Directed acyclic graph

From Wikipedia, the free encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

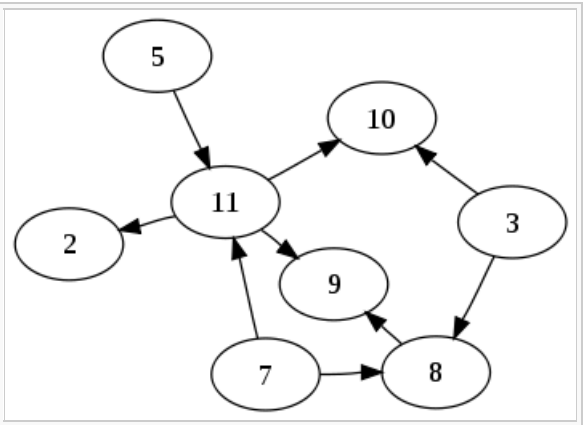
Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

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
Català
Dansk
Deutsch
Ελληνικά
Español
فارسی
Français
Hrvatski
Italiano
Magyar
日本語
Polski
Português
Русский
Suomi
Українська
中文
Edit links

In mathematics and computer science, a **directed acyclic graph** (**DAG** /ˈdæg/), is a directed graph with no directed cycles. That is, it is formed by a collection of vertices and directed edges, each edge connecting one vertex to another, such that there is no way to start at some vertex *v* and follow a sequence of edges that eventually loops back to *v* again.[1][2][3]



An example of a directed acyclic graph

DAGs may be used to model many different kinds of information. The reachability relation in a DAG forms a partial order, and any finite partial order may be represented by a DAG using reachability. A collection of tasks that must be ordered into a sequence, subject to constraints that certain tasks must be performed earlier than others, may be represented as a DAG with a vertex for each task and an edge for each constraint; algorithms for topological ordering may be used to generate a valid sequence. Additionally, DAGs may be used as a space-efficient representation of a collection of sequences with overlapping subsequences. DAGs are also used to represent systems of events or potential events and the causal relationships between them. DAGs may also be used to model processes in which data flows in a consistent direction through a network of processors, or states of a repository in a version-control system.

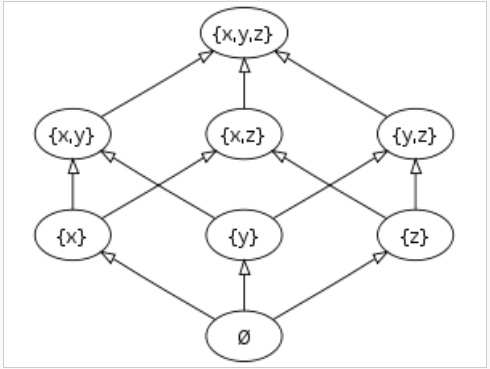
The corresponding concept for undirected graphs is a forest, an undirected graph without cycles. Choosing an orientation for a forest produces a special kind of directed acyclic graph called a polytree. However there are many other kinds of directed acyclic graph that are not formed by orienting the edges of an undirected acyclic graph. Moreover, every undirected graph has an acyclic orientation, an assignment of a direction for its edges that makes it into a directed acyclic graph. For these reasons it would be more accurate to call directed acyclic graphs **acyclic directed graphs** or **acyclic digraphs**.

Contents [hide]

Mathematical properties [edit]

Reachability, transitive closure, and transitive reduction [edit]

Each directed acyclic graph gives rise to a partial order ≤ on its vertices, where $u \leq v$ exactly when there exists a directed path from u to v in the DAG.[4] However, many different DAGs may give rise to this same reachability relation:[5] for example, the DAG with two edges $a \to b$ and $b \to c$ has the same reachability as the graph with three edges $a \to b$, $b \to c$, and $a \to c$. If G is a DAG, its transitive reduction is the graph with the fewest edges that represents the same reachability as G, and its transitive closure is the graph with the most edges that represents the same reachability. The transitive reduction and transitive closure are both uniquely defined for DAGs; in contrast, for a directed graph that is not acyclic, there can be more than one minimal subgraph with the same reachability relation.[6]



A Hasse diagram representing the partial order ⊆ among the subsets of a three-element set.

The transitive closure of G has an edge $u \to v$ for every related pair $u \leq v$ of distinct elements in the reachability relation of G, and may therefore be thought of as a direct translation of the reachability relation ≤ into graph-theoretic terms: every partially ordered set may be translated into a DAG in this way. If a DAG G represents a partial order ≤, then the transitive reduction of G is a subgraph of G with an edge $u \to v$ for every pair in the covering relation of ≤; transitive reductions are useful in visualizing the partial orders they represent, because they have fewer edges than other graphs representing the same orders and therefore lead to simpler graph drawings. A Hasse diagram of a partial order is a drawing of the transitive reduction in which the orientation of each edge is shown by placing the starting vertex of the edge in a lower position than its ending vertex.[7]

Topological ordering [edit]

Every directed acyclic graph has a topological ordering, an ordering of the vertices such that the starting endpoint of every edge occurs earlier in the ordering than the ending endpoint of the edge. In general, this ordering is not unique; a DAG has a unique topological ordering if and only if it has a directed path containing all the vertices, in which case the ordering is the same as the order in which the vertices appear in the path.[8] The family of topological orderings of a DAG is the same as the family of linear extensions of the reachability relation for the DAG,[9] so any two graphs representing the same partial order have the same set of topological orders.

Combinatorial enumeration [edit]

The graph enumeration problem of counting directed acyclic graphs was studied by Robinson (1973).[10] The number of DAGs on n labeled nodes, for $n = 0, 1, 2, 3, \ldots$, (allowing these numbers to appear in any order in a topological ordering of the DAG) is

1, 1, 3, 25, 543, 29281, 3781503, … (sequence A003024 in OEIS).

These numbers may be computed by the recurrence relation

$$a_n = \sum_{k=1}^{n} (-1)^{k-1} \binom{n}{k} 2^{k(n-k)} a_{n-k}.$$[10]

Eric W. Weisstein conjectured,[11] and McKay et al. (2004) proved,[12] that the same numbers count the (0,1) matrices in which all eigenvalues are positive real numbers. The proof is bijective: a matrix A is an adjacency matrix of a DAG if and only if $A + I$ is a (0,1) matrix with all eigenvalues positive, where I denotes the identity matrix. Because a DAG cannot have self-loops, its adjacency matrix must have a zero diagonal, so adding I preserves the property that all matrix coefficients are 0 or 1.

Related families of graphs [edit]

A polytree is a directed graph formed by orienting the edges of a free tree.[13] Every polytree is a DAG. In particular, this is true of the arborescences formed by directing all edges outwards from the root of a tree. A multitree (also called a strongly unambiguous graph or a mangrove) is a directed graph in which there is at most one directed path (in either direction) between any two nodes; equivalently, it is a DAG in which, for every node v, the set of nodes reachable from v forms a tree.[14]

Computational problems [edit]

Topological sorting and recognition [edit]

Main article: Topological sorting

Topological sorting is the algorithmic problem of finding topological orderings; it can be solved in linear time.[15] Kahn's algorithm for topological sorting builds the vertex ordering directly, by maintaining a list of vertices that have no edges connecting them to vertices that have not already been listed, and repeatedly adding one such vertex to the end of the list that is being built.[16] Alternatively, a topological ordering may be constructed by reversing a postorder numbering of a depth-first search graph traversal.[15]

It is also possible to check whether a given directed graph is a DAG in linear time, either by attempting to find a topological ordering and then testing for each edge whether the resulting ordering is valid[17] or alternatively, for some topological sorting algorithms, by verifying that the algorithm successfully orders all the vertices without meeting an error condition.[16]

Construction from cyclic graphs [edit]

Any undirected graph may be made into a DAG by choosing a total order for its vertices and orienting every edge from the earlier endpoint in the order to the later endpoint. However, different total orders may lead to the same acyclic orientation. The number of acyclic orientations is equal to $|\chi(-1)|$, where χ is the chromatic polynomial of the given graph.[18]

Any directed graph may be made into a DAG by removing a feedback vertex set or a feedback arc set. However, the smallest such set is NP-hard to find.[19] An arbitrary directed graph may also be transformed into a DAG, called its condensation, by contracting each of its strongly connected components into a single supervertex.[20] When the graph is already acyclic, its smallest feedback vertex sets and feedback arc sets are empty, and its condensation is the graph itself.

Transitive closure and transitive reduction [edit]

The transitive closure of a given DAG, with n vertices and m edges, may be constructed in time $O(mn)$ by using either breadth-first search or depth-first search to test reachability from each vertex.[21] Alternatively, it can be solved in time $O(n^\omega)$ where $\omega < 2.373$ is the exponent for fast matrix multiplication algorithms; this is a theoretical improvement over the $O(mn)$ bound for dense graphs.[22]

In all of these transitive closure algorithms, it is possible to distinguish pairs of vertices that are reachable by at least one path of length two or more from pairs that can only be connected by a length-one path. The transitive reduction consists of the edges that form length-one paths that are the only paths connecting their endpoints. Therefore, the transitive reduction can be constructed in the same asymptotic time bounds as the transitive closure.[23]

Closure problem [edit]

Main article: Closure problem

The closure problem takes as input a directed acyclic graph with weights on its vertices and seeks the minimum (or maximum) weight of a closure, a set of vertices with no outgoing edges. (The problem may be formulated for directed graphs without the assumption of acyclicity, but with no greater generality, because in this case it is equivalent to the same problem on the condensation of the graph.) It may be solved in polynomial time using a reduction to the maximum flow problem.[24]

Applications [edit]

Path algorithms [edit]

Some algorithms become simpler when used on DAGs instead of general graphs, based on the principle of topological ordering. For example, it is possible to find shortest paths and longest paths from a given starting vertex in DAGs in linear time by processing the vertices in a topological order, and calculating the path length for each vertex to be the minimum or maximum length obtained via any of its incoming edges.[25] In contrast, for arbitrary graphs the shortest path may require slower algorithms such as Dijkstra's algorithm or the Bellman–Ford algorithm,[26] and longest paths in arbitrary graphs are NP-hard to find.[27]

Scheduling [edit]

DAG representations of partial orderings have many applications in scheduling problems for systems of tasks with ordering constraints.[28] For instance, a DAG may be used to describe the dependencies between cells of a spreadsheet: if one cell is computed by a formula involving the value of a second cell, draw a DAG edge from the second cell to the first one. If the input values to the spreadsheet change, all of the remaining values of the spreadsheet may be recomputed with a single evaluation per cell, by topologically ordering the cells and re-

evaluating each cell in this order.[29] Similar problems of task ordering arise in makefiles for program compilation,[29] instruction scheduling for low-level computer program optimization,[30] and PERT scheduling for management of large human projects.[31] Dependency graphs without circular dependencies form directed acyclic graphs.

Data processing networks [edit]

A directed graph may be used to represent a network of processing elements; in this formulation, data enters a processing element through its incoming edges and leaves the element through its outgoing edges. Examples of this include the following:

- In electronic circuit design, static combinational logic blocks can be represented as an acyclic system of logic gates that computes a function of an input, where the input and output of the function are represented as individual bits. In general, the output of these blocks cannot be used as the input unless it is captured by a register or state element which maintains its acyclic properties.[32] Electronic circuit schematics either on paper or in a database are a form of directed acyclic graphs using instances or components to form a directed reference to a lower level component. Electronic circuits themselves are not necessarily acyclic or directed.
- Dataflow programming languages describe systems of values that are related to each other by a directed acyclic graph. When one value changes, its successors are recalculated; each value is evaluated as a function of its predecessors in the DAG.[33]
- In compilers, straight line code (that is, sequences of statements without loops or conditional branches) may be represented by a DAG describing the inputs and outputs of each of the arithmetic operations performed within the code; this representation allows the compiler to perform common subexpression elimination efficiently.[34]
- In most spreadsheet systems, the dependency graph that connects one cell to another if the first cell stores a formula that uses the value in the second cell must be a directed acyclic graph. Cycles of dependencies are disallowed because they cause the cells involved in the cycle to not have a well-defined value. Additionally, requiring the dependencies to be acyclic allows a topological sort to be used to schedule the recalculations of cell values when the spreadsheet is changed.[29]

Causal structures [edit]

Graphs that have vertices representing events, and edges representing causal relations between events, are often acyclic[35] – arranging the vertices in linear order of time, all arrows point in the same direction as time, from parent to child (due to causality affecting the future, not the past), and thus have no loops.

For instance, a Bayesian network represents a system of probabilistic events as nodes in a directed acyclic graph, in which the likelihood of an event may be calculated from the likelihoods of its predecessors in the DAG.[36] In this context, the moral graph of a DAG is the undirected graph created by adding an (undirected) edge between all parents of the same node (sometimes called *marrying*), and then replacing all directed edges by undirected edges.[37]

Another type of graph with a similar causal structure is an influence diagram, the nodes of which represent either decisions to be made or unknown information, and the edges of which represent causal influences from one node to another.[38] In epidemiology, for instance, these diagrams are often used to estimate the expected value of different choices for intervention.[39][40] The role of DAGs in these applications is to convert causal assumptions into conditional independencies constraints, which can be read from the DAG using Pearl's *d*-separation[41] and tested in the data.

Genealogy and version history [edit]

Family trees may also be seen as directed acyclic graphs, with a vertex for each family member and an edge for each parent-child relationship.[42] Despite the name, these graphs are not necessarily trees because of the possibility of marriages between distant relatives (so a child has a common ancestor on both the mother's and father's side) causing pedigree collapse. (The graphs of matrilineal descent ("mother" relationships between women) and patrilineal descent ("father" relationships between men) are trees within this graph.) Because no one can become their own ancestor, these graphs are acyclic.

For the same reason, the version history of a distributed revision control system generally has the structure of a directed acyclic graph, in which there is a vertex for each revision and an edge connecting pairs of revisions that were directly derived from each other; these are not trees in general due to merges.[43]

In many randomized algorithms in computational geometry, the algorithm maintains a *history DAG* representing the version history of a geometric structure over the course of a sequence of changes to the structure. For

instance in a randomized incremental algorithm for Delaunay triangulation, the triangulation changes by replacing one triangle by three smaller triangles when each point is added, and by "flip" operations that replace pairs of triangles by a different pair of triangles. The history DAG for this algorithm has a vertex for each triangle constructed as part of the algorithm, and edges from each triangle to the two or three other triangles that replace it. Tracing a path through this DAG representing the sequence of triangles that contain an individual point allows point location queries to be answered efficiently.[44]

Data compression [edit]

Another type of application of directed acyclic graphs arises in the concise representation of a set of sequences as paths in a graph. For example, the directed acyclic word graph is a data structure in computer science formed by a directed acyclic graph with a single source and with edges labeled by letters or symbols; the paths from the source to the sinks in this graph represent a set of strings, such as English words.[45] Any set of sequences can be represented as paths in a tree, by forming a tree node for every prefix of a sequence and making the parent of one of these nodes represent the sequence with one fewer element; the tree formed in this way for a set of strings is called a trie. A directed acyclic word graph saves space over a trie by allowing paths to diverge and rejoin, so that a set of words with the same possible suffixes can be represented by a single tree node.

The same idea of using a DAG to represent a family of paths occurs in the binary decision diagram,[46][47] a DAG-based data structure for representing binary functions. In a binary decision diagram, each non-sink vertex is labeled by the name of a binary variable, and each sink and each edge is labeled by a 0 or 1. The function value for any truth assignment to the variables is the value at the sink found by following a path, starting from the single source vertex, that at each non-sink vertex follows the outgoing edge labeled with the value of that vertex's variable. Just as directed acyclic word graphs can be viewed as a compressed form of tries, binary decision diagrams can be viewed as compressed forms of decision trees that save space by allowing paths to rejoin when they agree on the results of all remaining decisions.

References [edit]

1. ^ Christofides, Nicos (1975), *Graph theory: an algorithmic approach*, Academic Press, pp. 170–174.
2. ^ Thulasiraman, K.; Swamy, M. N. S. (1992), "5.7 Acyclic Directed Graphs", *Graphs: Theory and Algorithms*, John Wiley and Son, p. 118, ISBN 978-0-471-51356-8.
3. ^ Bang-Jensen, Jørgen (2008), "2.1 Acyclic Digraphs", *Digraphs: Theory, Algorithms and Applications*, Springer Monographs in Mathematics (2nd ed.), Springer-Verlag, pp. 32–34, ISBN 978-1-84800-997-4.
4. ^ Kozen, Dexter (1992), *The Design and Analysis of Algorithms*, Monographs in Computer Science, Springer, p. 9, ISBN 978-0-387-97687-7.
5. ^ Banerjee, Utpal (1993), "Exercise 2(c)", *Loop Transformations for Restructuring Compilers: The Foundations*, Springer, p. 19, ISBN 978-0-7923-9318-4.
6. ^ Bang-Jensen, Jørgen; Gutin, Gregory Z. (2008), "2.3 Transitive Digraphs, Transitive Closures and Reductions", *Digraphs: Theory, Algorithms and Applications*, Springer Monographs in Mathematics, Springer, pp. 36–39, ISBN 978-1-84800-998-1.
7. ^ Jungnickel, Dieter (2012), *Graphs, Networks and Algorithms*, Algorithms and Computation in Mathematics **5**, Springer, pp. 92–93, ISBN 978-3-642-32278-5.
8. ^ Sedgewick, Robert; Wayne, Kevin (2011), "4,2,25 Unique topological ordering", *Algorithms* (4th ed.), Addison-Wesley, pp. 598–599, ISBN 978-0-13-276256-4.
9. ^ Bender, Edward A.; Williamson, S. Gill (2005), "Example 26 (Linear extensions – topological sorts)", *A Short Course in Discrete Mathematics*, Dover Books on Computer Science, Courier Dover Publications, p. 142, ISBN 978-0-486-43946-4.
10. ^ [a] [b] Robinson, R. W. (1973), "Counting labeled acyclic digraphs", in Harary, F., *New Directions in the Theory of Graphs*, Academic Press, pp. 239–273. See also Harary, Frank; Palmer, Edgar M. (1973), *Graphical Enumeration*, Academic Press, p. 19, ISBN 0-12-324245-2.
11. ^ Weisstein, Eric W., "Weisstein's Conjecture", *MathWorld*.
12. ^ McKay, B. D.; Royle, G. F.; Wanless, I. M.; Oggier, F. E.; Sloane, N. J. A.; Wilf, H. (2004), "Acyclic digraphs and eigenvalues of (0,1)-matrices", *Journal of Integer Sequences* **7**, Article 04.3.3.
13. ^ Rebane, George; Pearl, Judea (1987), "The recovery of causal poly-trees from statistical data", *in Proc. 3rd Annual Conference on Uncertainty in Artificial Intelligence (UAI 1987), Seattle, WA, USA, July 1987* (PDF), pp. 222–228.
14. ^ Furnas, George W.; Zacks, Jeff (1994), "Multitrees: enriching and reusing hierarchical structure", *Proc. SIGCHI conference on Human Factors in Computing Systems (CHI '94)*, pp. 330–336, doi:10.1145/191666.191778.
15. ^ [a] [b] Cormen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford (2001) [1990]. *Introduction to Algorithms* (2nd ed.). MIT Press and McGraw-Hill. ISBN 0-262-03293-7. Section 22.4, Topological sort, pp. 549–552.
16. ^ [a] [b] Jungnickel (2012), pp. 50–51.

17. ^ For depth-first search based topological sorting algorithm, this validity check can be interleaved with the topological sorting algorithm itself; see e.g. Skiena, Steven S. (2009), *The Algorithm Design Manual*, Springer, pp. 179–181, ISBN 978-1-84800-070-4.
18. ^ Stanley, Richard P. (1973), "Acyclic orientations of graphs", *Discrete Mathematics* **5** (2): 171–178, doi:10.1016/0012-365X(73)90108-8.
19. ^ Garey, Michael R.; Johnson, David S. (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W. H. Freeman, ISBN 0-7167-1045-5, Problems GT7 and GT8, pp. 191–192.
20. ^ Harary, Frank; Norman, Robert Z.; Cartwright, Dorwin (1965), *Structural Models: An Introduction to the Theory of Directed Graphs*, John Wiley & Sons, p. 63.
21. ^ Skiena (2009), p. 495.
22. ^ Skiena (2009), p. 496.
23. ^ Bang-Jensen & Gutin (2008), p. 38.
24. ^ Picard, Jean-Claude (1976), "Maximal closure of a graph and applications to combinatorial problems", *Management Science* **22** (11): 1268–1272, doi:10.1287/mnsc.22.11.1268, MR 0403596.
25. ^ Cormen et al. 2001, Section 24.2, Single-source shortest paths in directed acyclic graphs, pp. 592–595.
26. ^ Cormen et al. 2001, Sections 24.1, The Bellman–Ford algorithm, pp. 588–592, and 24.3, Dijkstra's algorithm, pp. 595–601.
27. ^ Cormen et al. 2001, p. 966.
28. ^ Skiena (2009), p. 469.
29. ^ [a] [b] [c] Gross, Jonathan L.; Yellen, Jay; Zhang, Ping (2013), *Handbook of Graph Theory* (2nd ed.), CRC Press, p. 1181, ISBN 978-1-4398-8018-0.
30. ^ Srikant, Y. N.; Shankar, Priti (2007), *The Compiler Design Handbook: Optimizations and Machine Code Generation,* (2nd ed.), CRC Press, pp. 19–39, ISBN 978-1-4200-4383-9.
31. ^ Wang, John X (2002), *What Every Engineer Should Know About Decision Making Under Uncertainty*, CRC Press, p. 160, ISBN 978-0-8247-4373-4.
32. ^ Sapatnekar, Sachin (2004), *Timing*, Springer, p. 133, ISBN 978-1-4020-7671-8.
33. ^ Dennis, Jack B. (1974), "First version of a data flow procedure language", *Programming Symposium*, Lecture Notes in Computer Science **19**, pp. 362–376, doi:10.1007/3-540-06859-7_145.
34. ^ Touati, Sid; de Dinechin, Benoit (2014), *Advanced Backend Optimization*, John Wiley & Sons, p. 123, ISBN 978-1-118-64894-0.
35. ^ Gopnik, Alison; Schulz, Laura (2007), *Causal Learning*, Oxford University Press, p. 4, ISBN 978-0-19-803928-0.
36. ^ Shmulevich, Ilya; Dougherty, Edward R. (2010), *Probabilistic Boolean Networks: The Modeling and Control of Gene Regulatory Networks*, Society for Industrial and Applied Mathematics, p. 58, ISBN 978-0-89871-692-4.
37. ^ Cowell, Robert G.; Dawid, A. Philip; Lauritzen, Steffen L.; Spiegelhalter, David J. (1999), "3.2.1 Moralization", *Probabilistic Networks and Expert Systems*, Springer, pp. 31–33, ISBN 0-387-98767-3.
38. ^ Dorf, Richard C. (1998), *The Technology Management Handbook*, CRC Press, p. 9-7, ISBN 978-0-8493-8577-3.
39. ^ Boslaugh, Sarah (2008), *Encyclopedia of Epidemiology, Volume 1*, SAGE, p. 255, ISBN 978-1-4129-2816-8.
40. ^ Pearl, Judea (1995). "Causal diagrams for empirical research". *Biometrika* **82** (4): 669–709. doi:10.1093/biomet/82.4.669.
41. ^ Bayesian network
42. ^ Kirkpatrick, Bonnie B. (April 2011), "Haplotypes versus genotypes on pedigrees", *Algorithms for Molecular Biology* **6** (10), doi:10.1186/1748-7188-6-10, PMC 3102622, PMID 21504603.
43. ^ Bartlang, Udo (2010), *Architecture and Methods for Flexible Content Management in Peer-to-Peer Systems*, Springer, p. 59, ISBN 978-3-8348-9645-2.
44. ^ Pach, János; Sharir, Micha, *Combinatorial Geometry and Its Algorithmic Applications: The Alcalá Lectures*, Mathematical surveys and monographs **152**, American Mathematical Society, pp. 93–94, ISBN 978-0-8218-7533-9.
45. ^ Crochemore, Maxime; Vérin, Renaud (1997), "Direct construction of compact directed acyclic word graphs", *Combinatorial Pattern Matching*, Lecture Notes in Computer Science, Springer, pp. 116–129, doi:10.1007/3-540-63220-4_55.
46. ^ Lee, C. Y. (1959), "Representation of switching circuits by binary-decision programs", *Bell Systems Technical Journal* **38**: 985–999, doi:10.1002/j.1538-7305.1959.tb01585.x.
47. ^ Akers, Sheldon B. (1978), "Binary decision diagrams", *IEEE Transactions on Computers* **C–27** (6): 509–516, doi:10.1109/TC.1978.1675141.

External links [edit]

- Weisstein, Eric W., "Acyclic Digraph", *MathWorld*.

Categories: Directed graphs | Graphical models | Statistical models | Databases