

Graph-theoretic Models, Lecture 3, Segment 2

John Guttag

MIT Department of Electrical Engineering and
Computer Science

Class Node

```
class Node(object):
    def __init__(self, name):
        """Assumes name is a string"""
        self.name = name
    def getName(self):
        return self.name
    def __str__(self):
        return self.name
```

Class Edge

```
class Edge(object):
    def __init__(self, src, dest):
        """Assumes src and dest are nodes"""
        self.src = src
        self.dest = dest
    def getSource(self):
        return self.src
    def getDestination(self):
        return self.dest
    def __str__(self):
        return self.src.getName() + '->\n' + self.dest.getName()
```

Common Representations of Digraphs

- Adjacency matrix
 - Rows: source nodes
 - Columns: destination nodes
 - $\text{Cell}[s, d] = 1$ if there is an edge from s to d
0 otherwise
- Adjacency list
 - Associate with each node a list of destination nodes

Class Digraph, part 1

```
class Digraph(object):
    """edges is a dict mapping each node to a list of
    its children"""

    def __init__(self):
        self.edges = {}

    def addNode(self, node):
        if node in self.edges:
            raise ValueError('Duplicate node')
        else:
            self.edges[node] = []

    def addEdge(self, edge):
        src = edge.getSource()
        dest = edge.getDestination()
        if not (src in self.edges and dest in self.edges):
            raise ValueError('Node not in graph')
        self.edges[src].append(dest)
```

Class Digraph, part 2

```
def childrenOf(self, node):
    return self.edges[node]

def hasNode(self, node):
    return node in self.edges

def getNode(self, name):
    for n in self.edges:
        if n.getName() == name:
            return n
    raise NameError(name)

def __str__(self):
    result = ''
    for src in self.edges:
        for dest in self.edges[src]:
            result = result + src.getName() + '->'\
                + dest.getName() + '\n'
    return result[:-1] #omit final newline
```

Class Graph

```
class Graph(Digraph):  
    def addEdge(self, edge):  
        Digraph.addEdge(self, edge)  
        rev = Edge(edge.getDestination(), edge.getSource())  
        Digraph.addEdge(self, rev)
```

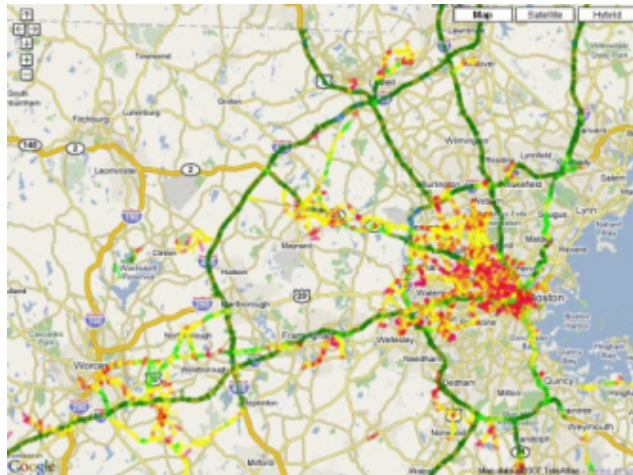
- Why is Graph a subclass of digraph?
- Remember the substitution rule from 6.00.1x?
 - If client code works correctly using an instance of the supertype, it should also work correctly when an instance of the subtype is substituted for the instance of the supertype
- Any program that works with a Digraph will also work with a Graph (but not *vice versa*)

A Classic Graph Optimization Problem

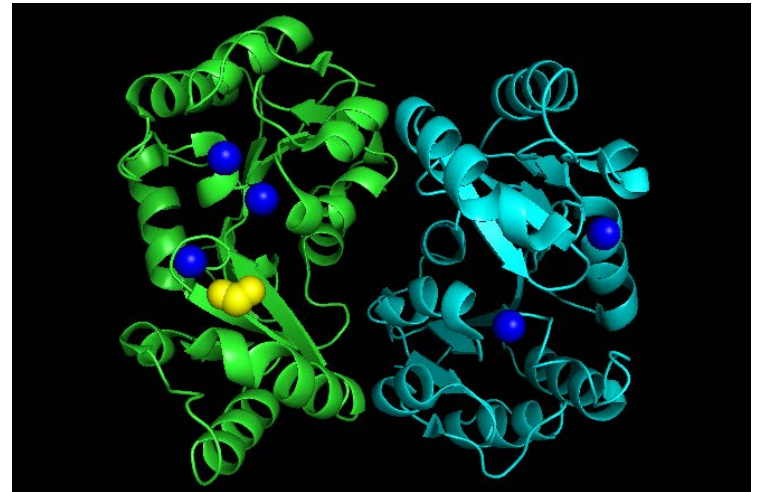
- Shortest path from n_1 to n_2
 - Shortest sequence of edges such that
 - Source node of first edge is n_1
 - Destination of last edge is n_2
 - For edges, e_1 and e_2 , in the sequence, if e_2 follows e_1 in the sequence, the source of e_2 is the destination of e_1
- Shortest weighted path
 - Minimize the sum of the weights of the edges in the path

Some Shortest Path Problems

- Finding a route from one city to another
- Designing communication networks
- Finding a path for a molecule through a chemical labyrinth
- ...

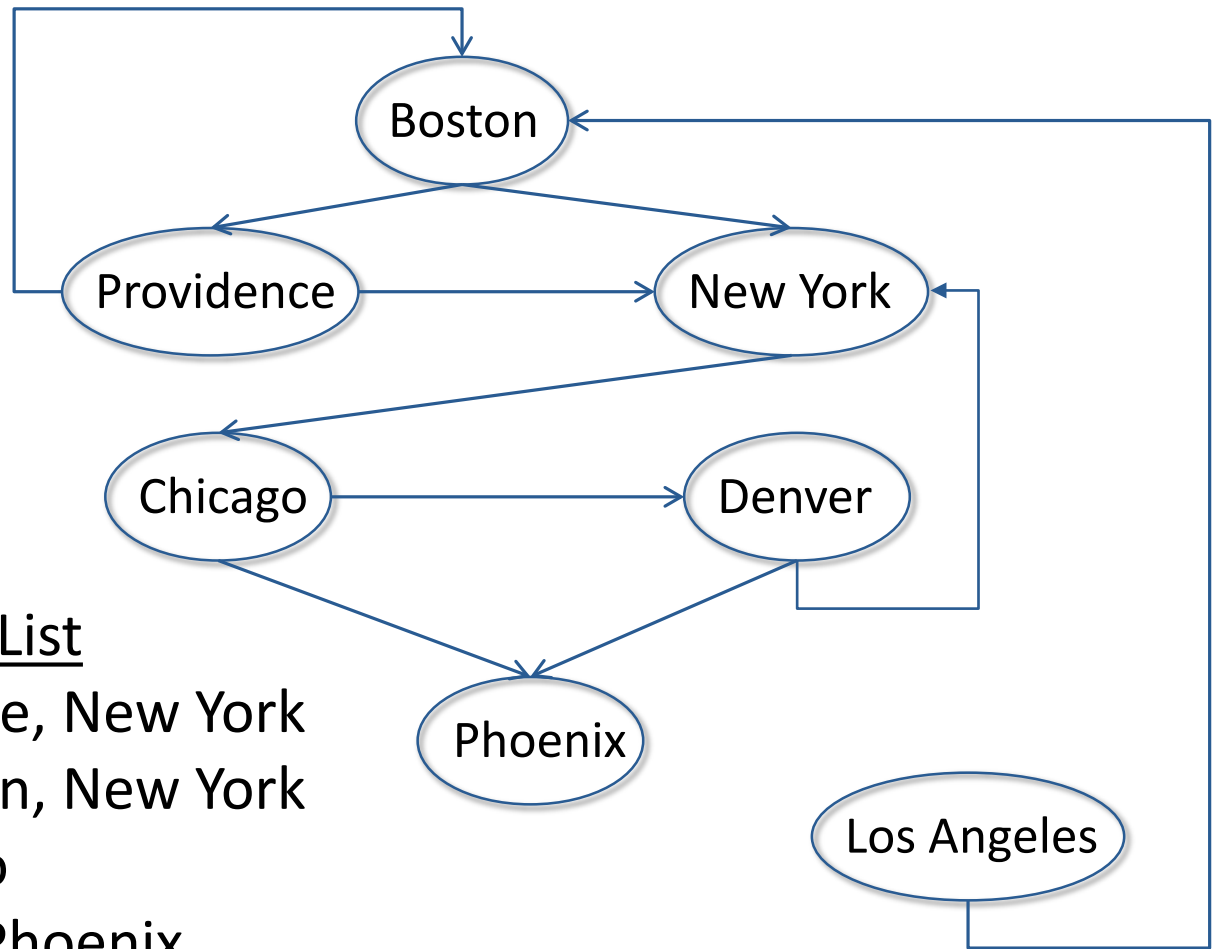


www.google.com



CC-BY Juliaytsai94

An Example



Adjacency List

Boston: Providence, New York

Providence: Boston, New York

New York: Chicago

Chicago: Denver, Phoenix

Denver: Phoenix, New York

Los Angeles: Boston

Build the Graph

```
def buildCityGraph():
    g = Digraph()
    for name in ('Boston', 'Providence', 'New York', 'Chicago',
                 'Denver', 'Phoenix', 'Los Angeles'): #Create 7 nodes
        g.addNode(Node(name))
    g.addEdge(Edge(g.getNode('Boston'), g.getNode('Providence')))
    g.addEdge(Edge(g.getNode('Boston'), g.getNode('New York')))
    g.addEdge(Edge(g.getNode('Providence'), g.getNode('Boston')))
    g.addEdge(Edge(g.getNode('Providence'), g.getNode('New York')))
    g.addEdge(Edge(g.getNode('New York'), g.getNode('Chicago')))
    g.addEdge(Edge(g.getNode('Chicago'), g.getNode('Denver')))
    g.addEdge(Edge(g.getNode('Denver'), g.getNode('Phoenix')))
    g.addEdge(Edge(g.getNode('Denver'), g.getNode('New York')))
    g.addEdge(Edge(g.getNode('Chicago'), g.getNode('Phoenix')))
    g.addEdge(Edge(g.getNode('Los Angeles'), g.getNode('Boston')))
```

Coming Up

- Solutions to shortest path problem