Undirected Graphs

Representing Graph $G = (V, E)$

- Adjacency List:
  - $O(n+m)$ words
  - $O(\log n)$ bits each

- Advantages:
  - Compact for sparse graphs

Directed Graphs

Representing Graph $G = (V, E)$

- Adjacency List:
  - $O(n+m)$ words
  - $O(\log n)$ bits each

- Advantages:
  - Back pointers and cross pointers allow easier traversal and deletion of edges
  - Usually assume this format
Graph Traversal

- Learn the basic structure of a graph
- Walk from a fixed starting vertex \( v \) to find all vertices reachable from \( v \)
- Three states of vertices
  - undiscovered
  - discovered
  - completely-explored

Breadth-First Search

- Completely explore the vertices in order of their distance from \( v \)
- Naturally implemented using a queue
**BFS analysis**
- Each edge is explored once from each end-point (at most)
- Each vertex is discovered by following a different edge
- Total cost $O(m)$ where $m$ = # of edges

**Graph Search Application: Connected Components**
- Want data structure that allows one to answer questions of the form:
  - given vertices $u$ and $v$ is there a path from $u$ to $v$?
- Idea: create array $A$ such that
  - $A[u]$ = smallest numbered vertex that is connected to $u$
Graph Search Application: Connected Components

```plaintext
for v = 1 to n do
    if state(v) = fully-explored then
        state(v) ← discovered
    BFS(v): setting A[u] ← v for each u found
endfor
```

- Total cost: $O(n+m)$
  - each vertex and each edge is touched a constant number of times
  - works also with DFS

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 go, recording all the vertices you will need to explore further as you go.

- Naturally implemented using recursive calls or a stack
Non-tree edges
- All non-tree edges join a vertex and its descendant in the DFS tree
- 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
Articulation Points

Articulation Points from DFS
- Every interior vertex of a tree is an articulation point
- Non-tree edges eliminate articulation points
- Root nodes are articulation points if they have more than one child

Non-leaf, non-root node $u$ is an articulation point $\iff$ no non-tree edges going from sub-tree below some child of $u$ to above $u$ in the tree

DFS Application: Articulation Points
- root has one child
- non-tree edges matched with vertices they eliminate
- leaves are not articulation points
- articulation points & reasons
  - 3 sub-tree at 4
  - 8 sub-tree at 9
  - 10 sub-tree at 11
  - 12 sub-tree at 13

No non-tree edges going from sub-tree below some child of $u$ to above $u$ in the tree