Eric Roberts
CS 106B

Handout #45
February 27, 2013

Graph Algorithms

Graph Algorithms

Eric Roberts
CS 106B
February 27, 2013

Outline

1. A review the **graphtypes.h** and **graph.h** interfaces
2. A tour of the Pathfinder assignment
3. Examples of depth-first and breadth-first search
4. Dijkstra's shortest-path algorithm
5. Kruskal's minimum-spanning-tree algorithm

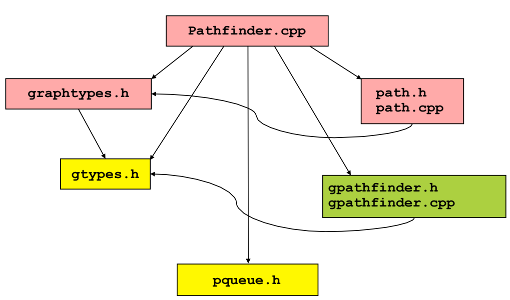
The **Node** and **Arc** Structures

```
struct Node;     /* Forward references to these two types so  */
struct Arc;      /* that the C++ compiler can recognize them. */

/*
 * Type: Node
 * -----------
 * This type represents an individual node and consists of the
 * name of the node and the set of arcs from this node.
 */

struct Node {
   string name;
   Set<Arc *> arcs;
};

/*
 * Type: Arc
 * ----------
 * This type represents an individual arc and consists of pointers
 * to the endpoints, along with the cost of traversing the arc.
 */

struct Arc {
   Node *start;
   Node *finish;
   double cost;
};
```

Entries in the **graph.h** Interface

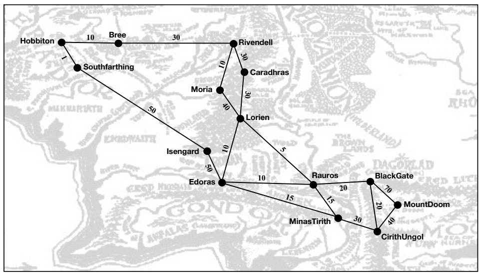
```
template <typename NodeType,typename ArcType>
class Graph {
public:
   Graph();
   ~Graph();

   void clear();

   NodeType *addNode(string name);
   NodeType *addNode(NodeType *node);

   ArcType *addArc(string s1, string s2);
   ArcType *addArc(NodeType *n1, NodeType *n2);
   ArcType *addArc(ArcType *arc);

   bool isConnected(NodeType *n1, NodeType *n2);
   bool isConnected(string s1, string s2);

   NodeType *getNode(string name);

   Set<NodeType *> & getNodeSet();
   Set<ArcType *> & getArcSet();
   Set<ArcType *> & getArcSet(NodeType *node);
};
```

Modules in the Pathfinder Assignment



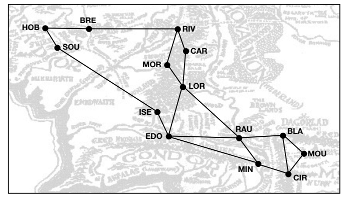
Frodo's Journey

The Middle Earth Graph



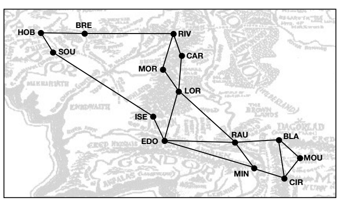
Exercise: Depth-First Search

Construct a depth-first search starting from Hobbiton (**HOB**):



Exercise: Breadth-First Search

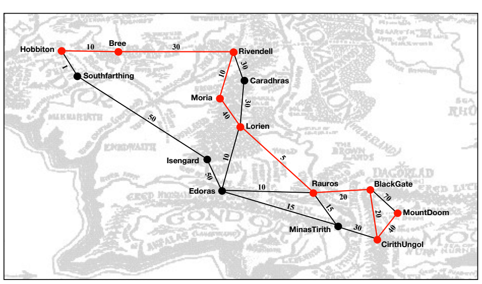
Construct a breadth-first search starting from Isengard (**ISE**):



Dijkstra's Algorithm

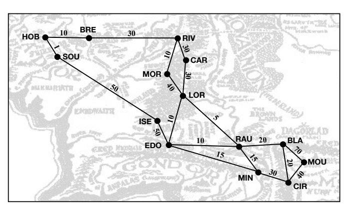
- One of the most useful algorithms for computing the shortest paths in a graph was developed by Edsgar W. Dijkstra in 1959.
- The strategy is similar to the breadth-first search algorithm you used to implement the word-ladder program in Assignment #2. The major difference are:
  - The queue used to hold the paths delivers items in increasing order of total cost rather than in the traditional first-in/first-out order. Such queues are called ***priority queues***.
  - The algorithm keeps track of all nodes to which the total distance has already been fixed. Distances are fixed whenever you dequeue a path from the priority queue.

Shortest Path



Exercise: Dijkstra's Algorithm

Find the shortest path from Hobbiton (**HOB**) to Lorien (**LOR**):



Kruskal's Algorithm

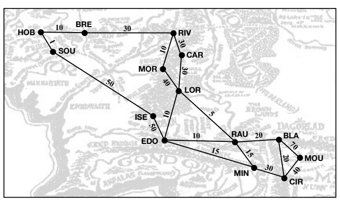
- In many cases, finding the shortest path is not as important as as minimizing the cost of a network as a whole. A set of arcs that connects every node in a graph at the smallest possible cost is called a ***minimum spanning tree***.
- The following algorithm for finding a minimum spanning tree was developed by Joseph Kruskal in 1956:
  - Start with a new empty graph with the same nodes as the original one but an empty set of arcs.
  - Sort all the arcs in the graph in order of increasing cost.
  - Go through the arcs in order and add each one to the new graph if the endpoints of that arc are not already connected by a path.
- This process can be made more efficient by maintaining sets of nodes in the new graph, as described on the next slide.

Combining Sets in Kruskal's Algorithm

- Implementing the Pathfinder version of Kruskal's algorithm requires you need to build a new graph containing the spanning tree. As you do, you will generate sets of disconnected graphs.
- When you choose a new arc, there are four possibilities for the sets formed by the nodes at the endpoints:
  1. *Neither node is yet in a set.* In this case, create a new set and add both nodes to it.
  2. *One node is in a set and the other isn't.* In this case, add the new node to the same set.
  3. *The endpoints are in different existing sets.* In this case, you need to merge the two sets to create a new one containing the union of the existing ones.
  4. *The endpoints are in the same set.* In this case, there is already a path between these two nodes, so you don't need this arc.

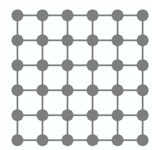
Exercise: Minimum Spanning Tree

Apply Kruskal's algorithm to find a minimum spanning tree:



An Application of Kruskal's Algorithm

- Suppose that you have a graph that looks like this:



- What would happen if you applied Kruskal's algorithm for finding a minimum spanning tree, assuming that you choose the arcs in a random order?