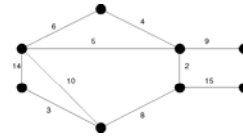


Graphs



- A set of nodes (vertices), denoted by V .
- A set of edges, denoted by E .
- What is an edge ?
 - » Conceptually, an edge specifies 2 nodes (u,v) that it connects.
 - » Can have associated direction. ("directed graph")
 - » Can have a "weight", i.e. a real number associated with it.
E.g. can represent distance or cost
- Representation:
 - » Adjacency matrix: $A(i,j) = 1$ iff edge (i,j) exists, i.e. $(i,j) \in E$, otherwise $A(i,j) = 0$
 - » Adjacency-list: For every node $v \in V$, create linked list of adjacent edges.
 - » Easy to convert from one representation to another, but takes time.
 - » Sparse vs. dense graphs

144

Types/properties of graphs

- Directed or undirected
- Weighted/unweighted
- Sparse/Dense
- Connected/ not connected
- Tree/Forest (no cycles)
- Bipartite
- Strongly connected (for directed graph)
- ...

145

Examples of graph problems

- Is there a cycle? Directed cycle? Negative weight cycle?
- Can you reach from node u to node v ?
- What is the shortest path from u to v ?
- All-pairs shortest paths
- Find spanning tree (tree, subset of original edges, touches all nodes)
- Find spanning tree of minimum weight (for weighted graphs)
- Maximum Bipartite matching
- ...
- We will discuss some of the above problems later.
Some other problems are in subsequent courses (e.g. CS261)

146

Graph Algorithms

- Examples of graph problems:
 - » Direct applications:
 - City streets map: reachability, shortest path, congestion management
 - Communication networks: planning, fault tolerance/reliability, topology augmentation
 - » Indirect applications:
 - Assigning MDs to hospitals
 - Scheduling jobs on a multiprocessor
 - Searching solution spaces
- Restate as a graph problem \longrightarrow solve \longrightarrow map back

147

Depth First Search

- **Visit(u)**
color(u) = gray; d(u) = time; time++;
for each neighbor w of u:
 if w is white then Visit(w)
 color(u) = black; f(u) = time; time++;
- Initially, set all nodes to be white,
examine nodes one-by-one, call **Visit** if node is still white.
- Node visited once, edge touched twice:
Running time $O(n+m)$

148

Edge Classification

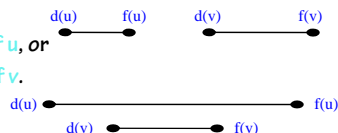
- Classification of uw according to (color of u) \rightarrow (color of w):
(when the edge is considered)
 - » Tree edge: gray \rightarrow white
 - » Back edge: gray \rightarrow gray
 - » Forward: gray \rightarrow black, u ancestor of w .
 - » Cross: other gray \rightarrow black edges.
- How to distinguish forward and cross edges ??
 - » We can use $d()$ time !
 - » $d(u) < d(w)$ - forward edge

149

Parenthesis Theorem

- **Theorem:**
For any two nodes u and v ,
the two intervals $[d(u), f(u)]$ and $[d(v), f(v)]$ either:

- » Do not intersect, or
- » $[d(u), f(u)]$ includes $[d(v), f(v)]$, v descendant of u , or
- » $[d(v), f(v)]$ includes $[d(u), f(u)]$, u descendant of v .



- **Proof:**
 - » Assume (wlog) $d(u) < d(v)$.
 - » If v was not discovered before finishing u , then we have case 1 above.
 - » If v was discovered, then we have to finish it before returning and finishing u , leading to case 2.
 - » Case 3 is symmetric.

150

White-Path Lemma

- In (directed or undirected) graph G , node v is descendant of u iff at $d(u)$ (time when u was discovered) there is a path from u to v using only currently white nodes.

- **Proof:**
 - » Assume v is descendant of u .
Let ww' be edge on the $u \rightarrow v$ path in the tree.
If w' was not white at $d(u)$, then ww' will not be tree edge.
Thus, all nodes on the $u \rightarrow v$ path are white when u is discovered.
 - » Assume that at $d(u)$ there is a white path from u to v .
Let ww' be the first edge on this path, with w' closest to u so that w is descendant of u but w' is not.
 - We have $f(u) > f(w) > d(w) > d(u)$.
 - But we have to discover w' after starting u and before finishing w :
 $d(u) < d(w') < f(w) < f(u)$
 - By parenthesis theorem, w' is also a descendant of u , contradiction.

151

Simple Lemma

- **Lemma:** if G undirected, then only tree and back edges.

Proof: Consider uv edge, WLOG $d(u) < d(v)$.

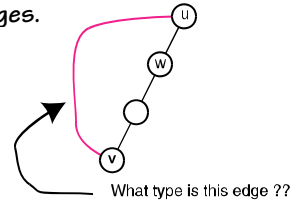
Thus v must be discovered and finished before finishing u , since uv exists.

If uv discovered from u , before v ,

it is **tree edge**

if v was discovered before uv ,

uv becomes a **back edge**.



- HW: Why does the proof break down in the directed case ?

152

Discovering Cycles

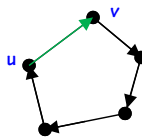
- **Claim:** G acyclic iff DFS yields no back edges.

- **Proof:**

» Trivial to observe that back edge implies a cycle.

» Assume there exists a cycle:

- Let v be the node with smallest d on the cycle and let uv be edge of the cycle.
- At $d(v)$ all nodes on the cycle, including u , are white.
- All these nodes, including u , become descendants of v .
- Why ?
- White-path lemma!
- Thus, when u is scanned, we will discover uv edge and mark it as "back edge".

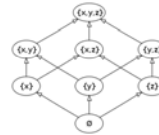


153

Partial Order

- Example: subsets ordered by inclusion

» Note implicit relationships, e.g. $x \in \{x,y,z\}$ but there is no edge between them.



- Acyclic directed graph, vertices ordered by reachability



- Topological sort: total order compatible with partial order.

» 7,5,3,11,8,2,10,9

» 3,7,8,5,11,10,9,2

- Numerous applications, mostly in scheduling: instructions, compilations in makefile, etc.