Principles of Autonomy & Decision Making

16.410/16.413

Java Tutorial

Useful Reference

- If you need to learn Java syntax:
  - Sun Java Tutorial
  - http://java.sun.com/docs/books/tutorial/

- If you want to know about available packages:
  - http://java.sun.com/javase/6/docs/api/

- You can find a link to these pages from the course website under “Materials”

Graph

- Directed Graph
  — A directed graph or digraph G is an ordered pair G:= (V, E) with
  - V is a set, whose elements are called vertices or nodes,
  - E is a set of ordered pairs of vertices, called directed edges, arcs, or arrows.

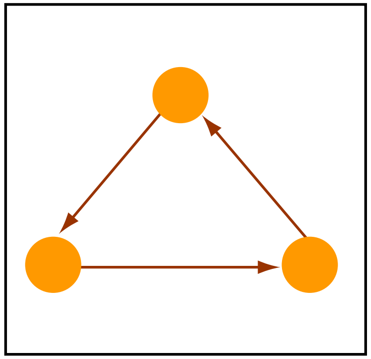

Image by MIT OpenCourseWare.

- Undirected Graph
  — An undirected graph G is an ordered pair G:= (V, E) that is subject to the following conditions:
  - V is a set, whose elements are called vertices or nodes,
  - E is a set of pairs (unordered) of distinct vertices, called edges or lines.

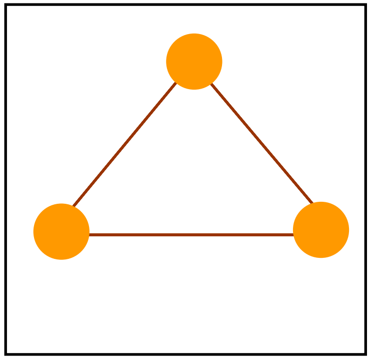

Image by MIT OpenCourseWare.

Graph as Adjacency Matrix

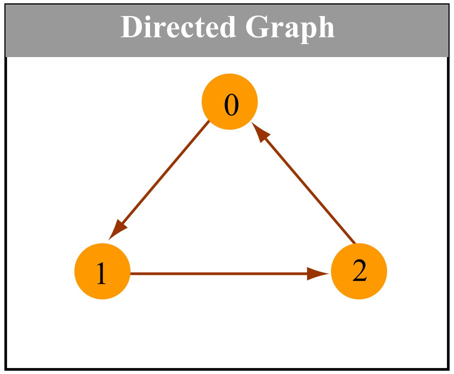

Image by MIT OpenCourseWare.

	0	1	2
0	false	true	false
1	false	false	true
2	true	false	false

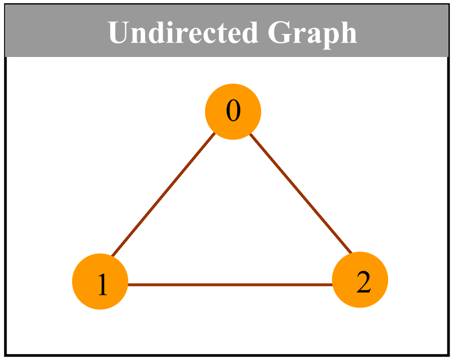

Image by MIT OpenCourseWare.

	0	1	2
0	false	true	true
1	true	false	true
2	true	true	false

Let’s Begin Coding

- Graph Class (i.e. Directed Graph)
  - Member Variables: Data stored in the object
    - **protected boolean** m_edges [][];
  - Constructors: How the object should be initialized
    - **public** Graph()
    - **public** Graph(**int** vertexCount);
  - Methods: Available operations on the object
    - **public void** addEdge(**int** from, **int** to)
    - **public void** deleteEdge(**int** from, **int** to)
    - **public boolean** isConnected(**int** from, **int** to)
    - **public** Set<Integer>getAdjacentVertices(**int** from)

Difference between Directed and Undirected

- UndirectedGraph Class
  - Member Variables: Data stored in the object
    - **protected boolean** m_edges[][];
  - Constructors: How the object should be initialized
    - **public** Graph()
    - **public** Graph(**int** vertexCount)
  - Methods: Available operations on the object
    - **public** void addEdge(**int** from, **int** to)
    - **public** void deleteEdge(int from, **int** to)
    - **public** boolean isConnected(**int** from, **int** to)
    - **public** Set<Integer> getAdjacentVertices(**int** from)

Difference between Directed and Undirected

- UndirectedGraph Class
  - Member Variables: Data stored in the object
    - **protected boolean** m_edges[][];
  - Constructors: How the object should be initialized
    - **public** Graph()
    - **public** Graph(**int** vertexCount)
  - Methods: Available operations on the object
    - **public** void addEdge(**int** from, **int** to)
    - **public** void deleteEdge(int from, **int** to)
    - **public** boolean isConnected(**int** from, **int** to)
    - **public** Set<Integer> getAdjacentVertices(**int** from)

Inheritance

- Let UndirectedGraph inherit Graph
  - Only implement the methods that are different.
  - The undefined methods will be inherited from the Graph class.

```
public class UndirectedGraph extends Graph {
        public Graph(int vertexCount) {
        }
        public void addEdge(int from, int to) {
        }
        public void deleteEdge(int from, int to) {
        }
}
```

Test Cases

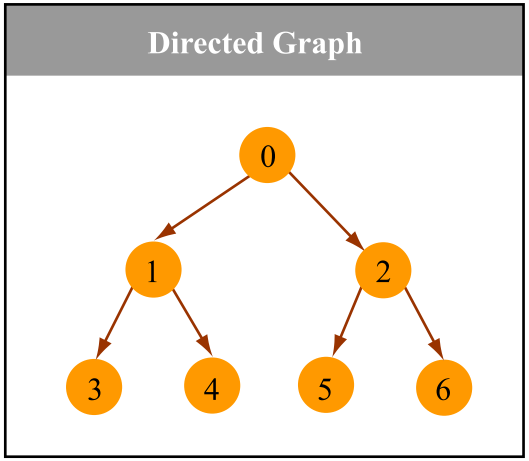

Image by MIT OpenCourseWare.

What is adjacent to 2?

— (5, 6)

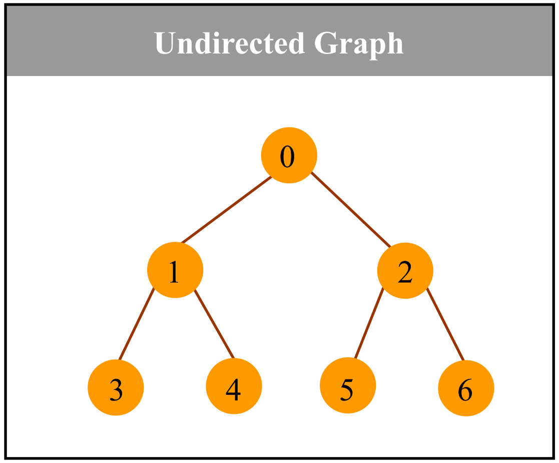

Image by MIT OpenCourseWare.

What is adjacent to 2?

— (0, 5, 6)

What You Should Know

- Basics of Programming
- Basic Object Oriented Programming:
  - Inheritance
  - Abstract Class
    - Some methods may be specified, but not implemented.
  - Interface
    - Methods are specified, but not implemented.

What You Should Know

- Collections (i.e. Containers)
  - Set <T>
    - Unordered collection of elements, without duplicates.
  - List <T>
    - Ordered collection of elements.
  - Queue <T>
    - Allow adding elements to the back and removing from the front.
  - Stack <T>
    - Allow pushing elements to the top and popping from the top.
- Templates
  - Allows the user to specify the object type of the elements, e.g. Set<Integer> is a set of integers.

16.410 / 16.413 Principles of Autonomy and Decision Making
Fall 2010