



WIKIPEDIA
The Free Encyclopedia

[Main page](#)

[Contents](#)

[Featured content](#)

[Current events](#)

[Random article](#)

[Donate to Wikipedia](#)

[Wikipedia store](#)

Interaction

[Help](#)

[About Wikipedia](#)

[Community portal](#)

[Recent changes](#)

[Contact page](#)

Tools

[What links here](#)

[Related changes](#)

[Upload file](#)

[Special pages](#)

[Permanent link](#)

[Page information](#)

[Wikidata item](#)

[Cite this page](#)

Print/export

[Create a book](#)

[Download as PDF](#)

[Printable version](#)

Languages

[Српски / srpski](#)

[ไทย](#)

[Edit links](#)

[Create account](#) [Log in](#)

Article

[Talk](#)

[Read](#)

[Edit](#)

[View history](#)



Lexicographic breadth-first search

From Wikipedia, the free encyclopedia

In [computer science](#), **lexicographic breadth-first search** or Lex-BFS is a [linear time](#) algorithm for ordering the vertices of a graph. The algorithm is different from [breadth first search](#), but it produces an ordering that is consistent with breadth-first search.

The lexicographic breadth-first search algorithm is based on the idea of [partition refinement](#) and was first developed by Donald J. Rose, [Robert E. Tarjan](#), and George S. Lueker ([1976](#)). A more detailed survey of the topic is presented by [Corneil \(2004\)](#). It has been used as a subroutine in other graph algorithms including the recognition of [chordal graphs](#), and optimal [coloring](#) of [distance-hereditary graphs](#).

Contents [hide]

1 Algorithm

2 Applications

 2.1 Chordal graphs

 2.2 Graph coloring

 2.3 Other applications

3 Notes

4 References

Graph and tree search algorithms

[α-β](#) · [A*](#) · [B*](#) · [Backtracking](#) · [Beam](#) · [Bellman–Ford](#) · [Best-first](#) · [Bidirectional](#) · [Borůvka](#) · [Branch & bound](#) · [BFS](#) · [British Museum](#) · [D*](#) · [DFS](#) · [Depth-limited](#) · [Dijkstra](#) · [Edmonds](#) · [Floyd–Warshall](#) · [Fringe search](#) · [Hill climbing](#) · [IDA*](#) · [Iterative deepening](#) · [Johnson](#) · [Jump point](#) · [Kruskal](#) · **Lexicographic BFS** · [Prim](#) · [SMA*](#)

Listings

[Graph algorithms](#) · [Search algorithms](#) · [List of graph algorithms](#)

Related topics

[Dynamic programming](#) · [Graph traversal](#) · [Tree traversal](#) · [Search games](#)

v · t · e

Algorithm [edit]

The lexicographic breadth-first search algorithm replaces the [queue](#) of vertices of a standard breadth-first search with an ordered sequence of sets of vertices. The sets in the sequence form a [partition](#) of the remaining vertices. At each step, a vertex *v* from the first set in the sequence is removed from that set, and if that removal causes the set to become empty then the set is removed from the sequence. Then, each set in the sequence is replaced by two subsets: the neighbors of *v* and the non-neighbors of *v*. The subset of neighbors is placed earlier in the sequence than the subset of non-neighbors. In [pseudocode](#), the algorithm can be expressed as follows:

- Initialize a sequence Σ of sets, to contain a single set containing all vertices.
- Initialize the output sequence of vertices to be empty.
- While Σ is non-empty:
 - Find and remove a vertex *v* from the first set in Σ
 - If the first set in Σ is now empty, remove it from Σ
 - Add *v* to the end of the output sequence.
 - For each edge *v*-*w* such that *w* still belongs to a set *S* in Σ :
 - If the set *S* containing *w* has not yet been replaced while processing *v*, create a new empty replacement set *T* and place it prior to *S* in the sequence; otherwise, let *T* be the set prior to *S*.
 - Move *w* from *S* to *T*, and if this causes *S* to become empty remove *S* from Σ .

Each vertex is processed once, each edge is examined only when its two endpoints are processed, and (with an appropriate representation for the sets in Σ that allows items to be moved from one set to another in constant time) each iteration of the inner loop takes only constant time. Therefore, like simpler graph search algorithms such as breadth-first search and [depth first search](#), this algorithm takes linear time.

The algorithm is called lexicographic breadth-first search because the order it produces is an ordering that could also have been produced by a breadth-first search, and because if the ordering is used to index the rows and columns of an [adjacency matrix](#) of a graph then the algorithm [sorts](#) the rows and columns into [Lexicographical order](#).

Applications [\[edit\]](#)

Chordal graphs [\[edit\]](#)

A graph G is defined to be **chordal** if its vertices have a *perfect elimination ordering*, an ordering such that for any vertex v the neighbors that occur later in the ordering form a clique. In a chordal graph, the reverse of a lexicographic ordering is always a perfect elimination ordering. Therefore, one can test whether a graph is chordal in linear time by the following algorithm:

- Use lexicographic breadth-first search to find a lexicographic ordering of G
- Reverse this ordering
- For each vertex v :
 - Let w be the neighbor of v occurring prior to v in the reversed sequence, as close to v in the sequence as possible
 - (Continue to the next vertex v if there is no such w)
 - If the set of earlier neighbors of v (excluding w itself) is not a subset of the set of earlier neighbors of w , the graph is not chordal
- If the loop terminates without showing that the graph is not chordal, then it is chordal.

This application was the original motivation that led [Rose, Tarjan & Lueker \(1976\)](#) to develop the lexicographic breadth first search algorithm.^[1]

Graph coloring [\[edit\]](#)

A graph G is said to be *perfectly orderable* if there is a sequence of its vertices with the property that, for any **induced subgraph** of G , a **greedy coloring** algorithm that colors the vertices in the induced sequence ordering is guaranteed to produce an optimal coloring.

For a chordal graph, a perfect elimination ordering is a perfect ordering: the number of the color used for any vertex is the size of the clique formed by it and its earlier neighbors, so the maximum number of colors used is equal to the size of the largest clique in the graph, and no coloring can use fewer colors. An induced subgraph of a chordal graph is chordal and the induced subsequence of its perfect elimination ordering is a perfect elimination ordering on the subgraph, so chordal graphs are perfectly orderable, and lexicographic breadth-first search can be used to optimally color them.

The same property is true for a larger class of graphs, the **distance-hereditary graphs**: distance-hereditary graphs are perfectly orderable, with a perfect ordering given by the reverse of a lexicographic ordering, so lexicographic breadth-first search can be used in conjunction with greedy coloring algorithms to color them optimally in linear time.^[2]

Other applications [\[edit\]](#)



[Bretscher et al. \(2008\)](#) describe an extension of lexicographic breadth-first search that breaks any additional ties using the **complement graph** of the input graph. As they show, this can be used to recognize **cographs** in linear time. [Habib et al. \(2000\)](#) describe additional applications of lexicographic breadth-first search including the recognition of **comparability graphs** and **interval graphs**.


Notes [\[edit\]](#)

- [^] [Corneil \(2004\)](#).
- [^] [Brandstädt, Le & Spinrad \(1999\)](#), Theorem 5.2.4, p. 71.

References [\[edit\]](#)

- Brandstädt, Andreas; Le, Van Bang; Spinrad, Jeremy (1999), *Graph Classes: A Survey*, SIAM Monographs on Discrete Mathematics and Applications, **ISBN 0-89871-432-X**.
- Bretscher, Anna; [Corneil, Derek](#); Habib, Michel; Paul, Christophe (2008), "A simple linear time LexBFS cograph recognition algorithm" [↗](#), *SIAM Journal on Discrete Mathematics* **22** (4): 1277–1296, doi:10.1137/060664690 [↗](#).
- [Corneil, Derek G.](#) (2004), "Lexicographic breadth first search – a survey", *Graph-Theoretic Methods in Computer Science: 30th International Workshop, WG 2004, Bad Honnef, Germany, June 21-23, 2004, Revised Papers*, Lecture Notes in Computer Science **3353**, Springer-Verlag, pp. 1–19, doi:10.1007/978-3-540-30559-0_1 [↗](#).
- Habib, Michel; McConnell, Ross; Paul, Christophe; Viennot, Laurent (2000), "Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones

testing"  (PDF), *Theoretical Computer Science* **234** (1–2): 59–84, doi:[10.1016/S0304-3975\(97\)00241-7](https://doi.org/10.1016/S0304-3975(97)00241-7) .

- Rose, D. J.; Tarjan, R. E.; Lueker, G. S. (1976), "Algorithmic aspects of vertex elimination on graphs", *SIAM Journal on Computing* **5** (2): 266–283, doi:[10.1137/0205021](https://doi.org/10.1137/0205021) .

Categories: [Graph algorithms](#) | [Search algorithms](#)

This page was last modified on 23 June 2015, at 16:31.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

