Graph Review

9 min

Let's put our Graph class to use and build out our train network. This time we will use it to map out all the train stations and routes that our rail service operates. We will also want to make sure that our routes can track the distance between each station.

For this exercise, we will build out the train network inside **trainNetwork.js**.

Instructions

1. Checkpoint 1 Passed

1.

Start by creating the train network as a weighted and directed Graph instance and assigning it to trainNetwork.

2. Checkpoint 2 Passed

2.

We just got funding to build out 6 train stations. Using the graph's .addVertex() method, add the following station vertices to our trainNetwork with the names:

- Los Angeles
- San Francisco
- New York
- Atlanta
- Denver
- Calgary

Hint

The .addVertex() returns the newly created verex. Don't forget to store the new vertices in variables. We will need the vertices so we can add edges between them later.

3. Checkpoint 3 Passed

3.

We only want to service the routes our customers will travel the most, so let's use .addEdge() to add the following route edges to the graph:

- From San Francisco to Los Angeles, which is 400mi
- From Los Angeles to San Francisco, which is 400mi
- From New York to Denver, which is 1800mi

- From Denver to New York, which is 1800mi
- From Calgary to Denver, which is 1000mi
- From Denver to Calgary, which is 1000mi
- From Los Angeles to Atlanta, which is 2100mi
- From Atlanta to Los Angeles, which is 2100mi

Hint

In order to create an edge between two vertices, you will need the vertices that were created in the 2nd checkpoint. Make sure the vertices that are returned from .addVertex() are set to variables.

4. Checkpoint 4 Passed

4.

Darn! As we were building out our routes, there was a huge snowstorm that hit Calgary and New York. We were able to salvage the route from Denver to New York, but all of the routes to and from Calgary broke down.

Using the graph's .removeEdge() and .removeVertex() methods, remove the route from New York to Denver, all the routes to and from Calgary, and the Calgary station.

5. Checkpoint 5 Passed

5.

We're finally all aboard the same page. Print out our final graph and check that we built the following routes:

- San Francisco to and from Los Angeles
- Los Angeles to and from Atlanta
- Denver to New York

This wraps up our graph implementation! There are still some edge (pardon the pun) cases that we have not yet accounted for. If you're feeling up for it, try to challenge yourself with the following:

- Currently, it is possible to add duplicate edges between two vertices. How will you improve this Graph implementation to avoid adding duplicate edges?
- How would you iterate through a directed graph? What about an undirected graph?
- How would you create a cycle with a directed graph?

trainNetwork.js

```
const Graph = require('./Graph.js');


const trainNetwork = new Graph(true, true);


const losAngeles = trainNetwork.addVertex('Los Angeles');
const sanFrancisco = trainNetwork.addVertex('San Francisco');
const newYork = trainNetwork.addVertex('New York');
const atlanta = trainNetwork.addVertex('Atlanta');
const denver = trainNetwork.addVertex('Denver');
const calgary = trainNetwork.addVertex('Calgary');


trainNetwork.addEdge(sanFrancisco, losAngeles, 400);
trainNetwork.addEdge(losAngeles, sanFrancisco, 400);
trainNetwork.addEdge(newYork, denver, 1800);
trainNetwork.addEdge(denver, newYork, 1800);
trainNetwork.addEdge(calgary, denver, 1000);
trainNetwork.addEdge(denver, calgary, 1000);
trainNetwork.addEdge(losAngeles, atlanta, 2100);
trainNetwork.addEdge(atlanta, losAngeles, 2100);


trainNetwork.removeEdge(newYork, denver);
trainNetwork.removeEdge(calgary, denver);
trainNetwork.removeEdge(denver, calgary);
trainNetwork.removeVertex(calgary);


trainNetwork.print();
```

Graph.js

```
const Edge = require('./Edge.js');
const Vertex = require('./Vertex.js');

class Graph {
  constructor(isWeighted = false, isDirected = false) {
    this.vertices = [];
    this.isWeighted = isWeighted;
    this.isDirected = isDirected;
  }

  addVertex(data) {
    const newVertex = new Vertex(data);
    this.vertices.push(newVertex);

    return newVertex;
  }

  removeVertex(vertex) {
    this.vertices = this.vertices.filter(v => v !== vertex);
  }

  addEdge(vertexOne, vertexTwo, weight) {
    const edgeWeight = this.isWeighted ? weight : null;

    if (vertexOne instanceof Vertex && vertexTwo instanceof Vertex) {
      vertexOne.addEdge(vertexTwo, edgeWeight);

      if (!this.isDirected) {
```

```
      vertexTwo.addEdge(vertexOne, edgeWeight);
    }
   } else {
    throw new Error('Expected Vertex arguments.');
   }
  }

  removeEdge(vertexOne, vertexTwo) {
   if (vertexOne instanceof Vertex && vertexTwo instanceof Vertex) {
    vertexOne.removeEdge(vertexTwo);

    if (!this.isDirected) {
     vertexTwo.removeEdge(vertexOne);
    }
   } else {
    throw new Error('Expected Vertex arguments.');
   }
  }

  print() {
   this.vertices.forEach(vertex => vertex.print());
  }
}

module.exports = Graph;
```

Vertex.js

```
const Edge = require('./Edge.js');

class Vertex {
  constructor(data) {
    this.data = data;
    this.edges = [];
  }

  addEdge(vertex, weight) {
    if (vertex instanceof Vertex) {
      this.edges.push(new Edge(this, vertex, weight));
    } else {
      throw new Error('Edge start and end must both be Vertex');
    }
  }

  removeEdge(vertex) {
    this.edges = this.edges.filter(edge => edge.end !== vertex);
  }

  print() {
    const edgeList = this.edges.map(edge =>
        edge.weight !== null ? `${edge.end.data} (${edge.weight})` : edge.end.data);

    const output = `${this.data} --> ${edgeList.join(', ')}`;
    console.log(output);
  }
}
```

```
module.exports = Vertex;
```

Edge.js

```
class Edge {
  constructor(start, end, weight = null) {
    this.start = start;
    this.end = end;
    this.weight = weight;
  }
}

module.exports = Edge;
```

Output-only Terminal

Output:

Los Angeles --> San Francisco (400), Atlanta (2100)

San Francisco --> Los Angeles (400)

New York -->

Atlanta --> Los Angeles (2100)

Denver --> New York (1800)