Directed Graphs

Computing 2 COMP1927 16x1

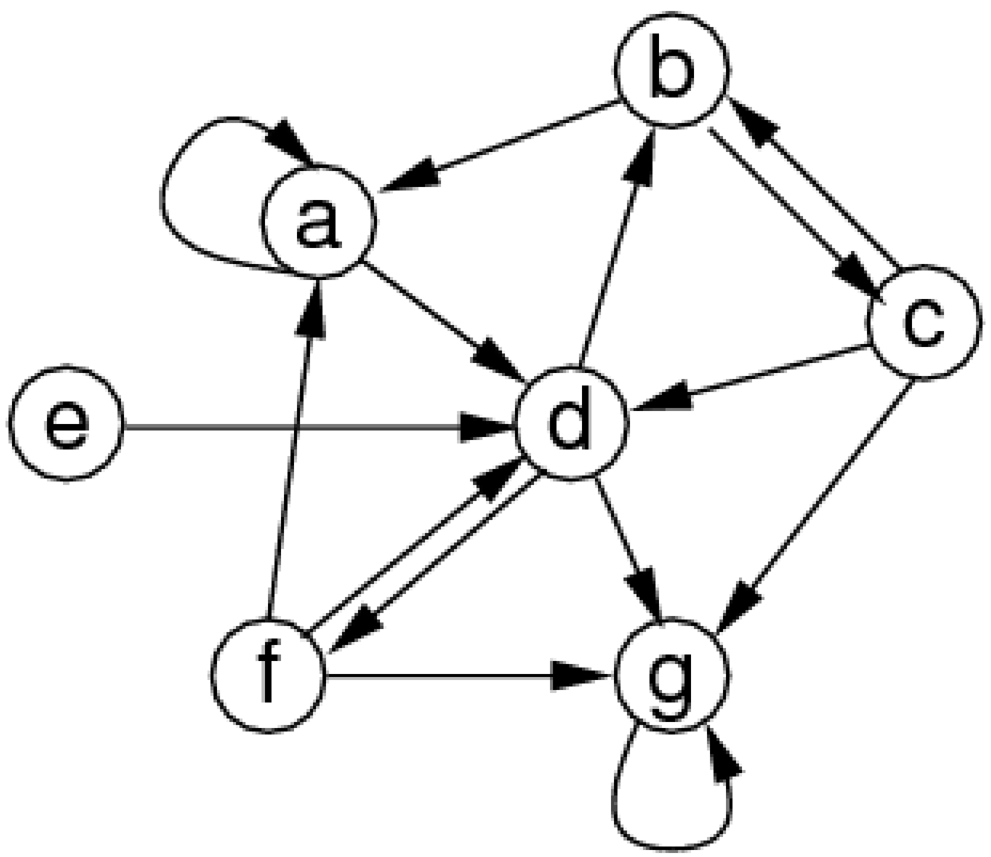
Directed Graphs

- In our previous discussion of graphs:
  - an edge indicates a relationship between two vertices
  - an edge indicates nothing more than a relationship
- In many real-world applications of graphs:
  - edges are directional ($v \rightarrow w \neq w \rightarrow v$)
  - For example a one way street
  - Liking a fan page on facebook, following someone on twitter
- Directed graphs include
  - edges that are directional
  - Self -loops

Potential Digraph Application Areas

Domain	Vertex	Edge
Web	Web page	Hyperlink
Chess	Board Pos	Legal Move
Scheduling	Task	Precedence
Program	Function	Function Call
Science	Journal Article	Citation

EXAMPLE OF A DIRECTED GRAPH

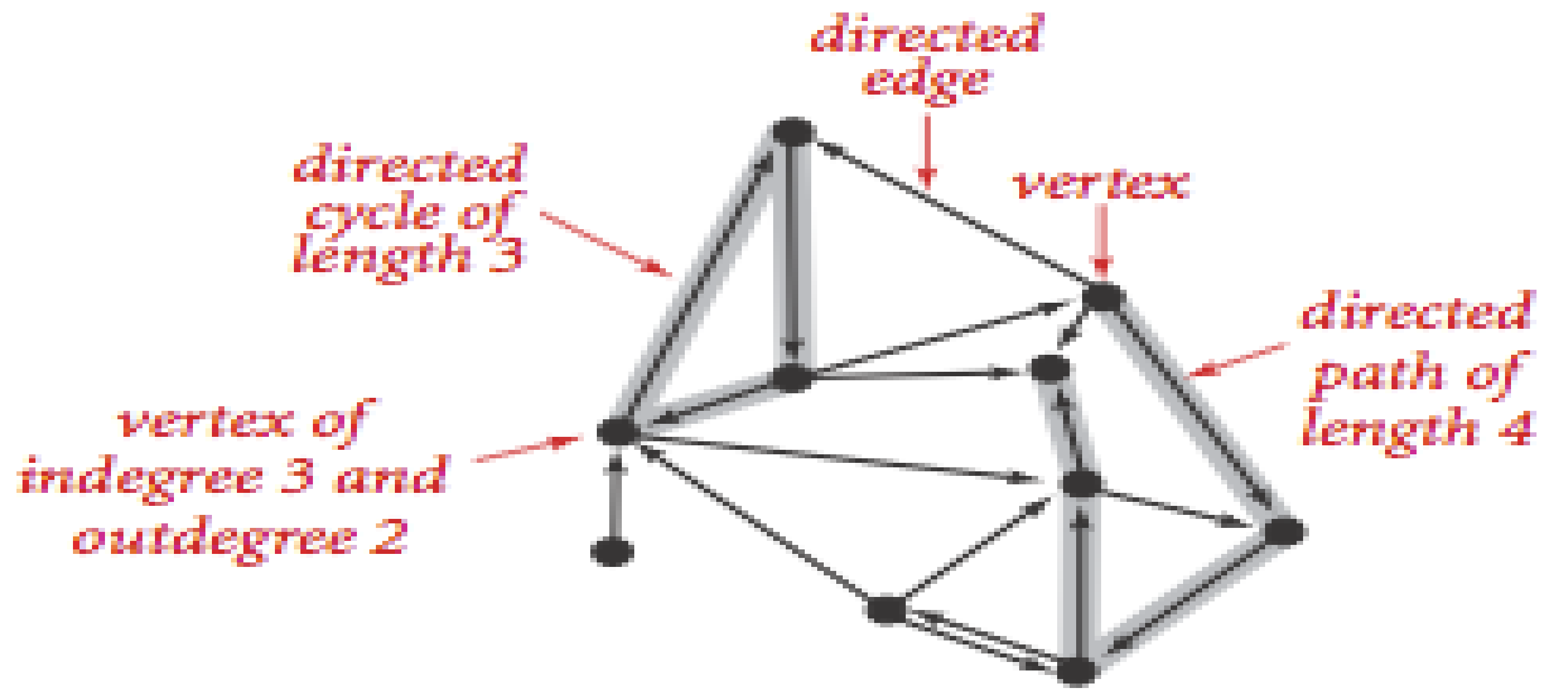


	a	b	c	d	e	f	g
a	1	0	0	1	0	0	0
b	1	0	1	0	0	0	0
c	0	1	0	1	0	0	1
d	0	1	0	0	0	1	1
e	0	0	0	1	0	0	0
f	1	0	0	1	0	0	1
g	0	0	0	0	0	0	1

adjacency matrix

TERMINOLOGY FOR DIRECTED GRAPHS

- Out-degree (d(v))
  - The number of directed edges leading out of the vertex
- In-degree ($d^{-1}(v)$)
  - The number of directed edges leading into a vertex
- Directed acyclic graph (DAG):
  - graph containing no directed cycles

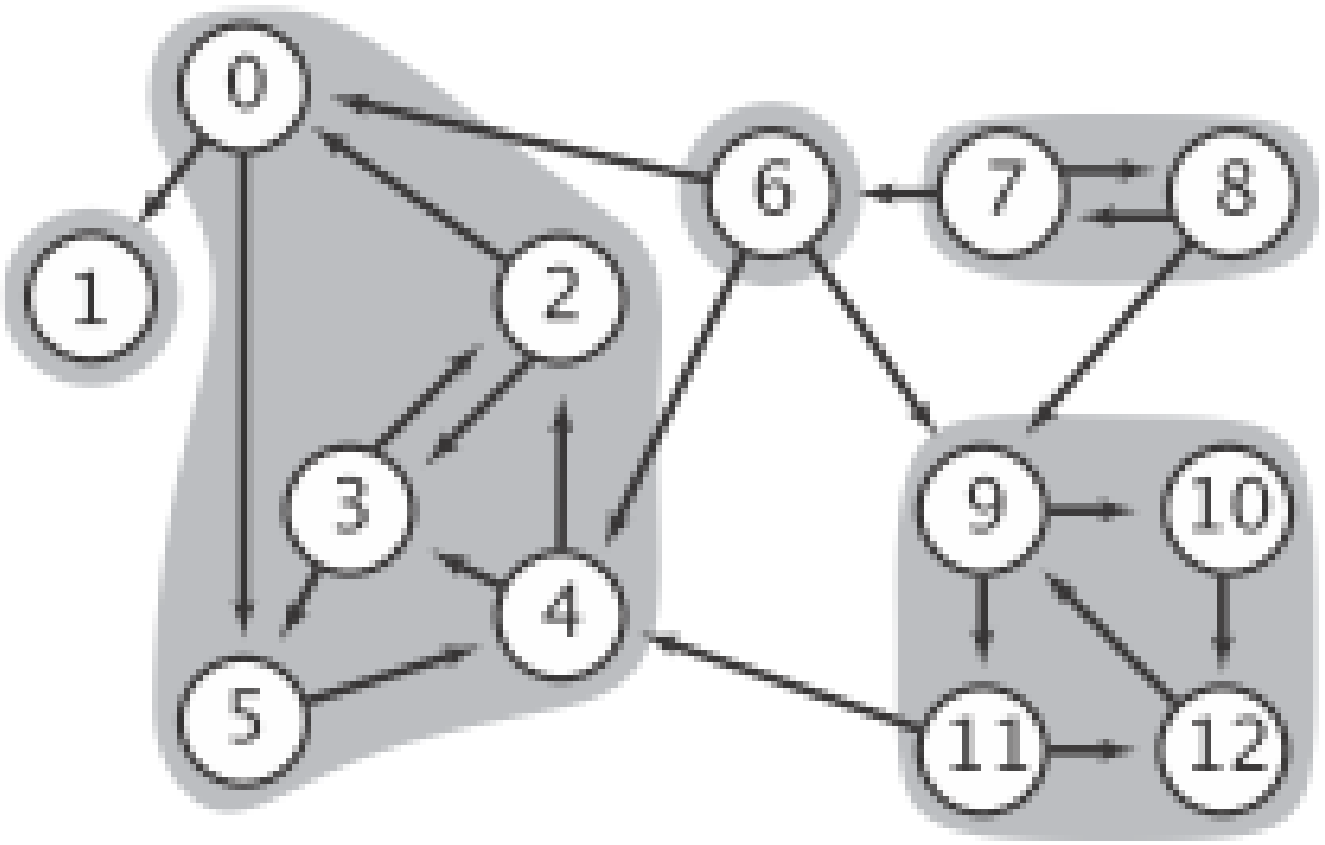


Anatomy of a digraph

TERMINOLOGY FOR DIRECTED GRAPHS

- Reachability:
  - *w* is reachable from *v* if *there exists a* directed path *v,…,w*
- Strongly Connected:
  - Two vertices v and w are *strongly connected* if they are mutually reachable: there is a directed path from v to w and a directed path from w to v.
- Strong connectivity:
  - every vertex is reachable from every other vertex
- Strongly connected components:
  - A digraph that is not strongly connected consists of a set of *strongly-connected components*, which are maximal strongly-connected subgraphs.

Strong Connected Components



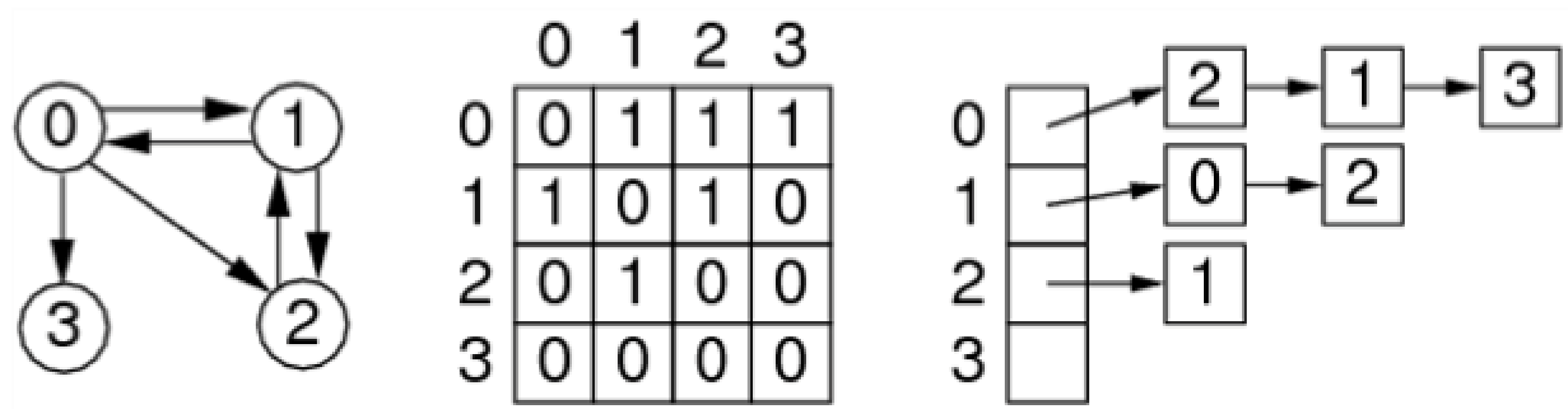
A digraph and its strong components

Problems To Solve on Digraphs

- is there a directed path from *s* to *t*? (transitive closure)
- what is the shortest path from *s* to *t*? (shortest path)
- are all vertices mutually reachable? (strong connectivity)
- how to organise a set of tasks? (topological sort)
- how to build a web crawler? (graph traversal)
- which web pages are "important"? (PageRank)

DIGRAPH REPRESENTATION

- Similar set of choices as for non-directional graphs:
  - *V* vertices identified by *0 .. V-1*
  - vertex-indexed adjacency matrix (non-symmetric)
  - vertex-indexed adjacency lists
- What needs to be modified to turn our undirected graph implementations into directed graphs?



digraph *adj matrix* *adj lists*

Cost of Representations

	Storage	Add Edge	Edge Exist?	Get edges leaving v
Adj matrix	$V + V^2$	1	1	V
Adj list	V + E	d(v)	d(v)	d(v)

- Where d(v) is the degree (out degree) of vertex v.

Directed Graph Traversal

- Can use some of the same algorithms as for non-directed graphs
  - depth-first searching (DFS)
  - breadth-first searching (BFS)
- Example: Web Crawling
  - visit every page on the web
  - Solution:
    - breadth-first search with "implicit" graph
    - visit operation scans page and collects e.g. keywords and links
  - Assumption:
    - web is fully connected

Web Crawling Pseudo-Code

```
webCrawl(startingURL):
   mark startingURL as alreadySeen
      enqueue(Q, startingURL)
      while not empty(Q)
         nextPage = dequeue(Q)
         visit nextPage
    foreach (hyperLink in nextPage)
      if (hyperLink not alreadySeen)
         mark hyperLink as alreadySeen
         enqueue(Q, hyperLink)
```

- visit scans page and collects e.g. keywords and links