Building Crawlers Using Built-in Services in Scrapy

Janani Ravi
CO-FOUNDER, LOONYCORN
www.loonycorn.com

Overview

Scrapy has multiple features which are useful when crawling websites at scale

Logging events to console and to file

Using the telnet utility to debug crawlers

Working with broad crawlers to concurrently crawl multiple sites

Auto throttling crawls so as to not fall afoul of website policies

Demo

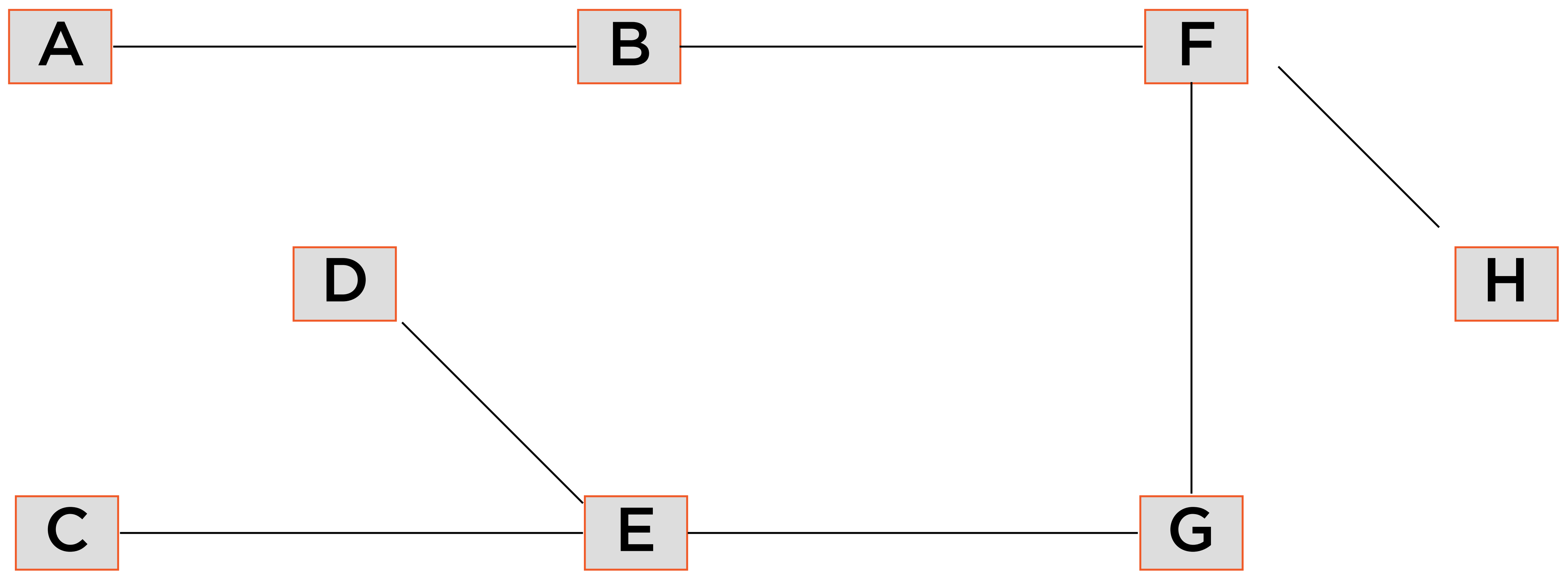
Logging to console and file

Demo

Sending email notifications based on crawl results

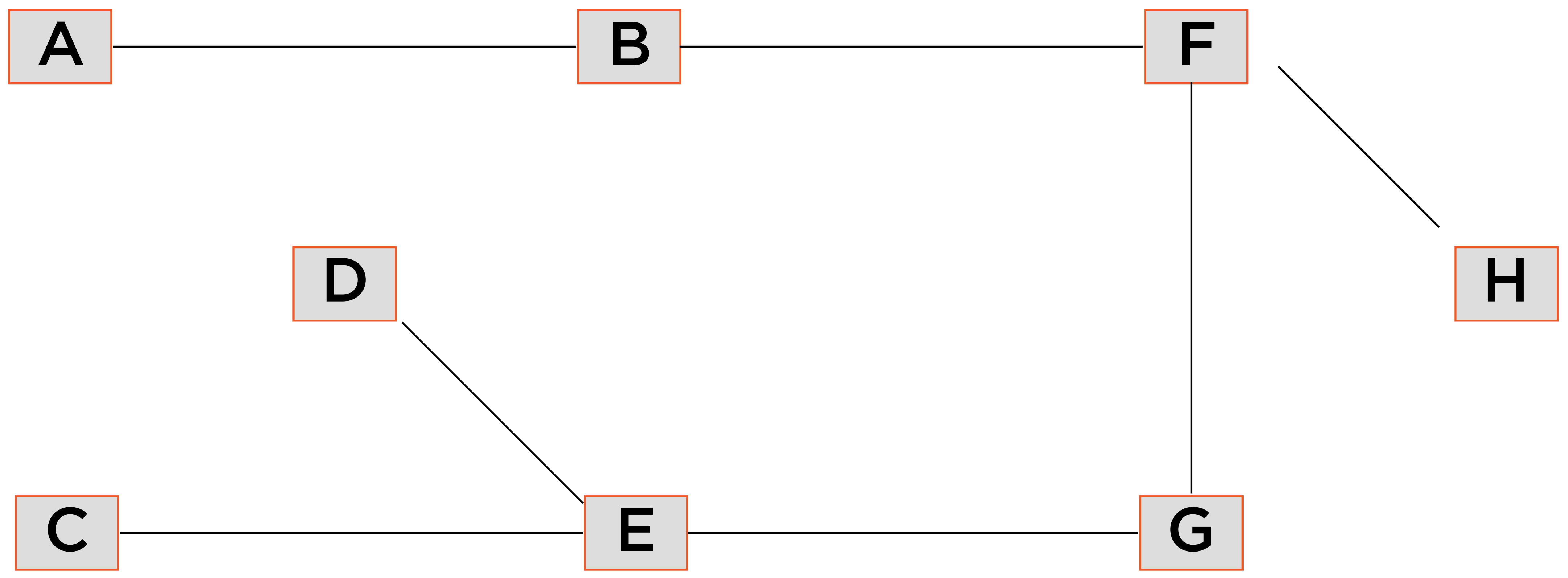
Broad Crawls

Connected Graph with No Cycle



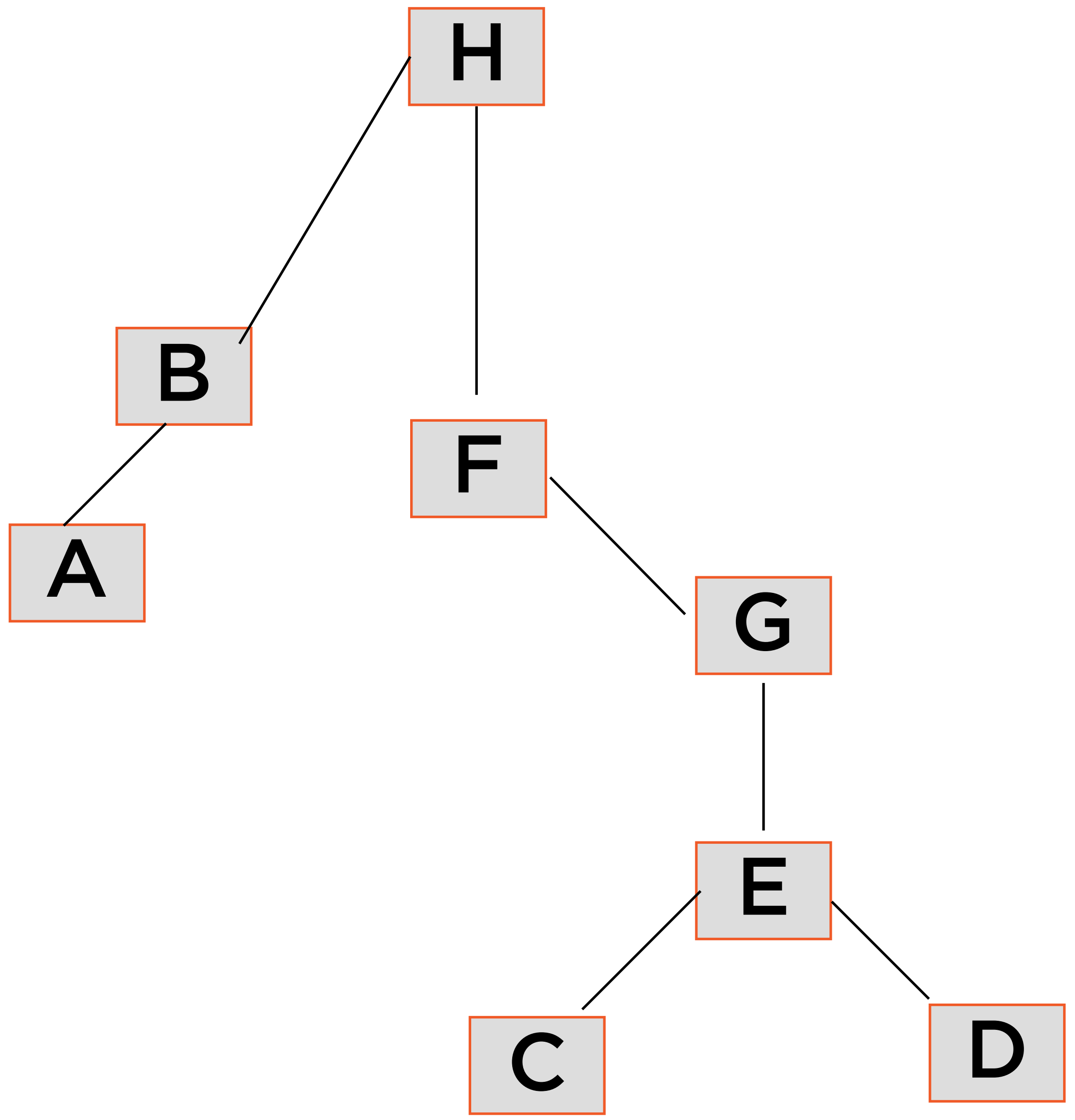
Such a graph is called a tree

Connected Graph with No Cycle



Trees are great for depicting hierarchical relationships

Tree



Trees are great for depicting **hierarchical relationships**

Two Ways of Traversing Graphs

Breadth-first

All nodes at same distance from origin visited together

Depth-first

All nodes in certain direction from origin visited together

Crawling web pages is quite similar to traversing a tree where the initial web page is the root node

Two Ways of Traversing Graphs

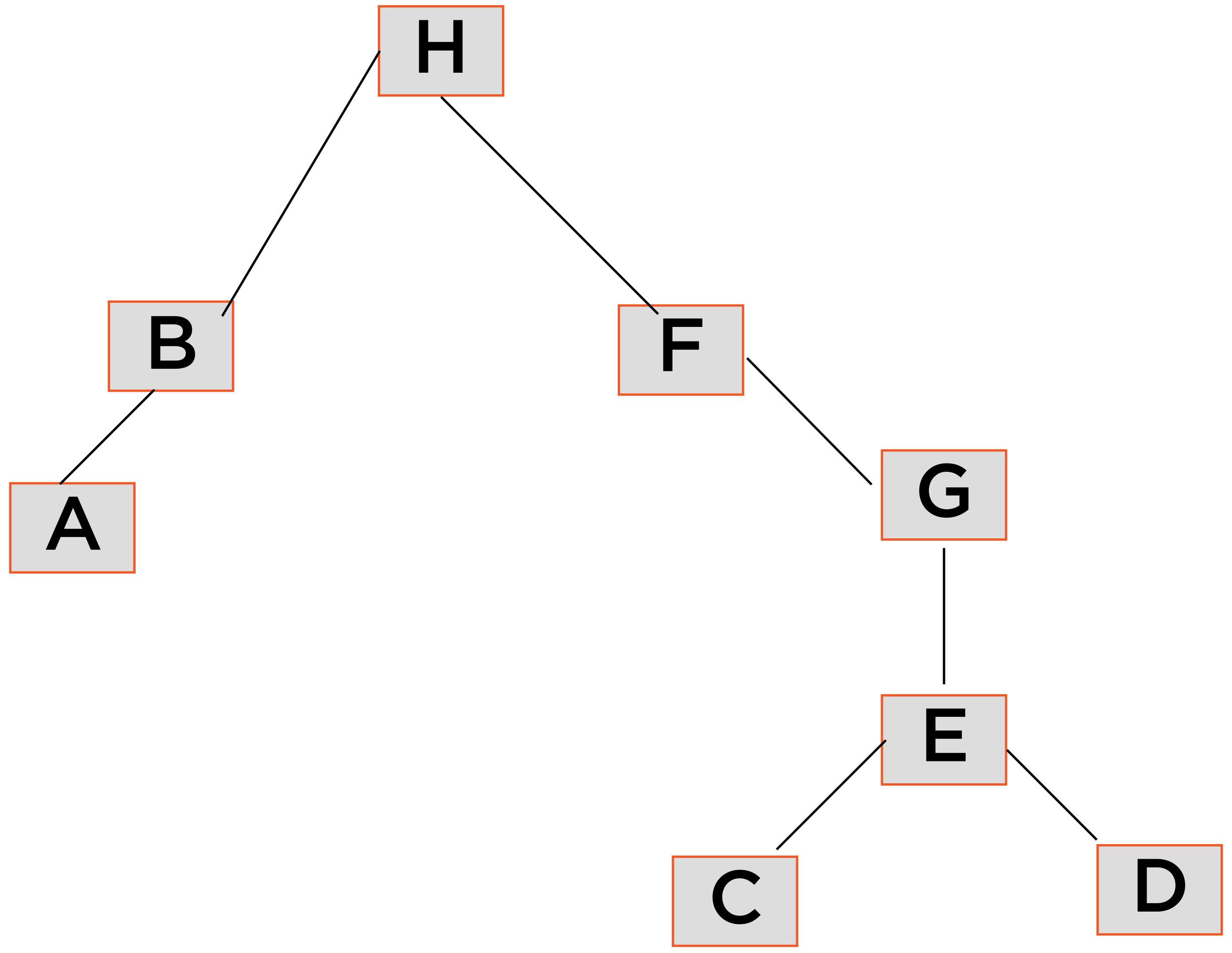
Breadth-first

All nodes at same distance from origin visited together

Depth-first

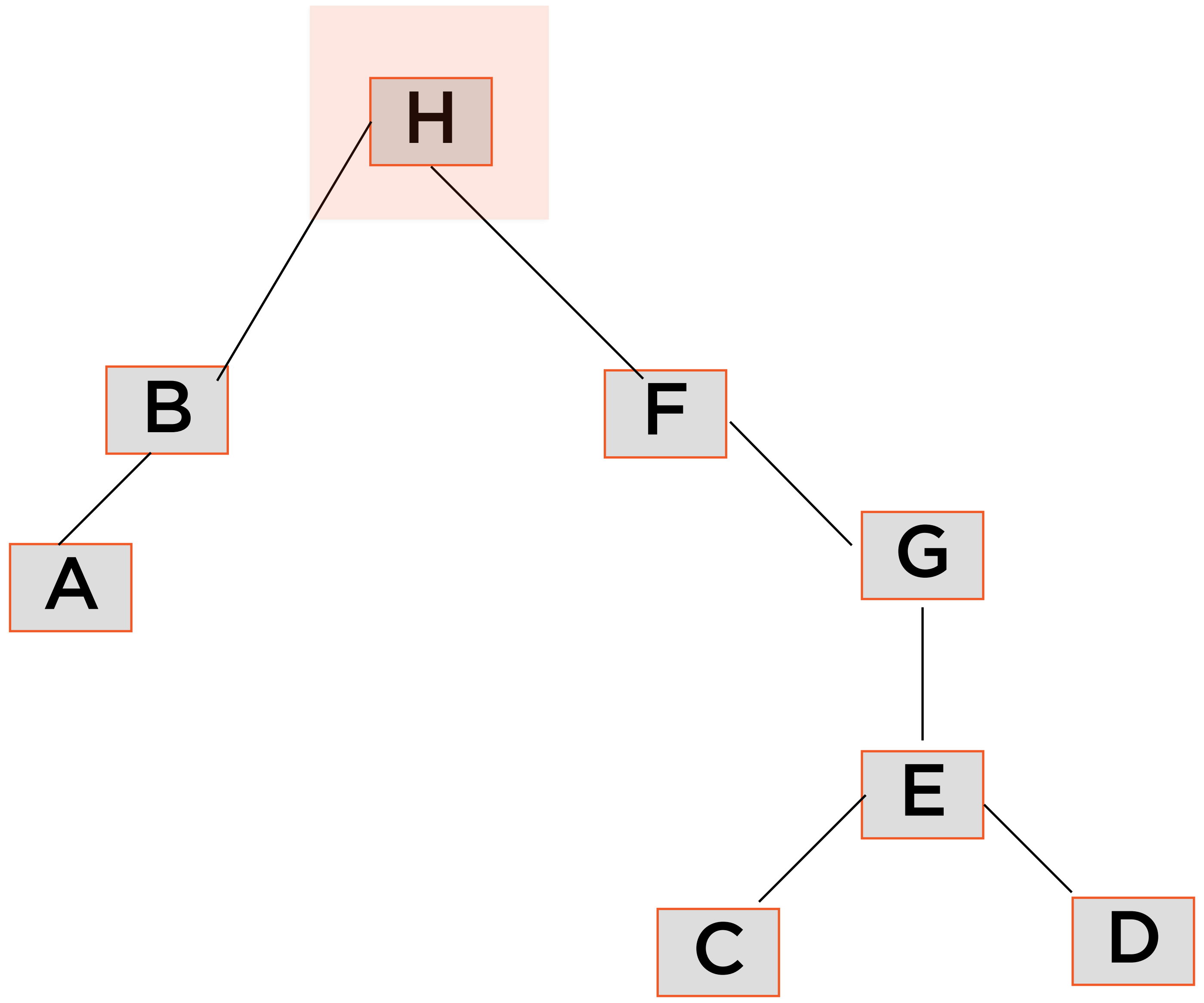
All nodes in certain direction from origin visited together

“Breadth-first” Tree Traversal



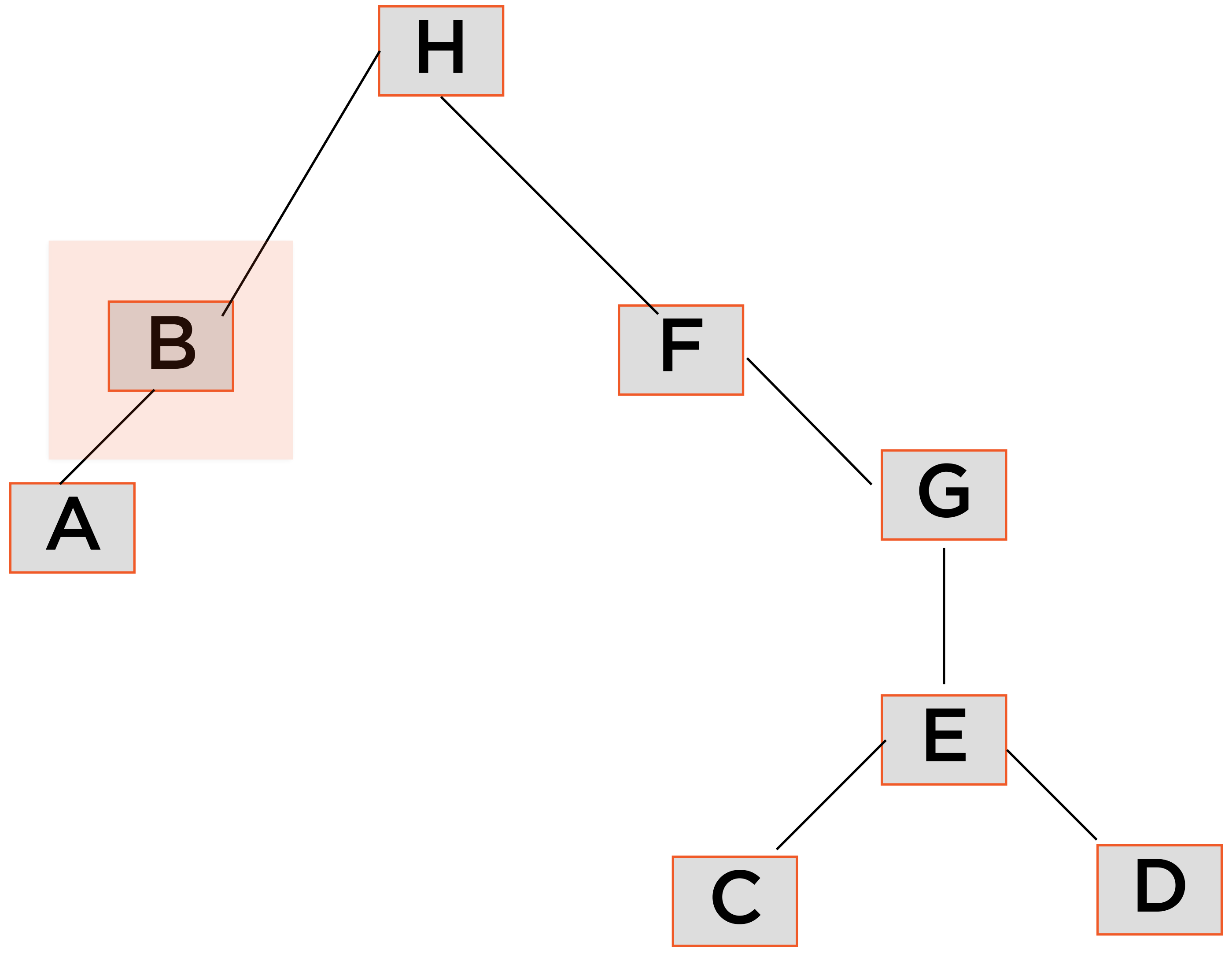
Nodes are visited level-by-level

"Breadth-first" Tree Traversal



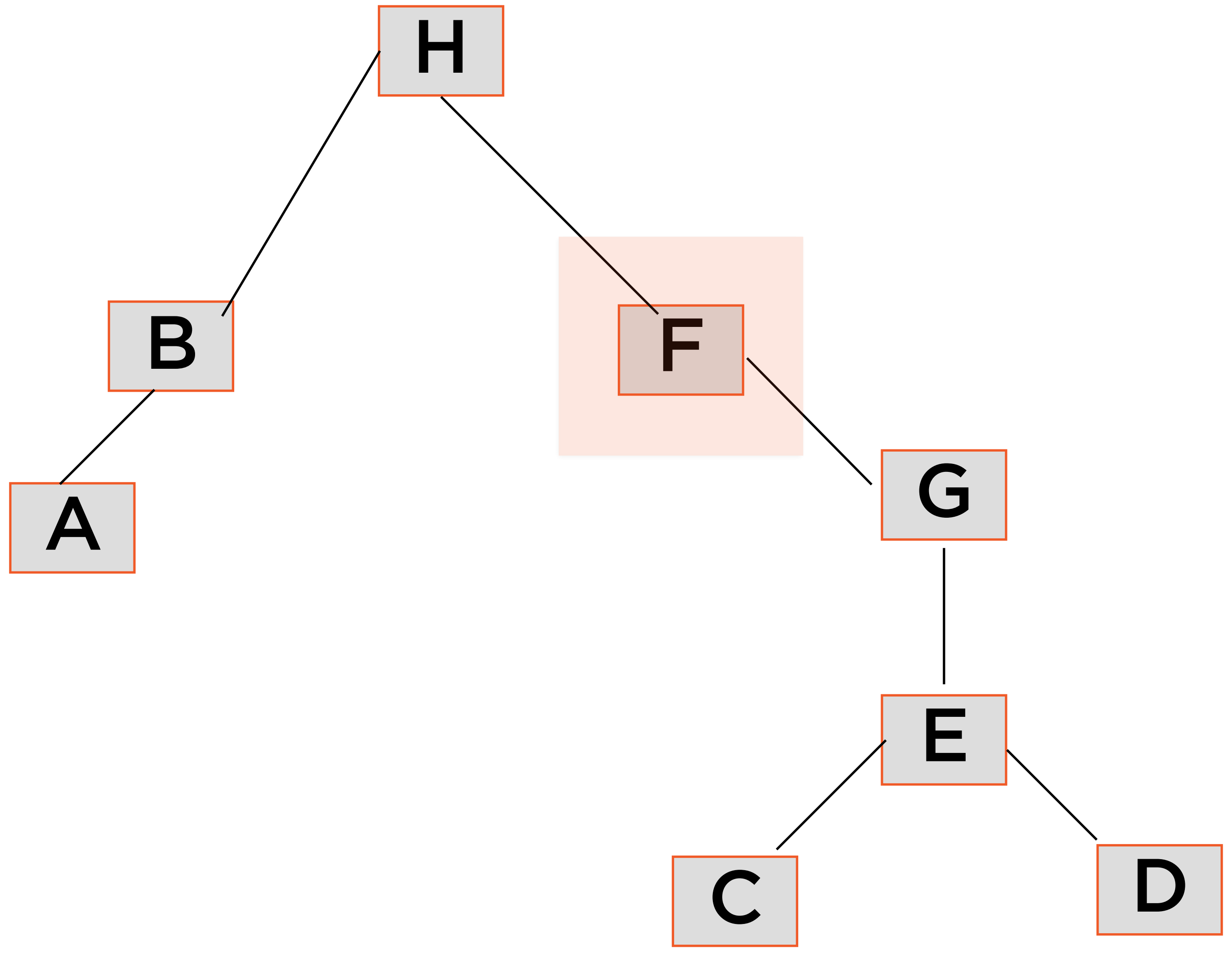
Visited H

“Breadth-first” Tree Traversal



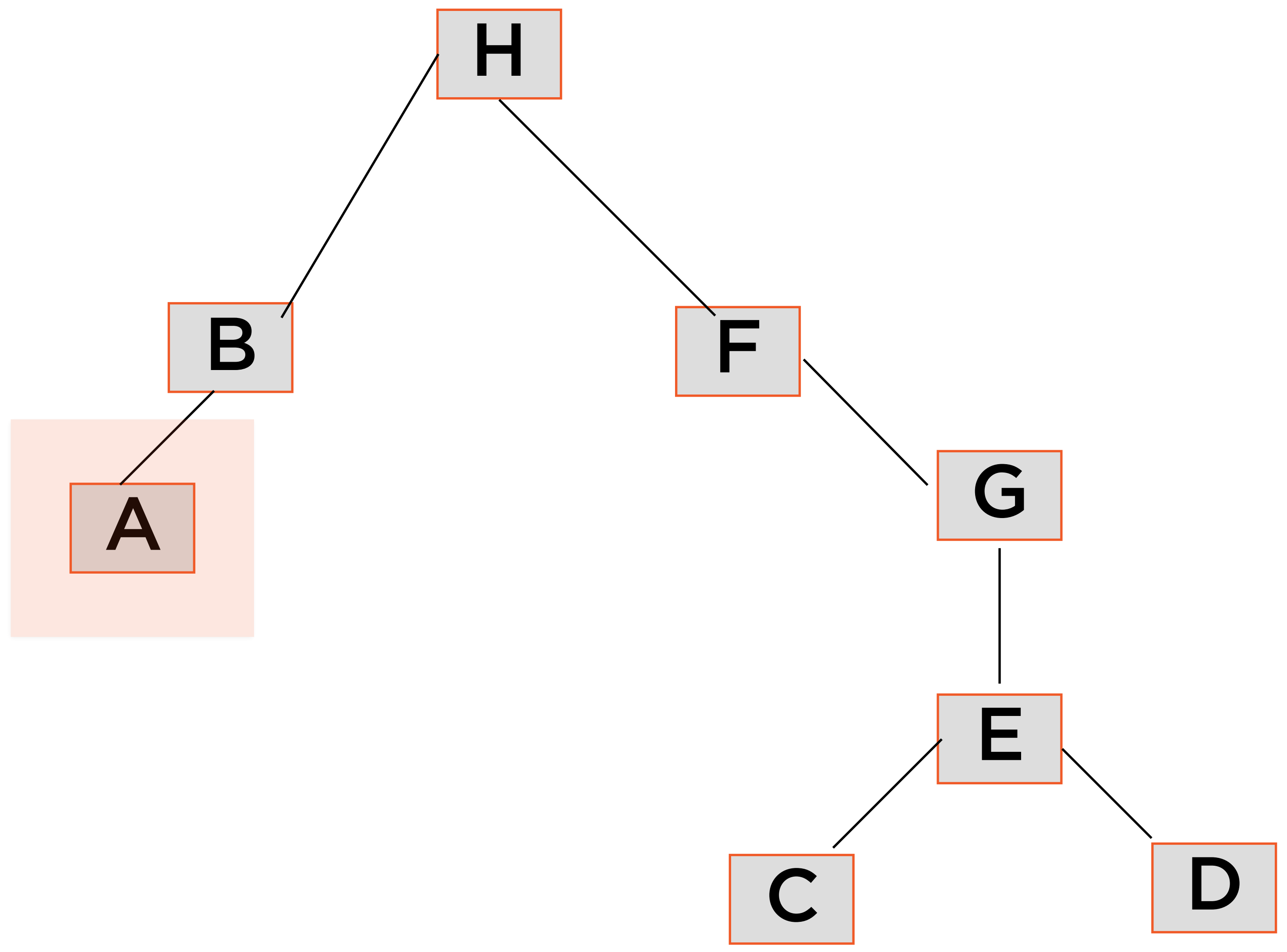
Visited H - B

"Breadth-first" Tree Traversal



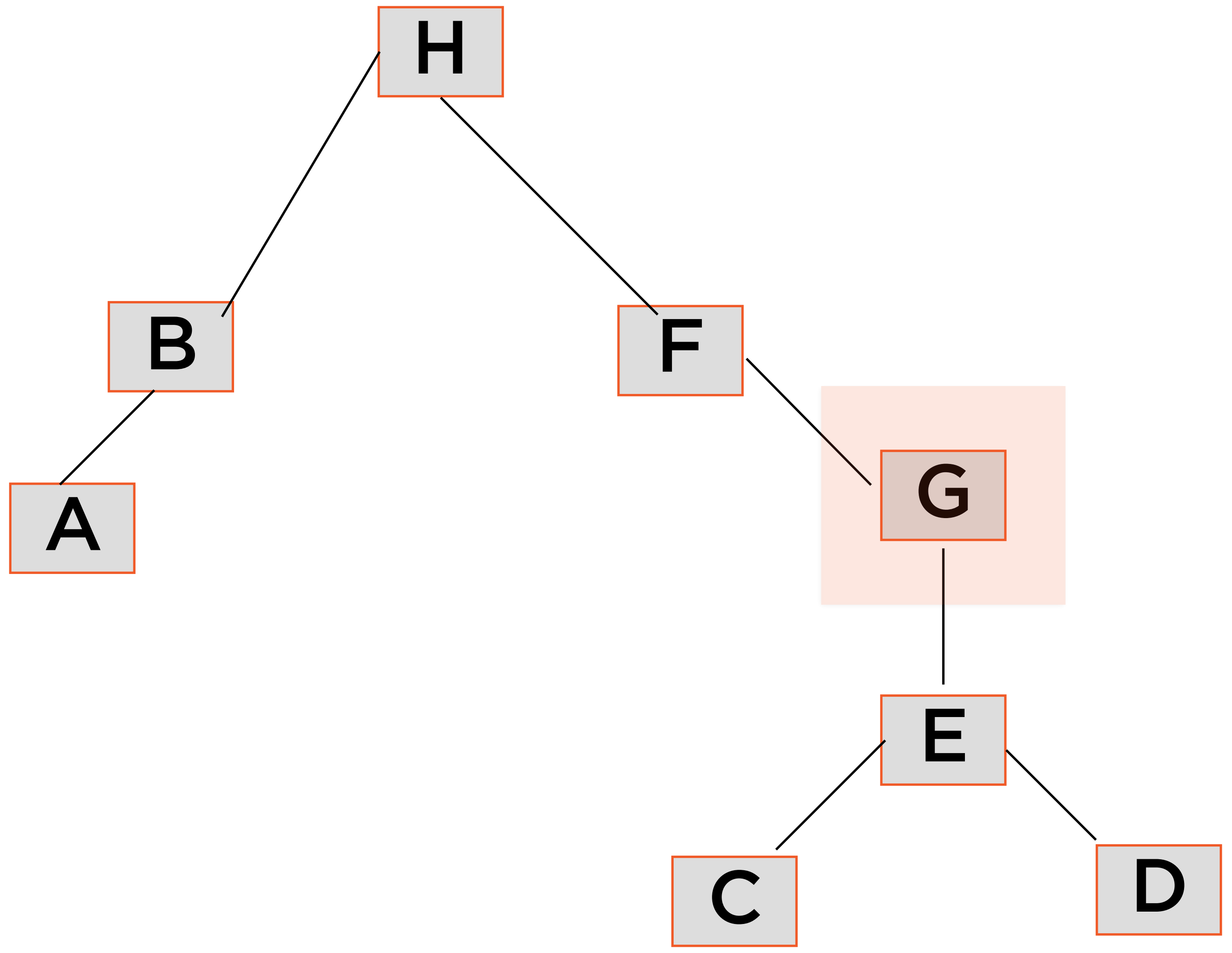
Visited H - B - F

“Breadth-first” Tree Traversal



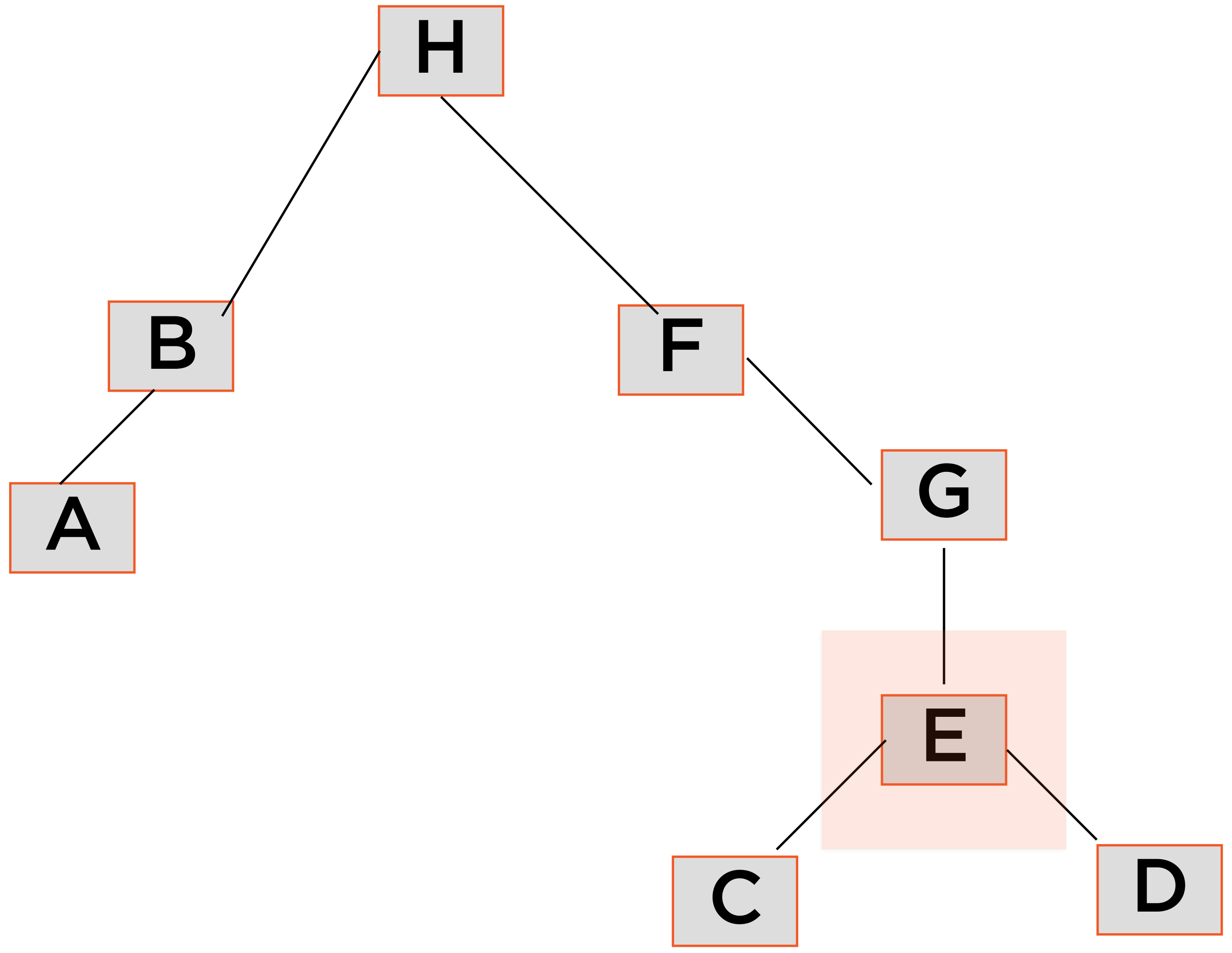
Visited H - B - F - A

“Breadth-first” Tree Traversal



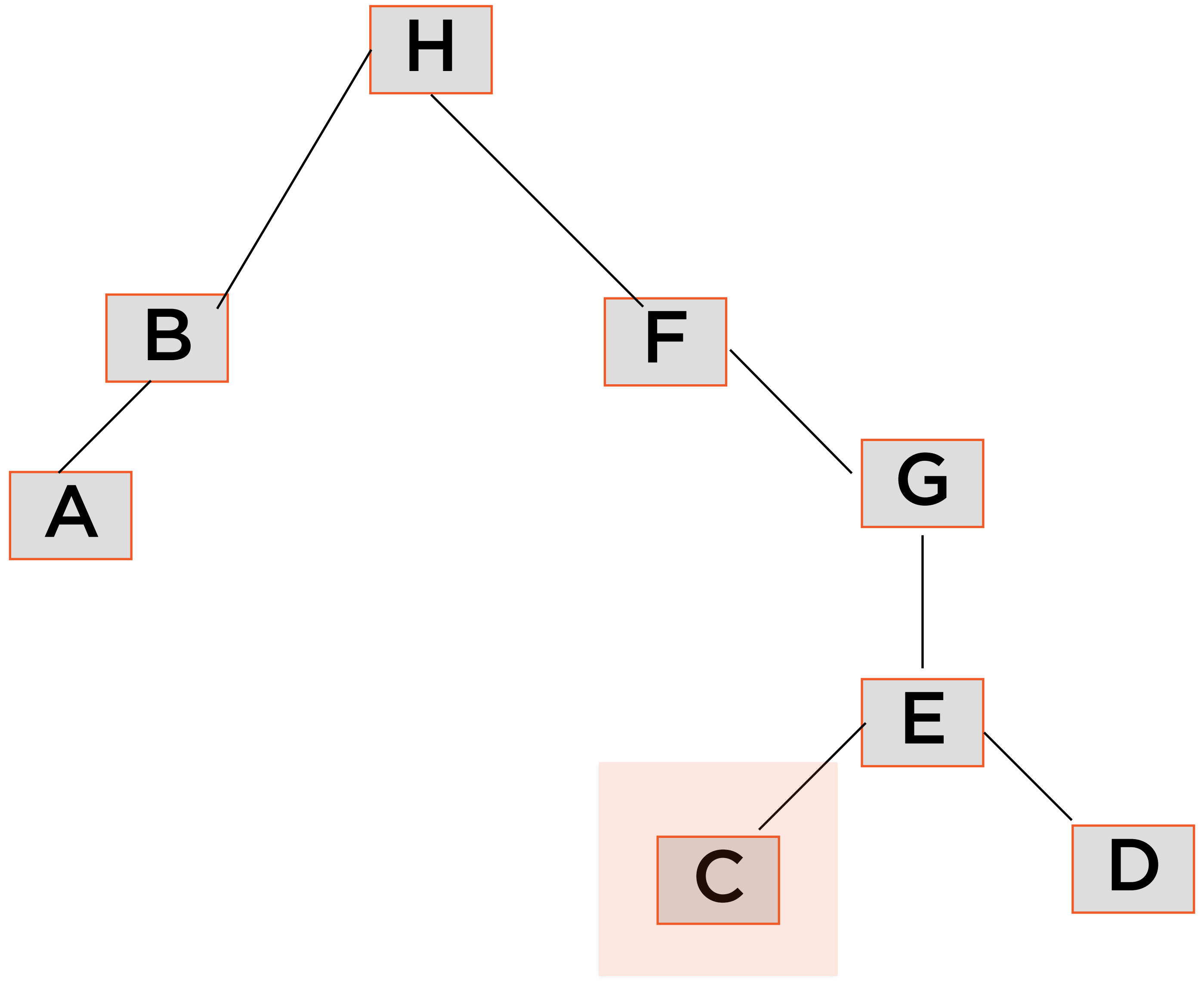
Visited H - B - F - A - G

“Breadth-first” Tree Traversal



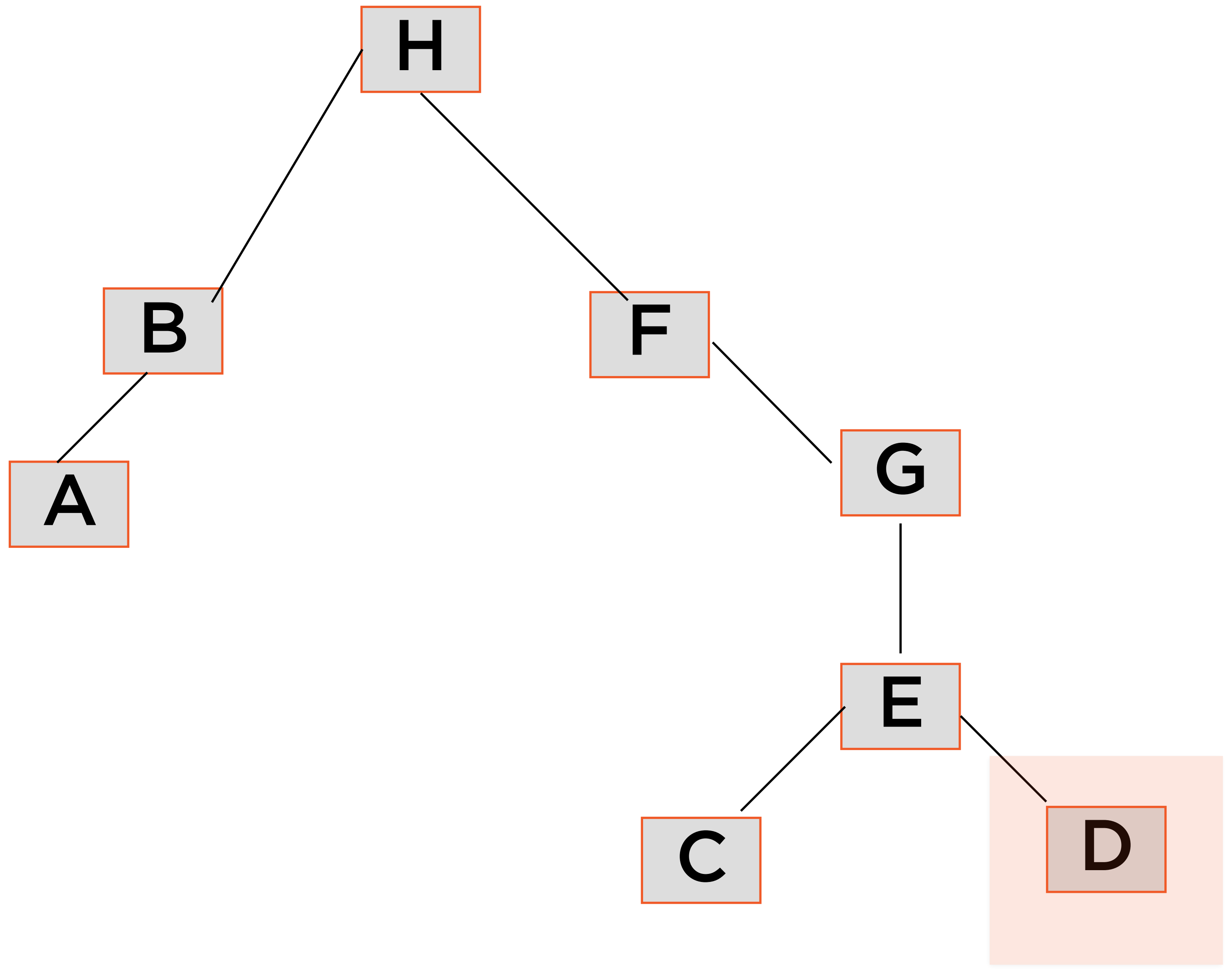
Visited H - B - F - A - G - E

“Breadth-first” Tree Traversal



Visited H - B - F - A - G - E - C

“Breadth-first” Tree Traversal



Visited H - B - F - A - G - E - C - D

Two Ways of Traversing Graphs

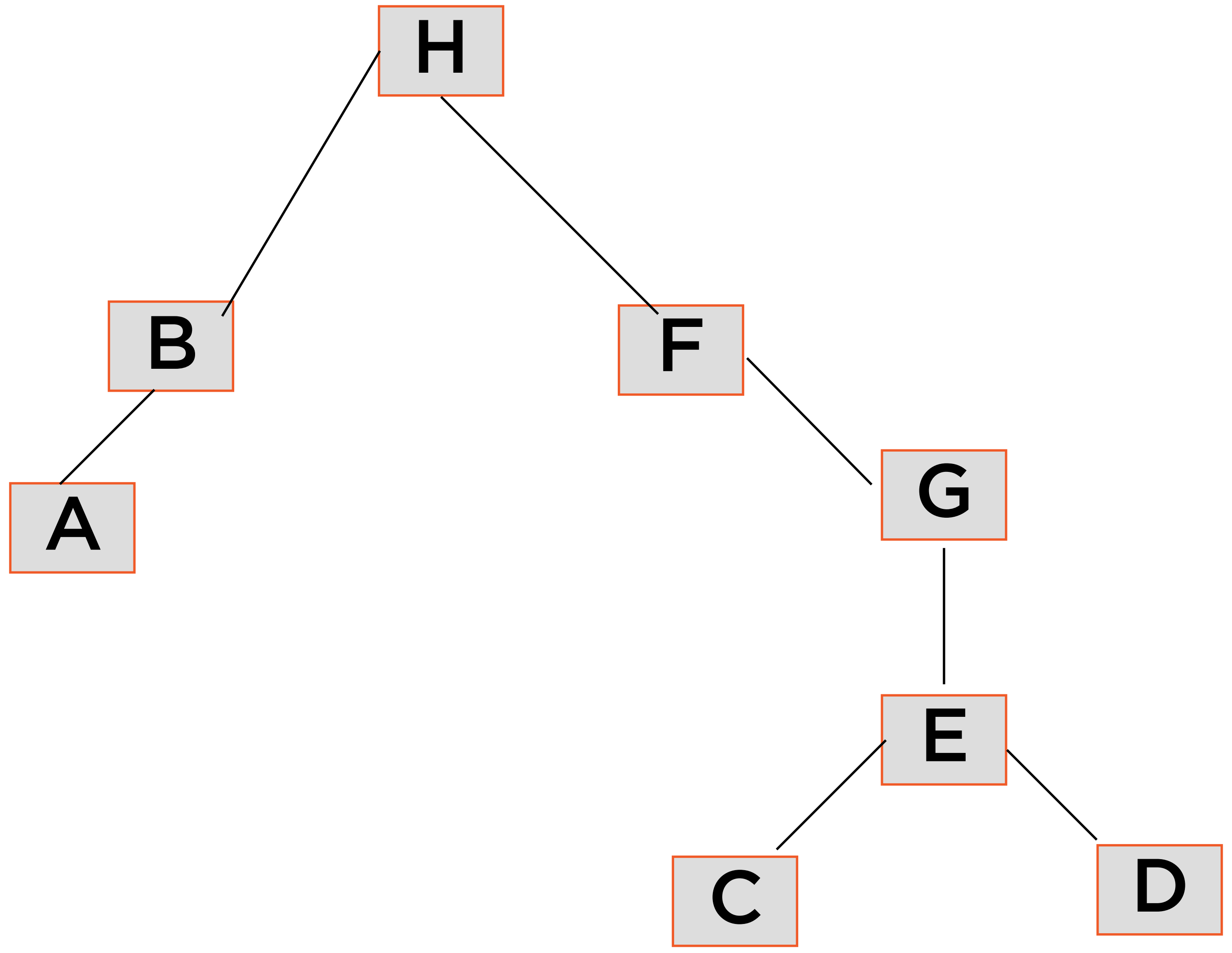
Breadth-first

All nodes at same distance from origin visited together

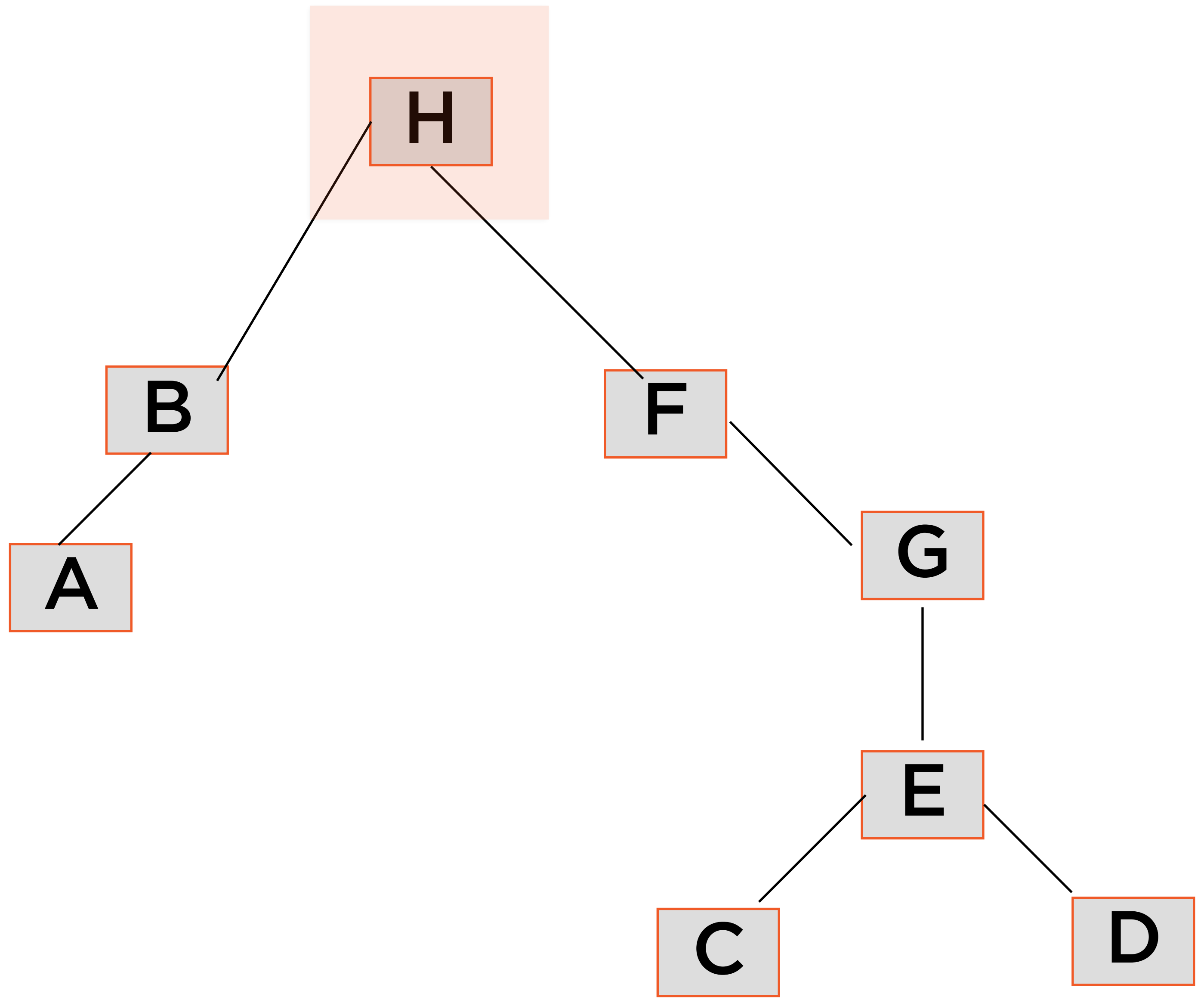
Depth-first

All nodes in certain direction from origin visited together

“Depth-first” Tree Traversal

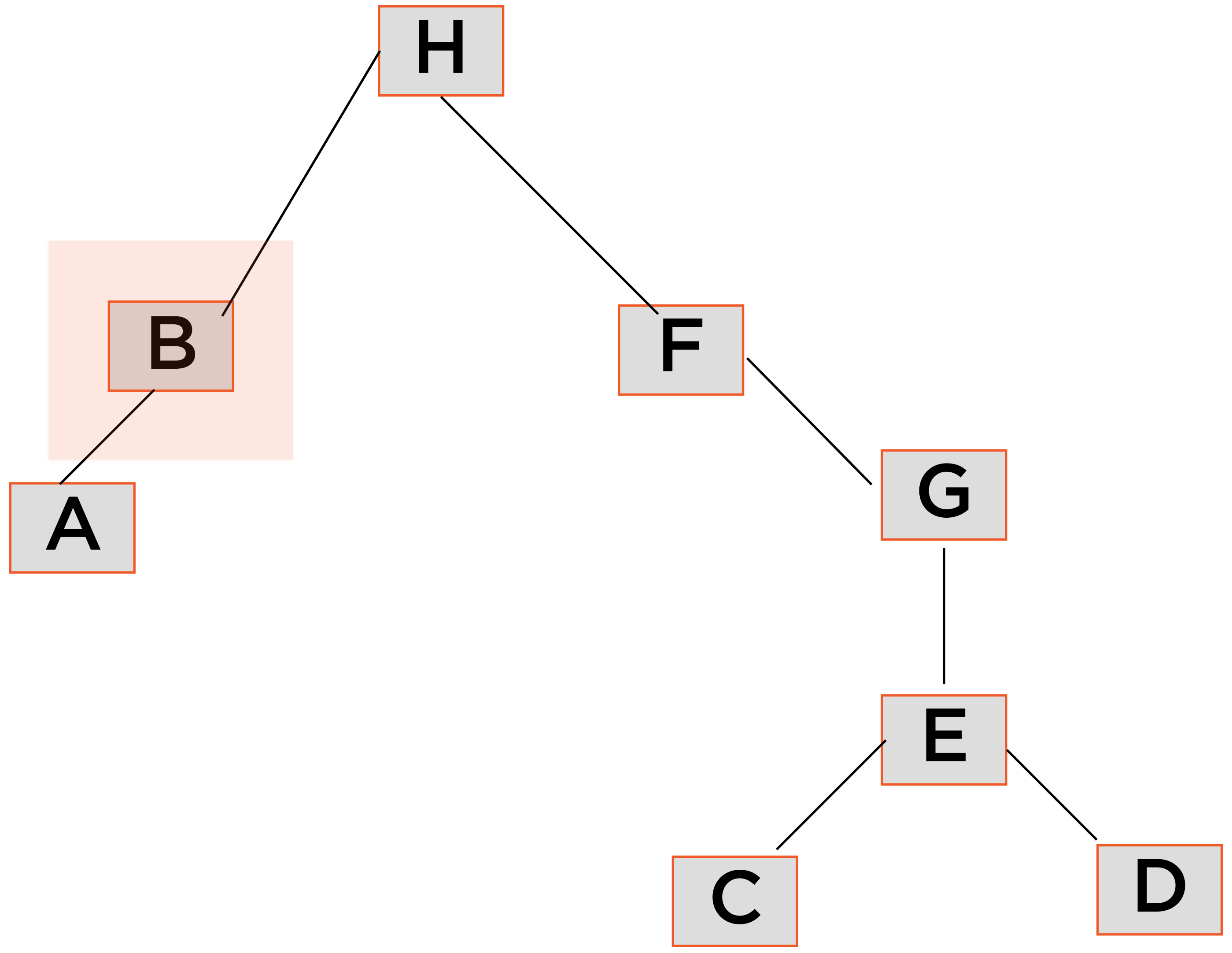


“Depth-first” Tree Traversal



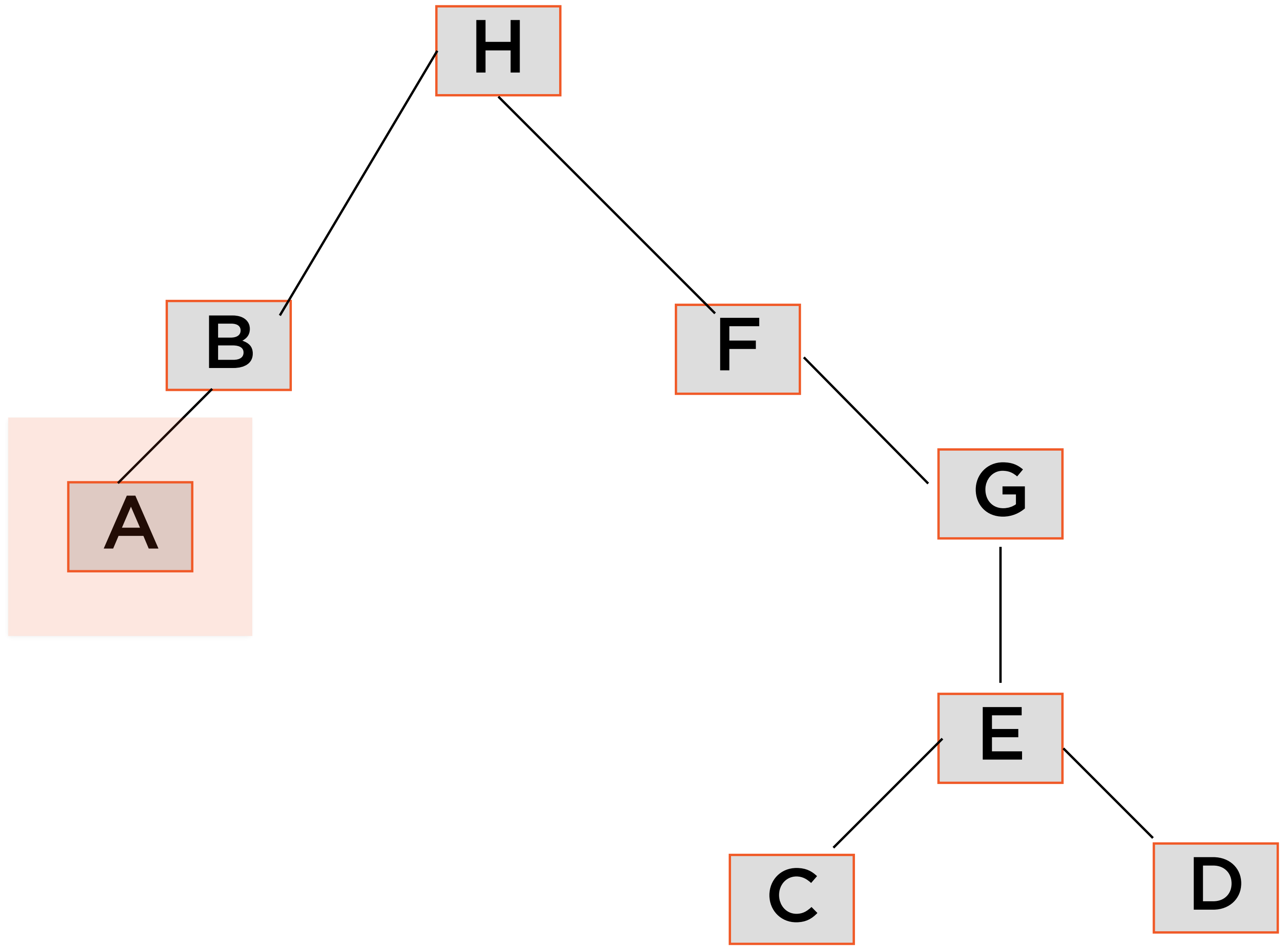
Visited H

“Depth-first” Tree Traversal



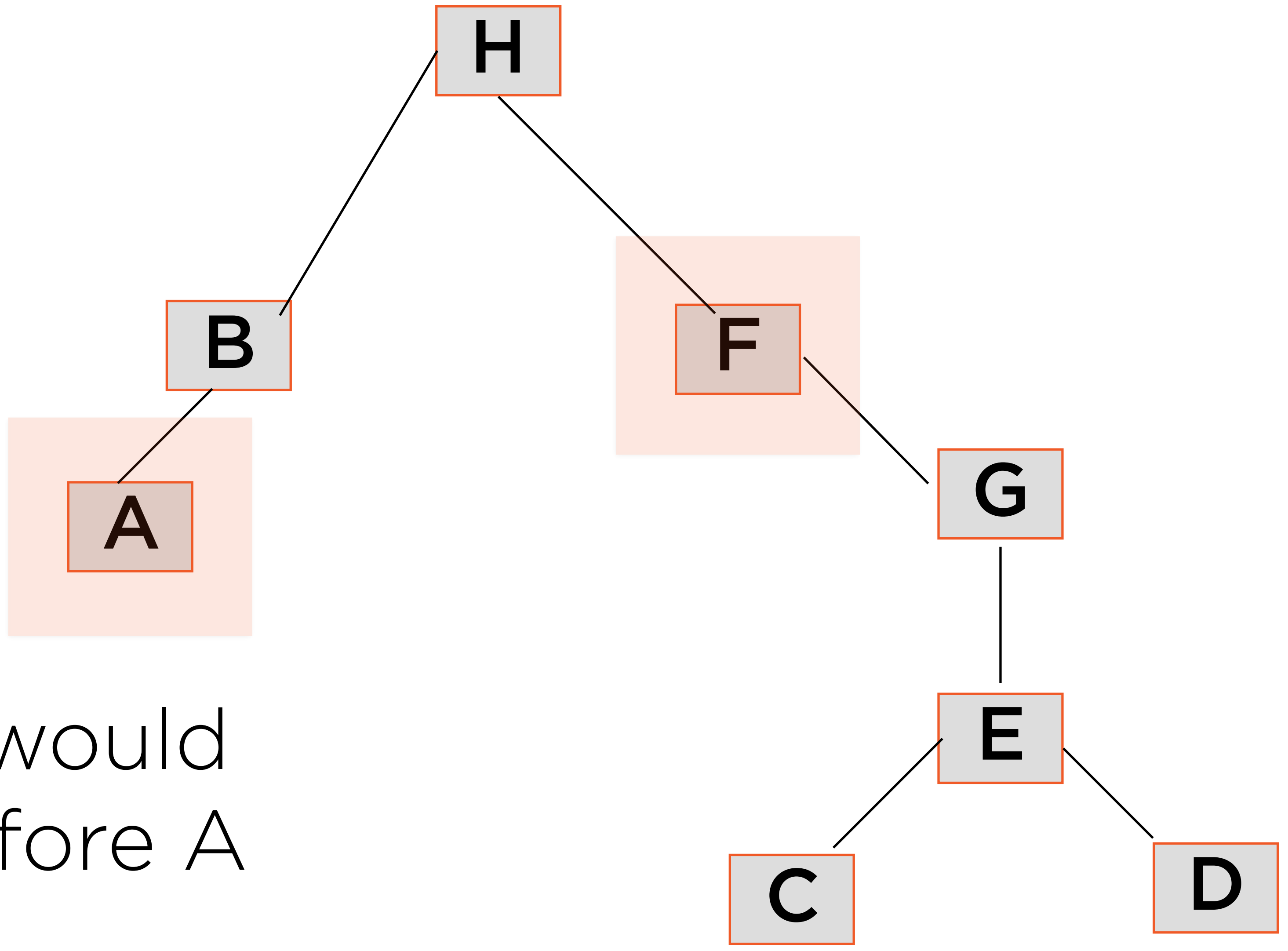
Visited H - B

“Depth-first” Tree Traversal



Visited H - B - A

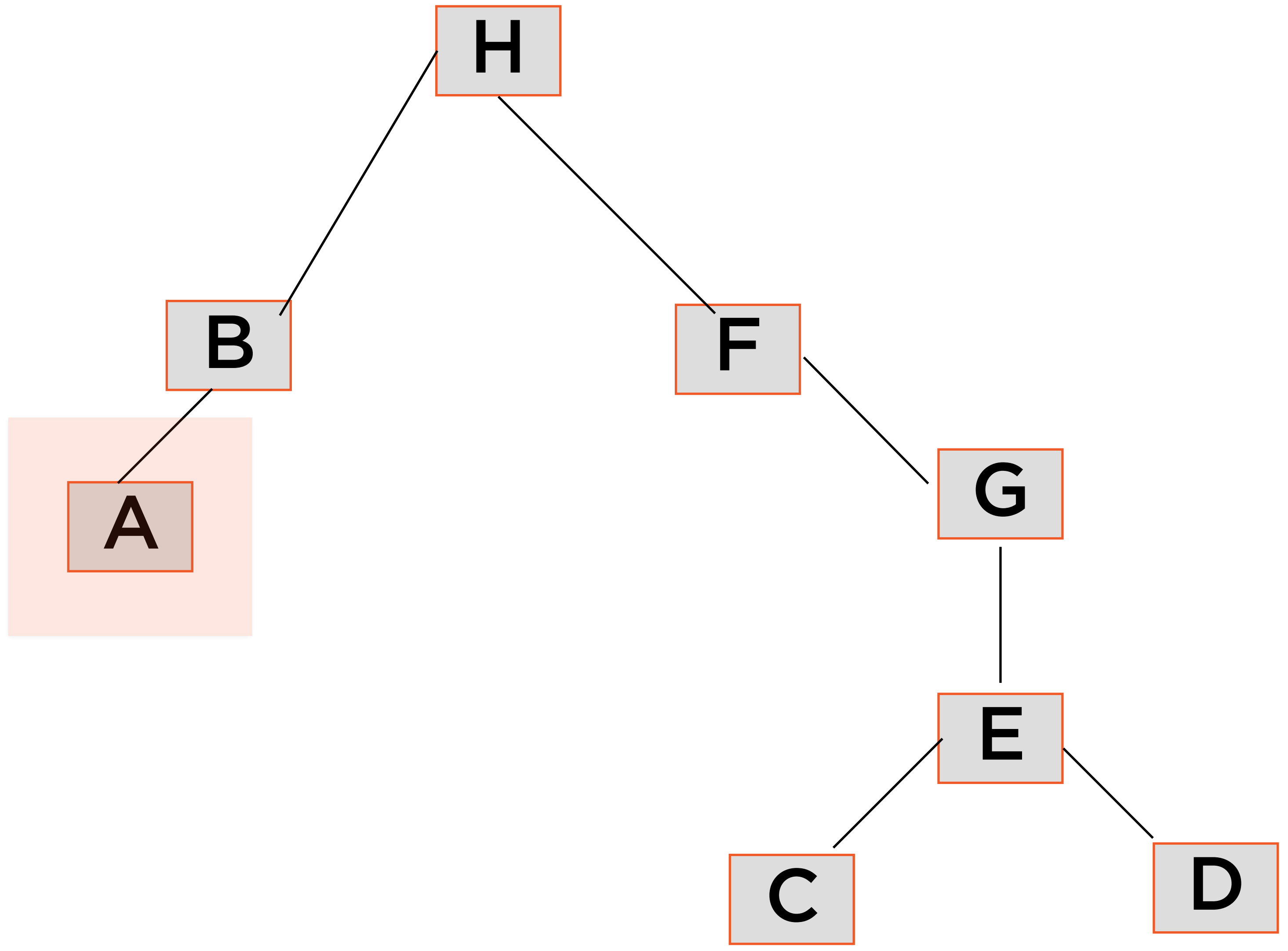
“Depth-first” Tree Traversal



In breadth-first, would have visited F before A

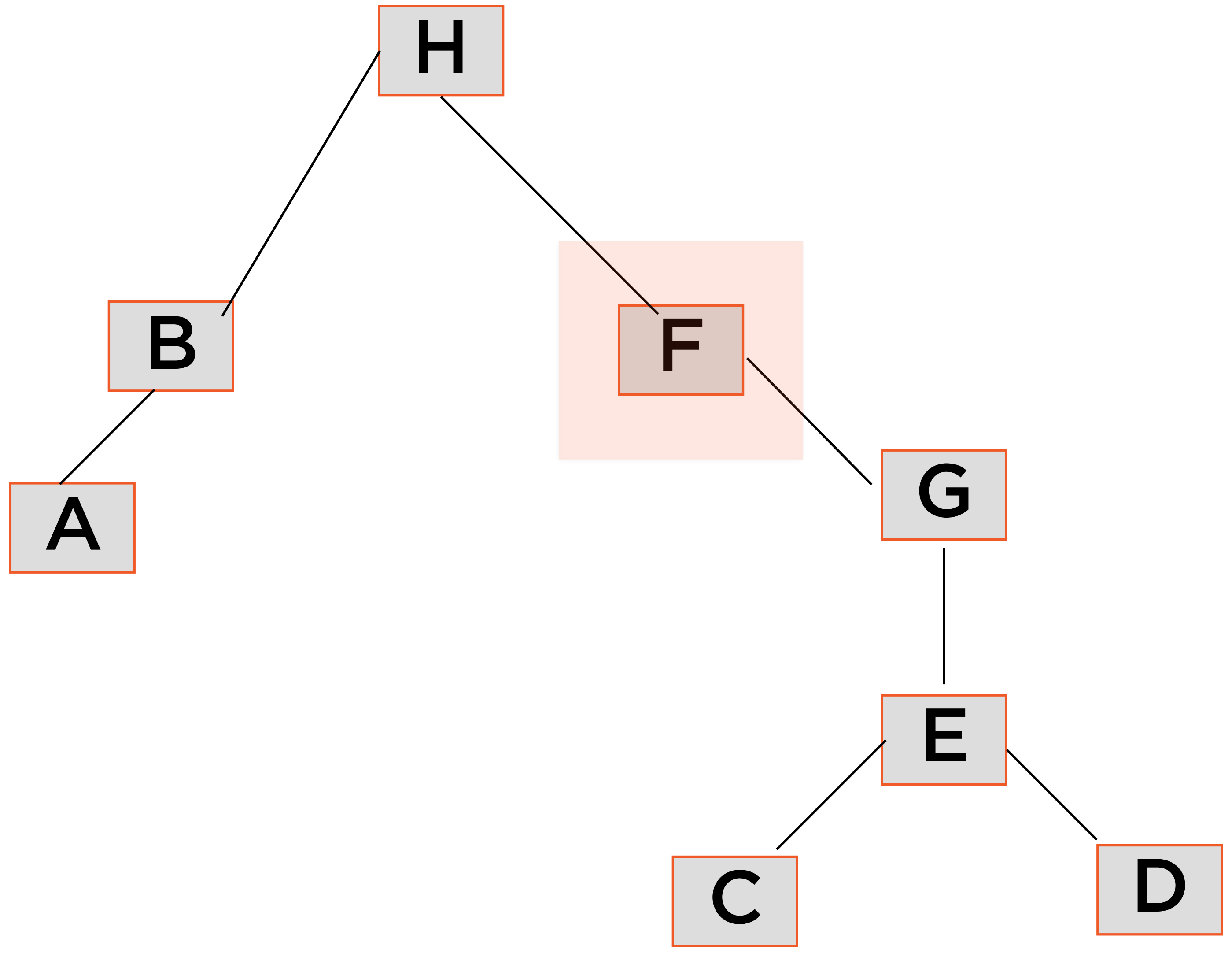
Visited H - B - A

“Depth-first” Tree Traversal



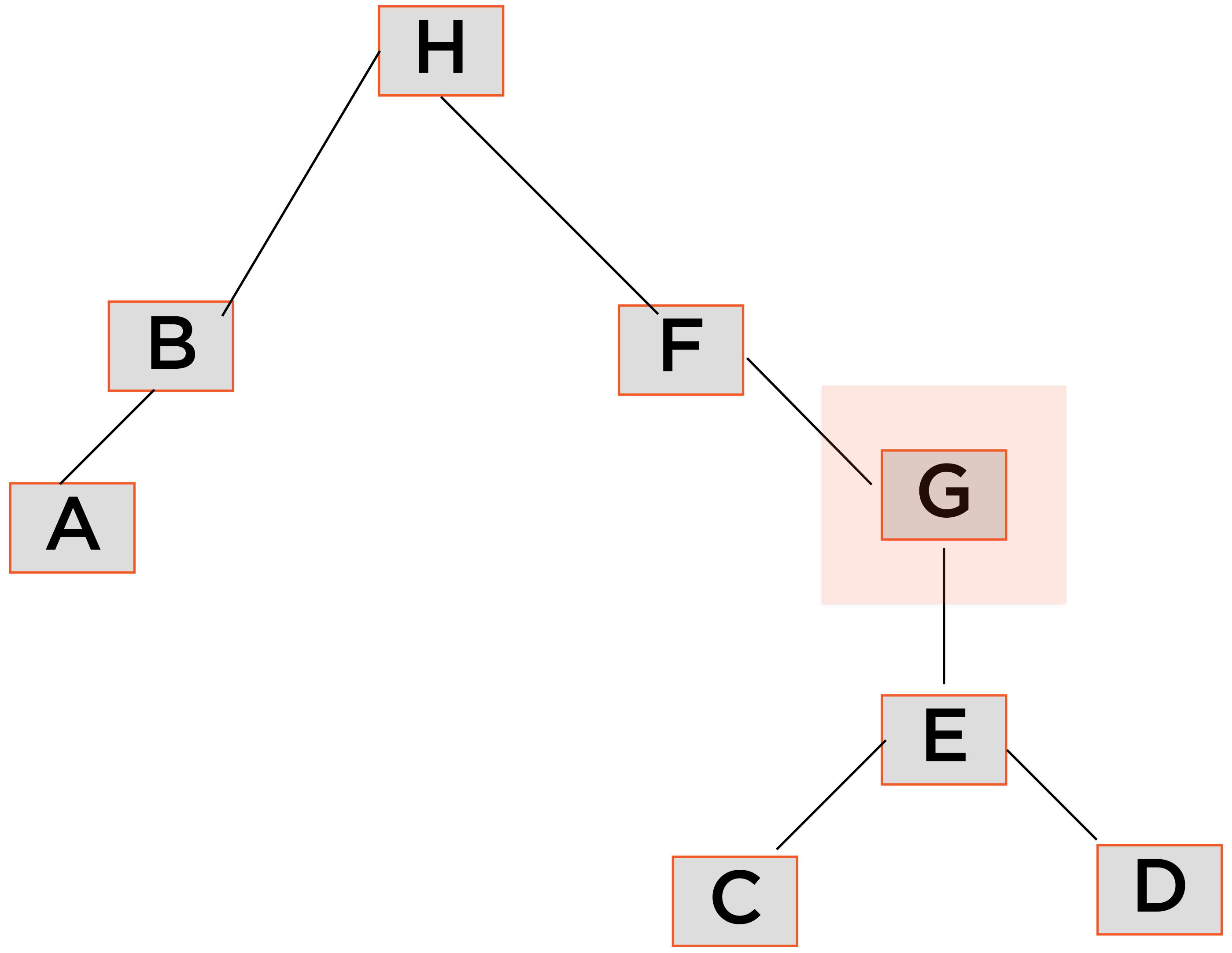
Visited H - B - A

“Depth-first” Tree Traversal



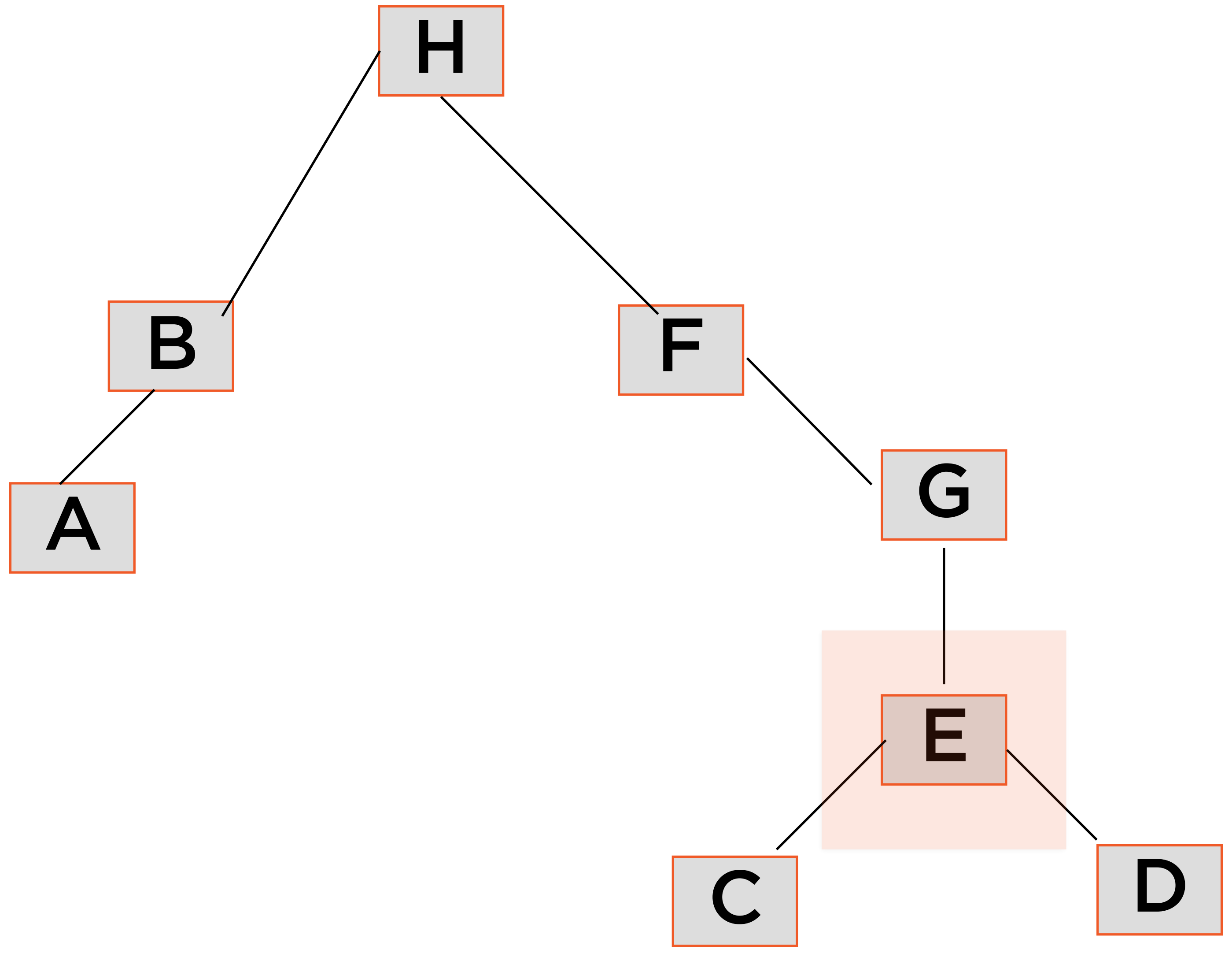
Visited H - B - A - F

“Depth-first” Tree Traversal



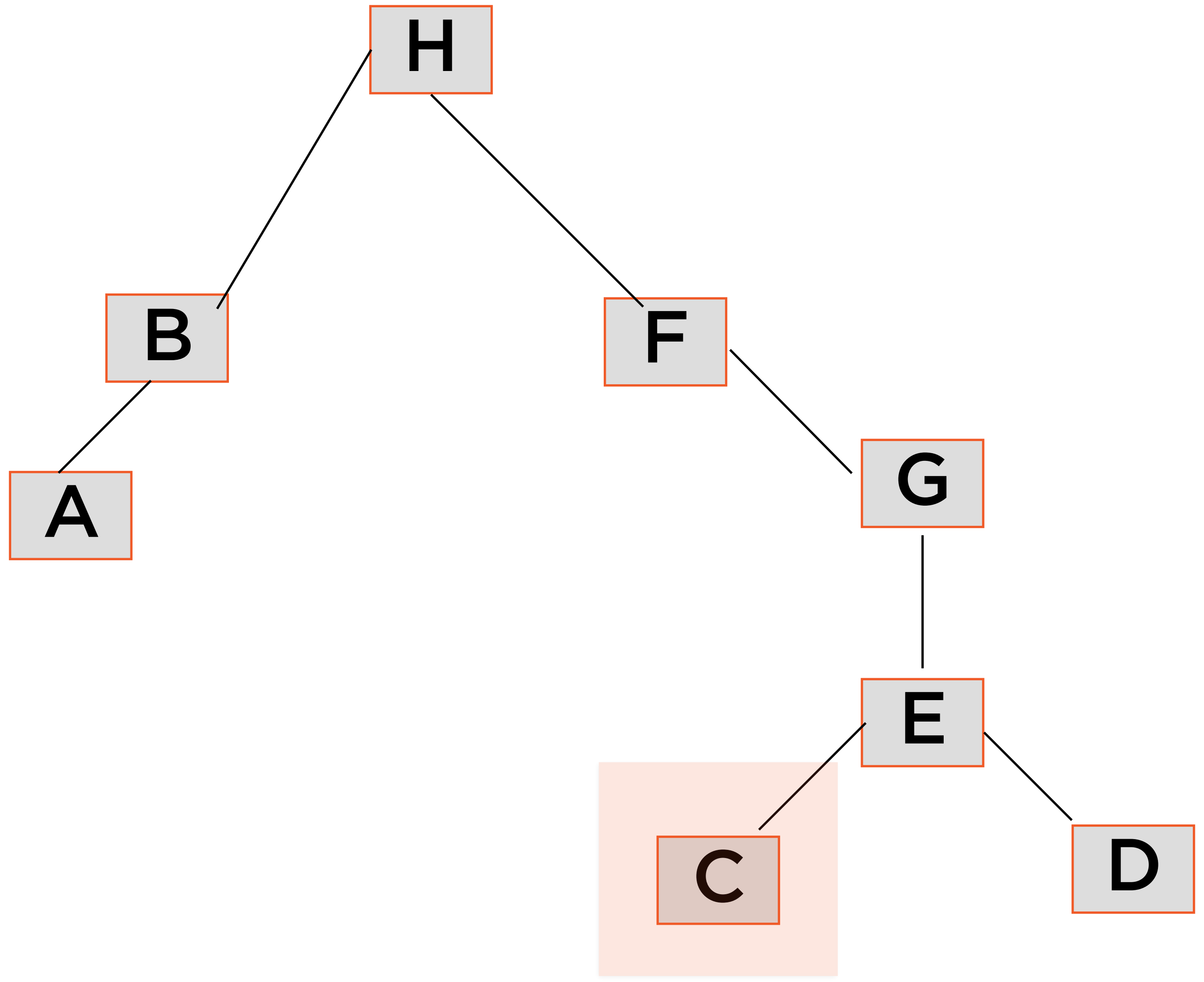
Visited H - B - A - F - G

“Depth-first” Tree Traversal



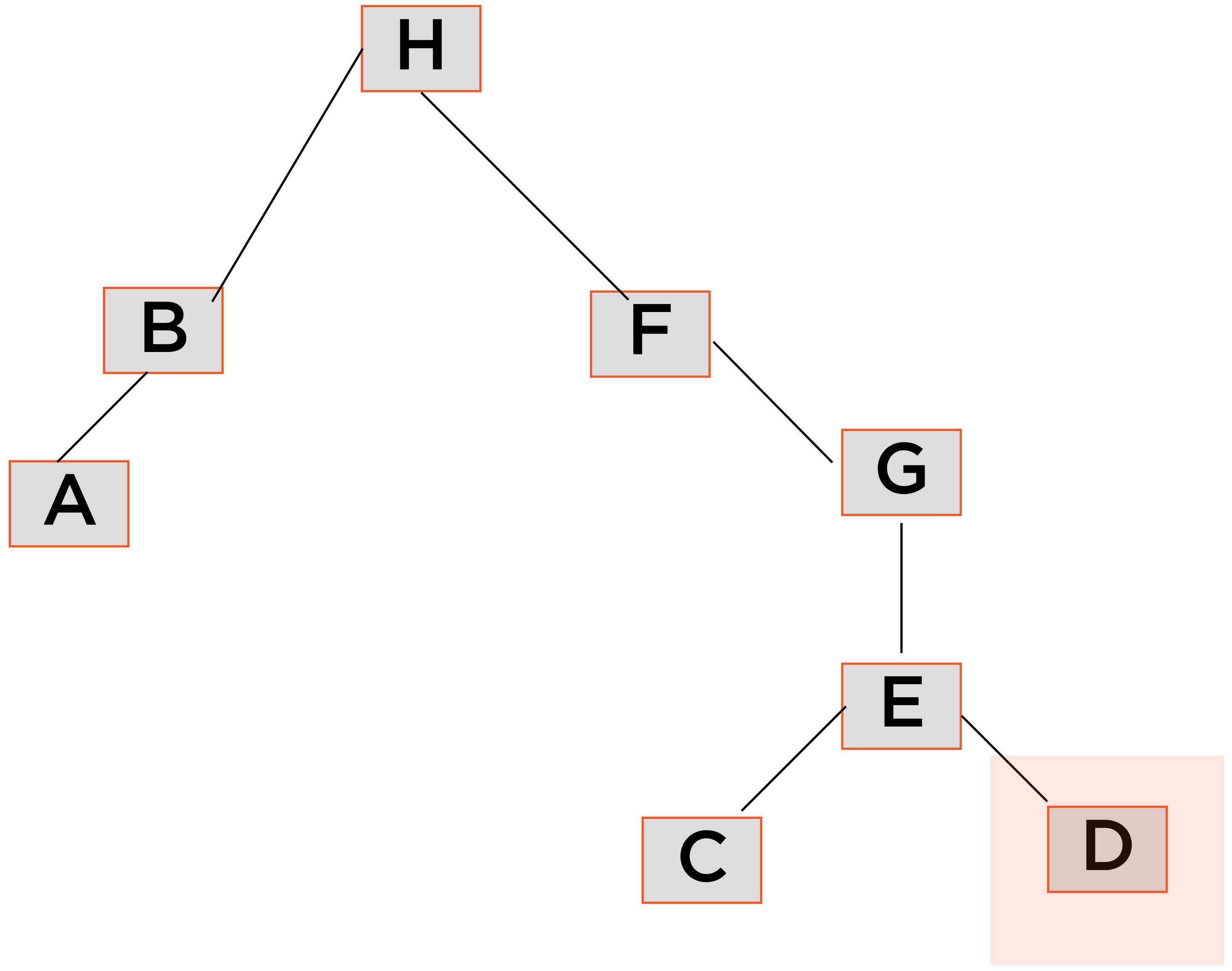
Visited H - B - A - F - G - E

“Depth-first” Tree Traversal



Visited H - B - A - F - G - E - C

“Depth-first” Tree Traversal



Visited H - B - A - F - G - E - C

Two Ways of Traversing Graphs

Breadth-first

All nodes at same distance from origin visited together

Depth-first

All nodes in certain direction from origin visited together

Two Ways of Using Scrapy

Broad Crawls

Crawl a forest of domains, stopping only when resources exhausted; used in search engines

Depth-first

All nodes in certain direction from origin visited together

Two Ways of Using Scrapy

Broad Crawls

Crawl a forest of domains, stopping only when resources exhausted; used in search engines

Focused Crawls

Crawl a specific site, usually with a single Scrapy Spider

Most Scrapy defaults are for focused crawls, which are similar logically to depth-first traversal

Two Types of Crawls

Focused Crawls	Broad Crawls
Usually crawl a single, specific domain	Usually used to crawl an unbounded set of domains
Crawl to completion	Limit crawl by number of pages or time
Logic is usually quite complex - handled inside single Spider	Logic is usually simple - data processed in a separate stage
Parallelism in scraping a single domain restricted by politeness	Parallelism in scraping multiple domains is not limited, hence very fast

Broad Crawls are very fast because the full parallelism of Scrapy comes into play

Broad Crawl Best Practices

Increase global concurrency

Increase max thread pool size

Set up your own DNS

Reduce log level

Disable cookies

Disable retries

Reduce download timeout

Disable redirects

Enable crawl of Ajax “crawlable pages”

Broad Crawl Best Practices

- Increase global concurrency
- Increase max thread pool size
- Set up your own DNS
- Reduce log level
- Disable cookies
- Disable retries
- Reduce download timeout
- Disable redirects
- Enable crawl of Ajax “crawlable pages”

Broad Crawl Best Practices

Increase global concurrency

Increase max thread pool size

Set up your own DNS

Reduce log level

Disable cookies

Disable retries

Reduce download timeout

Disable redirects

Enable crawl of Ajax “crawlable pages”

Broad Crawl Best Practices

Increase global concurrency

Increase max thread pool size

Set up your own DNS

Reduce log level

Disable cookies

Disable retries

Reduce download timeout

Disable redirects

Enable crawl of Ajax “crawlable pages”

Broad Crawl Best Practices

Increase global concurrency

Increase max thread pool size

Set up your own DNS

Reduce log level

Disable cookies

Disable retries

Reduce download timeout

Disable redirects

Enable crawl of Ajax “crawlable pages”

Demo

Building broad crawlers using Scrapy

Demo

Debugging crawlers using telnet

Demo

Auto throttling broad crawlers

Summary

Scrapy has multiple features which are useful when crawling websites at scale

Logging events to console and to file

Using the telnet utility to debug crawlers

Working with broad crawlers to concurrently crawl multiple sites

Auto throttling crawls so as to not fall afoul of website policies