AP Computer Science — Project 35b - Outline Main Classes

Dr. Paul L. Bailey — Thursday, March 26, 2020

A *graph* is a pair $(V, \mathcal{E})$, where V is a set whose members are called *vertices*, and $\mathcal{E}$ is a set whose members are called *edges*, where an *edge* is a subset of V which contains exactly two (distinct) vertices.

We view the vertices as points and the edges as line segments between them; however, this viewpoint only models the abstract notion of graph.

Two vertices are *adjacent* is they are contained in the same edge. Adjacent vertices are called *neighbors*. The *degree* of a vertex is the number of adjacent vertices.

A *subgraph* of $(V, \mathcal{E})$ is a graph $(W, \mathcal{F})$ such that $W \subset V$ and $\mathcal{F}\mathcal{E}$. Note that if $(W, \mathcal{F})$ is a subgraph of $(V, \mathcal{W})$, the edges in $\mathcal{F}$ must contain only vertices that come from W.

A *walk* in a graph $(V, \mathcal{E})$ is a finite sequence of at least three vertices such that their exists an edge between consecutive vertices. We say that the walk *visits* each of the vertices in the sequence, and that it *traverses* each edge between consecutive vertices. The *length* of the walk is the number of vertices visited, minus one. The first vertex in the sequence is called the *initial* vertex of the walk, and the last is called the *terminal* vertex of the walk.

The graph is *connected* if their exists a walk between any two vertices. A *component* of a graph is a maximal connected subgraph.

Your assignment is to build three low-level classes that can be used to model a graph, and are effective is writing software to determine things such as, "is the graph connected?"

- Each vertex should have a unique label, to identity it.
  Let the label be an integer. Call the label an "id".
- The same vertex may not occur twice in the same graph.
- An edge may only contain vertices in the graph.
- An edge may not connect a vertex to itself.
- The same edge may not occur twice in the same graph.

Outline three classes:

- `public class Vertex`
- `public class Edge`
- `public class Graph`

Write initialization code that obeys these rules:

- Each vertex should have a unique label, to identity it. Let the label be and integer.
- The same vertex may not occur twice in the same graph.
- An edge may only contain vertices in the graph.
- An edge may not connect a vertex to itself.
- The same edge may not occur twice in the same graph.

Include the following methods.

```
public class Vertex
{
    public Vertex(int a)
    public String toString()
    public boolean equals(Vertex v)
    public int hashCode()
}

public class Edge
{
    public Edge(Vertex v, Vertex w)
    public Edge(int a, int b)
    public boolean involves(Vertex u) // Is the vertex in the edge
    public Vertex other(Vertex u) // Get the other vertex in the edge
    public String toString()
    public boolean equals(Edge e)
    public int hashCode()
}

public class Graph
{
    public String toString()
    public int sizeV() // Number of vertices
    public int sizeE() // Number of edges
    public boolean add(Vertex v, List<Vertex> L)
    public boolean add(Vertex v)
    public boolean add(int a) // Make a vertex with this id and add it
    public boolean add(Edge e)
    public boolean add(int a, int b)
    public boolean contains(Vertex v)
    public boolean contains(Edge e)
    public void print()
    public int degree(Vertex v)
}
```