Directed Graphs

Stephen B. Maurer , '67
Swarthmore College, smaurer1@swarthmore.edu

8.2.1 ATTRIBUTES OF A GRAPH MODEL

Definitions:

A ***mathematical representation*** of a physical or behavioral phenomenon is a correspondence between the parts and processes of that phenomenon and a mathematical system of objects and functions.

A ***model*** of a physical or behavioral phenomenon is the mathematical object or function assigned to that phenomenon under a mathematical representation.

Modeling is the mathematical activity of designing models and comprehensive mathematical representations of physical and behavioral phenomena.

A ***graph model*** is a mathematical representation that involves a graph.

Examples:

1. Table 1 gives many examples of graph models. Each example states what the vertices and edges (or arcs) represent and where in the *Handbook* details on the application can be found.

8.3 DIRECTED GRAPHS

Assigning directions to the edges of a graph greatly enhances modeling capability, and is natural whenever order is important, e.g., in a hierarchical structure or a one-way road system. Also, any graph may be viewed as a digraph, by replacing each edge with two directed edges, one in each direction. Many graph problems are best solved as special cases of digraph problems, for instance, finding shortest paths, maximum flows, and connectivity.

8.3.1 DIGRAPH MODELS AND REPRESENTATIONS

Most graph terminology applies equally well to digraphs, e.g., subgraph, self-loop, bipartite, isomorphic, empty. The definitions below are special to digraphs or take on a somewhat different meaning for digraphs. In context, where it is clear that only digraphs are being discussed, "directedness" is often an implicit attribute of an "edge", "path", and other terms.

Definitions:

A ***directed graph***, or ***digraph***, consists of:

- a set V, whose elements are called ***vertices***,
- a set E, whose elements are called ***directed edges*** or ***arcs***, and
- an ***incidence function*** that assigns to each edge a ***tail*** and a ***head***.

The ***tail*** of an arc is the vertex it leaves, and the ***head*** is the vertex it enters.

A ***strict digraph*** has no self-loops or multi-arcs.

The ***underlying graph of a digraph*** is the graph obtained from the digraph by replacing every directed edge by an undirected edge.

Table 1 **Directory of graph models.**

subject area and application	*vertex attributes and meaning edge/arc attributes and meaning*	*reference*
computer programming flowcharts	vertex labels are program steps edge directions show flow	§8.1.1
social organization social networks	vertices are persons edges represent interactions	§8.1.1
civil engineering road networks	vertices are road intersections edges are roads	§8.1.1, §8.3.1
operations research scheduling	vertices are activities arcs show operational precedence	§8.3.1
sociology hierarchical dominance	vertices are individuals arcs show who reports to whom	§8.3.1
computer programming subprogram calling diagram	vertices are subprograms arcs show calling direction	§8.3.1
ecology food webs	vertices are species arcs show who eats whom	§8.3.1
operations research scheduling	vertices are activities to be scheduled edges are activity conflicts	§8.3.1, §8.6.1
genealogy "family trees"	vertices are family members arcs show parenthood	§8.3.1
set theory binary relations	vertices are elements arcs show relatedness	§8.3.1
probabilistic analysis Markov models	vertices are process states edges are state transitions	§8.3.2
traffic control assigning one-way streets	vertices are intersection edges are streets	§8.3.3
partially ordered sets Hasse diagrams	vertices are elements arcs show covering relation	§8.3.4
computer engineering communications networks	vertices are computational nodes arcs are communications links	§8.4.2
operations research transportation networks	vertices are supply and demand nodes arcs are supply lines	§8.4.2
walking tours Seven Bridges of Königsberg	vertices are land masses edges are bridges	§8.4.3
postal delivery routing Chinese Postman Problem	vertices are street intersections edges are streets	§8.4.3
information theory Gray codes	vertices are binary strings edges are single-bit changes	§8.4.4
radio broadcasting assignment of frequencies	vertices are broadcast stations edges are potential interference	§8.6.1
chemistry preventing explosions	vertices are chemicals edges are co-combustibility	§8.6.1

subject area *and application*	*vertex attributes and meaning* *edge/arc attributes and meaning*	*reference*
cartography map-coloring	regions are countries edges are borders	§8.6.4
highway construction avoiding overcrossings	vertices are road intersections edges are roads	§8.7.1
electrical network boards avoiding insulation	vertices are circuit components edges are wires	§8.7.1
VLSI computer chips minimizing layering	vertices are circuit components edges are wires	§8.7.4
information management binary search trees	vertices are data records edges are decisions	§17.1.4
computer operating systems priority trees	vertices are prioritized jobs edges are priority relations	§17.1.5
physical chemistry counting isomers	vertices are atoms edges are molecular bonds	§9.3.2
network optimization min-cost spanning trees	edges are connections edge-labels are costs	§10.1.1
bipartite matching personnel assignment	parts are people and jobs edges are job-capabilities	§10.2.2
network optimization shortest path	vertices are locations edge-labels are distances	§10.3.1
traveling salesman routing shortest complete tour	vertices are locations edge-labels are distances	§10.7.1

The ***out-degree*** of vertex v, denoted $\delta^+(v)$, is the number of arcs with tail at v.

The ***in-degree*** of vertex v, denoted $\delta^-(v)$, is the number of arcs with head at v.

A digraph D is ***transitive*** if whenever it contains an arc from u to v and an arc from v to w, it also contains an arc from u to w.

The ***adjacency matrix*** A_D of a digraph D is

$$A_D = [a_{ij}], \text{ where } a_{ij} = \text{number of arcs from } v_i \text{ to } v_j.$$

The ***incidence matrix*** M_D of a digraph D with no self-loops is $M_D = [b_{ij}]$, where

$$b_{ij} = \begin{cases} +1, & \text{if } v_i \text{ is the tail of } e_j \text{ but not the head} \\ -1, & \text{if } v_i \text{ is the head of } e_j \text{ but not the tail} \\ 0, & \text{otherwise.} \end{cases}$$

There is no standard convention for self-loops.

Facts:

1. *Strict-digraph terminology*: In a context focusing primarily on strict digraphs, there is often a different terminological convention:

- "digraph" refers to a strict digraph;
- a directed graph with multi-arcs is called a *multidigraph*;
- a directed graph with self-loops is called a *pseudodigraph*;
- an arc with tail u and head v is designated uv.

2. *Alternative "path" terminology*: There is an alternative convention in which a (directed) "path" may use vertices and arcs more than once, but an "elementary path" does not repeat arcs, and a "simple path" does not repeat vertices (and, hence, does not repeat arcs either). See §8.3.2.

3. The incidence structure of a digraph is frequently represented by an *arc list*, in which each arc is represented by an ordered pair uv, where u is its tail and v is its head. For each arc with tail u and head v, there is a separate entry, so that uv occurs as often as the number of such arcs. A list of the isolated vertices plus such an arc list completely specifies a digraph.

4. Another common specification of a digraph is the *lists-of-neighbors representation*. For each vertex u, there is a corresponding row, which has as an entry the head of each arc whose tail is u. Thus a vertex v occurs in that row as many times as there are arcs from u to v.

5. The incidence matrix is another common way to represent a digraph. Since all but one or two of the entries in every column are zero, the incidence matrix is a highly inefficient form of representation.

6. The adjacency matrix is also a common way to specify a digraph in some contexts when there is no reason to identify the arcs by name.

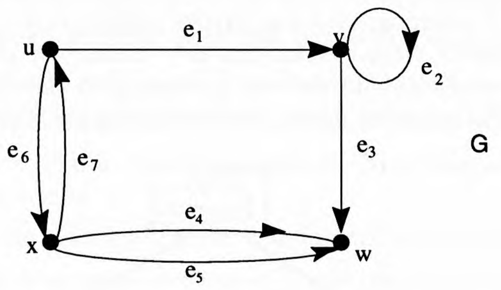
7. A digraph can be represented by a $2 \times |E|$ *incidence table* in which the tail and head of each arc e appear in column e. Direction on an arc can be indicated by a convention as to whether tail or head appears in the first row, which requires swapping the two column entries if the direction is changed. Alternatively, direction can be indicated by marking one of the two entries in each column as the head, and then moving the marker if the direction changes.

8. A row-sum in a directed adjacency matrix equals the out-degree of the corresponding vertex. A column-sum equals the in-degree.

9. In any digraph, the sum of the in-degrees, the sum of the out-degrees, and the number of edges are all equal to each other; i.e., $\sum_{v \in V} \delta^-(v) = \sum_{v \in V} \delta^+(v) = |E|$.

Examples:

1. The following arc list, incidence table, list-of-neighbors, and adjacency matrix all represent the digraph G.



arc list:

$uv,\ vv,\ vw,\ xw,\ xw,\ ux,\ xu$

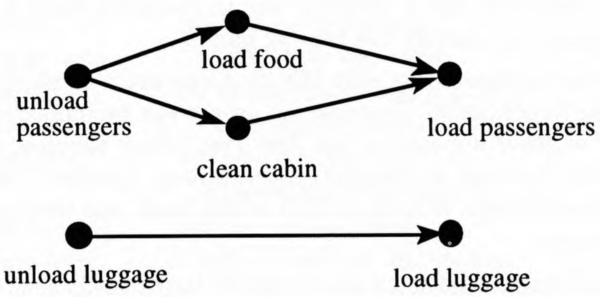
incidence table:

e_1	e_2	e_3	e_4	e_5	e_6	e_7
u	v	v	x	x	u	x
v	v	w	w	w	x	u

lists-of-neighbors: $\left\{\begin{array}{l} u : v, x \\ v : v, w \\ w : \emptyset \\ x : w, w, u \end{array}\right\}$ adjacency matrix: $\begin{array}{c|cccc} & u & v & w & x \\ \hline u & 0 & 1 & 0 & 1 \\ v & 0 & 1 & 1 & 0 \\ w & 0 & 0 & 0 & 0 \\ x & 1 & 0 & 2 & 0 \end{array}$

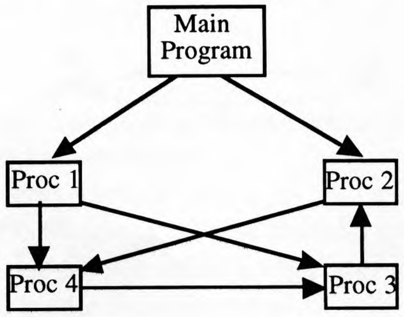
2. *Civil Engineering*: A road network in which at least some of the roads are one-way can be modeled by a digraph. The nodes are road junctures; each two-way road is represented by a pair of arcs, one in each direction. Loops are allowed, and they may represent "circles" that occur in housing developments and in industrial parks. Similarly, multiarcs may occur.

3. *Operations Research*: A large project consists of many smaller tasks with a *precedence relation* — some tasks must be completed before certain others can begin. The vertices represent tasks, and there is an arc from u to v if task u must be completed before v can begin. For instance, in the following figure it is necessary both that food is loaded and the cabin is cleaned before passengers are loaded.

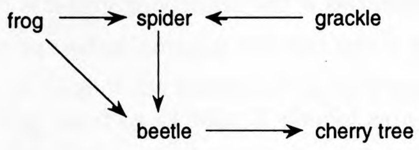


4. *Sociology and Sociobiology*: A business (or army, or society, or ant colony) has a *hierarchical dominance* structure. The nodes are the employees (soldiers, citizens, ants) and there is an arc from u to v if u dominates v. If the chain of command is unique, with a single leader, and if only arcs representing immediate authority are included, then the result is a *rooted tree*. (See §9.1.2.)

5. *Computer Software Design*: A large program consists of many subprograms, some of which can invoke others. Let the nodes of D be the subprograms, and let there be an arc from u to v if subprogram u can invoke subprogram v. Then the *call graph* D encapsulates all possible ways control can flow within the program. Directed cycles represent indirect recursion, and serve as a warning to the designer to ensure against infinite loops. See the following figure, where subprogram 2 can call itself indirectly.



6. *Ecology*: A *food web* is a strict digraph in which nodes represent species and in which there is an arc from u to v if species u eats species v. The following figure shows a small food web.



7. *Operations Research*: A sequence of books must be printed and bound, using one press and one binding machine. Suppose that book i requires time p_i for printing and time b_i for binding. It is desired to print the books in such an order that the binding machine is never idle: when it finishes one book, the next book should already be printed. The vertices of a digraph D can represent the books. There is an arc from book i to book j if $p_j \leq b_i$. Then any path through all the vertices corresponds to a permissible ordering.

8. *Genealogy*: A "family tree" is a digraph where the orientation is traditionally given not by arrows but by the direction down for later generations. Despite the name, a family tree is usually not a tree, since people commonly marry distant cousins, knowingly or unknowingly.

9. *Binary relations*: To any binary relation R on a set V (see §12.1) a digraph $D(V, R)$ can be associated: the vertices are the elements of V, and there is an arc from u to v if $(u, v) \in R$. Conversely, every digraph without multiple arcs defines a binary relation on its vertices. The relation R is transitive (see §12.1.2) if and only if the digraph $D(V, R)$ is transitive.

8.3.2 PATHS, CYCLES, AND CONNECTEDNESS

Definitions:

A ***directed walk*** is a sequence of arcs such that the head of one arc is the tail of the next arc.

The ***length*** of a directed walk is the number of arcs in the sequence.

A ***closed directed walk*** is a directed walk that begins and ends at the same vertex.

A ***directed trail*** is a directed walk in which no arc is repeated.

A ***directed path*** is a directed trail in which no vertex is repeated.

A ***directed cycle*** is a closed directed trail in which no vertices are repeated, except the starting and stopping vertex.

Vertex v is ***reachable*** from vertex u if there is a directed path from u to v.

A ***basis for a digraph*** is a set of vertices V' such that every vertex not in V' is reachable from V' and such that no proper subset of V' has this property.

The ***distance*** from a vertex u to a vertex v in a digraph D is the length of the shortest directed path from u to v.

A digraph is ***strongly connected*** (or ***diconnected***, or ***strong***) if every vertex is reachable from every other vertex.

A digraph is ***unilaterally connected*** (or ***unilateral***) if for every pair of vertices u and v, there is either a uv-path or a vu-path.

A digraph D is ***weakly connected*** (or ***weak***) if the underlying graph is connected.

The digraph D is ***disconnected*** if the underlying graph is disconnected.

A ***strong component*** of a digraph is a maximal subgraph that is strongly connected.

A digraph $D(V, E)$ is ***reducible*** if the vertex set V may be partitioned into a disjoint union $V_1 \cup V_2$ so that all arcs joining V_1 and V_2 go from V_1 to V_2.

The ***condensation*** D^* of a digraph D is the strict digraph whose nodes are the strong components $\{V_1, V_2, \ldots, V_k\}$ of D, with an arc $V_iV_j \in E_{D^*}$ if and only if there is an arc vv' in D such that $v \in V_i$ and $v' \in V_j$.

The ***converse of a digraph*** D is obtained by reversing the directions of all the arcs of D.

The ***directional dual*** of a theorem about digraphs is the statement obtained by replacing each property in the theorem statement by its converse.

Facts:

1. Using a pencil on a drawing of a digraph, a directed walk can be traversed by following the arrows without lifting the pencil from the graph.

2. Distance in digraphs need not be symmetric. That is, the distance from u to v might be different from the distance from v to u.

3. If A is the adjacency matrix of D, then the ij entry of A^n is the number of n-arc walks from v_i to v_j.

4. Let δ^+ be the smallest out-degree of a strict digraph D. If $\delta^+ > 0$, then D has a cycle of length at least $\delta^+ + 1$.

5. Let δ^- be the smallest in-degree of a strict digraph D. If $\delta^- > 0$, then D has a cycle of length at least $\delta^- + 1$.

6. The directional dual of a theorem about digraphs is a theorem about digraphs.

7. Fact 5 is the directional dual of Fact 4.

8. A digraph D is Eulerian (§8.4.3) if and only if the underlying graph is connected and in-degree equals out-degree at every vertex.

9. A digraph D has an Euler uv-trail (where $u \neq v$) if the following conditions hold:

- the out-degree of vertex u exceeds the in-degree by one;
- the in-degree of v exceeds the out-degree by one;
- at every other vertex, the in-degree equals the out-degree.

That is,

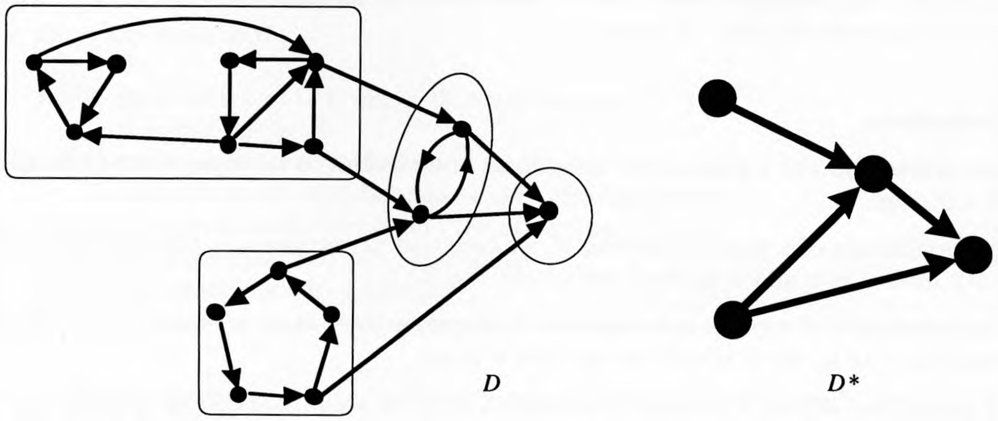
- $d^+(u) = d^-(u) + 1$;
- $d^-(v) = d^+(v) + 1$;
- $(\forall w \neq u, v)\ [d^-(w) = d^+(w)]$.

Other Euler-type results for graphs generalize to digraphs as well.

10. Let δ be the minimum of all in- and out-degrees of D. If D is strict and $\delta \geq \frac{|V|}{2} > 1$, then D contains a Hamilton cycle (§8.4.4).

11. Hamilton theory is much harder and less complete than Euler theory, for digraphs as for graphs.

12. The strong components of a digraph D partition the vertices of D, but not the arcs, since some arcs go from one component to another. However, the maximal unilateral subgraphs do partition the arcs. If V_1, V_2 are the vertex sets of two strong components of D, then all arcs between V_1 and V_2 face the same way — either all are from V_1 or all are to V_1. See the following figure.



13. The condensation of any digraph is an acyclic digraph (§8.3.4). See the figure for Fact 12.

14. A digraph is reducible if and only if its condensation has at least two vertices.

15. A digraph is unilateral if and only if its condensation is a path.

16. A set V' is a basis of a digraph D if and only if V' consists of one vertex from each strong component of D that has in-degree 0 in D^*. Thus, every basis of a digraph has the same number of vertices.

17. The eigenvalues of a digraph D are the union (counting multiplicities) of the eigenvalues of its strong components. (See §8.9.3.)

Examples:

1. Let u, v be vertices of an n-vertex digraph D with adjacency matrix A. If v is reachable from u, then some uv-path has length $\leq n-1$. Thus, D is strong if and only if every entry of $\sum_{k=0}^{n-1} A^k$ is positive. There are more computationally efficient tests for diconnectivity: Warshall's algorithm (§14.2) and directed depth first search (§13.3.2).

2. Let M be an arbitrary square matrix. Computation of the eigenvalues of M can sometimes be speeded up as follows. Create matrix A by replacing each nonzero entry of M by a '1', and then let D be the digraph with adjacency matrix A. The eigenvalues of M are the union of the eigenvalues of the minors of M indexed by the strong components of D. (If one component has vertices v_1, v_3, v_7, then one minor has rows and columns $1, 3, 7$ of M.) If M is *sparse* (few nonzeros), then digraph D will usually have many small components and this approach will be efficient.

3. *Markov models*: Let V represent a set of states and E the possible transitions of a Markov process (§7.7). Then walks through D represent "histories" that the process can follow.

8.3.3 ORIENTATION

There are many natural questions concerning when the edges of an undirected graph could be assigned directions so as to obtain a certain sort of digraph. For instance, when can a graph be oriented to obtain a strong digraph? An application of this last question is to determine when a set of roads could all be made one-way, while keeping all points reachable from all others.

Definitions:

An ***orientation*** of a graph is an assignment of directions to its edges, thereby making it a digraph.

An orientation of a graph is ***strong*** if, for each pair of vertices u, v, there is a directed path from u to v and a directed path from v to u.

An orientation of a graph is ***transitive*** if, whenever there is an arc from u to v and an arc from v to w, there is also an arc from u to w.

A graph that admits a transitive orientation is called a ***comparability graph***.

A ***cut-edge*** (or ***bridge***) of a graph is an edge whose removal would increase the number of components (§8.4.1).

A ***2-edge-connected*** graph G is connected and has no cut-edge.

A ***generalized circuit*** in a graph is a closed walk (§8.4.1) that uses each edge at most once in each direction.

A ***triangular chord*** for a closed walk (§8.4.1) $u_1, u_2, \ldots, u_k, u_1$ is a proper edge that joins two vertices exactly two apart on the walk.

Facts:

1. Let $\chi(G)$ be the chromatic number (§8.6.1) of graph G. Then every orientation of G has a path of length at least $\chi(G) - 1$.

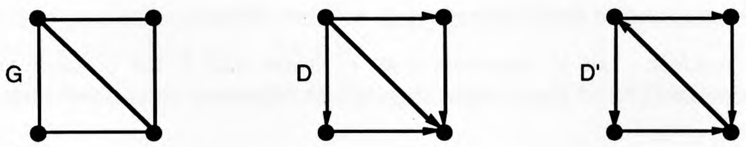
2. A graph G has a strong orientation if and only if G is 2-edge-connected. (H. Robbins, 1939)

3. A graph G is a comparability graph if and only if every generalized circuit of G of odd length > 3 has a triangular chord.

4. Algorithms 1 and 2 give ways of creating a strong orientation in a 2-edge-connected graph.

Examples:

1. In the figure below the digraph D is a weak transitive orientation of the graph G and D' is a strong nontransitive orientation.



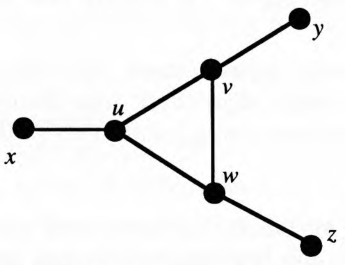
Algorithm 1: Naive algorithm for creating a strong orientation.

```
{This algorithm is good to use by hand for small graphs.}
input: a 2-edge-connected graph $G$
output: a strong orientation of $G$
$H$ := any cycle in $G$
direct $H$
while some vertex of $G$ is not in directed subgraph $H$
   $v$ := a vertex not in $H$
   find two edge-disjoint paths from $v$ to $H$
      {Two such paths exist because $G$ is 2-edge-connected}
   direct one path from $v$ to $H$ and the other from $H$ to $v$
   $H$ := $H$ with these two subgraph added
orient any remaining edges arbitrarily
```

Algorithm 2: Better algorithm for creating a strong orientation.

```
{A good algorithm for large graphs or for computer implementation.}
input: a 2-edge-connected graph
output: a strong orientation
select an arbitrary vertex as root
construct the Depth-First-Search spanning tree from that root  {See §9.2.2.}
orient the tree edges downward from the root
orient all back edges upward toward the root
orient all cross edges arbitrarily
```

2. The following graph is not transitively orientable, and $x, u, v, y, v, w, z, w, u, x$ are the vertices of a generalized circuit without a triangular chord.



3. *Traffic control*: The flow of traffic on crowded city streets can sometimes be improved by making streets one-way. When this is done, it is necessary that a car can travel legally between any two locations. Assigning directions to the edges of the graph representing the street grid is an orientation of this graph, and cars can travel legally between any two points if and only this graph has a strong orientation. Consequently, by Robbins' theorem (Fact 2), to make all the streets one-way without losing mutual accessibility of locations, it is necessary and sufficient that the grid of streets be 2-edge-connected.

```
Algorithm 3: Naive topological sort.
  {Construct a linear extension ordering for a DAG.}
  input: a DAG $D$
  output: a numbering of the vertices in a topsort order
  $H := D$; $k := 1$
    while $V_H \neq \emptyset$
      $v_k :=$ a vertex of $H$ of in-degree 0  {This exists. See Fact 1.}
      $H := H - v_k$  {Remaining graph is still a DAG.}
      $k := k + 1$
```

8.3.4 DIRECTED ACYCLIC GRAPHS

Definitions:

A digraph is ***acyclic*** if it has no *directed* cycles. A directed acyclic graph is sometimes called a DAG.

A ***source*** of a digraph is a vertex of in-degree zero.

A ***sink*** of a digraph is a vertex of out-degree zero.

A ***linear extension ordering*** of the n vertices of a digraph is a consecutive labeling $v_1, v_2, \ldots, v_n$ so that, if there is an arc from v_i to v_j, then $i < j$. (See also §11.2.5.)

A ***topological sort***, or ***topsort***, is an algorithm that assigns a linear extension ordering to a DAG. This traditional name belies the facts that it is not quite a sort, in the usual sense of sorting, and that its relation to topology (in the sense understood by topologists) is obscure.

Facts:

1. Every DAG has at least one source, and by duality, at least one sink.

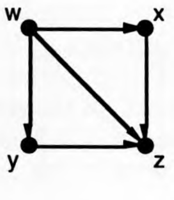
2. Every DAG has a unique basis (§8.3.2), namely, the set of all its sources.

3. Topsort yields a linear ordering for the vertices that makes the adjacency matrix of a DAG upper-triangular.

4. Doing a preliminary topsort permits many optimization problems about paths to be solved subsequently by a single algorithmic pass through the vertices in the topsort order; see §15.2.2 (dynamic programming) and §15.5 (critical paths). See Algorithm 3.

Examples:

1. In the following digraph vertex w is a source and vertex z is a sink. It is a DAG, even though the underling graph has cycles. Labeling the vertices either in the order w, x, y, z or w, y, x, z is a linear extension ordering.



2. Consider any digraph whose vertices represent discrete events, and whose arcs go from earlier events to later events. Any such digraph is acyclic. Conversely, any digraph whose vertices represent procedural steps and whose arcs represent required precedence can be scheduled (using a topological sort) so that arcs do in fact go forward in time.

3. The Hasse diagram of a poset (§12.3.5) is a DAG, as is the entire graph of a poset (arc from u to v if and only if $u \geq v$).

8.3.5 TOURNAMENTS

Definitions:

A ***tournament*** is a digraph with exactly one arc between each pair of distinct vertices. An ***n-tournament*** has n vertices.

The ***score vector*** of a tournament is the sequence of out-degrees of the vertices (number of arcs leaving each vertex), usually in ascending order.

A tournament T is ***regular*** if every vertex has the same outdegree.

A tournament T is ***strong*** if there is a directed path between each pair of vertices in both directions.

A tournament T is ***transitive*** if, whenever there is an arc from u to v and from v to w, there is also an arc from u to w.

A tournament T is ***irreducible*** if there is no bipartition V_1, V_2 of the vertices such that all arcs between V_1 and V_2 go from V_1 to V_2.

Vertex u of a tournament ***dominates*** vertex v if there is an arc from u to v.

A ***transmitter*** in a digraph is a vertex that has an arc to every other vertex.

A ***king*** in a digraph is a vertex from which there is a path of length 1 or 2 to all other vertices.

A ***single-elimination competition*** is a contest from which a competitor is eliminated after the first loss.

Facts:

1. Every tournament has a Hamilton path (§8.4.4), in fact an odd number of them.

2. The following statements are equivalent for any n-tournament T:

- T is strong;
- T is irreducible;
- T has a Hamilton cycle (§8.4.4);
- T has cycles of all lengths $3, 4, \ldots, n$.;
- Every vertex of T is on cycles of all lengths $3, 4, \ldots, n$.

3. Almost all tournaments are strong, in the sense that, as $n \to \infty$, the fraction of labeled n-tournaments that are strong approaches 1.

4. The following are equivalent for a tournament:

- the tournament is transitive;
- the tournament contains no cycles;
- the tournament contains no 3-cycles;
- the tournament is a total (i.e. linear) order;
- the tournament has a unique Hamilton path.

5. Every tournament has a king.

6. The king of a tournament is unique if and only if it is a transmitter. Otherwise, there are at least three kings.

7. In a large tournament, almost every vertex is a king, for as $n \to \infty$, the fraction of n-tournaments in which every vertex is a king approaches 1.

8. *Score vector characterizations*: A nondecreasing sequence S of nonnegative integers $s_1, s_2, \ldots, s_n$ is the score vector of an n-tournament if and only if

$$\sum_{i=1}^{k} s_i \geq \binom{k}{2}, \quad \text{for } k = 1, 2, \ldots, n-1, \quad \text{and} \quad \sum_{i=1}^{n} s_i = \binom{n}{2},$$

or equivalently, if and only if

the sequence S' obtained by deleting any one s_i and reducing the largest remaining $n - s_i - 1$ terms by 1 is a score vector of an $(n-1)$-tournament.

9. The second characterization of Fact 8 leads to a recursive algorithm to construct a tournament having a specified score vector. See Example 4.

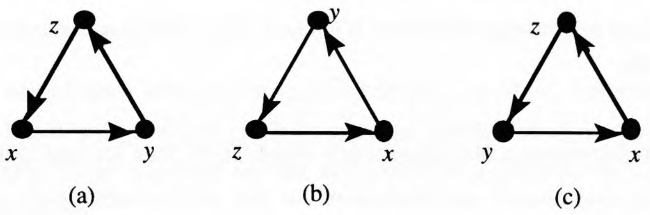
10. A nonnegative integer sequence $s_1 \leq s_2 \leq \ldots \leq s_n$ is the score vector of a *strong* n-tournament if and only if

$$\sum_{i=1}^{k} s_i > \binom{k}{2}, \quad \text{for } k = 1, 2, \ldots, n-1, \quad \text{and} \quad \sum_{i=1}^{n} s_i = \binom{n}{2}.$$

11. There are $2^{\binom{n}{2}}$ distinct *labeled* tournaments, because for each pair of vertices $\{u, v\}$, there are two choices which way to direct the edge. If c_n is the numbered of distinct *unlabeled* n-tournaments, then

$$c_n > \frac{2^{\binom{n}{2}}}{n!} \quad \text{and} \quad \lim_{n \to \infty} \frac{c_n}{2^{\binom{n}{2}}/n!} = 1.$$

The distinction between labeled and unlabeled tournaments is the same as between labeled and unlabeled graphs; see §8.9.1. The two tournaments in the following figure are isomorphic as unlabeled tournaments, but distinct as labeled tournaments.



12. *Ranking real tournaments*: When a tournament models a competition, there is an obvious desire to rank the teams, or at least to pick a clear winner. Many ranking methods have been proposed, and continue to be proposed.

13. When a tournament is acyclic, it corresponds to a unique total ordering (Fact 4), so the ranking is unequivocal. However, almost all tournaments are strong (Fact 3). Moreover, in a large tournament, almost every vertex is a king (Fact 7). These are reasons why it is considered difficult to give a satisfactory general method to rank tournaments.

14. *Scheduling tournaments*: To speed up the play of an n-tournament, games can be scheduled in parallel. If n is even, then at most $\frac{n}{2}$ of the $\frac{n(n-1)}{2}$ games may be played at once, so at least $n-1$ rounds are needed. However, if n is odd, then only $\frac{n-1}{2}$ games can be played at once, so at least n rounds are needed. In fact, this minimum number of rounds can be obtained, and several methods of scheduling tournaments, subject to various additional conditions, have been devised. See [Mo68].

Examples:

1. A round-robin sports tournament in which there are no ties is a tournament in the mathematical sense defined above. However, a single-elimination competition (e.g., most tennis tournaments) is not a tournament as defined above.

2. It has been observed that in every small flock of hens, *every* pair of hens establish a dominance relation — the weaker of the two allows the stronger to peck her. Thus, this pecking order is a tournament.

3. In a "paired comparison experiment", a subject is asked to state a preference in each pair chosen from n items. This amounts to a tournament, where there is an arc ij if item i is preferred to item j.

4. Is there a tournament on vertices (a, b, c, d, e) with respective scores $(1, 2, 2, 2, 3)$? Deleting e according to the second part of Fact 8 leaves vertices (a, b, c, d) with scores $(1, 2, 2, 1)$. Next deleting d leaves (a, b, c) with scores $(1, 1, 1)$. The obvious tournament with such a score vector is a 3-cycle. Next reinsert vertex d, making it dominate vertex a only. Then reinsert vertex e, making it dominate a, b, c. This 5-tournament has the specified score vector $(1, 2, 2, 2, 3)$.

5. *Ranking real tournaments*: Ranking teams by their order along a Hamilton path (see Fact 1) is rarely satisfactory, because that order is unique only for transitive tournaments (Fact 4); in most cases, there are a great many Hamilton paths. Ranking by score vector usually creates ties, and a team with few wins may deserve a better rank if those teams it beats have many wins. So one may consider the *second-order score vector*, where each team's score is the sum of the out-degrees of the teams it beats. This can be continued to nth-order score vectors. There is a limit ranking obtained this way (often quite satisfactory), related to the eigenvalues of the digraph. See [Mo68] for more detail and references.

8.4 DISTANCE, CONNECTIVITY, TRAVERSABILITY

Movement from one node to another in the network corresponds to the graph-theoretic notion of a walk. Graphs often serve as models for transportation and communication network problems. The capability for any two nodes in a network to communicate corresponds to connectedness. The connectivity of a graph is a measure of resistance to a communications cutoff.

8.4.1 WALKS, DISTANCE, AND CYCLE RANK

Definitions:

A ***walk*** in a graph is an alternating sequence $v_0, e_1, v_1, \ldots, e_r, v_r$ of vertices and edges in which each edge e_i joins vertices v_{i-1} and v_i. Such a walk is also called a v_0, v_r-*walk*.

The ***length*** of a walk is the number of occurrences of edges in it. An edge that occurs more than once is counted each time it occurs.

A ***trail*** is a walk in which all of the edges are different.

A ***path*** is a trail in which all the vertices are different, except that the initial and final vertices may be the same. A path from v_0 to v_r is called a v_0, v_r-*path*.

A walk, trail, or path is ***open*** if its final vertex is different from its initial vertex.