Removing Vertex Connections

7 min

Now that we can connect vertices together, we want to make the Graph more flexible by giving it the ability to remove connections.

We will use the .removeEdge()

Preview: Docs Loading link description

method

to remove any Edge between the given vertex instances.

Instructions

1. Checkpoint 1 Passed

1.

In our Vertex class, create the .removeEdge() method that expects an ending vertex parameter. In order to remove the edge that leads to the given vertex, iterate through its list of edges and filter out the Edge whose end property is strictly equal to the ending vertex.

Hint

To verify that we can successfully remove an edge between two vertices, create two Vertex instances under the Vertex class. Then add an edge going from the first vertex to the second vertex using the first vertex's .addEdge() method and giving it the second vertex.

Add another edge going in the other direction, from the second to the first vertex using the second vertex's .addEdge() method and giving it the first vertex.

Call the .removeEdge() method on the first vertex to remove the edge going to the second vertex.

Print out the connection by calling the first vertex's .print() method. We should see that there is no edge going from the first to the second vertex, but the edge going from the second to the first vertex still exists.

2. Checkpoint 2 Passed

2.

We're ready to remove an edge between vertices through our Graph class. In the Graph class, create the .removeEdge() method that removes the edge between two given vertices.

It should expect the vertices as two parameters: vertexOne and vertexTwo. Throw an error if either of them are not Vertex instances. Then use the vertices' .removeEdge() method to remove the edge between the other vertex. Remember to do this for both vertices.

3. Checkpoint 3 Passed

3.

Let’s verify that we can successfully remove an edge between two vertices through the Graph class. After the edge between Atlanta and New York is added, remove the edges between the two cities. Call the trainNetwork’s .removeEdge() with atlantaStation and newYorkStation.

We should see that the atlantaStation and newYorkStation vertices have no edge connections.

Vertex.js

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


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

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

 removeEdge(vertex) {
  this.edges = this.edges.filter(v => v.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;
```

Graph.js

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

class Graph {
  constructor() {
    this.vertices = [];
  }

  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) {
    if (vertexOne instanceof Vertex && vertexTwo instanceof Vertex) {
      vertexOne.addEdge(vertexTwo);
      vertexTwo.addEdge(vertexOne);
    } else {
      throw new Error('Expected Vertex arguments.');
    }
  }

  removeEdge(vertexOne, vertexTwo) {
     if (vertexOne instanceof Vertex && vertexTwo instanceof Vertex) {
      vertexOne.removeEdge(vertexTwo);
      vertexTwo.removeEdge(vertexOne);
    } else {
      throw new Error('Expected Vertex arguments.');
    }
  }

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

const trainNetwork = new Graph();
const atlantaStation = trainNetwork.addVertex('Atlanta');
const newYorkStation = trainNetwork.addVertex('New York');
trainNetwork.addEdge(atlantaStation, newYorkStation);
trainNetwork.removeEdge(atlantaStation, newYorkStation);
trainNetwork.print();
```

```
module.exports = Graph;
```

Edge.js

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

module.exports = Edge;
```