Data Structures and Algorithms

Spring 2009-2010

Outline

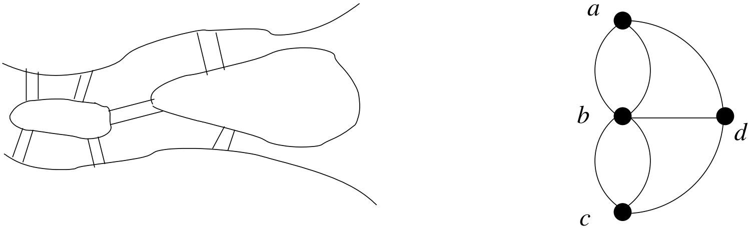
Outline

Introduction

- A *graph* $G = (V, E)$ consists of a set of vertices, V, and a set of edges, E
- An edge connects two vertices in V so that: $\forall e \in E, e = (u, v),$ where $u, v \in V$
- If $e = (v, v) \in E$ we say that e is a *loop* or a *self-edge* – not common
- A *digraph* is a graph where edge vertices are ordered so that $(u, v) \neq (v, u)$; edge (u, v) is drawn as an arrow pointing from u to v
- Vertex u is adjacent to $v \Leftrightarrow (u, v) \in E$; if the graph is *undirected*, v is also adjacent to u
- Edges will often have a *weight* or *cost* associated with them

Origins of Graph Theory

The Bridges of Königsberg:



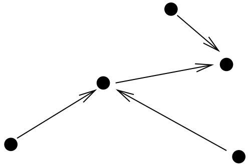
Wikipedia's historical account is here

Introduction (contd.)

- A **cycle** in a directed graph is a path of length at least 1 where $w_1 = w_n$; the cycle will be simple if the path is simple
- If graph is undirected all edges must be distinct so that u, v, u is not considered a cycle
- An **acyclic** directed graph (DAG) is a graph with no cycles
- A **complete** graph is a graph where every pair of vertices has an edge between them: if undirected $|E| = \binom{n}{2}$; if directed $|E| = n(n-1)$

Introduction (contd.)

- An undirected graph is called **connected** if there is some path from every vertex to every other vertex
- A directed graph is called **strongly connected** if there is some path from every vertex to every other vertex
- A directed graph is called **weakly connected** if the graph is not strongly connected but the underlying undirected graph is connected

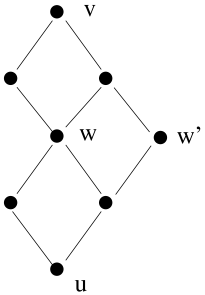


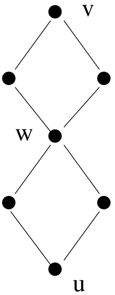
Graph is not strongly connected but **is** weakly connected.

Introduction (contd.)

- A graph is called *k-connected* if there are *k* vertex-disjoint paths between every pair of nodes

Graph is 2–connected.



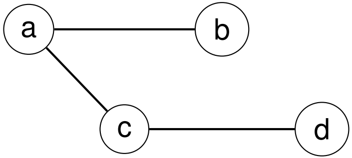


Graph is 1–connected since there is only one vert–disjoint path from u to v.

Representing Graphs Internally

- We can represent the adjacancies of a graph using a $|V| \times |V|$ array, `adj`
- `adj[1][2]` will be `1` if there is an edge between v_1 and v_2, `0` otherwise
- If the graph is *weighted* then we can easily store the weight of the edge in `adj[1][2]`
- Space requirement: $\Theta(|V|^2)$ – not unreasonable if graph has many edges but, as is more often the case, for *sparse* graphs this is very wasteful
- Instead, use *adjacancy list*: keep a list of the nodes that each node is connected to
- Storage requirement is now $O(|V| + |E|)$

Representing Graphs Internally (contd.)



	a	b	c	d
a	0	1	1	0
b		0	0	0
c			0	1
d				0

Symmetric

Adjacency Matrix

a : $\boxed{b} \rightarrow \boxed{c}$

b : $\boxed{a}$

c : $\boxed{a} \rightarrow \boxed{d}$

d : $\boxed{c}$

Adjacency List