```
function Graph() {
  this.vertices = [];
  this.edges = [];
  this.numberOfEdges = 0;
}

Graph.prototype.addVertex = function(vertex) {
  this.vertices.push(vertex);
  this.edges[vertex] = [];
};
Graph.prototype.removeVertex = function(vertex) {
  var index = this.vertices.indexOf(vertex);
  if(~index) {
    this.vertices.splice(index, 1);
  }
  while(this.edges[vertex].length) {
    var adjacentVertex = this.edges[vertex].pop();
    this.removeEdge(adjacentVertex, vertex);
  }
};
Graph.prototype.addEdge = function(vertex1, vertex2) {
  this.edges[vertex1].push(vertex2);
  this.edges[vertex2].push(vertex1);
  this.numberOfEdges++;
};
Graph.prototype.removeEdge = function(vertex1, vertex2) {
  var index1 = this.edges[vertex1] ? this.edges[vertex1].indexOf(vertex2) : -1;
  var index2 = this.edges[vertex2] ? this.edges[vertex2].indexOf(vertex1) : -1;
  if(~index1) {
    this.edges[vertex1].splice(index1, 1);
    this.numberOfEdges--;
  }
  if(~index2) {
    this.edges[vertex2].splice(index2, 1);
  }
};
Graph.prototype.size = function() {
  return this.vertices.length;
};
Graph.prototype.relations = function() {
  return this.numberOfEdges;
};
Graph.prototype.traverseDFS = function(vertex, fn) {
  if(!~this.vertices.indexOf(vertex)) {
    return console.log('Vertex not found');
  }
  var visited = [];
  this._traverseDFS(vertex, visited, fn);
};
Graph.prototype._traverseDFS = function(vertex, visited, fn) {
  visited[vertex] = true;
  if(this.edges[vertex] !== undefined) {
    fn(vertex);
  }
  for(var i = 0; i < this.edges[vertex].length; i++) {
    if(!visited[this.edges[vertex][i]]) {
      this._traverseDFS(this.edges[vertex][i], visited, fn);
    }
  }
};
```

```
61 Graph.prototype.traverseBFS = function(vertex, fn) {
62   if(!~this.vertices.indexOf(vertex)) {
63     return console.log('Vertex not found');
64   }
65   var queue = [];
66   queue.push(vertex);
67   var visited = [];
68   visited[vertex] = true;
69
70   while(queue.length) {
71     vertex = queue.shift();
72     fn(vertex);
73     for(var i = 0; i < this.edges[vertex].length; i++) {
74       if(!visited[this.edges[vertex][i]]) {
75         visited[this.edges[vertex][i]] = true;
76         queue.push(this.edges[vertex][i]);
77       }
78     }
79   }
80 };
81 Graph.prototype.pathFromTo = function(vertexSource, vertexDestination) {
82   if(!~this.vertices.indexOf(vertexSource)) {
83     return console.log('Vertex not found');
84   }
85   var queue = [];
86   queue.push(vertexSource);
87   var visited = [];
88   visited[vertexSource] = true;
89   var paths = [];
90
91   while(queue.length) {
92     var vertex = queue.shift();
93     for(var i = 0; i < this.edges[vertex].length; i++) {
94       if(!visited[this.edges[vertex][i]]) {
95         visited[this.edges[vertex][i]] = true;
96         queue.push(this.edges[vertex][i]);
97         // save paths between vertices
98         paths[this.edges[vertex][i]] = vertex;
99       }
100    }
101  }
102  if(!visited[vertexDestination]) {
103    return undefined;
104  }
105
106  var path = [];
107  for(var j = vertexDestination; j != vertexSource; j = paths[j]) {
108    path.push(j);
109  }
110  path.push(j);
111  return path.reverse().join('-');
112 };
113 Graph.prototype.print = function() {
114   console.log(this.vertices.map(function(vertex) {
115     return (vertex + ' -> ' + this.edges[vertex].join(', ')).trim();
116   }, this).join(' | '));
117 };
118
119 var graph = new Graph();
120 graph.addVertex(1);
```

```
graph.addVertex(2);
graph.addVertex(3);
graph.addVertex(4);
graph.addVertex(5);
graph.addVertex(6);
graph.print(); // 1 -> | 2 -> | 3 -> | 4 -> | 5 -> | 6 ->
graph.addEdge(1, 2);
graph.addEdge(1, 5);
graph.addEdge(2, 3);
graph.addEdge(2, 5);
graph.addEdge(3, 4);
graph.addEdge(4, 5);
graph.addEdge(4, 6);
graph.print(); // 1 -> 2, 5 | 2 -> 1, 3, 5 | 3 -> 2, 4 | 4 -> 3, 5, 6 | 5 -> 1, 2, 4 | 6 -> 4
console.log('graph size (number of vertices):', graph.size()); // => 6
console.log('graph relations (number of edges):', graph.relations()); // => 7
graph.traverseDFS(1, function(vertex) { console.log(vertex); }); // => 1 2 3 4 5 6
console.log('---');
graph.traverseBFS(1, function(vertex) { console.log(vertex); }); // => 1 2 5 3 4 6
graph.traverseDFS(0, function(vertex) { console.log(vertex); }); // => 'Vertex not found'
graph.traverseBFS(0, function(vertex) { console.log(vertex); }); // => 'Vertex not found'
console.log('path from 6 to 1:', graph.pathFromTo(6, 1)); // => 6-4-5-1
console.log('path from 3 to 5:', graph.pathFromTo(3, 5)); // => 3-2-5
graph.removeEdge(1, 2);
graph.removeEdge(4, 5);
graph.removeEdge(10, 11);
console.log('graph relations (number of edges):', graph.relations()); // => 5
console.log('path from 6 to 1:', graph.pathFromTo(6, 1)); // => 6-4-3-2-5-1
graph.addEdge(1, 2);
graph.addEdge(4, 5);
console.log('graph relations (number of edges):', graph.relations()); // => 7
console.log('path from 6 to 1:', graph.pathFromTo(6, 1)); // => 6-4-5-1
graph.removeVertex(5);
console.log('graph size (number of vertices):', graph.size()); // => 5
console.log('graph relations (number of edges):', graph.relations()); // => 4
console.log('path from 6 to 1:', graph.pathFromTo(6, 1)); // => 6-4-3-2-1

```