Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

- CallBack< E > ................................................................. 5
- PrintCallBack< E > ......................................................... 8
- Graph< E > ................................................................. 5
Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

CallBack\(<\ E\>\) .............................................................. 5
Graph\(<\ E\>\) ............................................................... 5
PrintCallBack\(<\ E\>\) ..................................................... 8
Chapter 3

Class Documentation

3.1 CallBack< E > Interface Reference

Inheritance diagram for CallBack< E >:

```
CallBack< E >
   |
   v
PrintCallBack< E >
```

3.1.1 Detailed Description

Represents the processing we apply to a vertex of a graph.

The documentation for this interface was generated from the following file:

- CallBack.java

3.2 Graph< E > Class Reference

Public Member Functions

- Graph ()
- void addVertex (String vertexName, E data)
- void addDirectedEdge (String startVertexName, String endVertexName, int cost)
- String toString ()
- Map<String, Integer> getAdjacentVertices (String vertexName)
- int getCost (String startVertexName, String endVertexName)
- Set<String> getVertices ()
- E getData (String vertex)
- void doDepthFirstSearch (String startVertexName, CallBack< E > callback)
- void doBreadthFirstSearch (String startVertexName, CallBack< E > callback)
- int doDijkstras (String startVertexName, String endVertexName, ArrayList<String> shortestPath)
3.2.1 Detailed Description

Implements a graph. We use two maps: one map for adjacency properties (adjancencyMap) and one map (dataMap) to keep track of the data associated with a vertex.

Author

cmsc132

Parameters

\[ E \]

3.2.2 Constructor & Destructor Documentation

3.2.2.1 Graph()

Initializes the adjacency and data maps.

3.2.3 Member Function Documentation

3.2.3.1 void addDirectedEdge ( String startVertexName, String endVertexName, int cost )

Adds or updates a directed edge with the specified cost.

Parameters

\begin{itemize}
  \item \texttt{startVertexName} \quad \text{vertex's name}
  \item \texttt{endVertexName} \quad \text{vertex's name}
  \item \texttt{cost} \quad \text{cost}
\end{itemize}

Exceptions

\begin{itemize}
  \item \textit{IllegalArgumentException} \quad If any of the vertices are not part of the graph. Use any error message.
\end{itemize}

3.2.3.2 void addVertex ( String vertexName, E data )

Adds a vertex to the graph by adding to the adjacency map an entry for the vertex. This entry will be an empty map. An entry in the dataMap will store the provided data.

Parameters

\begin{itemize}
  \item \texttt{vertexName} \quad \text{vertex's name}
  \item \texttt{data} \quad \text{data associated with the vertex}
\end{itemize}

Exceptions

\begin{itemize}
  \item \textit{IllegalArgumentException} \quad If the vertex already exists in the graph. Use any error message.
\end{itemize}

3.2.3.3 void doBreadthFirstSearch ( String startVertexName, CallBack< E > callback )

Computes Breadth-First Search of the specified graph.
3.2 Graph< E > Class Reference

Parameters

<table>
<thead>
<tr>
<th>startVertexName</th>
<th>Represents the processing to apply to each vertex</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td></td>
</tr>
</tbody>
</table>

Exceptions

| IllegalArgumentException | If the vertex is not part of the graph. Use any error message. |

3.2.3.4 void doDepthFirstSearch ( String startVertexName, CallBack< E > callback )

Computes Depth-First Search of the specified graph.

Parameters

<table>
<thead>
<tr>
<th>startVertexName</th>
<th>Represents the processing to apply to each vertex</th>
</tr>
</thead>
<tbody>
<tr>
<td>callback</td>
<td></td>
</tr>
</tbody>
</table>

Exceptions

| IllegalArgumentException | If the vertex is not part of the graph. Use any error message. |

3.2.3.5 int doDijkstras ( String startVertexName, String endVertexName, ArrayList< String > shortestPath )

Computes the shortest path and shortest path cost using Dijkstra's algorithm. It initializes shortestPath with the names of the vertices corresponding to the shortest path. If there is no shortest path, shortestPath will be have entry "None".

Parameters

<table>
<thead>
<tr>
<th>startVertexName</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>endVertexName</td>
<td></td>
</tr>
<tr>
<td>shortestPath</td>
<td>Initialized by the method with the shortest path or &quot;None&quot;.</td>
</tr>
</tbody>
</table>

Returns

Shortest path cost or -1 if no path exist.

Exceptions

| IllegalArgumentException | If any of the vertices are not part of the graph. Use any error message. |

3.2.3.6 Map< String, Integer > getAdjacentVertices ( String vertexName )

Returns a map with information about vertices adjacent to vertexName. If the vertex has no adjacents, an empty map is returned.

Parameters

<table>
<thead>
<tr>
<th>vertexName</th>
</tr>
</thead>
</table>

Returns

map

3.2.3.7 int getCost ( String startVertexName, String endVertexName )

Returns the cost associated with the specified edge.
Parameters

<table>
<thead>
<tr>
<th>startVertexName</th>
</tr>
</thead>
<tbody>
<tr>
<td>endVertexName</td>
</tr>
</tbody>
</table>

Returns

edge cost

Exceptions

IllegalArgumentException | If any of the vertices are not part of the graph. Use any error message.

3.2.3.8 E getData ( String vertex )

Returns the data component associated with the specified vertex.

Parameters

| vertex |

Returns

data

Exceptions

IllegalArgumentException | If the vertex is not part of the graph. Use any error message.

3.2.3.9 Set<String> getVertices ( )

Returns a Set with all the graph vertices.

Returns

set with vertices.

3.2.3.10 String toString ( )

Returns a string with information about the Graph. Notice that vertices are printed in sorted order and information about adjacent edges is printed in sorted order (by vertex name). You may not use Collections.sort or Arrays.sort in order to implement this method. See the sample output for formatting details.

return string with graph information

The documentation for this class was generated from the following file:

- Graph.java

3.3 PrintCallBack<E> Class Reference

Inheritance diagram for PrintCallBack<E>:
3.3.1 Detailed Description

Implements a processor that appends the name of a vertex to a result string. It is used to generate the string associated with a BFS or DFS traversal. We append the data of the vertex to the vertex’s name.

The documentation for this class was generated from the following file:

- PrintCallBack.java
Index

addDirectedEdge
graphs::Graph<E>, 6

addVertex
graphs::Graph<E>, 6

CallBack<E>, 5

doBreadthFirstSearch
graphs::Graph<E>, 6
doDepthFirstSearch
graphs::Graph<E>, 7
doDijkstras
graphs::Graph<E>, 7

getAdjacentVertices
graphs::Graph<E>, 7

getCost
graphs::Graph<E>, 7

getData
graphs::Graph<E>, 8

getVertices
graphs::Graph<E>, 8

Graph
graphs::Graph<E>, 6

Graph<E>, 5

graphs::Graph<E>
addDirectedEdge, 6
addVertex, 6
doBreadthFirstSearch, 6
doDepthFirstSearch, 7
doDijkstras, 7
getAdjacentVertices, 7
getCost, 7
getData, 8
getVertices, 8

Graph, 6
toString, 8

PrintCallBack<E>, 8

toString
graphs::Graph<E>, 8