Digital Logic Design: a rigorous approach ©

Chapter 4: Directed Graphs

Guy Even Moti Medina

School of Electrical Engineering Tel-Aviv Univ.

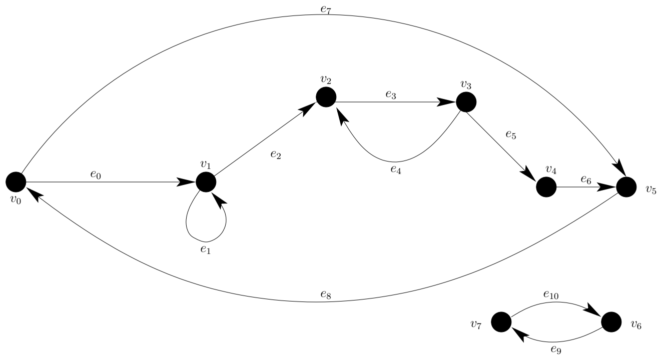
April 10, 2012

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

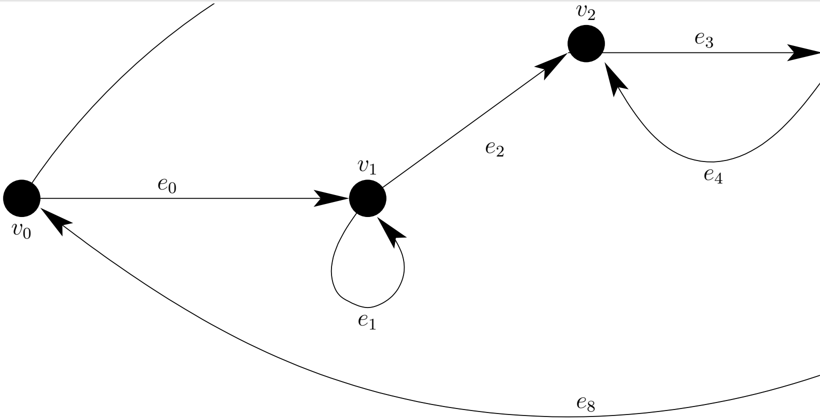
Directed Graphs

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



paths

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.

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:

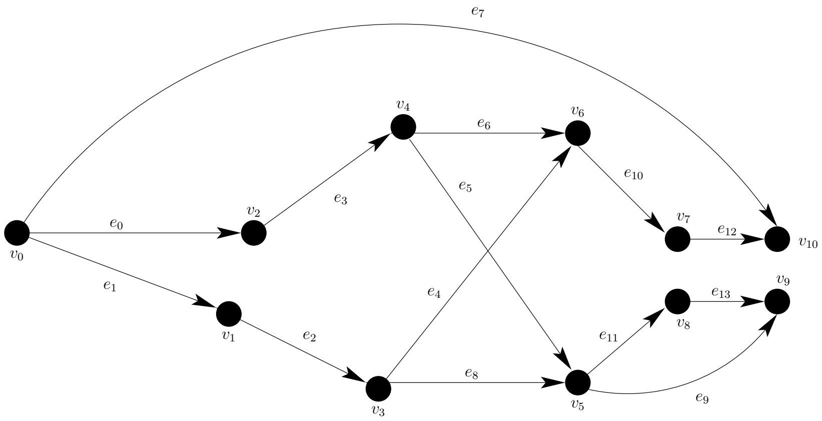
$$deg_{in}(v) \triangleq |\{e \in E \mid e \text{ enters } v\}|,$$
$$deg_{out}(v) \triangleq |\{e \in E \mid e \text{ emanates from } v\}|.$$

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 DAG has at least one sink.

Corollary

Every DAG has at least one source.

topological ordering

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

Definition (topological ordering)

Let $G = (V, E)$ denote a DAG with $|V| = n$. A bijection $\pi : V \to \{0, \ldots, n-1\}$ is a topological ordering if $(u, v) \in E$ implies that $\pi(u) < \pi(v)$.

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

ordering is a permutation

A bijection $\pi : V \rightarrow \{0, \ldots, n-1\}$ can be represented by an n-tuple $(v_0, \ldots, v_{n-1})$ in which each vertex appears exactly once. Such an n-tuple is called a permutation of the vertices. Hence, topological ordering means that we are looking for a permutation that satisfies a special condition.

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.

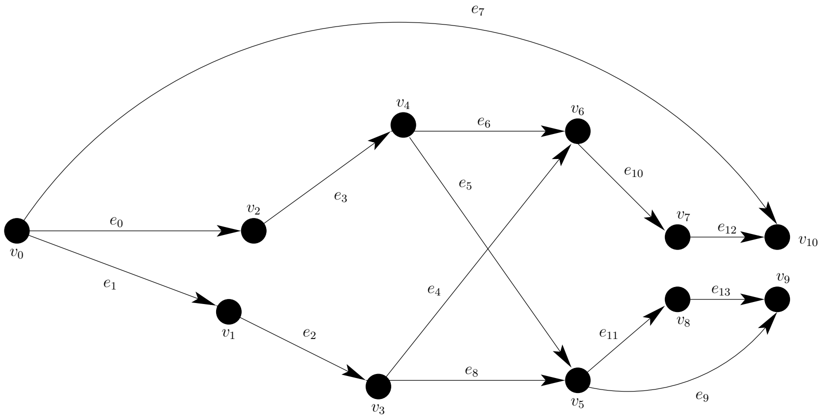
```
1. Base Case: If $|V| = 1$, then let $v \in V$ and return $(\pi(v) = 0)$.
2. Reduction Rule:
    1. Let $v \in V$ denote a sink.
    2. return (TS$(V \setminus \{v\}, E \setminus E_v)$ extended by $(\pi(v) = |V| - 1)$)
```

algorithm correctness

Theorem

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

example: longest paths in DAGs



longest paths

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.

Definition

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

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 idea: (1) The length of every path in a DAG is at most $|V| - 1$. (2) 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.

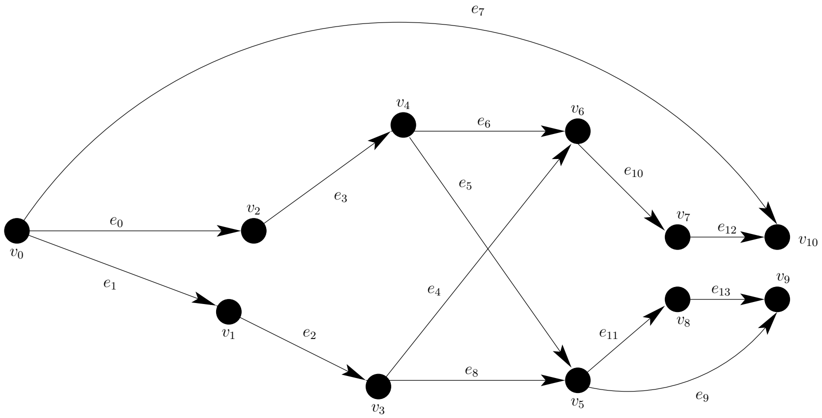
input: A DAG $G = (V, E)$.

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

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

Note: $d(source) = 0$.
"Delay Function": application for bounding the delay of a combinational circuit. Model circuits by DAGs, and the delay of the output of a gate equals $d(v)$ (if all gates have unit delays).

example: delay function



algorithm: longest path lengths

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.

algorithm correctness

Theorem

Algorithm longest-path-lengths(V, E) *satisfies the specification.*

Need to prove that $d(v_i)$ equals the length of a longest path ending in v_i.

algorithm correctness - cont.

Consider a node v_{j+1} that is not a source and a longest path Γ that ends in v_{j+1}. Let $\ell = |\Gamma|$. Clearly, $\ell \geq 1$. Denote the vertices and edges in Γ by

$$u_0 \xrightarrow{e_0} u_1 \xrightarrow{e_1} u_2 \cdots u_{\ell-1} \xrightarrow{e_{\ell-1}} u_\ell = v_{j+1}.$$

observation 1

If $v_i \xrightarrow{e} v_{j+1}$ is an arc in E, then $i \leq j$ and $d(v_i) \leq \ell - 1$.

observation 2

The path $\Gamma \setminus \{e_{\ell-1}, u_\ell\}$ is a longest path that ends in $u_{\ell-1}$. In particular, $d(u_{\ell-1}) = \ell - 1$.

rooted trees

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 a 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

Theorem

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

decomposition of rooted trees

The following claim states that every rooted tree G can be decomposed into rooted trees that are connected to $r(G)$.

claim

Let $G = (V, E)$ denote a rooted tree. Let $\{r_i \xrightarrow{e_i} r\}_{i=1}^{k}$ denote the set of arcs that enter the root $r = r(G)$. Define the sets V_i and E_i by

$$V_i \triangleq \{v \in V : \text{there exists a path from } v \text{ to } r_i \text{ in } G\}.$$
$$E_i \triangleq \{e \in E : \text{the arc } e \text{ emanates from a vertex in } V_i \setminus \{r_i\}\}.$$

Then,

1. The sets $V_1, \ldots V_k$ are pairwise disjoint and $V = V_1 \cup \cdots \cup V_k \cup \{r\}$.
2. The graph $G_i \triangleq (V_i, E_i)$ is a rooted tree with $r(G_i) = r_i$, for every $1 \leq i \leq k$.

decomposition: example

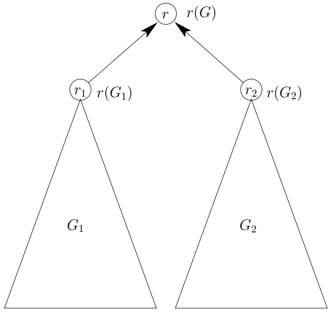


Figure: A decomposition of a rooted tree G into two rooted trees G_1 and G_2.

Composing rooted trees

claim

If $G_i = (V_i, E_i)$ are disjoint rooted trees, for $1 \le i \le k$, then the directed graph $G = (V, E)$ defined below is a rooted tree.

$$V \triangleq V_1 \cup \cdots \cup V_k \cup \{r\}, \text{ where } \forall i : \; r \notin V_i. \tag{1}$$

$$E \triangleq E_1 \cup \cdots \cup E_k \cup \{r(G_i) \longrightarrow r\}_{i=1}^{k}. \tag{2}$$

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 and root $=$ output of the circuit.