



Graph Traversals Coding

🔖 Marcar esta página

Graph Traversals Coding

For this coding question, you will be tasked with implementing **Breadth-First Search**. The structure of this coding question will be similar to that of the coding assignments in the course. Here are a few important differences between the coding assignments and this coding question:

- Rather than implementing multiple graph traversal algorithms, you will only need to implement Breadth-First Search.
- The cooldown period between submissions has been removed.

IMPORTANT:

- **You will be given unlimited attempts on this assignment, with no 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.**

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

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!

Graph Traversals Coding (External resource)

GraphAlgorithms.java

Submit

Run

Grades

Reset

```
37  * @param start the vertex to begin the bfs on.
38  * @param graph The graph to search through.
39  * @return List of vertices in visited order.
40  */
41  public static <T> List<Vertex<T>> bfs(Vertex<T> start, Graph<T> graph) {
42      // WRITE YOUR CODE HERE (DO NOT MODIFY METHOD HEADER)!
43      if (graph == null) {
44          throw new IllegalArgumentException("Cannot search null graph.");
45      } else if (!graph.getVertices().contains(start)) {
46          throw new IllegalArgumentException(
47              "Graph does not contain start vertex");
48      }
49      List<Vertex<T>> list = new ArrayList<>();
50      Set<Vertex<T>> VisitedSet = new HashSet<>();
51      Queue<Vertex<T>> queue = new ArrayDeque<>();
52      VisitedSet.add(start);
53      queue.add(start);
54      while (!queue.isEmpty()) {
55          Vertex<T> v = queue.remove();
56          list.add(v);
57          for (VertexDistance<T> verticeDistancia : graph.getAdjList().get(v)){
58              Vertex<T> w = verticeDistancia.getVertex();
59              if (!VisitedSet.contains(w)){
60                  queue.add(w);
61                  VisitedSet.add(w); // Add w to VisitedSet
62              }
63          }
64      }
65      return list;
66  }
67  }
68  }
```

[Executed at: Sun Nov 19 14:28:02 PST 2023]

=====

GraphAlgorithms.java successfully compiled.

=====

Success: All Tests Passed.

=====

Score: 10.0 / 10.0

=====

< Anterior

Siguiente >

© Todos los Derechos están Reservados



edX

Acerca de
Afiliados
edX para negocios
Open edX
Carreras
Noticias

Legal

Condiciones de Servicio y Código
de Honor
Política de privacidad
Políticas de Accesibilidad
Política de marcas
Mapa del Sitio
Política de cookies
Opciones de privacidad

Contáctanos

Centro de ideas
Contáctenos
Centro de Ayuda
Seguridad
Kit Multimedia



© 2023 edX LLC. All rights reserved.
深圳市恒宇博科技有限公司 粤ICP备
17044299号-2