all permutations of X.



Algorithms on Interval Graphs

Let F be a collection of subsets of a finite set X, and let $\Pi(F)$ denote the collection of all permutations π of X such that the elements of each subset $S \in F$ occur as a consecutive subsequence of π.

Example *Let* $F = \{\{A,B,C\},\{A,D\}\}$. *Then,*
$\Pi(F) = \{[D,A,B,C],[D,A,C,B],[C,B,A,D],[B,C,A,D]\}$.

Theorem (Booth and Lueker, 1976)

i. *For every collection of subsets* F *of* X *there is a PQ-tree* T *such that* $\Pi(F) = consistent(T)$.

ii. *For every PQ-tree* T *there is a collection of subsets* F *of* X *such that* $\Pi(F) = consistent(T)$.

Example *Let* $F = \{\{A,B\},\{C,D\},\{D,E\},\{B,C,D,E\},\{I,J\},\{G,H\},\{G,H,I\}\}$. *Then,* $[A,B,C,D,E,F,G,H,I,J] \in \Pi(F)$.

Algorithms on Interval Graphs

The following algorithm calculates $\Pi(F)$.

1: **procedure** *consecutive* (X, F, Π)
2: let Π be the set of all permutations of X
3: **for all** $S \in F$ **do**
4: remove from Π those permutations in which the elements of S do not occur as a subsequence
5: **end procedure**

Algorithms on Interval Graphs

Despite the initially exponential size of Π, PQ-trees allow to represent Π using only $O(|X|)$ space.

```
1: procedure consecutive(X, F, Π)
2:     let T be the universal PQ-tree over X
3:     for all S ∈ F do
4:         reduce T using S
5: end procedure
```

The pattern matching procedure *reduce* attempts to apply from the bottom to the top of the PQ-tree a set of 11 templates, consisting of a pattern to be matched against the current PQ-tree and a replacement to be substituted for the pattern.

Algorithms on Interval Graphs

Theorem (Booth and Lueker, 1976) *The PQ-tree representation T of the class of permutations $\Pi(F)$ can be computed in $O(|F|+|X|+\sum_{S\in F}|S|)$ time.*

- X is the set of maximal cliques of G.
- Each $S \in F$ is the set of all maximal cliques of G containing a given vertex of G.
- F is the set of S for all the vertices of G.

Corollary *Let M be a zero-one matrix with r rows, c columns, and f nonzero entries. Then, M can be tested for the consecutive ones property for columns in $O(r+c+f)$ time.*

Theorem (Booth and Lueker, 1976) *Interval graphs can be recognized in $O(n+m)$ time. Moreover, if G is an interval graph, then there is an algorithm taking $O(n+m)$ time to construct a proper PQ-tree T such that consistent(T) is the set of orderings of the maximal cliques of G in which, for every vertex v of G, the maximal cliques containing vertex v occur consecutively.*

Algorithms on Interval Graphs

Let $T(G)$ denote the proper PQ-tree constructed for an interval graph G by the recognition algorithm.
It turns out that isomorphic interval graphs will have equivalent PQ-trees.

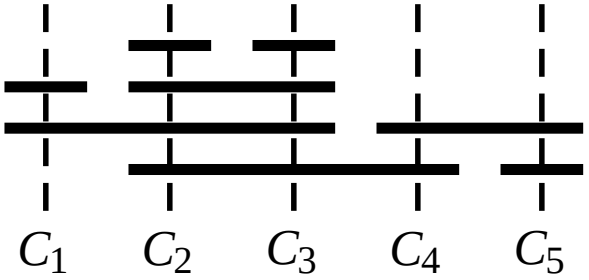
Theorem *If T_1 and T_2 are PQ-trees, with the same number of leaves, such that $consistent(T_1) = consistent(T_2)$, then $T_1 \equiv T_2$.*

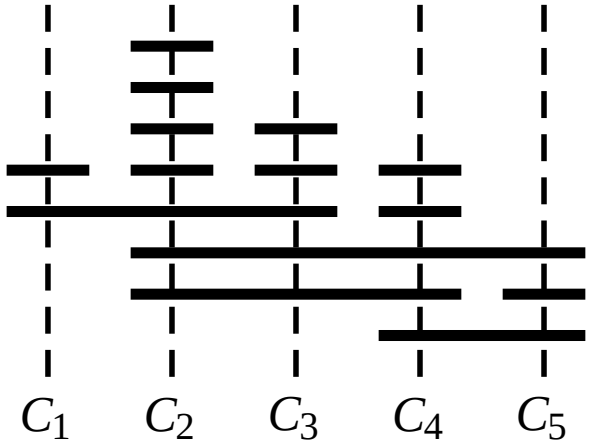
- George S. Lueker, Kellogg S. Booth. A Linear Time Algorithm for Deciding Interval Graph Isomorphism. Journal of the ACM, 26(2):183–195, 1979.

It is possible, though, for interval graphs which are not isomorphic to have equivalent PQ-trees.

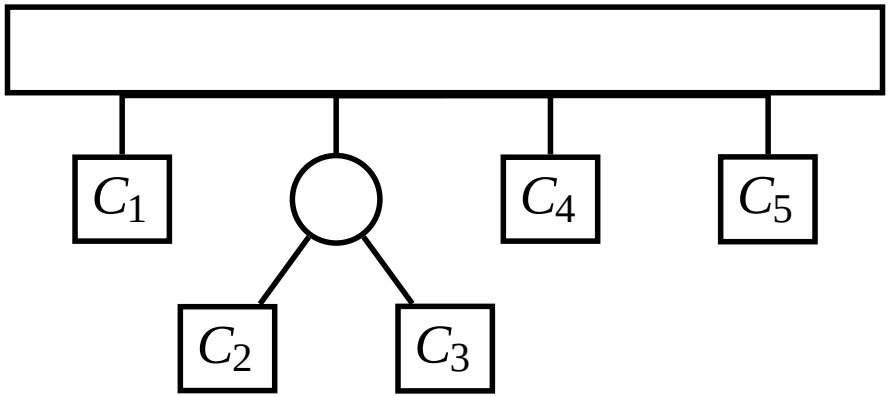
Algorithms on Interval Graphs

Example *The interval graphs with the following interval representation are not isomorphic,*





but the following tree is a proper PQ-tree of either.

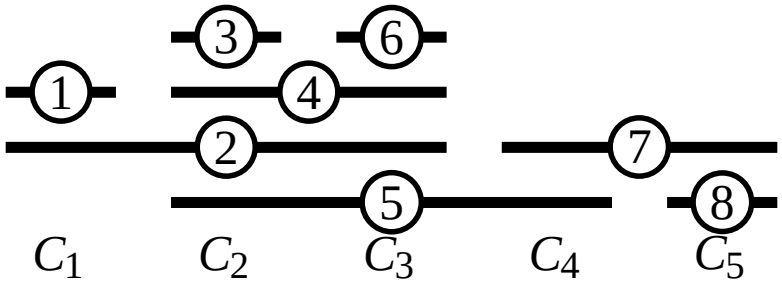


Therefore, the *PQ*-tree will have to be extended with more information about the structure of the interval graph.

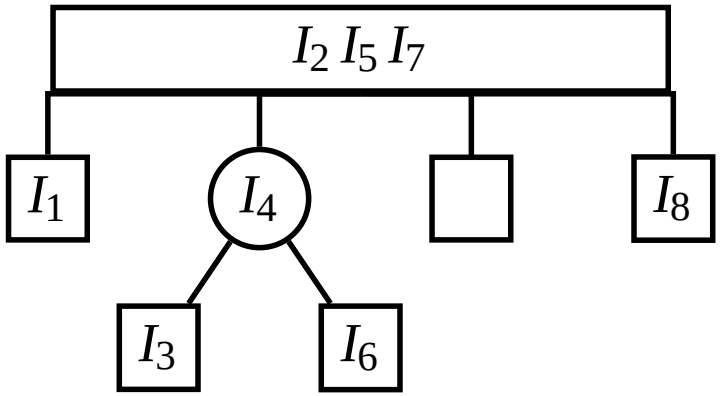
Algorithms on Interval Graphs

Definition *For any vertex v in an interval graph G, the characteristic node of v in a PQ-tree T of G is the deepest node x in T such that the frontier of node x includes the set of maximal cliques containing vertex v.*

Example *Given the following interval representation of the previous example,*



the characteristic nodes of all vertices (intervals) are the following:



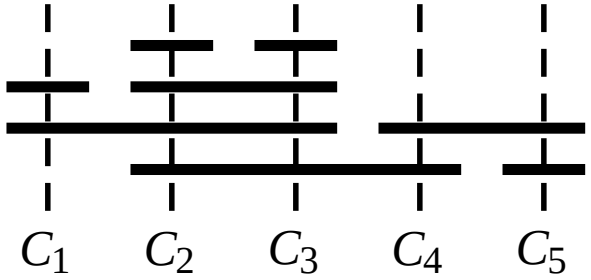
Definition *A labeled PQ-tree is a PQ-tree whose nodes are labeled by strings of integers which indicate how the sets of all maximal cliques containing each vertex are distributed over the frontier of the tree, as follows:*

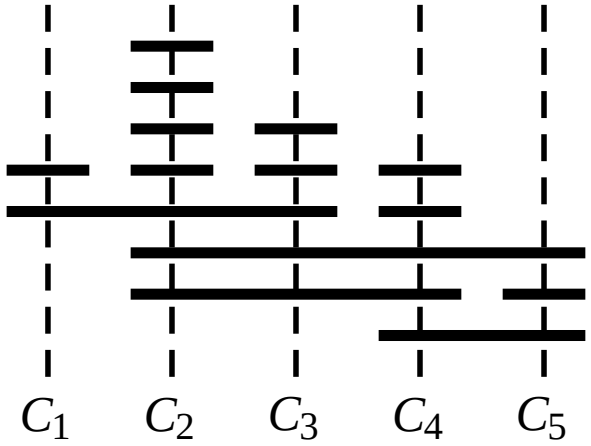
- *If x is a P-node or a leaf, $label[x]$ is set to the number of vertices of G which have x as their characteristic node.*
- *If x is a Q-node, number the children of x as $x_1, x_2, \ldots, x_k$ from left to right. For each vertex v of G having x as characteristic node, form a pair (i, j) such that x_i and x_j are the leftmost and rightmost child of x, respectively, whose frontier belongs to the set of maximal cliques containing vertex v. Sort all these pairs into lexicographically nondecreasing order and concatenate them to form $label[x]$.*

The resultant labeled PQ-tree is denoted $T_L(G)$.

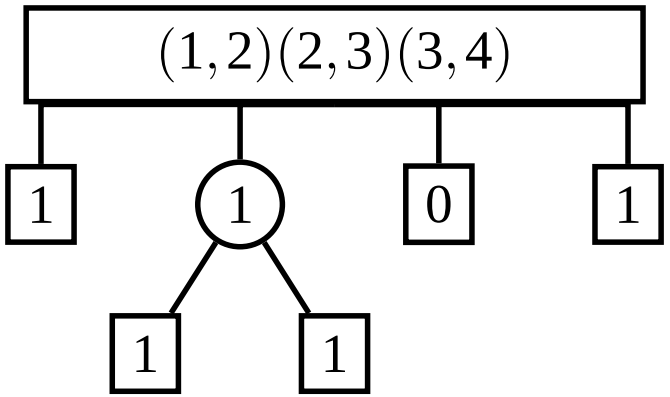
Algorithms on Interval Graphs

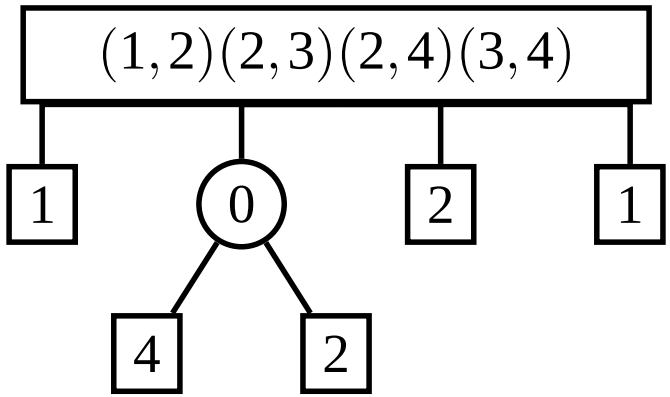
Example *The interval graphs with the following interval representation are not isomorphic,*





and their labeled PQ-trees are not equivalent:





Algorithms on Interval Graphs

Theorem (Lueker and Booth, 1979) *A labeled PQ-tree contains enough information to reconstruct an interval graph up to isomorphism.*

Definition *Two labeled PQ-trees are identical if they are isomorphic as rooted ordered trees and corresponding nodes have identical labels.*

Definition *Two labeled PQ-trees and are equivalent if one can be made identical to the other by a sequence of equivalence transformations, provided labels of Q-nodes whose children are reversed are appropriately modified as follows, for a Q-node x with k children:*

- *Replace each pair (i, j) in label$[x]$ by the pair $(k+1-j, k+1-i)$.*
- *Resort the pairs into lexicographically nondecreasing order.*

Theorem (Lueker and Booth, 1979) *Two interval graphs G_1 and G_2 are isomorphic if and only if $T_L(G_1) \equiv T_L(G_2)$.*

Remark *Equivalence of labeled PQ-trees can be tested using a modification of the Aho-Hopcroft-Ullman tree isomorphism algorithm.*

Algorithms on Interval Graphs

Definition *An interval graph is called proper if it has an interval representation such that no interval is properly contained in another interval.*

Remark *Isomorphism of proper interval graphs can be tested by just computing and comparing canonical labels for the PQ-trees corresponding to adjacency matrices augmented by adding ones along the main diagonal, without need to find any maximal cliques.*

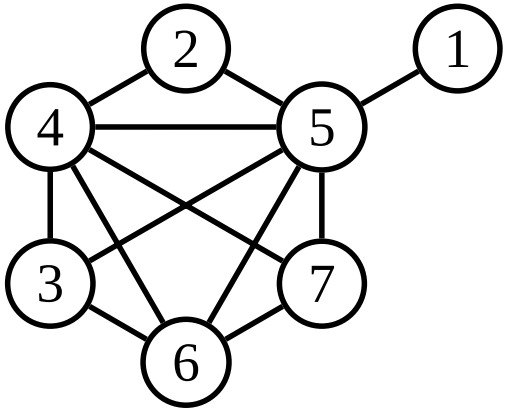
- Lin Chen. Graph Isomorphism and Identification Matrices: Parallel Algorithms. IEEE Transactions on Parallel and Distributed Systems, 7(3):308–319, 1996.
- Lin Chen. Graph Isomorphism and Identification Matrices: Sequential Algorithms. Journal of Computing and System Sciences, 59(3):450–475, 1999.

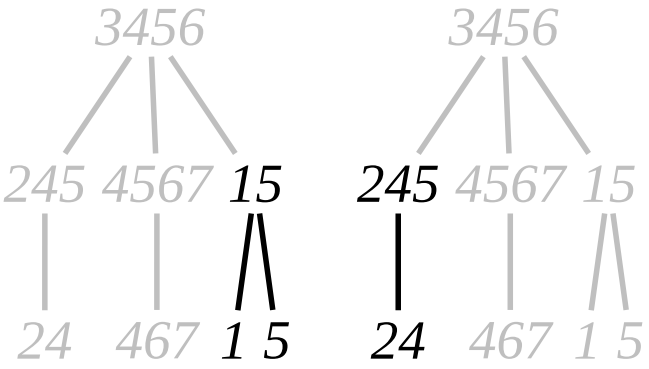
Algorithms on Chordal Graphs

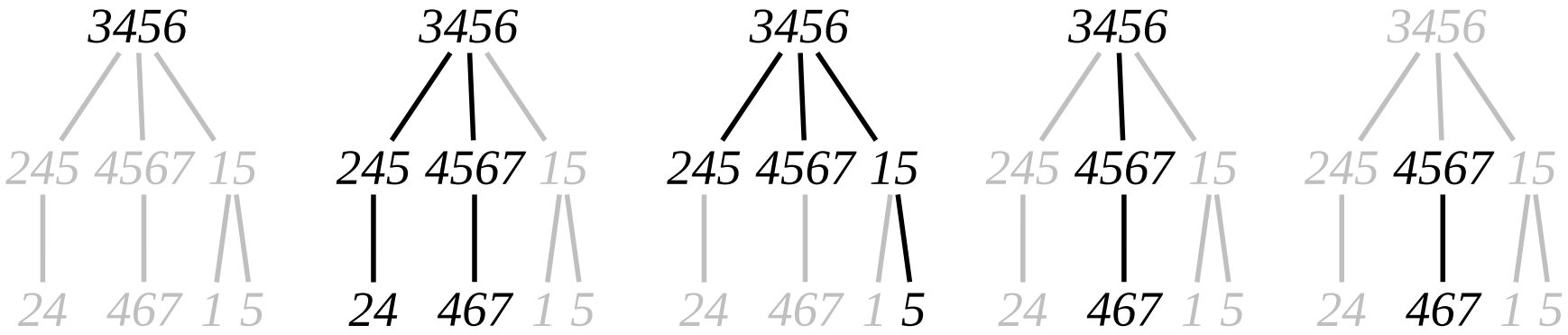
Algorithms on Chordal Graphs

Definition *A chordal graph is a graph that is isomorphic to the intersection graph of a finite set of subtrees of a tree.*

Example *The following family of subtrees of a tree is a subtree model of the chordal graph shown to the left.*







Algorithms on Chordal Graphs

- Recognition of Chordal Graphs

  $O(n+m)$ Donald J. Rose, Robert E. Tarjan, George S. Lueker. Algorithmic Aspects of Vertex Elimination on Graphs. SIAM Journal on Computing, 5(2):266–283, 1976.

  $O(n+m)$ Robert E. Tarjan, Mihalis Yannakakis. Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs. SIAM Journal on Computing, 13(3):566–579, 1984. Addendum. SIAM Journal on Computing, 14(1):254–255, 1985.

Algorithms on Chordal Graphs

- Maximum Independent Set of Chordal Graphs

  $O(n^2)$ Fanica Gavril. Algorithms for Minimum Coloring, Maximum Clique, Minimum Covering by Cliques, and Maximum Independent Set of a Chordal Graph. SIAM Journal on Computing, 1(2):180–187, 1972.

Algorithms on Chordal Graphs

Definition *An undirected graph is called chordal if every cycle of length greater than three posseses a chord, that is, an edge joining two nonconsecutive vertices of the cycle.*

- An undirected graph is chordal if it does not contain an induced subgraph isomorphic to C_n for $n > 3$.

Remark *Being chordal is a hereditary property, inherited by all the induced subgraphs of a chordal graph.*

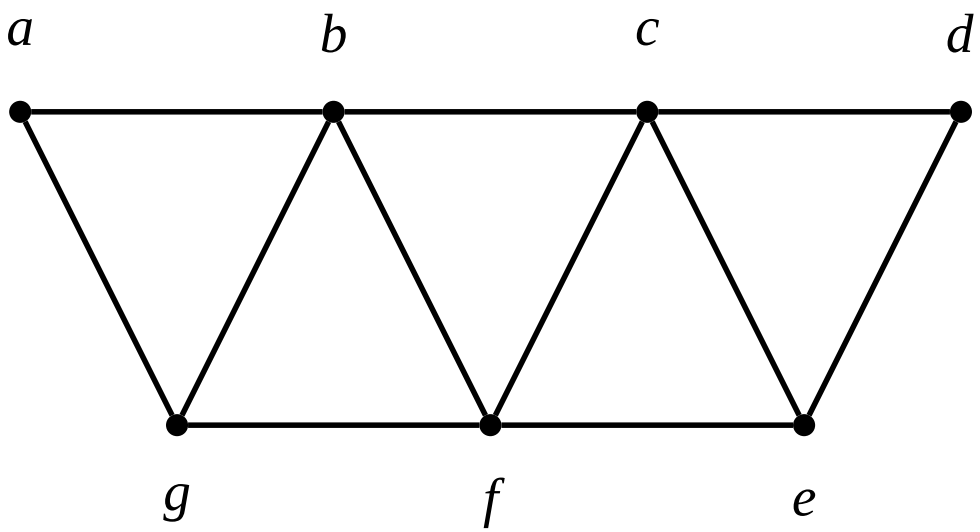
Chordal graphs are also called triangulated and perfect elimination graphs.

Algorithms on Chordal Graphs

Definition *A vertex is called simplicial if its adjacency set induces a complete subgraph, that is, a clique (not necessarily maximal).*

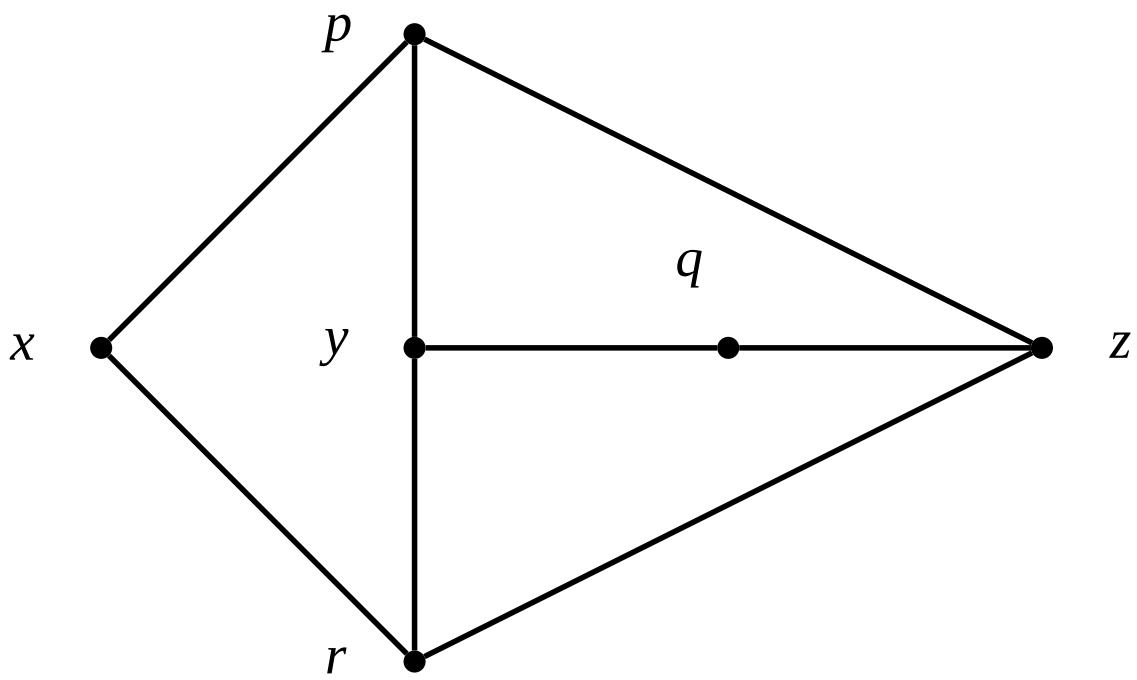
Definition *A permutation $\sigma = [v_1, v_2, \ldots, v_n]$ of the vertices of an undirected graph G, or a bijection $\sigma : V \to \{1, \ldots, n\}$, is called a perfect elimination order if each v_i is a simplicial vertex of the subgraph of G induced by $\{v_i, \ldots, v_n\}$.*

Example *The following chordal graph has 96 different perfect elimination orders, one of which is $\sigma = [a, g, b, f, c, e, d]$.*



Algorithms on Chordal Graphs

Example *The following undirected graph is not chordal. It has no simplicial vertex.*



Algorithms on Chordal Graphs

Theorem *An undirected graph is chordal if and only if it has a perfect elimination order. Moreover, in such a case, any simplicial vertex can start a perfect elimination order.*

- D. R. Fulkerson, O. A. Gross. Incidence Matrices and Interval Graphs. Pacific Journal of Mathematics, 15(3):835–855, 1965.

Lemma *Every chordal graph has a simplicial vertex and, if it is not a clique, then it has two nonadjacent simplicial vertices.*

- Gabriel A. Dirac. On Rigid Circuit Graphs. Abh. Math. Sem. Univ. Hamburg, 25(1–2):71–76, 1961.

Corollary *Chordal graphs can be recognized by the following iterative procedure: repeatedly locate a simplicial vertex and eliminate it from the graph, until no vertices remain (and the graph is chordal) or at some stage no simplicial vertex exists (and the graph is not chordal).*

Algorithms on Chordal Graphs

Remark *A naïve implementation of the previous procedure takes $O(n^4)$ time. As a matter of fact, testing whether a given vertex is simplicial takes $O(n^2)$ time, finding a simplicial vertex requires eventually testing all remaining vertices and thus takes total $O(n^3)$ time, and such a step must be repeated $O(n)$ times when the graph is chordal.*

Chordal graphs can be recognized in linear time, though. Since a chordal graph G which is not a clique has two nonadjacent simplicial vertices, and every induced subgraph of a chordal graph is chordal, it is possible to choose any vertex v of G and decide it will be last in the perfect elimination order, then choose any vertex w adjacent to v and put it in position $n-1$, and so on, choosing vertices backward to the perfect elimination order.

Algorithms on Chordal Graphs

Several linear-time algorithms for recognizing chordal graphs are based on this idea.

- Donald J. Rose, Robert E. Tarjan, George S. Lueker. Algorithmic Aspects of Vertex Elimination on Graphs. SIAM Journal on Computing, 5(2):266–283, 1976.

  Lexicographic search: Number the vertices from n to 1 in decreasing order. For each unnumbered vertex v, maintain a list of the numbers of the numbered vertices adjacent to v, with the numbers in each list arranged in decreasing order. As the next vertex to number, select the vertex whose list is lexicographically greatest, breaking ties arbitrarily.

Algorithms on Chordal Graphs

Several linear-time algorithms for recognizing chordal graphs are based on this idea.

- Robert E. Tarjan, Mihalis Yannakakis. Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs. SIAM Journal on Computing, 13(3):566–579, 1984. Addendum. SIAM Journal on Computing, 14(1):254–255, 1985.

  Maximum cardinality search: Number the vertices from n to 1 in decreasing order. As the next vertex to number, select the vertex adjacent to the largest number of previously numbered vertices, breaking ties arbitrarily.

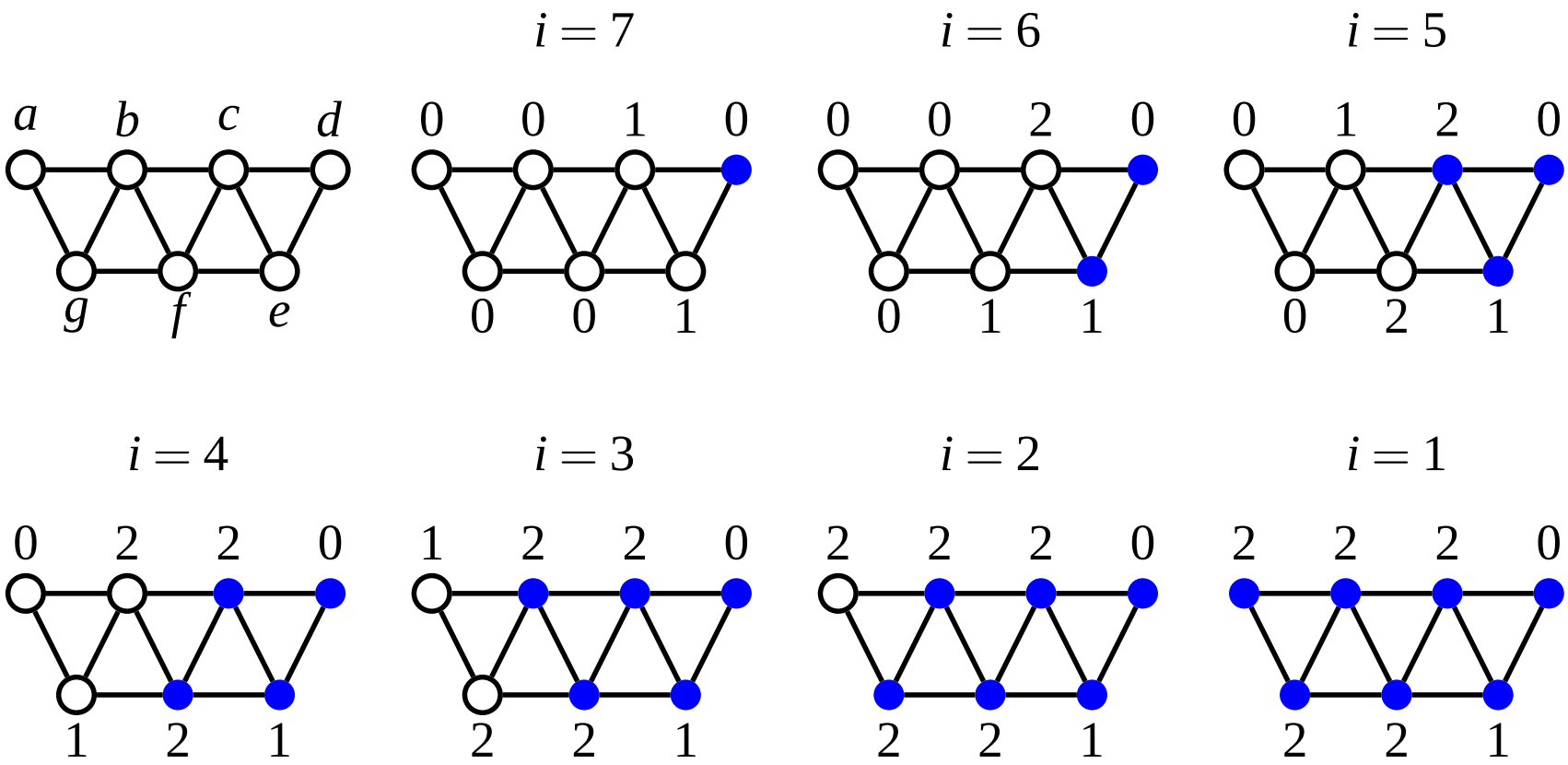
Algorithms on Chordal Graphs

The following algorithm performs a maximum cardinality search on an undirected graph $G = (V, E)$, computing also a perfect elimination order σ when G is chordal.

procedure *maximum cardinality search* (G, σ)
  for all vertices v of G **do**
    set $label[v]$ to zero
  for all i **from** n **downto** 1 **do**
    choose an unnumbered vertex v with largest label
    set $\sigma(v)$ to i {number vertex v}
    for all unnumbered vertices w adjacent to vertex v **do**
      increment $label[w]$ by one
end procedure

Algorithms on Chordal Graphs

Example *The following chordal graph has 96 different perfect elimination orders, one of which,* $\sigma = [a, g, b, f, c, e, d]$*, can be found by maximum cardinality search as follows.*



Algorithms on Chordal Graphs

Theorem *The previous algorithm can be implemented to perform maximum cardinality search in $O(n+m)$ time.*

Proof *Let vertices be labeled by the number of adjacent numbered vertices, and let S_i be the set of unnumbered vertices with label i. Represent each set S_i by a doubly linked list of vertices, to facilitate deletion, and maintain an array of sets of vertices S_i, for $0 \leqslant i \leqslant m-1$. Maintain also an index j to the nonempty set S_j with the largest label, and maintain for each vertex a pointer to the position of the vertex in the corresponding set. Maximum cardinality search proceeds by removing some vertex v from set S_j, numbering vertex v and, for all unnumbered vertices w adjacent to vertex v, moving vertex w from the set containing it, say S_i, to set S_{i+1}. Since this takes $O(1+\deg(v))$ time, maximum cardinality search takes $O(n+m)$ time.*

Algorithms on Chordal Graphs

Theorem (Tarjan and Yannakakis, 1984) *An undirected graph G is chordal if and only if maximum cardinality search(G,σ) produces a perfect elimination order σ.*

Remark *A naïve procedure for testing whether σ is a perfect elimination order, and thus G is chordal, consists of simulating the vertex elimination process and, for each vertex to be eliminated, testing whether its remaining neighbors form a clique. However, this procedure takes $O(nm)$ time.*

Algorithms on Chordal Graphs

Let σ be the permutation of the vertices of G produced by *maximum cardinality search*(G,σ). The following algorithm tests whether σ is a perfect elimination order.

function *perfect elimination order* (G,σ)
  for all i **from** 1 **to** $n-1$ **do**
    set v to $\sigma^{-1}(i)$
    set $m[v]$ to $\sigma^{-1}(\min\{\sigma(w) \mid w \in adj(v), \sigma(w) > \sigma(v)\}$
    for all vertices w adjacent to vertex v **do**
      if $\sigma(m(v)) < \sigma(w)$ **and** $w \notin adj(m(v))$ **then**
        return *false*
  return *true*
end function

Algorithms on Chordal Graphs

Theorem *The previous algorithm can be implemented to test a perfect elimination order in $O(n+m)$ time.*

Proof *The algorithm returns false during the $\sigma(u)$-th iteration if and only if there are vertices v,u,w with $\sigma(v) < \sigma(u) < \sigma(w)$, where $u = m(v)$ is defined during the $\sigma(v)$-th iteration, such that u and w are adjacent to v but u is not adjacent to w. In this case, σ is clearly not a perfect elimination order.*

Conversely, suppose σ is not a perfect elimination order and the algorithm returns true. Let v be the vertex with the largest $\sigma(v)$ possible such that $W = \{w \in adj(v) \mid \sigma(w) > \sigma(v)\}$ is not complete. Let also $u = m(v)$ be the vertex of W defined during the $\sigma(v)$-th iteration. Since during the $\sigma(u)$-th iteration the body of the conditional (return false) is not executed, every vertex $w \in W \setminus \{u\}$ is adjacent to vertex u, and every vertex pair $\{w,z\} \in W \setminus \{u\}$ is adjacent, because of the maximality of $\sigma(v)$, contradicting the assumption that W is not complete.

Regarding time complexity, the body of the algorithm takes $O(\deg(v))$ time for each vertex v. The algorithm takes thus $O(n+m)$ time.

Algorithms on Chordal Graphs

Corollary *Chordal graphs can be recognized in linear time.*

Let $X(v) = \{w \in adj(v) \mid \sigma(v) < \sigma(w)\}$.

Lemma *Every maximal clique of a chordal graph $G = (V,E)$ is of the form $\{v\} \cup X(v)$, for some vertex $v \in V$.*

Proof *The set $\{v\} \cup X(v)$ is complete, because σ is a perfect elimination order. On the other hand, if w is the vertex of a maximal clique C with smallest σ number, then C is of the form $\{w\} \cup X(w)$.*

Lemma *A chordal graph has at most n maximal cliques.*

Remark

- *Suppose* $A = \{v\} \cup X(v)$ *is not a maximal clique. Then, there is another clique* $B = \{w\} \cup X(w)$ *such that* $A \subseteq B$ *and, by the previous to last lemma, it must be* $\sigma(w) < \sigma(v)$.
- *Recall that* $m(w)$ *is defined by* $\sigma(m(w)) = \min\{\sigma(z) \mid z \in X(w)\} = \min\{\sigma(z) \mid z \in adj(w), \sigma(w) < \sigma(z)\}$.
- *Among all cliques* $B = \{w\} \cup X(w)$ *with* $A \subseteq B$, *the one with largest* $\sigma(w)$ *implies* $m(w) = v$ *and, therefore, the clique* $A = \{v\} \cup X(v)$ *is not maximal if and only if* $m(w) = v$ *and* $|X(v)| \leqslant |X(w)| - 1$ *for all vertices* $w \in V$.

Algorithms on Chordal Graphs

Let σ be the perfect elimination order produced by *maximum cardinality search*(G,σ). The following algorithm enumerates all maximal cliques of G.

procedure *all maximal cliques* (G,σ)
  for all vertices v of G **do**
    set $size[v]$ to zero
  for all i **from** 1 **to** n **do**
    set v to $\sigma^{-1}(i)$
    if $\deg(v) = 0$ **then**
      output maximal clique $\{v\}$
    set X to $\{w \in adj(v) \mid \sigma(v) < \sigma(w)\}$
    if $X \neq \emptyset$ **then**
      if $size[v] < |X|$ **then**
        output maximal clique $\{v\} \cup X$
      set $m[v]$ to $\sigma^{-1}(\min\{\sigma(w) \mid w \in X\}$
      set $size[m[v]]$ to $\max\{size[m[v]], |X| - 1\}$
end procedure

Algorithms on Chordal Graphs

Lemma *The previous algorithm can be implemented to enumerate all maximal cliques in $O(n+m)$ time.*

Algorithms on Chordal Graphs

Theorem (Lueker and Booth, 1979) *Graph isomorphism is polynomially reducible to chordal graph isomorphism.*

Proof *A chordal graph $M(G) = G' = (V', E')$ can be associated to an arbitrary graph $G = (V, E)$ as follows.*

- *Let $V' = V \cup E$.*
- *Let $E' = \{\{v, w\} \mid v, w \in V\} \cup \{\{v, e\} \mid v \in V, e \in E, v \text{ is incident with } e\}$.*

The construction can be implemented to take $O(n+m)$ time.

- *Consider any cycle of length greater that three in G'. If the cycle contains only V-vertices, then it has a chord, since all V-vertices are adjacent. Otherwise, the cycle contains an E-vertex, and then the two vertices adjacent to this E-vertex must be V-vertices, and they are adjacent. Therefore, G' is chordal.*
- *Assume now $n \geqslant 4$. It turns out that G' contains enough structure to allow to reconstruct G, up to isomorphism.*

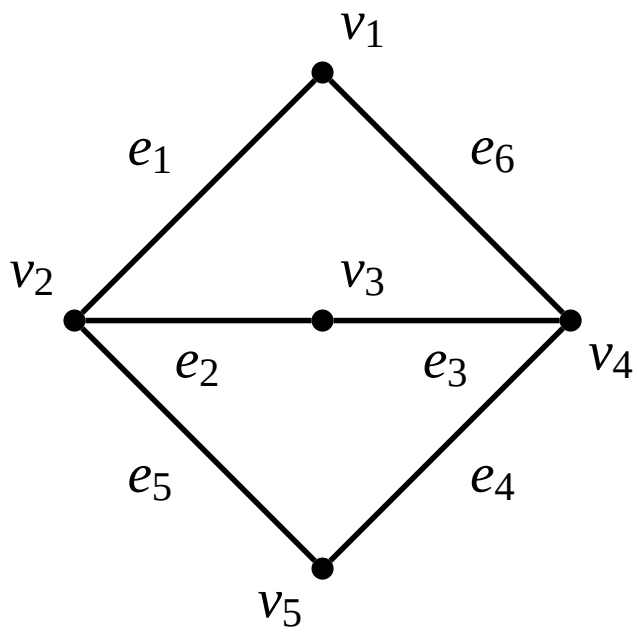
Proof *(Continued.)*

- *As a matter of fact, since all V-vertices are adjacent, all have degree at least equal to $n-1$, which is greater than two, while E-vertices always have degree two, because an E-vertex is adjacent to exactly two V-vertices.*
- *Furthermore, two vertices of G are adjacent if the corresponding V-vertices are adjacent to a common E-vertex.*

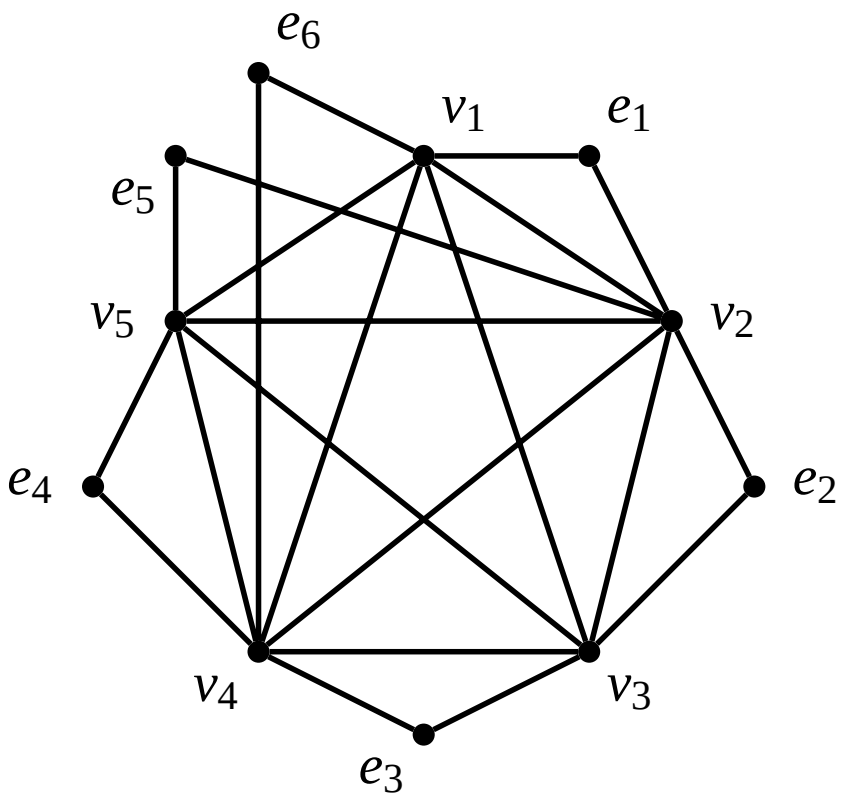
The problem of testing isomorphism of G_1 and G_2 is then polynomially reduced to the problem of testing isomorphism of $M(G_1)$ and $M(G_2)$.

Algorithms on Chordal Graphs

Example *Construction underlying the reduction of graph isomorphism to chordal graph isomorphism.*



G



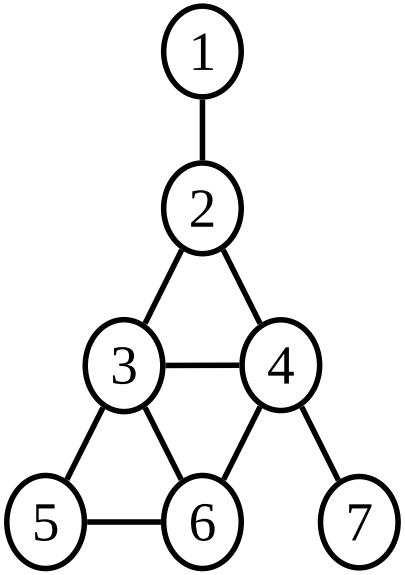
$G' = M(G)$

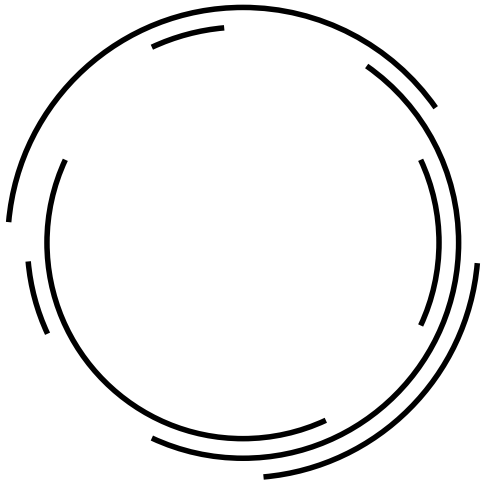
Algorithms on Circular-Arc Graphs

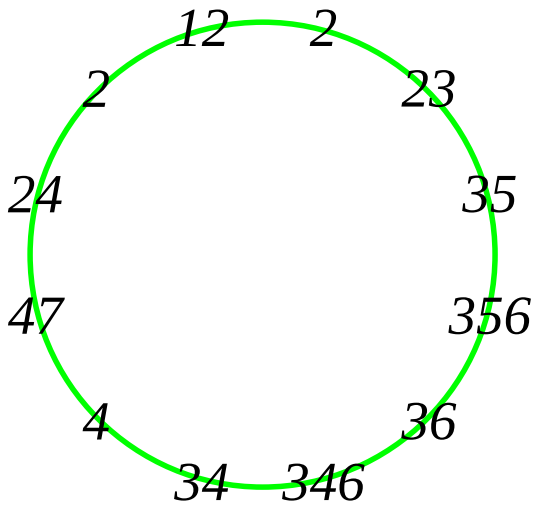
Algorithms on Circular-Arc Graphs

Definition *A circular-arc graph is a graph that is isomorphic to the intersection graph of a finite set of arcs along a circle.*

Example *The following finite set of arcs along a circle is an arc model of the circular-arc graph shown to the left.*







Algorithms on Circular-Arc Graphs

- Recognition of Circular-Arc Graphs

  $O(n^3)$ Alan Tucker. An Efficient Test for Circular-Arc Graphs. SlAM Journal on Computing, 9(1):1–24, 1980.

  $O(mn)$ Wen-Lian Hsu. $O(mn)$ Algorithms for the Recognition and Isomorphism Problems on Circular-Arc Graphs. SIAM Journal on Computing, 24(3):411–439, 1995.

  $O(n^2)$ Elaine M. Eschen, Jeremy Spinrad. An $O(n^2)$ Algorithm for Circular-Arc Graph Recognition. Proc. 4th Annual ACM-SIAM Symposium on Discrete Algorithms, 1993, pp. 128–137.

  $O(n+m)$ Ross M. McConnell. Linear-Time Recognition of Circular-Arc Graphs. Proc. 42nd Annual Symposium on Foundations of Computer Science, pp. 386–394. IEEE Computer Society, 2001.

Algorithms on Circular-Arc Graphs

- Maximum Independent Set of Circular-Arc Graphs

  $O(n^4)$ Fanica Gavril. Algorithms on Circular-Arc Graphs. Networks, 3(3):357–369, 1974.

  $O(n^2)$ U. I. Gupta, D. T. Lee, Joseph Y.-T. Leung. Efficient Algorithms for Interval Graphs and Circular-Arc Graphs. Networks, 12(4):459–467, 1982.

  $O(n^2)$ Toshinobu Kashiwabara, Sumio Masuda. Polynomial Time Algorithms on Circular-Arc Overlap Graphs. Networks, 21(2):195–203, 1991.

  $O(n \log n)$ Sumio Masuda, Kazuo Nakajima. An Optimal Algorithm for Finding a Maximum Independent Set of a Circular-Arc Graph. SIAM Journal on Computing, 17(1):41–52, 1988.

  $O(n+m)$ Wen-Lian Hsu, Jeremy Spinrad. Independent Sets in Circular-Arc Graphs. Journal of Algorithms, 19(2):145–160, 1995.

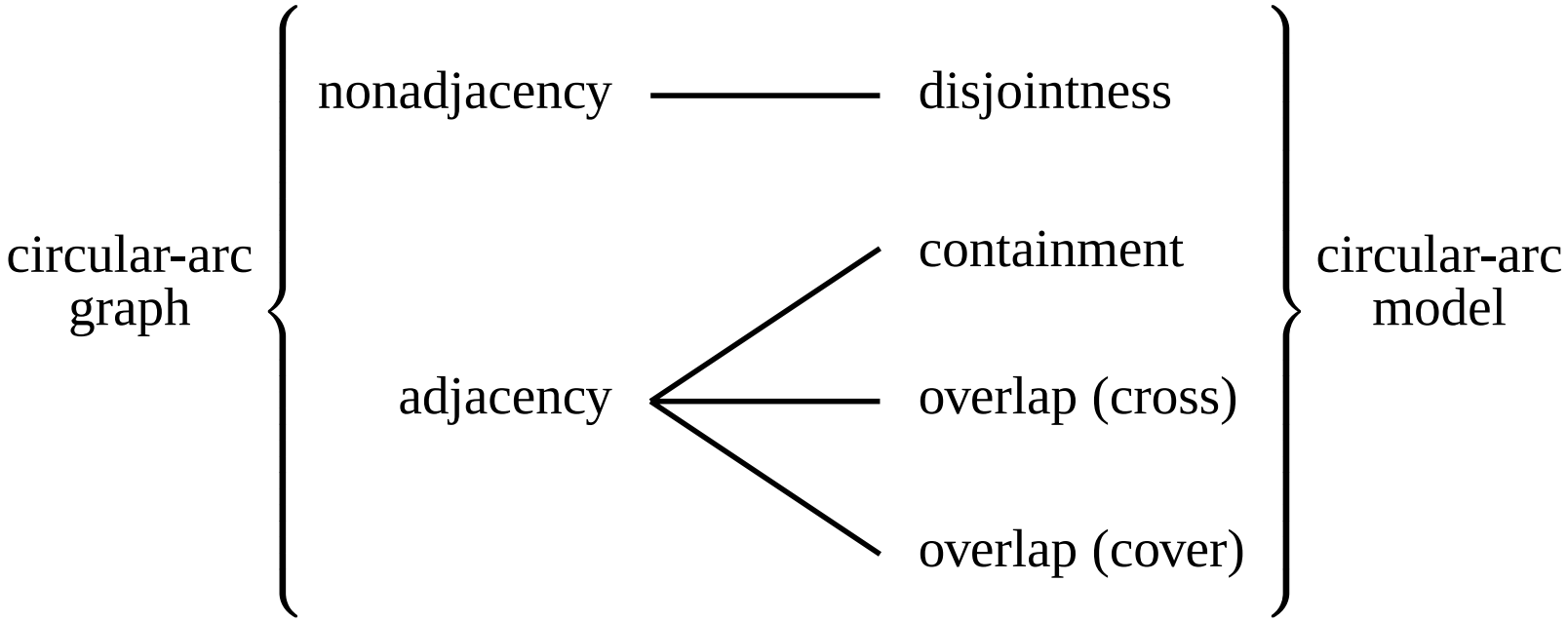
Algorithms on Circular-Arc Graphs

- Isomorphism of Circular-Arc Graphs

  $O(n^2)$ Elaine M. Eschen, Jeremy Spinrad. An $O(n^2)$ Algorithm for Circular-Arc Graph Recognition. Proc. 4th Annual ACM-SIAM Symposium on Discrete Algorithms, 1993, pp. 128–137.

  $O(mn)$ Wen-Lian Hsu. $O(mn)$ Algorithms for the Recognition and Isomorphism Problems on Circular-Arc Graphs. SIAM Journal on Computing, 24(3):411–439, 1995.

Algorithms on Circular-Arc Graphs



- A set of mutually disjoint arcs of $I(G)$ models an independent set of G.
- A set of mutually overlapping, or contained in each other, arcs of $I(G)$ models a clique of G.

Algorithms on Circular-Arc Graphs

Definition *The open neighborhood of a vertex v is $N(v) = \{w \in V \mid vw \in E\}$. The closed neighborhood of a vertex v is the set of vertices $N[v] = N(v) \cup \{v\}$.*

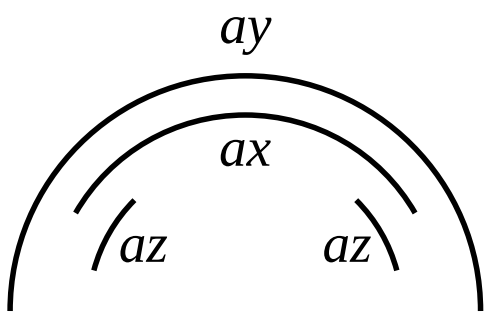
Remark *The overlap relationship between a pair of arcs in any circular-arc model of a circular-arc graph is determined by the neighborhood containment relation between the corresponding vertices.*

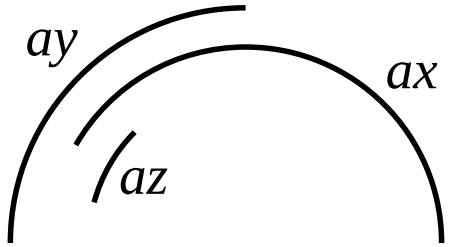
- Arcs ax and ay are disjoint if and only if $xy \notin E$.
- Arc ax can be contained in arc ay if and only if $xy \in E$ and $N(x) \subseteq N[y]$.
- Arcs ax and ay can cover the circle (they can intersect at both endpoints) if and only if $xy \in E$ and, for every $w \in V - N[x]$, $wy \in E$ and $N(w) \subseteq N[y]$.
- Arc ax must cross arc ay (they must intersect at one endpoint only) if and only if $xy \in E$ and neither of the previous two conditions are satisfied.

Lemma *$N(x) \subseteq N[y]$ if and only if $|N(x) - y| = A^2[x,y]$.*

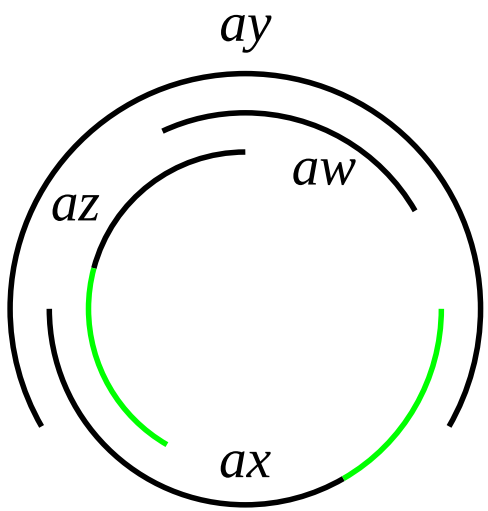
Algorithms on Circular-Arc Graphs

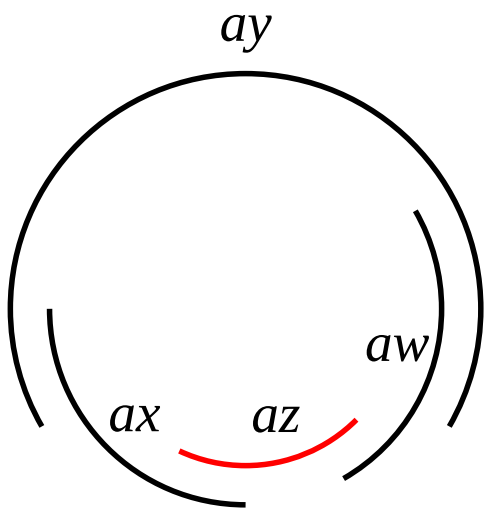
- Arc ax can be contained in arc ay if and only if $xy \in E$ and $N(x) \subseteq N[y]$.





- Arcs ax and ay can cover the circle if and only if $xy \in E$ and, for every $w \in V - N[x]$, $wy \in E$ and $N(w) \subseteq N[y]$.





Algorithms on Circular-Arc Graphs

Theorem *Circular-arc graphs can be recognized in $O(n^{2.37})$ time.*

Proof (Sketch) *Compute the neighborhood containment relation by matrix multiplication. There are two cases to consider.*

1. *$\overline{G}$ is bipartite. (G partitions into two cliques.)*
2. *$\overline{G}$ is not bipartite. (G contains and odd-length cycle as induced subgraph.)*
   (a) *$\overline{G}$ has a triangle.*
   (b) *$\overline{G}$ has a cycle of odd length $\geqslant 5$ as induced subgraph.*

The algorithm consists of three stages.

- *Find a valid placement for an initial set of arcs. (The endpoints of these arcs divide the circle into sections such that no remaining arc can have both its endpoints in the same section.)*
- *Place the endpoints of the remaining arcs in the appropriate sections.*
- *Order the arc endpoints in each section.*