Digital Logic Design: a rigorous approach ©

Chapter 4: Directed Graphs

Guy Even Moti Medina

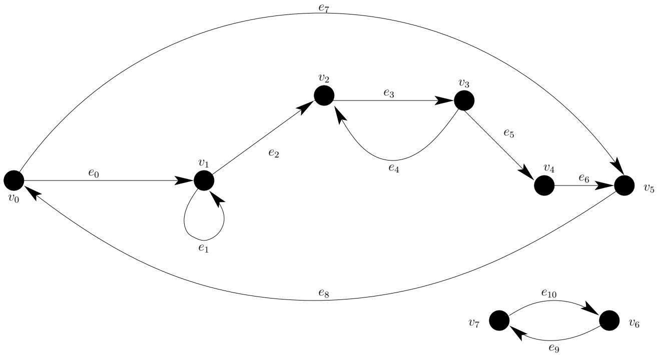
School of Electrical Engineering Tel-Aviv Univ.

March 19, 2020

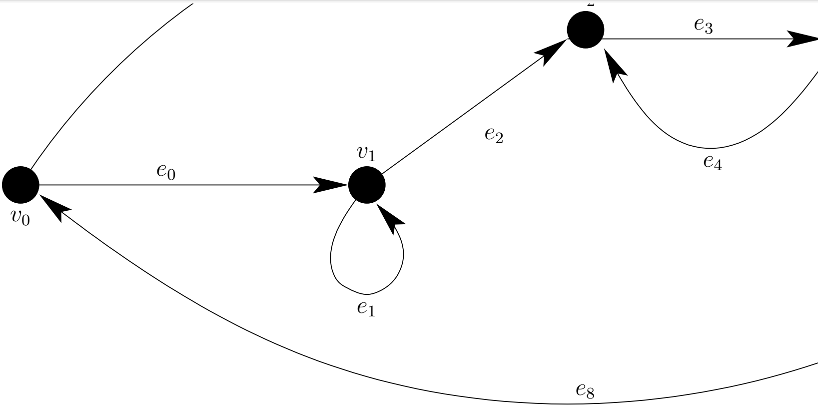
Book Homepage:
http://www.eng.tau.ac.il/~guy/Even-Medina

Definition (directed graph)

Let V denote a finite set and $E \subseteq V \times V$. The pair (V, E) is called a directed graph and is denoted by $G = (V, E)$. An element $v \in V$ is called a vertex or a node. An element $(u, v) \in E$ is called an arc or a directed edge.



Directed Graphs



Definition (path)

A path or a walk of length ℓ in a directed graph $G = (V, E)$ is a sequence $(v_0, e_0, v_1, e_1, \ldots, v_{\ell-1}, e_{\ell-1}, v_\ell)$ such that:

1. $v_i \in V$, for every $0 \leq i \leq \ell$,
2. $e_i \in E$, for every $0 \leq i < \ell$, and
3. $e_i = (v_i, v_{i+1})$, for every $0 \leq i < \ell$.

We denote an arc $e = (u, v)$ by $u \xrightarrow{e} v$ or simply $u \longrightarrow v$. A path of length ℓ is often denoted by

$$v_0 \xrightarrow{e_0} v_1 \xrightarrow{e_1} v_2 \cdots v_{\ell-1} \xrightarrow{e_{\ell-1}} v_\ell.$$

path terminology

1. A path is closed if the first and last vertices are equal.
2. A path is open if the first and last vertices are distinct.
3. An open path is simple if every vertex in the path appears only once in the path.
4. A closed path is simple if every interior vertex appears only once in the path. (A vertex is an interior vertex if it is not the first or last vertex.)
5. A self-loop is a closed path of length 1, e.g., $v \xrightarrow{e} v$.

To simplify terminology, we refer to a closed path as a cycle, and to a simple closed path as a simple cycle.

directed acyclic graph (DAG)

Definition (DAG)

A directed acyclic graph (DAG) is directed graph that does not contain any cycles.

Question

What do you think about the suggestion to turn all the streets into one-way streets so that the resulting directed graph is acyclic?

directed graph terminology

We say that an arc $u \xrightarrow{e} v$ enters v and emanates (or exits) from u.

Definition (indegree/outdegree)

The in-degree and out-degree of a vertex v are denoted by $deg_{in}(v)$ and $deg_{out}(v)$, respectively, and defined by:

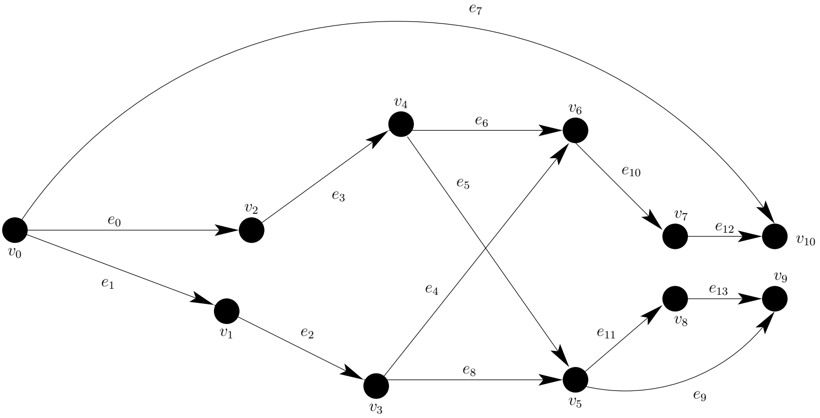
$$deg_{in}(v) \triangleq |\{e \in E \mid e \text{ enters } v\}|,$$
$$deg_{out}(v) \triangleq |\{e \in E \mid e \text{ emanates from } v\}|.$$

Definition (source/sink)

A vertex is a source if $deg_{in}(v) = 0$. A vertex is a sink if $deg_{out}(v) = 0$.

In circuits, sources correspond to inputs and sinks correspond to outputs.

DAG example



Is this a DAG? How many paths are there from v_0 to v_6? What is the in-degree of v_5? What is the out-degree of v_4? Which vertices are sources? sinks?

DAG properties

Lemma

Every non-simple path contains a (simple) cycle.

Lemma

Let G denote a DAG over n vertices. The length of every path in G is at most $n-1$.

Lemma

Every DAG has at least one sink.

Corollary

Every DAG has at least one source.

Proof?

ordering is a permutation

Question

Suppose we want to list the vertices. How can we specify the order of the vertices in the list?

Answer

A bijection $\pi : V \to \{0, \dots, n-1\}$ defines an order. Let v_i denote the vertex such that $\pi(v) = i$. Then π specifies the ordering $(v_0, \dots, v_{n-1})$.

- Note that each vertex appears exactly once in this n-tuple. Such an n-tuple is called a permutation of the vertices.
- We are interested in permutations of the vertices that satisfy a special condition...

- Order the vertices of a DAG so that if u precedes v, then (v, u) is not an arc.
- This means that no arc will "point to the left".
- Our main application of topological ordering is for simulating digital circuits.

Let $G = (V, E)$ denote a DAG with $|V| = n$.

Definition (topological ordering)

A bijection $\pi : V \to \{0, \ldots, n-1\}$ is a topological ordering of the vertices of a directed graph (V, E) if

$$(u, v) \in E \quad \Rightarrow \quad \pi(u) < \pi(v).$$

Note that by contraposition, $\pi(v) \leq \pi(u)$ implies that $(u, v) \notin E$.

Why order a DAG in topological ordering?

- consider a DAG where vertices denote assembly steps and arcs denote order.
- example: how to assemble a couch? An arc (u, v) signifies that the action represented by node v cannot begin before the action represented by node u is completed: "put the skeleton together" $\rightarrow$ "put pillows on the couch".
- Assembly must use a "legal" schedule of assembly steps: cannot "put the pillows" before "skeleton is constructed".
- Such a schedule is a topological ordering of the assembly instructions.
- Suppose each assembly step can be performed only by a single person. Does it help to have more than one worker? Will they build the couch faster?

algorithm for topological ordering

Notation:

$$E_v \triangleq \{e \mid e \text{ enters } v \text{ or emanates from } v\}.$$

Algorithm 1 TS(V, E) - An algorithm for sorting the vertices of a DAG $G = (V, E)$ in topological ordering.

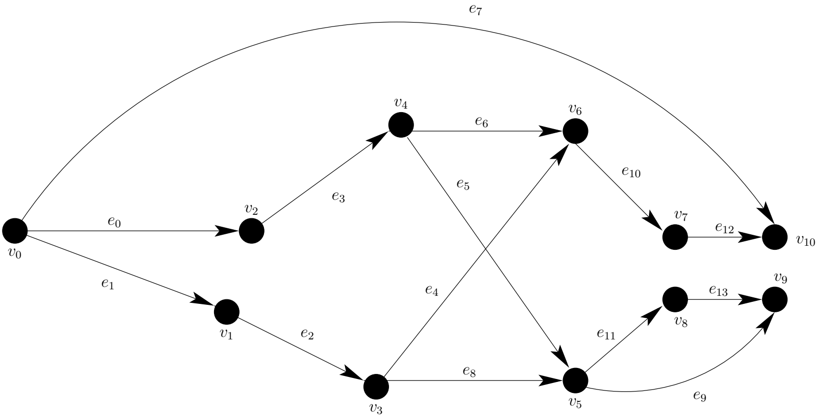
---

1. Base Case: If $|V| = 1$, then let $v \in V$ and return $(\pi(v) = 0)$.
2. Reduction Rule:
   1. Let $v \in V$ denote a sink.
   2. return (TS$(V \setminus \{v\}, E \setminus E_v)$ extended by $(\pi(v) = |V| - 1)$)

Theorem

Algorithm $TS(V, E)$ computes a topological ordering of a DAG $G = (V, E)$.

example: longest paths in DAGs



We denote the length of a path Γ by $|\Gamma|$.

Definition

A path Γ that ends in vertex v is a longest path ending in v if $|\Gamma'| \leq |\Gamma|$ for every path Γ' that ends in v.

Note: there may be multiple longest paths ending in v (hence "a longest path" rather than "the longest path").

Definition

A path Γ is a longest path in G if $|\Gamma'| \leq |\Gamma|$, for every path Γ' in G.

Question

Does a longest path always exist in a directed graph?

longest paths in DAGs

If a directed graph has a cycle, then there does not exist a longest path. Indeed, one could walk around the cycle forever. However, longest paths do exist in DAGs.

Lemma

If $G = (V, E)$ is a DAG, then there exists a longest path that ends in v, for every v. In addition, there exists a longest path in G.

Proof: The length of every path in a DAG is at most $|V| - 1$. [Or, every path is simple, hence, the number of paths is finite.]

computing longest paths: specification

Goal: compute, for every v in a DAG, a longest path that ends in v. We begin with the simpler task of computing the length of a longest path.

Specification

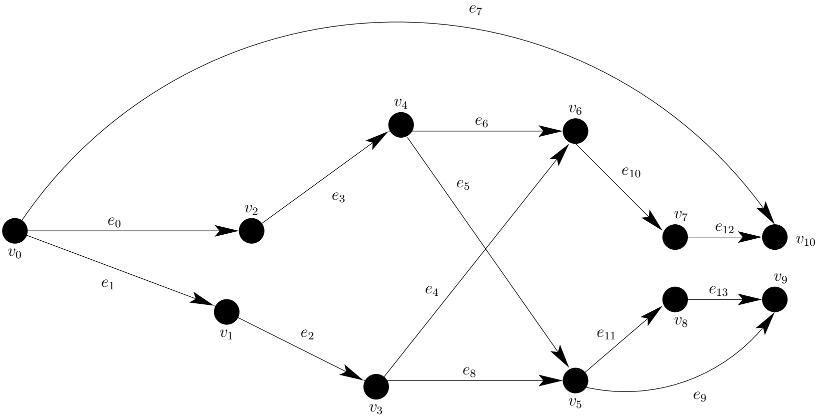
Algorithm longest-path is specified as follows.

input: A DAG $G = (V, E)$.

output: A delay function $d : V \to \mathbb{N}$.

functionality: For every vertex $v \in V$: $d(v)$ equals the length of a longest path that ends in v.

Application: Model circuits by DAGs. Assume all gates complete their computation in one unit of time. The delay of the output of a gate v equals $d(v)$


e_7
v_4
e_6
v_6
e_{10}
e_5
e_3
v_2
v_7
e_{12}
v_{10}
e_0
v_0
e_1
v_9
e_4
e_{13}
v_8
v_1
e_2
e_{11}
e_8
v_3
v_5
e_9

Algorithm 2 longest-path-lengths(V, E) - An algorithm for computing the lengths of longest paths in a DAG. Returns a delay function $d(v)$.

1. topological sort: $(v_0, \ldots, v_{n-1}) \leftarrow TS(V, E)$.
2. For $j = 0$ to $(n-1)$ do
    1. If v_j is a source then $d(v_j) \leftarrow 0$.
    2. Else
    $$d(v_j) = 1 + \max \left\{ d(v_i) \mid i < j \text{ and } (v_i, v_j) \in E \right\}.$$

One could design a "single pass" algorithm; the two pass algorithm is easier to prove.

Let

$$d(v) \triangleq \text{output of algorithm}$$
$$\delta(v) \triangleq \text{the length of a longest path that ends in } v$$

Theorem

Algorithm correct: $\forall j : d(v_j) = \delta(v_j)$.

Proof: Complete induction on j. Basis for sources easy.

We prove now that

1. $\delta(v_{j+1}) \geq d(v_{j+1})$, namely, there exists a path Γ that ends in v_{j+1} such that $|\Gamma| \geq d(v_{j+1})$.
2. $\delta(v_{j+1}) \leq d(v_{j+1})$, namely, for every path Γ that ends in v_{j+1} we have $|\Gamma| \leq d(v_{j+1})$.

In the following definition we consider a directed acyclic graph $G = (V, E)$ with a single sink called the root.

Definition

A DAG $G = (V, E)$ is a rooted tree if it satisfies the following conditions:

1. There is a single sink in G.
2. For every vertex in V that is not the sink, the out-degree equals one.

The single sink in rooted tree G is called the root, and we denote the root of G by $r(G)$.

paths to the root

Definition

A DAG $G = (V, E)$ is a rooted tree if it satisfies the following conditions:

1. There is a single sink in G.
2. For every vertex in V that is not a sink, the out-degree equals one.

Theorem

In a rooted tree there is a unique path from every vertex to the root.

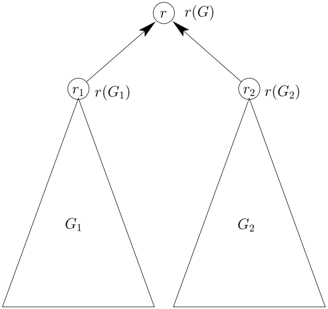


Figure: A decomposition of a rooted tree G into two rooted trees G_1 and G_2.

Terminology

- each the rooted tree $G_i = (V_i, E_i)$ is called a tree hanging from $r(G)$.
- Leaf : a source node.
- interior vertex : a vertex that is not a leaf.
- parent : if $u \longrightarrow v$, then v is the parent of u.
- Typically maximum in-degree$= 2$.

Applications

- The rooted trees hanging from $r(G)$ are ordered. Important in parse trees.
- Arcs are oriented from the leaves towards the root. Useful for modeling circuits:
  - leaves = inputs
  - root = output of the circuit.