Due to the print book page limit, we cannot inlcude all good CheckPoint questions in the physical book. The CheckPoint on this Website may contain extra questions not printed in the book. The questions in some sections may have been reordered as a result. Nevertheless, it is easy to find the CheckPoint questions in the book on this Website. Please send suggestions and errata to Dr. Liang at y.daniel.liang@gmail.com. Indicate the book, edition, and question number in your email. Thanks!

Chapter 28 Check Point Questions

Section 28.2

▼28.2.1

What is the famous Seven Bridges of Königsberg problem?

See §28.1.

Hide Answer Read Answer

▼28.2.2

What is a graph? Explain the following terms: undirected graph, directed graph, weighted graph, degree of a vertex, parallel edge, simple graph, complete graph, connected graph, cycle, subgraph, tree, and spanning tree.

See §28.2.

Hide Answer Read Answer

▼28.2.3

How many edges are in a complete graph with 5 vertices? How many edges are in a tree of 5 vertices?

A complete graph of n vertices has 10 edges. A tree always has 4 edges.

Hide Answer Read Answer

▼28.2.4

How many edges are in a complete graph with n vertices? How many edges are in a tree of n vertices?

A complete graph of n vertices has n(n-1)/2 edges. A tree always has n-1 edges.

Hide Answer Read Answer

Section 28.3

▼28.3.1

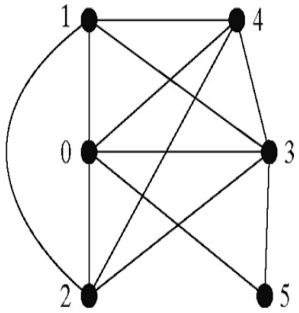
How do you represent vertices in a graph? How do you represent edges using an edge array? How do you represent an edge using an edge object? How do you represent edges using an adjacency matrix? How do you represent edges using adjacency lists?

See §28.3.

Hide Answer Read Answer

▼28.3.2

Represent the following graph using an edge array, a list of edge objects, an adjacency matrix, an adjacency vertex list, and an adjacency edge list, respectively.



Edge array:

```
int[][] edges = {
  {0, 1}, {0, 2}, {0, 3}, {0, 4}, {0, 5},
  {1, 0}, {1, 2}, {1, 3}, {1, 4},
  {2, 0}, {2, 1}, {2, 3}, {2, 4},
  {3, 0}, {3, 1}, {3, 2}, {3, 4}, {3, 5},
  {4, 0}, {4, 1}, {4, 2}, {4, 3},
  {5, 0}, {5, 3}
}
```

List of edge objects:

```
java.util.ArrayList<Edge> list = new java.util.ArrayList<Edge>();
list.add(new Edge(0, 1));
list.add(new Edge(0, 2));
list.add(new Edge(0, 3));
list.add(new Edge(0, 4));
list.add(new Edge(0, 5));
…
…
…
```

Adjacency matrix:

```
int[][] adjacencyMatrix = {
  {0, 1, 1, 1, 1, 1}, // node 0
  {1, 0, 1, 1, 1, 0}, // node 1
  {1, 1, 0, 1, 1, 0}, // node 2
  {1, 1, 1, 0, 1, 1}, // node 3
  {1, 1, 1, 1, 0, 0}, // node 4
  {1, 0, 0, 1, 0, 0}  // node 5
};
```

Adjacency list:

```
LinkedList<Integer> list[] = new LinkedList<>();
list[0].add(1); list[0].add(2); list[0].add(3); list[0].add(4);
            list[0].add(5);
list[1].add(0); list[1].add(2); list[1].add(3); list[1].add(4);
list[2].add(0); list[2].add(1); list[2].add(3); list[2].add(4);
list[3].add(0); list[3].add(1); list[3].add(2); list[3].add(4);
            list[3].add(5);
list[4].add(0); list[4].add(1); list[4].add(2); list[4].add(3);
list[5].add(0); list[5].add(3);
```

Adjacency edge list:

```
ArrayList<ArrayList<Edge>> list = new ArrayList<>();
list.add(new ArrayList<>());
list.get(0).add(new Edge(0, 1));
list.get(0).add(new Edge(0, 2));
list.get(0).add(new Edge(0, 3));
list.get(0).add(new Edge(0, 4));
list.get(0).add(new Edge(0, 5));

list.add(new ArrayList<>());
list.get(1).add(new Edge(1, 0));
list.get(1).add(new Edge(1, 2));
list.get(1).add(new Edge(1, 3));
list.get(1).add(new Edge(1, 4));

list.add(new ArrayList<>());
list.get(2).add(new Edge(2, 1));
list.get(2).add(new Edge(2, 2));
list.get(2).add(new Edge(2, 3));
list.get(2).add(new Edge(2, 4));

list.add(new ArrayList<>());
list.get(3).add(new Edge(3, 0));
list.get(3).add(new Edge(3, 1));
list.get(3).add(new Edge(3, 2));
list.get(3).add(new Edge(3, 4));
list.get(3).add(new Edge(3, 5));

list.add(new ArrayList<>());
list.get(4).add(new Edge(4, 0));
list.get(4).add(new Edge(4, 1));
list.get(4).add(new Edge(4, 2));
list.get(4).add(new Edge(4, 3));

list.add(new ArrayList<>());
list.get(5).add(new Edge(5, 0));
list.get(5).add(new Edge(5, 3));
```

Hide Answer Read Answer

Section 28.4

▼28.4.1

Describe the relationships between Graph, UnweightedGraph, and WeightedGraph.

Graph is an interface that defines the common features for graphs. UnweightedGraph is a concreate graph that represents unweighted graphs. WeightedGraph is a subtype of UnweightedGraph that represents weighted graphs.

Hide Answer Read Answer

▼28.4.2

For the code in Listing 28.2, TestGraph.java, what is graph1.getIndex("Seattle")? What is graph1.getDegree(5)? What is graph1.getVertex(4)?

What is graph1.getIndex("Seattle")? 0
What is graph1.getDegree(5)? 5

What is graph1.getVertex(4)? Kansas City

Hide Answer Read Answer

▼28.4.3

Show the output of the following code.

```
public class Test {
  public static void main(String[] args) {
    Graph<Character> graph = new UnweightedGraph<>();
    graph.addVertex('U');
    graph.addVertex('V');
    int indexForU = graph.getIndex('U');
    int indexForV = graph.getIndex('V');
    System.out.println("indexForU is " + indexForU);
    System.out.println("indexForV is " + indexForV);
    System.out.println("Degree of U is " +
      graph.getDegree(indexForU));
    System.out.println("Degree of V is " +
      graph.getDegree(indexForV));
  }
}
```

indexForU is 0
indexForV is 1
Degree of U is 1
Degree of V is 0

Hide Answer Read Answer

▼28.4.4

What will getIndex(v) return if v is not in the graph? What happens to getVertex(index) if index is not in the graph? What happens to addVertex(v) if v is already in the graph? What happens to addEdge(u, v) if u or v is not in the graph?

getIndex(v) returns -1 if v is not in the graph.
What happens to getVertex(index) if index is not in the graph?
IndexOutBoundsException
What happens to addVertex(v) if v is already in the graph? v is not added to graph and the method returns false.
What happens to addEdge(u, v) if u or v is not in the graph? Throws IlleagalArgumentException.

Hide Answer Read Answer

Section 28.5

▼28.5.1

Will Listing 28.7 DisplayUSMap.java work, if the code in lines 38-42 in Listing 28.6 GraphView.java is replaced by the following code?

```
if (i < v) {
  double x2 = graph.getVertex(v).getX();
  double y2 = graph.getVertex(v).getY();
```

```
    // Draw an edge for (i, v)
    getChildren().add(new Line(x1, y1, x2, y2));
  }
```

Yes. Because the graph in Figure 18.1 is undirected. The edge from v to u and the edge from u to v are both contained in the graph. You need to draw just one such edge.

Hide Answer Read Answer

▼28.5.2

For the graph1 object created in Listing 28.1, TestGraph.java, can you create a GraphView object as follows?

```
GraphView view = new GraphView(graph1);
```

No, because graph1 is not an instance of Displayable.

Hide Answer Read Answer

Section 28.6

▼28.6.1

Does UnweightedGraph<V>.SearchTree implement the Tree interface defined in Listing 25.3 Tree.java?

No.

Hide Answer Read Answer

▼28.6.2

What method do you use to find the parent of a vertex in the tree?

The getParent(int index) method.

Hide Answer Read Answer

Section 28.7

▼28.7.1

What is depth-first search?

See the text.

Hide Answer Read Answer

▼28.7.2

Draw a DFS tree for the graph in Figure 28.3b starting from node A.

See the text.

Hide Answer Read Answer

▼28.7.3

Draw a DFS tree for the graph in Figure 28.1 starting from vertex Atlanta.

See the text.

Hide Answer Read Answer

▼28.7.4

What is the return type from invoking dfs(v)?

The return is an instance of AbstractGraph.Tree.

Hide Answer Read Answer

▼28.7.5

The depth-first search algorithm described in Listing 28.8 uses recursion. Alternatively, you can use a stack to implement it, as shown below. Point out the error in this algorithm and give a correct algorithm.

```
// Wrong version
Tree dfs(vertex v) {
  push v into the stack;
  mark v visited;

  while (the stack is not empty) {
    pop a vertex, say u, from the stack
    visit u;
    for each neighbor w of u
      if (w has not been visited)
        push w into the stack;
  }
}
```

There are several problems in this algorithm: 1. There is a possibility that a vertex may be pushed into the stack more than once. Can you give such an example? 2. The neighbors of a vertex is pushed into the stack. You should only push one neighbor into the stack and continue to search for the next vertex from this new vertex. The correct algorithm is given below:

```
* Input: G = (V, E) and a starting vertex v
* Output: A DFS tree rooted at v
*
* Tree dfs(v) {
*   push v to the stack;
*   mark x visited;
*   while (the stack is not empty) {
*     peek a vertex from the stack, say x;
*     if (x still has an unvisited neighbor, say y) {
*       parent[y] = x;
*       push y into the stack;
*       mark y visited;
*     }
*     else {
*       pop a vertex from the stack;
*     }
*   }
* }
```

Hide Answer Read Answer

Section 28.8

▼28.8.1

How is a graph created for the connected circles problem?

Each circle corresponds to a vertex and the vertices are connected if the corresponding two circles overlap.

Hide Answer Read Answer

▼28.8.2

When you click the mouse inside a circle, does the program create a new circle?

No.

Hide Answer Read Answer

▼28.8.3

How does the program know if all circles are connected?

This is done by performing a DFS. If the DFS tree contains the same number of vertices as the number of vertices, the graph is connected and all the circles are connected.

Hide Answer Read Answer

Section 28.9

▼28.9.1

What is the return type from invoking bfs(v)?

The return is an instance of AbstractGraph.Tree.

Hide Answer Read Answer

▼28.9.2

What is breadth-first search?

See the text.

Hide Answer Read Answer

▼28.9.3

Draw a BFS tree for the graph in Figure 28.3b starting from node A.

See the text.

Hide Answer Read Answer

▼28.9.4

Draw a BFS tree for the graph in Figure 28.1 starting from vertex Atlanta.

See the text.

Hide Answer Read Answer

▼28.9.5

Prove that the path between the root and any node in the BFS tree is the shortest path between the root and the node.

You may prove it using induction on the path length.

Hide Answer Read Answer

Section 28.10

▼28.10.1

How are the nodes created for the graph in NineTailModel?

See the text.

Hide Answer Read Answer

▼28.10.2

How are the edges created for the graph in NineTailModel?

See the text.

Hide Answer Read Answer

▼28.10.3

What is returned after invoking getIndex("HTHTTTHHH".toCharArray()) in Listing 28.13? What is returned after invoking getNode(46) in Listing 28.13?

getIndex("HTHTTTHHH".toCharArray()) returns 184.
getNode(46) returns HHHTHTTTH

Hide Answer Read Answer

▼28.10.4

If lines 26 and 27 are swapped in Listing 28.13, NineTailModel.java, will the program work? Why not?

It will not work, because node is changed in the getFlippedNode(node, k).

Hide Answer Read Answer