Module 13 Assignment: Graph Traversals

Marcar esta página

Coding Assignment fecha límite Nov 9, 2023 01:01 -05 Vencidos

Graph Traversals

For this assignment, you will be implementing the **Breadth-First Search** and **Depth-First Search** graph traversal algorithms. This assignment has quite a few files in it, so please be sure to read **ALL** of the documentation given to you.

IMPORTANT:

- **You will be given 5 attempts on this assignment, with a 30 minute cooldown between submissions.**
- **Please run your code before each submission to ensure that there are no formatting errors! If there are formatting errors in your code, your code will not be graded and a submission attempt will be logged. For more information, please review the Vocareum overview below.**

Graph Data Structure

You are provided a **Graph** class. The important methods to note from this class are:

- **getVertices** provides a Set of **Vertex** objects (another class provided to you) associated with a graph.
- **getAdjList** provides a Map that maps **Vertex** objects to Lists of **VertexDistance** objects. This Map is especially important for traversing the graph, as it will efficiently provide you the edges associated with any vertex. For example, consider an adjacency list map where vertex A is associated with a list that includes a **VertexDistance** object with vertex B and distance 2 and another **VertexDistance** object with vertex C and distance 3. This implies that in this graph, there is an edge from vertex A to vertex B of weight 2 and another edge from vertex A to vertex C of weight 3.

Vertex Distance Data Structure

In the **Graph** class, you will be using the **VertexDistance** class implementation that we have provided. This data structure is used by the adjacency list to represent which vertices a vertex is connected to.

Search Algorithms

Breadth-First Search is a search algorithm that visits vertices in order of "level", visiting all vertices one edge away from start, then two edges away from start, etc. Similar to levelorder traversal in BSTs, it utilizes a Queue data structure.

Depth-First Search is a search algorithm that visits vertices in a depth based order. In your implementation of DFS, you will be using the recursive stack rather than a Stack data structure. It searches along one path of vertices from the start vertex and backtracks once it hits a dead end or a visited vertex until it finds another path to continue along. **Your implementation of DFS must be recursive.**

Self-Loops and Parallel Edges

In this framework, self-loops and parallel edges work as you would expect. If you recall, self-loops are edges from a vertex to itself. Parallel edges are multiple edges with the same orientation between two vertices. These cases are valid test cases, and you should expect them to be tested. However, most implementations of these algorithms handle these cases automatically, so you shouldn't have to worry too much about them when implementing the algorithms.

General Tips

- Keeping track of visited vertices is important. It allows for your traversal to be efficient and to avoid traversing through cycles over and over again. Using a Set to store visited vertices will allow for constant time look up, versus a list where searching is done in linear time.
- Your BFS implementation should utilize three data structures, while your DFS implementation should utilize two.
- We highly recommend copying the starter code and working in your preferred IDE in order to have access to features such as code completion, auto-formatting, and much more!

Here are general assignment guidelines that should be followed.

- Do not include any package declarations in your classes.
- Do not change any existing class headers, constructors, instance/global variables, or method signatures. For example, do not add throws to the method headers since they are not necessary. Instead, exceptions should be thrown as follows: `throw new InsertExceptionHere("Error: some exception was thrown");`
- All helper methods you choose to write should be made private. Recall the idea of Encapsulation in Object-Oriented Programming!
- Do not use anything that would trivialize the assignment. (e.g. Don't import/use java.util.ArrayList for an ArrayList assignment.)
- Always be very conscious of efficiency. Even if your method is to be $O(n)$, traversing the structure multiple times is considered inefficient unless that is absolutely required (and that case is extremely rare).
- If applicable, use the generic type of the class; do not use the raw type of the class. For example, use `new LinkedList<Integer>()` instead of `new LinkedList()`.

Use of the following statements should be avoided at all times.

package	System.arraycopy()	clone()
assert()	Arrays class	Array class
Thread class	Collections class	Collection.toArray()
Reflection APIs	Inner or nested classes	Lambda Expressions

The Vocareum (code editor) interface has six main components:

- The **Drop-Down** in the top left. This lets you choose from multiple available files. Note that this drop-down will only be visible in assignments that require multiple files.
- The **Run** button. This will compile your code and run a file scan. Running your code will not count towards your total allowed submission attempts, therefore you are free to run as many times as needed.

- The **Submit** button. This will compile your code, run a file scan, grade your assignment, and output results to console. Note that for most assignments in this class, you will only be allowed a limited number of submissions. A submission is counted when the submit button is clicked, regardless of whether or not your code can compile or if there are any file issues. Therefore, we **highly recommend** that you run your code before submitting to ensure that there are no issues that will prevent your code from being graded and that every submission attempt will generate meaningful results.
- The **Reset** button. This will revert all your changes and reset your code to the default code template.
- The **Code Window**. This is where you will write your code. For large coding assignments, we highly recommend copying the starter code and working in your preferred IDE to have access to features such as code completion, auto-formatting, and much more!
- The **Output Window**. This window will appear whenever you run or submit your code and will display the output for you to view.

For additional help, please visit the Vocareum information page located in the course information module!

‹ Anterior Siguiente ›