COM2160/2860: Data Structures

The University of Sydney
School of Information Technologies

Week 9:
- Graph Traversal
- Topological Sort
- Spanning Trees

Graphs - Terminology

- \( G = (V, E) \)
- A graph \( G \) consists of two sets
  -- A set \( V \) of vertices, or nodes
  -- A set \( E \) of edges
- A subgraph
  -- Consists of a subset of a graph's vertices and a subset of its edges
- Adjacent vertices
  -- Two vertices that are joined by an edge

Graphs - Terminology

Terminology

- A path between two vertices
  -- A sequence of edges that begins at one vertex and ends at another vertex
  -- May pass through the same vertex more than once
- A simple path
  -- A path that passes through a vertex only once
- A cycle
  -- A path that begins and ends at the same vertex
- A simple cycle
  -- A cycle that does not pass through a vertex more than once

First Invention of Graphs

- Exists a path in Koenigsberg, such that each each bridge is used only once?
- Euler solved this problem in the year 1736 and founded graph theory.

Proof

- The river divides the town Koenigsberg into 4 districts
- Every district is reachable by an odd number of bridges
- A district that is reachable by an odd number of bridges can only be at the beginning or end of the path
- There can only be at most two districts lying on the beginning or end of a path.

Terminology

- A connected graph
  -- A graph that has a path between each pair of distinct vertices
- A disconnected graph
  -- A graph that has at least one pair of vertices without a path between them
- A complete graph
  -- A graph that has an edge between each pair of distinct vertices
Terminology

- Directed graph
  - Can have two edges between a pair of vertices, one in each direction
- Directed path
  - A sequence of directed edges between two vertices
- Vertex y is adjacent to vertex x if
  - There is a directed edge from x to y

Implementing Graphs

- Most common implementations of a graph
  - Adjacency matrix
  - Adjacency list

Graph Traversals

- A graph-traversal algorithm
  - Visits all the vertices that it can reach
  - Visits all vertices of the graph if and only if the graph is connected
    - A connected component
      - The subset of vertices visited during a traversal that begins at a given vertex
    - Can loop indefinitely if a graph contains a loop
      - To prevent this, the algorithm must
        - Mark each vertex during a visit, and
        - Never visit a vertex more than once

Depth-First Search

- Depth-first search (DFS) traversal
  - Visit vertex then explore all adjacent vertices recursively
  - A last visited, first explored strategy
  - An iterative form uses a queue
  - A recursive form is also possible

Algorithm

```plaintext
function DFS(u)
  if not marked(u) then
    marked(u) = true;
    for all s ∈ succ(u) do
      DFS(s)
  endif
endfunction
```

- Set of successors
  \( \text{succ}(u) = \{ v | (u, v) \in E \} \)
- Initialization
  - for all nodes u
    - marked(u) = false;
    - DFS(start)
Breath-First Search

- Breath-First Search (BFS) traversal
  - Visits neighbors first before exploring depth.
  - A first visited, first explored strategy
  - An iterative form uses a queue
  - A recursive form is possible, but not simple

Algorithm

```java
void BFS(Node u) {
    q = new Queue();
    q.enq(u); marked[u] = true;
    while(!q.isEmpty()) {
        u = q.deq();
        for all v in succ(u)
            if (!marked[v]) {
                q.enq(v);
                marked[v] = true;
            }
    }
}
```

Implementing a BFS Iterator Class Using the JCF

- BFSIterator class uses the ListIterator class
  - As a queue to keep track of the order the vertices should be processed
- BFSIterator constructor
  - Initiates methods used to determine BFS order of vertices for the graph
- Graph is searched by processing vertices from each vertex’s adjacency list
  - In the order that they were pushed onto the queue

Applications of Graphs: Topological Sorting

- Topological order
  - A list of vertices in a directed graph without cycles such that vertex x precedes vertex y if there is a directed edge from x to y in the graph
  - There may be several topological orders in a given graph
- Topological sorting
  - Arranging the vertices into a topological order

Simple algorithms for finding a topological order

- topSort
  - Find a vertex that has no successor
  - Remove from the graph that vertex and all edges that lead to it, and add the vertex to the beginning of a list of vertices
  - Add each subsequent vertex that has no successor to the beginning of the list
  - When the graph is empty, the list of vertices will be in topological order

Figure 13.14
A directed graph without cycles

Figure 13.15
The graph in Figure 13.14 arranged according to the topological orders a), b), c), d), and e). a) a, p, d, b, e, c, f, and b) a, b, g, d, e, f, c.
Topological Sorting

• Simple algorithms for finding a topological order (Continued)
  – topSort2
    • A modification of the iterative DFS algorithm
    • Strategy
      – Push all vertices that have no predecessor onto a stack
      – Each time you pop a vertex from the stack, add it to the
        beginning of a list of vertices
      – When the traversal ends, the list of vertices will be in
        topological order

Algorithm for Top-Sort

• Find start node with no successor

Algorithm

function TopSort(u)
  if not marked(u) then
    marked(u) = true;
    for all s ∈ succs(u)
      TopSort(s)
    endfor
    order[i]=u;
    i=i-1;
  endif
endfunction

Spanning Trees

• A tree
  – An undirected connected graph without cycles
• A spanning tree of a connected undirected graph G
  – A subgraph of G that contains all of G’s vertices and enough
    of its edges to form a tree
• To obtain a spanning tree from a connected
  undirected graph with cycles
  – Remove edges until there are no cycles

Spanning Trees

• You can determine whether a connected graph
  contains a cycle by counting its vertices and edges
  – A connected undirected graph that has n vertices must have
    at least n – 1 edges
  – A connected undirected graph that has n vertices and
    exactly n – 1 edges cannot contain a cycle
  – A connected undirected graph that has n vertices and more
    than n – 1 edges must contain at least one cycle

Spanning Trees

Figure 13.19
Connected graphs that each have four vertices and three edges

The DFS Spanning Tree

• To create a depth-first search (DFS) spanning tree
  – Traverse the graph using a depth-first
    search and mark the edges that you follow
  – After the traversal is complete, the graph’s
    vertices and marked edges form the
    spanning tree
The BFS Spanning Tree

- To create a breadth-first search (BFS) spanning tree
  - Traverse the graph using a breadth-first search and mark the edges that you follow
  - When the traversal is complete, the graph’s vertices and marked edges form the spanning tree

Minimum Spanning Tree

- Minimum spanning tree
  - A spanning tree for which the sum of its edge weights is minimal
- Prim’s algorithm
  - Finds a minimal spanning tree that begins at any vertex
  - Strategy
    - Find the least-cost edge (v, u) from a visited vertex v to some unvisited vertex u
    - Mark u as visited
    - Add the vertex u and the edge (v, u) to the minimum spanning tree
    - Repeat the above steps until there are no more unvisited vertices

Shortest Paths

- Shortest path between two vertices in a weighted graph
  - The path that has the smallest sum of its edge weights
- Dijkstra’s shortest-path algorithm
  - Determines the shortest paths between a given origin and all other vertices
  - Uses
    - A set vertexSet of selected vertices
    - An array weight, where weight[v] is the weight of the shortest (cheapest) path from vertex 0 to vertex v that passes through vertices in vertexSet

Circuits

- A circuit
  - A special cycle that passes through every vertex (or edge) in a graph exactly once
- Euler circuit
  - A circuit that begins at a vertex v, passes through every edge exactly once, and terminates at v
  - Exists if and only if each vertex touches an even number of edges

Figure 14-27
a) Euler’s bridge problem
b) Its multigraph representation

Some Difficult Problems

- Three applications of graphs
  - The traveling salesperson problem
  - The three utilities problem
  - The four-color problem
- A Hamilton circuit
  - Begins at a vertex v, passes through every vertex exactly once, and terminates at v

Summary

- The two most common implementations of a graph are the adjacency matrix and the adjacency list
- Graph searching
  - Depth-first search goes as deep into the graph as it can before backtracking
  - Breadth-first search visits all possible adjacent vertices before traversing further into the graph
- Topological sorting produces a linear order of the vertices in a directed graph without cycles
Summary

- Trees are connected undirected graphs without cycles
  - A spanning tree of a connected undirected graph is a subgraph that contains all the graph's vertices and enough of its edges to form a tree
  - A minimum spanning tree for a weighted undirected graph is a spanning tree whose edge-weight sum is minimal
  - The shortest path between two vertices in a weighted directed graph is the path that has the smallest sum of its edge weights

- An Euler circuit in an undirected graph is a cycle that begins at vertex v, passes through every edge in the graph exactly once, and terminates at v
- A Hamilton circuit in an undirected graph is a cycle that begins at vertex v, passes through every vertex in the graph exactly once, and terminates at v

Graph Terminology
- Directed graphs
- Undirected graphs

Graph Implementation
- Adjacency Matrix
- Adjacent List

Graph searching
- Depth-first search goes as deep into the graph as it can before backtracking
- Breath-first search visits all possible adjacent vertices before traversing further into the graph
- Topological sorting produces a linear order of the vertices in a directed graph without cycles
- Trees are connected undirected graphs without cycles
  - A spanning tree of a connected undirected graph is a subgraph that contains all the graph's vertices and enough of its edges to form a tree
  - A minimum spanning tree for a weighted undirected graph is a spanning tree whose edge-weight sum is minimal

Summary