Breadth-First Search

- Completely explore the vertices in order of their distance from \( v \)
- Naturally implemented using a queue
- Works on general graphs, not just trees

BFS(v)

Global initialization: mark all vertices "undiscovered"

BFS(v)
mark \( v \) "discovered"
queue = \( v \)
while queue not empty
\( u = \) remove_first(queue)
for each edge \( (u,x) \)
if \( x \) is undiscovered
mark \( x \) discovered
append \( x \) on queue
mark \( u \) completed

Exercise: modify code to number vertices & compute level numbers

BFS analysis

- Each edge is explored once from each end-point
- Each vertex is discovered by following a different edge
- Total cost \( O(m) \) where \( m = \) # of edges
- Disconnected? Restart @ undiscovered vertices: \( O(m+n) \)

Properties of (Undirected) BFS(v)

- BFS(v) visits \( x \) if and only if there is a path in \( G \) from \( v \) to \( x \).
- Edges into then-undiscovered vertices define a tree – the "breadth first spanning tree" of \( G \).
- Level \( i \) in this tree are exactly those vertices \( u \) such that the shortest path (in \( G \), not just the tree) from the root \( v \) is of length \( i \).
- All non-tree edges join vertices on the same or adjacent levels
BFS Application: Shortest Paths

Tree gives shortest paths from start vertex. Can label by distances from start.

Depth-First Search

- Follow the first path you find as far as you can
- Back up to last unexplored edge when you reach a dead end, then go as far you can
- Naturally implemented using recursive calls or a stack
- Works on general graphs, not just trees

DFS(v) – Recursive version

Global Initialization:
mark all vertices v "undiscovered" via v.dfs# = -1
dfscounter = 0

DFS(v)
v.dfs# = dfscounter++ // mark v "discovered"
for each edge (v,x)
  if (x.dfs# = -1) // tree edge (x previously undiscovered)
    DFS(x)
  else … // code for back-, fwd-, parent,
          // edges, if needed
// mark v "completed," if needed

Properties of (Undirected) DFS(v)

- Like BFS(v):
  - DFS(v) visits x ⇔ there is a path in G from v to x
    (through previously unvisited vertices)
  - Edges into then-undiscovered vertices define a tree –
    the "depth first spanning tree" of G
- Unlike the BFS tree:
  - the DF spanning tree isn't minimum depth
  - Its levels don't reflect min distance from the root
  - Non-tree edges never join vertices on the same or adjacent levels
- BUT…
Non-tree edges

- All non-tree edges join a vertex and one of its descendents/ancestors in the DFS tree
- Called back/forward edges (depending on end)
- No cross edges!

Application: Articulation Points

- A node in an undirected graph is an articulation point iff removing it disconnects the graph
- Articulation points represent vulnerabilities in a network – single points whose failure would split the network into 2 or more disconnected components

Articulation Points

Exercise

- draw a graph, ~ 10 nodes, A-J
- redraw as via DFS
- add dfs#s & tree/back edges (solid/dashed)
- find cycles
- give alg to find cycles via dfs; does G have any?
- find articulation points
- what do cycles have to do with articulation points?
- alg to find articulation points via DFS???

Articulation Points from DFS

- Root node is an articulation point iff it has more than one child
- Leaf is never an articulation point
- non-leaf, non-root node u is an articulation point ⇔ no non-tree edge goes above u from a sub-tree below some child of u

Articulation Points: the "LOW" function

- Definition: LOW(v) is the lowest dfs# of any vertex that is either in the dfs subtree rooted at v (including v itself) or connected to a vertex in that subtree by a back edge.
- Key idea 1: if some child x of v has LOW(x) ≤ dfs#(v) then v is an articulation point.
- Key idea 2: LOW(v) = min { LOW(w) | w a child of v } U { dfs#(x) | {v,x} is a back edge from v }
Properties of DFS Vertex Numbering

• If \( u \) is an ancestor of \( v \) in the DFS tree, then

\[
\text{dfs#}(u) \leq \text{dfs#}(v).
\]

DFS(v) for Finding Articulation Points

Global initialization: \( v.\text{dfs#} = -1 \) \( \forall v \); DFS(v) \( \forall \)unvisited v.

DFS(v)

\[
v.\text{dfs#} = \text{dfscounter}++
\]

// initialization

for each edge \( (v,x) \)

if \( x.\text{dfs#} == -1 \) // x is undiscovered

DFS(x)

\[
v.\text{low} = \min(v.\text{low}, x.\text{low})
\]

if \( x.\text{low} \geq v.\text{dfs#} \)

print ‘v is art. pt., separating x’

else if \( (x \text{ is not v’s parent}) \)

\[
v.\text{low} = \min(v.\text{low}, x.\text{dfs#})
\]

Equiv: “If (v,x) is a back edge)”

Why?